Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 7316954, 13 pages
https://doi.org/10.1155/2018/7316954

Research Article

Deep Residual Bidir-LSTM for Human Activity Recognition Using Wearable Sensors

Yu Zhao,[1] **Rennong Yang**,[1] **Guillaume Chevalier**,[2] **Ximeng Xu**,[1] **and Zhenxing Zhang**[1]

[1]*Aeronautics and Astronautics Engineering College, Air Force Engineering University, Xi'an 710038, China*
[2]*Laval University, 2325 Rue de l'Universite, Quebec G1V 0A6, Canada*

Correspondence should be addressed to Yu Zhao; zhaoyuafeu@gmail.com

Received 19 June 2018; Accepted 9 December 2018; Published 30 December 2018

Academic Editor: Javier Martinez Torres

Human activity recognition (HAR) has become a popular topic in research because of its wide application. With the development of deep learning, new ideas have appeared to address HAR problems. Here, a deep network architecture using residual bidirectional long short-term memory (LSTM) is proposed. The advantages of the new network include that a bidirectional connection can concatenate the positive time direction (forward state) and the negative time direction (backward state). Second, residual connections between stacked cells act as shortcut for gradients, effectively avoiding the gradient vanishing problem. Generally, the proposed network shows improvements on both the temporal (using bidirectional cells) and the spatial (residual connections stacked) dimensions, aiming to enhance the recognition rate. When testing with the Opportunity dataset and the public domain UCI dataset, the accuracy is significantly improved compared with previous results.

1. Introduction

In real life, many problems can be described as time series problems. Indeed, human activity recognition (HAR) is of value in both theoretical research and actual practice. It can be used widely, including health monitoring [1, 2], smart homes [3, 4], and human–computer interactions [5, 6]. For example, long short-term memory (LSTM) networks are a good choice for solving HAR problems. Unlike traditional algorithms, LSTM is able to catch relationship in data on the temporal dimension without mixing the time steps together as convolutional neural network (CNN). As more of what is commonly called "big data" emerges, LSTM network offers great performance and many potential applications.

More specifically, HAR is a process of obtaining action data with sensors. It symbolizes action information and then allows understanding and extraction of the motion characteristics, which is what activity recognition refers to. Because of the spatial complexity and temporal divergence of behavior, there is no unified recognition method. A public domain benchmark of HAR has been introduced, and different methods of recognition have been analyzed [7]. The results showed that the K-nearest neighbor (KNN) algorithm outperformed other algorithms in most recognition tasks. Support vector machine (SVM) is another outstanding algorithm. A multiclass hardware-friendly support vector machine (MC-HF-SVM), which uses fixed-point arithmetic for HAR instead of the typical floating-point arithmetic, has been proposed for sensor data [8]. Unlike the manual filtering features in previous algorithms, a systematic feature learning method that combines feature extraction with CNN has also been proposed [9]. Subsequently, Deep ConvLSTM networks outperformed previous algorithms in the Opportunity Challenge by an average of 4% of the F1 score [10].

Although researchers have made great strides in HAR, room for improvement remains. Inspired by previous neural networks, we describe a novel deep residual bidirectional long short-term memory LSTM (Res-Bidir-LSTM) network. The deep architecture has improved learning ability and, despite the time required to reach maximum accuracy, shows good accuracy early in training. It is especially suitable for complex, large-scale HAR problems where sensor fusion is

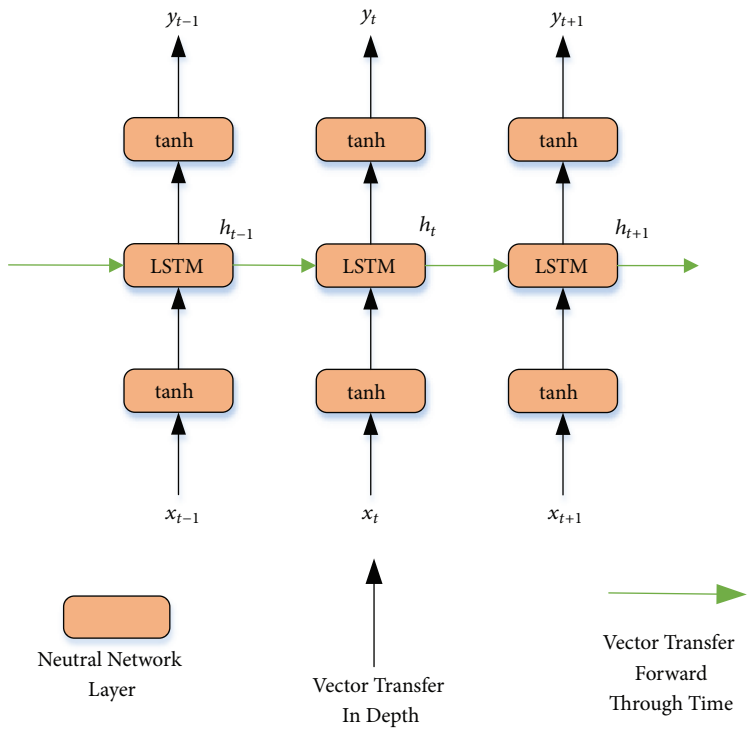

Figure 1: The unfolded structure of one-layer baseline LSTM is shown. Baseline LSTM structure operating through the time axis, from left to right.

required. Residual connections and bidirectional communication through time are available to ensure the integrity of information flowing deeply through the neural network.

In recent years, deep learning has shown applicability to many fields, such as image processing [11, 12], speech recognition [13–15], and natural language processing [16, 17]. In ILSVRC 2012, AlexNet, proposed by Krizhevsky [18], took the first place. Since then, deep learning has been considered to be applicable to solving real problems and has done so with impressive accuracy. Indeed, deep learning has become a popular area for scientists and engineers.

Another event in 2016 that drew considerable attention was the century man–machine war at the end of the game in which AlphaGo achieved victory. This event also demonstrated that deep learning, based on big data, is a feasible way to solve the nondeterministic polynomial problem.

LSTM netowrk, first proposed by Juergen Schmidhuber in 1997 [19], is variants of recurrent neural networks (RNNs). It has special inner gates that allow for consistently better performance than RNN for time series. Compared with other networks, such as CNN, restricted Boltzmann machine (RBM), and auto-encoder (AE), the structure of the LSTM renders it especially good at solving problems involving time series, such as those related to natural language processing, speech recognition, and weather prediction, because its design enables gradients to flow through time readily.

Section 2 presents the theory of baseline LSTM, Bidir-LSTM, and residual networks. In Section 3, we provide an explicit introduction to the preprocessing in HAR and describe Res-Bidir-LSTM. Several experiments are performed with HAR benchmarks: the public domain UCI dataset and the Opportunity dataset. We compare the accuracy of recognition of our algorithm with those of other algorithm. Finally, we summarize the research and discuss our future work.

2. Background

2.1. Baseline LSTM. LSTM [18] is an extension of recurrent neural networks. Due to its special architecture, which combats the vanishing and exploding gradient problems, it is good at handling time series problems up to a certain depth.

In Figure 1, we define the input set as $\{x_0, x_1, \ldots, x_t, x_{t+1}, \ldots\}$, the output set as $\{y_0, y_1, \ldots, y_t, y_{t+1}, \ldots\}$, and hidden layers as $\{h_0, h_1, \ldots, h_t, h_{t+1}, \ldots\}$. Then, $\boldsymbol{U}, \boldsymbol{W}, \boldsymbol{V}$ denote weight metrics from the input layer to the hidden layer, from the hidden layer to the hidden layer, and from the hidden layer to the output layer, respectively. The transfer process of the network can be described as follows: the input tensor is transformed, along with the tensor of the hidden layer (at the last stage), to the hidden layer by a matrix transformation. Then, the output of the hidden layer passes through an activation function to the final value of the output layer.

Formally, outputs of the hidden layer and output layer can be defined as follows:

$$h_i = \begin{cases} g\left(\boldsymbol{U}x_i + b_i^h\right) & i = 0 \\ g\left(\boldsymbol{U}x_i + \boldsymbol{W}h_{i-1} + b_i^h\right) & i = 1, 2, \ldots \end{cases} \quad (1)$$

$$y_i = g\left(\boldsymbol{V}h_i + b_i^y\right) \quad i = 0, 1, \ldots$$

where $g(\cdot)$ represents tanh function.

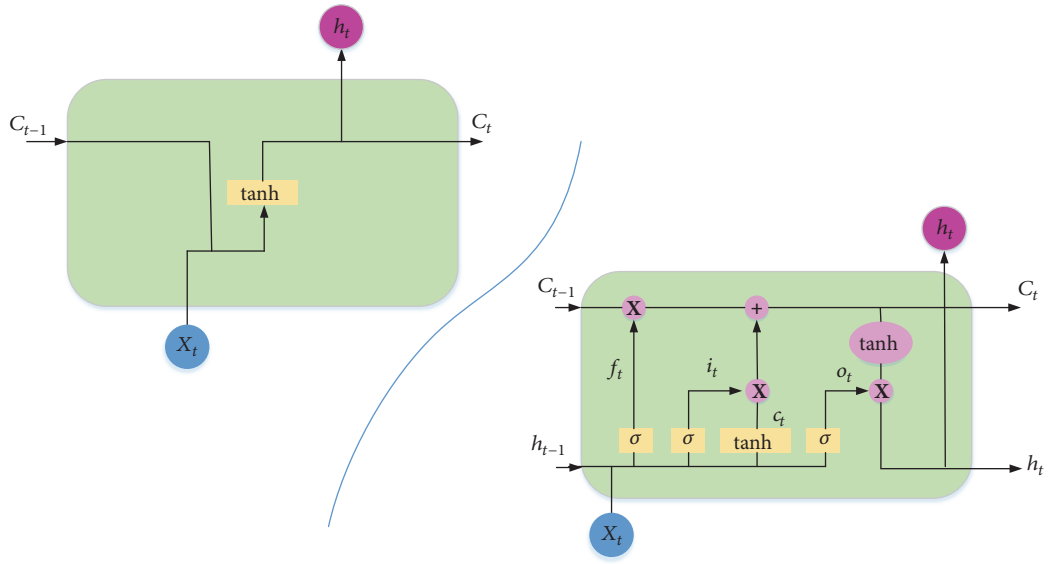

FIGURE 2: Comparison between a RNN and a LSTM cell. Notations are operators and arrows are information flows. The three σ operators from left to right are forget gate, input gate, and output gate, respectively. All the three gates have the same operator in form and the same inputs. But they have different weights and biases, which also leads to different values of forgot information f_t, input information i_t, and output information o_t. More details are described in text and (2).

In theory, RNN can estimate the output of current time based on past information. However, Bengio [20] found that RNN could remember the information for only a short time, because of the vanishing and exploding gradient problems. When backpropagation with a deep network is used, gradients will vanish rapidly if preventative measures that permit gradients to flow deeply are not taken. Compared with the simple input concatenation and activation used in RNNs, LSTM has a particular structure for remembering information for a longer time as an input gate and a forget gate control how to overwrite the information by comparing the inner memory with the new information arriving. It enables gradients to flow through time easily.

As shown in Figure 2, the input gate, the forget gate, and the output gate of LSTM are designed to control what information should be forgotten, remembered, and updated. Gating is a method to selectively pass the information that is needed. It consists of a sigmoid function and an element-wise multiplication function. The output value is within $[0, 1]$ to allow the multiplication to then happen to let information flow or not. It is considered good practice to initialize these gates to a value of 1, or close to 1, so as to not impair training at the beginning.

In the LSTM cell, each parameter at moment t can be defined as follows:

$$
\begin{aligned}
f_t &= \sigma\left(\boldsymbol{W}_f\left[h_{t-1}, x_t\right] + b_f\right) \\
i_t &= \sigma\left(\boldsymbol{W}_i\left[h_{t-1}, x_t\right] + b_i\right) \\
\widetilde{C}_t &= \tanh\left(\boldsymbol{W}_c\left[h_{t-1}, x_t\right] + b_C\right) \\
C_t &= f_t C_{t-1} + i_t \widetilde{C}_t \\
o_t &= \sigma\left(\boldsymbol{W}_o\left[h_{t-1}, x_t\right] + b_o\right) \\
h_t &= o_t \tanh\left(C_t\right)
\end{aligned}
\tag{2}
$$

The information process of LSTM cell is described in (2). First, there is a need to forget old information, which involves the forget gate. The next step is to determine what new information needs to keep in memory with an input gate. From that, it is possible to update the old cell state, C_{t-1}, to the new cell state, C_t. Finally, it decides which information should be output to the layer above with an output gate.

2.2. Bidirectional LSTM. In real life, human trajectories are continuous. Baseline LSTM cells predict the current status based only on former information. It is clear that some important information may not be captured properly by the cell if it runs in only one direction.

The improvement in bidirectional LSTM is that the current output is not only related to previous information but also related to subsequent information. For example, it is appropriate to predict a missing word based on context. Bidirectional LSTM [21] is made up of two LSTM cells, and the output is determined by the two together.

In Figure 3, there are forward sequences $\overrightarrow{\boldsymbol{h}}$ and backward sequences $\overleftarrow{\boldsymbol{h}}$ in the hidden layer. For the moment $t(t = 0, 1, 2\ldots)$, the hidden layer and the input layer can be defined as follows:

$$
\begin{aligned}
\overrightarrow{h}_t &= g\left(\boldsymbol{U}_{\overrightarrow{h}} x_t + \boldsymbol{W}_{\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right) \\
\overleftarrow{h}_t &= g\left(\boldsymbol{U}_{\overleftarrow{h}} x_t + \boldsymbol{W}_{\overleftarrow{h}} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}\right) \\
y_t &= g\left(\boldsymbol{V}_{\overrightarrow{h}} \overrightarrow{h}_t + \boldsymbol{V}_{\overleftarrow{h}} \overleftarrow{h}_t + b_y\right)
\end{aligned}
\tag{3}
$$

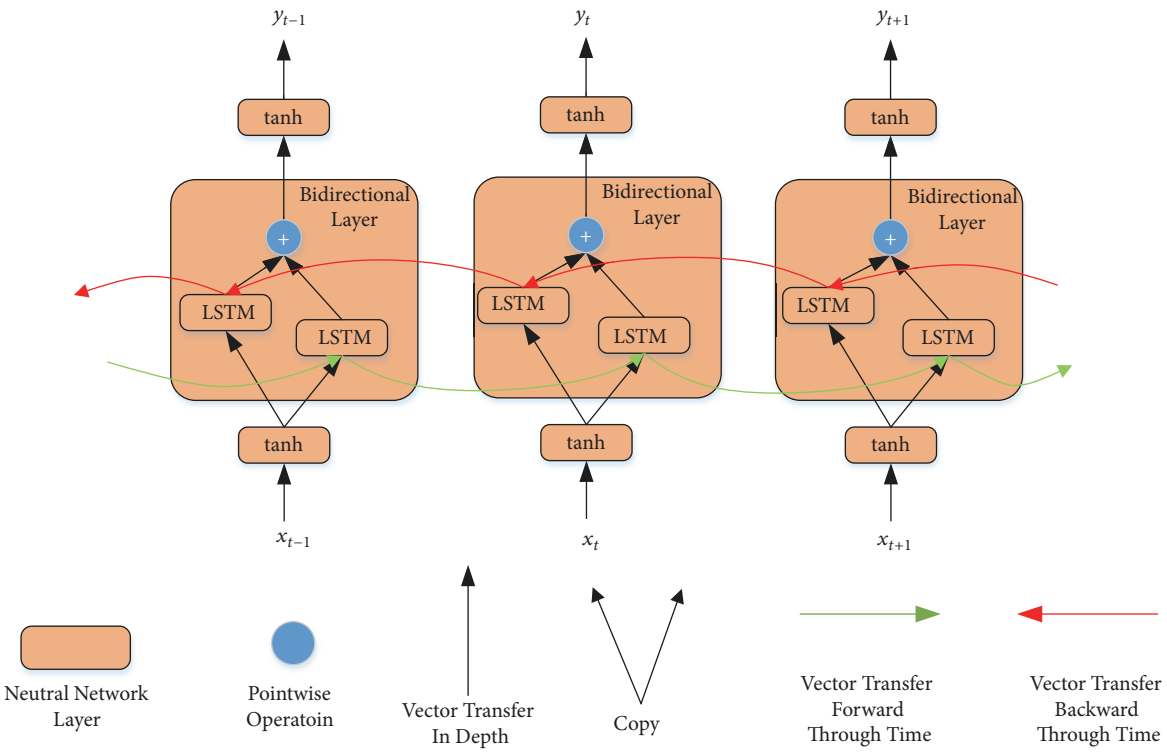

FIGURE 3: Standard bidirectional LSTM structure. For a bidirectional layer, it gets information from vertical direction (lower layer) and horizontal direction (past and future) and finally outputs the processed information for the upper layer.

Our bidirectional LSTM cell differs slightly from this. We concatenate the results of the two h_t to then reduce the number of features in half with a ReLU fully connected hidden layer as follows:

$$y_t = ReLU\left(\boldsymbol{W} * concat\left(\overrightarrow{h}_t, \overleftarrow{h}_t\right) + b\right), \tag{4}$$

where $concat(\cdot)$ means concatenating sequences.

2.3. Residual Network. The Microsoft research Asia (MSRA) team built a 152-layer network, which is about eight times that of the VGG network [22]. Due to its excellent performance, they took first place in the 2015 ILSVRC competition owing to an absolute advantage in image classification, image location, and image detection.

As the network deepens, the research emphasis shifts to how to overcome the obstruction of information and gradient transmission. The MSRA uses residual networks. The main idea is that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. Compared to highway network [23], residual network has no gates but residual connections. So the network is parameter-free and the shortcut will never close. In addition, all information is always passed through with additional residual functions to be learned.

An important advantage of residual networks is that they are much easier to train because the gradients can be passed through the layers more directly with the addition operator that enables them to bypass some layers that would have otherwise been restrictive. It enables both better training and a deeper network, because residual connections do not impede gradients and still contribute to refining the output of several stacked layers composed of such residual connections. On top of a collection of residual connections is a bottleneck where the next layers stop being residual and where batch normalization is generally applied to normalize and restrict the usage of the feature space represented by the layer [24].

A residual network is shown in Figure 4. The lower information can transmit to upper layer directly through shortcut connection, which increases the freedom of the information flowing. The shortcut connects many supplementary $n(n = 0, 1, 2, \ldots)$ layers in height before the bottleneck. When n equals 0, there is no residual connection: it becomes like the baseline deep-stacked LSTMs layers. In Figure 4, n is 0, and the output of the hidden layer $i(i = 1, 2, \ldots, \mathrm{L})$ can be defined as follows:

$$\begin{aligned} \boldsymbol{h}_1 &= \sigma\left(W_1\boldsymbol{x} + b_1\right) \quad i = 1 \\ \boldsymbol{h}_i &= \sigma\left(W_i\boldsymbol{h}_{i-1} + b_i\right) + \boldsymbol{h}_{i-2} \quad i = 2, 3, \ldots, L-1 \\ \boldsymbol{y} &= \sigma\left(W_y\boldsymbol{h}_i + b_y\right) + \boldsymbol{h}_{i-1} \quad i = L \end{aligned} \tag{5}$$

In the code implementation, indexing in the configuration file starts at 1 rather than 0, because we include the count of the first layer that acts as a basis before the residual cells. The same counting rule applies for indicating how many blocks of residual connection stacked one on top of the other.

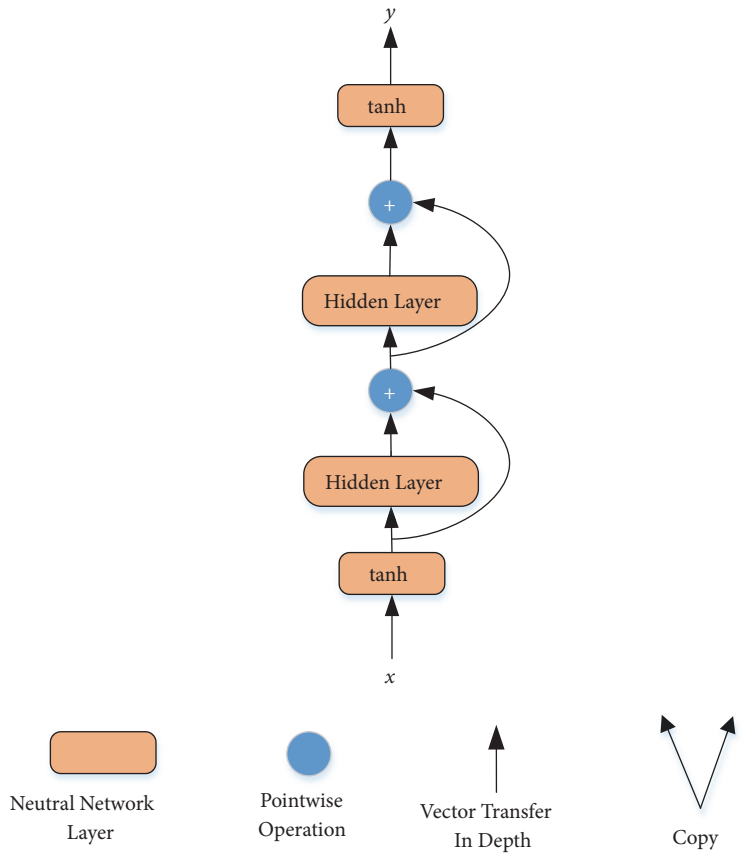

Figure 4: Isolated residual neural network.

3. Our Model: Deep Residual Bidir-LSTM Network

3.1. Process Pipeline for HAR. The process pipeline of HAR can be divided into three parts: preprocessing, training, and testing. In our case, testing is modified in parallel with training. First, testers perform activities of daily living with wearable sensors and gathered information to form the raw dataset. The missing data is interpolated and normalized to a mean of zero and standard deviation of 0.5. We then reshape the data to fit the designed network, with windows of 128 time steps. The dataset is split into training and testing datasets.

Second, a training tensor is added to the designed network so it could output a prediction. The difference between the predicted value and the real value is compared with a sigmoid cross entropy loss with L2 weight decay to backpropagate errors backward into the network layer by layer with the Adam optimizer [25]. Thus, we could adjust the hyperparameters in networks, such as the learning rate and L2 weight decay, to reduce the difference.

Finally, during testing, we add the testing tensor to the neural network without affecting the learned parameters, so as to not corrupt the test. Testing does not affect the training. Predictions obtained from the neural network are compared with the real values. The metrics of accuracy and of the F1 score of HAR are calculated throughout learning. Both the best in-training metrics and the final metrics obtained are kept for comparison.

3.2. Architecture of Deep Res-Bidir-LSTM Network. Inspired by the networks in Section 2, we propose the Res-Bidir-LSTM to deal with HAR. Although residual connections for CNN have been used [22], the methodology is also available for LSTM.

Similar to building blocks, we select modules and combine them to build a network based on our mission. The input of HAR should be a time series, and the basic structure of the LSTM guarantees that it can preserve the characteristics on the temporal dimension.

Additionally, a large network can be optimized correctly for a problem with sufficient regularization, such as L2 weight decay and dropout. However, if no regularization is used, results trend to overfitting and bad operations on the test set. Complexity is good but only if countered with regularization. Too many layers and cells per layer will increase the computational complexity and waste computational resources. When the layer number and cell number reach a certain scale, the recognition accuracy will remain at a certain scale instead of increasing. By adding more depth, regularization is then needed to avoid overfitting while still improving accuracy.

Our deep LSTM neural network is limited in terms of how many data points it can access: it has access to only 128 time steps when making its predictions. Especially

when deepened, the next forward/backward duo will see output from the other pass "in advance", because, logically, a backward pass for our bidirectional LSTM reverses the input and the output before the concatenation. Thus, the Bidir-LSTM has the same input and output shape as the baseline LSTM. But at a given time step, it has access to more information in advance because of the backward passes.

In general, gradient vanishing is a widespread problem for deep networks. The residual, bidirectional, and stacked layers (hence, the name "Deep Residual Bidir-LSTM") help counter this problem, because some bottom layers would otherwise be too hard to optimize when using backpropagation. Combined with batch normalization on the top of each residual layer, residual connections act as shortcut for gradients. It prevents restrictions in the hidden layer feature space from being too complex and avoids outlier values at test time, against overfitting.

In Figure 5, the information flows in the horizontal direction (temporal dimension) and in the vertical direction (depth dimension). With the exception of the input and output layers, there are 2 hidden layers which have residual connection inside (hence, called "residual layer"). Moreover, each residual layer contains 2 bidirectional layers. The network in Figure 5 has 2×2 architecture, which can also be thought of as 8 LSTM cells in sum. In our network, the activity function is unified with ReLU, because it always outperforms tanh with deep networks to counter gradient vanishing. Although the output is a tensor for a given time window T, the time axis has been crunched by the neural network. That is, we need only the last element of the output and can discard the others. Thus, only the gradient from the prediction at the last time step is applied. This also causes a LSTM cell to be unnecessary: the uppermost backward LSTM in the bidirectional pass. Hopefully, this is not of great concern because TensorFlow should evaluate what to compute and what not to compute. Additionally, the training dataset should be shuffled during the training process. The state of the neural network is reset at each new window for each new prediction.

3.3. Tricks for Optimization. Our Res-Bidir-LSTM for HAR makes it possible to see that the accuracy during testing is much worse than that during training. Overfitting is likely to occur, and balancing the regularization hyper-parameters becomes difficult because they are so numerous. The L2 norm of the weights for weight decay is added in the loss function.

Also, dropout is applied between each layer on the depth axis or, sometimes, just at the output, depending on what is specified in the configuration file, which is another hyper-parameter. Dropout refers to the fact that parts of tensors that are output by the hidden layer are shut down to a zero value to a certain probability for each value in each training epoch, while other values scale up accordingly to keep the same geometric norm of the tensor's values. The inoperative nodes can be regarded as dead nodes (or neurons) that are temporarily not in the network, which means that the weights and biases behind these dead notes temporarily neither learns nor contributes to the predictions during that training step for a batch. The weights are kept intact.

To avoid a sudden leveling off in the accuracy during learning, gradient clipping [24] is added with a maximal gradient step norm of 15. The threshold, $v(v > 0)$, for the gradient helps not to overshoot the weight update during training due to having sharp cliffs in the weight space, a characteristic of RNNs:

$$\text{if } \|g\| > v \quad g \longleftarrow \frac{gv}{\|g\|}, \tag{6}$$

where g is the gradient and $\|g\|$ is normed gradient.

Batch normalization [26] can also be useful in training residual connections. The fundamental idea of batch normalization is that layers are simply normalized by mean and variance such that they have a mean of zero and a standard deviation of 1 over the whole batch. One big rescaling factor multiplies the whole batch, and one big bias is also added. The result is then normalized and offset in a linear fashion. The scaling multiplier α and the offset parameter β are learned to rescale inputs in a custom way, and β is initialized to 1, as is commonly done. The formula can be defined as follows:

$$\begin{aligned} \hat{x}^{(k)} &= \frac{x^{(k)} - E\left[x^{(k)}\right]}{\sqrt{\text{var}\left[x^{(k)}\right]}} \\ y^{(k)} &= \alpha^{(k)} \hat{x}^{(k)} + \beta^{(k)}, \end{aligned} \tag{7}$$

where $x^{(k)}$ means the k_{th} parameter in the parameters vector and $y^{(k)}$ is normed value of $x^{(k)}$.

We add many tricks to the network to provide better results. Generally, L2 norm for weight decay and dropout are used to prevent overfitting, and gradient clipping and batch normalization are used to prevent gradient vanishing or explosion as well as overshooting the learning updates.

4. Experiments

We test the Res-Bidir-LSTM network with the public domain UCI dataset [27] and the Opportunity dataset [7]. It also compares with the outcome of other methods. The computer for testing has an i7 CPU with 8 GB RAM as well as an NVIDIA GTX 960m GPU. The GPU and CPU are used alternatively depending on the size of the neural network, which sometimes exceeded the available amount of memory on the graphics card during training.

4.1. Datasets. The research objects of recognition are activities in daily life. The benchmark for HAR should meet two conditions. First, it should contain most behavioral classes to reflect real life. Second, it should abstract features and labels for modeling and calculations. Human actions can be divided into several layers in terms of granularity, such as the gesture layer (including left-arm lift, trunk-back), the action layer (including jumping, running, sitting), and the behavior layer (such as drinking water, typing, and sleeping). A good HAR benchmark should include a clear understanding of

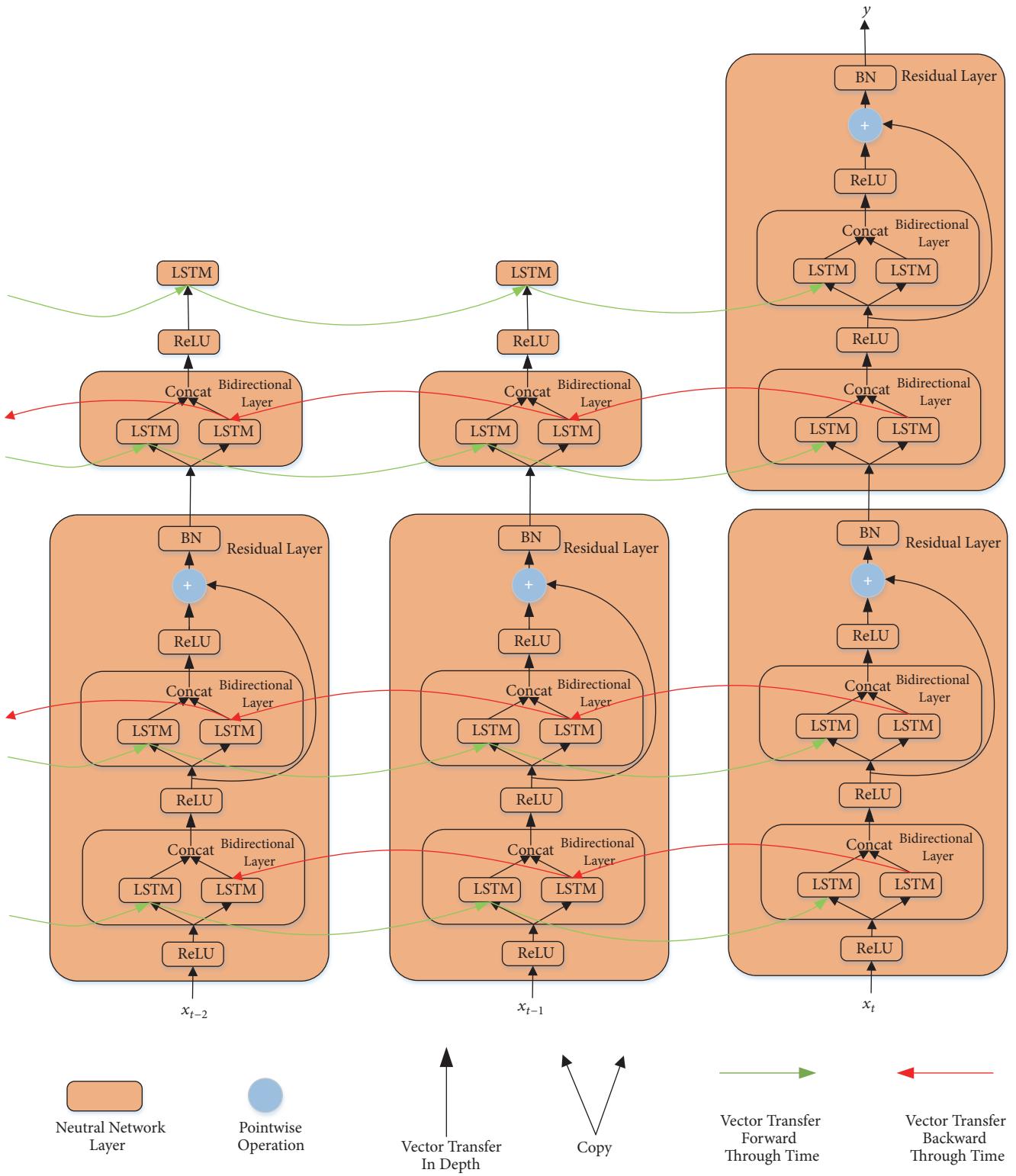

Figure 5: Unfold Res-Bidir-LSTM network architecture. "ReLU" represents the full connection layer and a ReLU function followed. And "BN" is short for batch normalization.

the hierarchy. There are several open datasets that can be benchmarked, such as the public domain UCI [27], the Opportunity [7], and the KTH datasets [28]. Many studies have used these benchmarks. We choose the public domain UCI and the Opportunity datasets for our experiments. The neural network should be readily adaptable to a new dataset with an architecture module and a changeable configuration file that also loads the dataset.

Public Domain UCI Dataset. Experiments are carried out with a group of 30 volunteers aged 19–48 years. Each person performs six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, and LAYING_DOWN) wearing a smart phone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we are able to make three-axial linear acceleration and three-axial angular velocity available at a constant rate of 50 Hz and trim into windows of 128 time steps for a 2.56-second window. It is enough to capture two steps, in the case of walking, for the classification. The experiments are video-recorded to label the data manually and obtain balanced classes. The dataset obtained is partitioned randomly into two sets: 70% of the volunteers are selected for generating the training data, and the others are selected for generating the testing data. Each sample has 561 linear (time-independent) hand-made, preprocessed features from signal analysis (e.g., window's peak frequency), but only nine features are used in our study: triaxial gravity acceleration from the accelerometer (from a 0.3 Hz Butterworth low-pass filter) and triaxial body acceleration and triaxial angular velocity from the gyroscope. These are raw signals with a time component. The sensor data are preprocessed by applying denoising median filters, clipping the approximately 20 Hz mark. They are then sampled in fixed-width sliding window of 2.56 seconds. The window is provided with an overlap of 50% to ease training. Additionally, all features are prenormalized and bound within $[-1, 1]$.

Opportunity Dataset. The Opportunity dataset for human activity recognition from wearable, object, and ambient sensors is a dataset devised to benchmark human activity recognition algorithms. A subset of this dataset is used for the "OPPORTUNITY Activity Recognition Challenge". From then on, the subset has been used by numerous third party publications [29–31]. In order to compare with benchmark methods, we use the same subset, which has 3 subjects and 6 runs per subject. The testing set is composed of Run 4 and Run 5 for Subjects 2 and 3, and the others are left for training set. The activities of the user in the scenario were annotated on different levels. Notably, among others, 17 mid-level gesture classes are identified and used for our predictions; this group included the "NULL" class, which is common, for a total of 18 classes. Due to the use of wireless sensors to transfer data, there may be missing data. We use linear interpolation to fill in the missing data. Also, the data is provided with a custom scale and different value ranges and resolutions for each feature. There are sometimes magnitudes of difference according to the cell used. Our architecture use mean and variance (standard deviation) normalization on the z-score scale with a standard deviation of 0.5. Such a small standard deviation is often useful in deep learning [32]. The transition function was defined as follows:

$$x^* = \frac{x - \mu}{\sigma}, \tag{8}$$

where μ is mean value and σ is the standard deviation.

To summarize, we use only a subset of the full dataset to simulate the conditions of the competition, using 113 sensor channels and classifying the 17 categories of output (and the NULL class). Our LSTM's inner representation is always reset to 0 between series. We use mean and variance normalization rather than min-to-max rescaling.

Because of class imbalance in the Opportunity dataset, we use F1 score as a measurement of recognition. The F1 score is defined as follows:

$$F_1 = 2\sum_c \frac{N_c}{N_{total}} \frac{prec \times recall}{prec + recall}, \tag{9}$$

where *prec* and *recall* indicate precision and recall, respectively. N_c is the sample count of class c, and N_{total} is the total sample count of the dataset.

4.2. Hyper-Parameters Setting. The hyper-parameters in the Res-Bidir-LSTM network affect the final result. Generally used methods of tuning parameters include experimental methods, grid searches [33], genetic algorithm (GA) [34], and particle swarm optimization (PSO) [35, 36]. As experimental methods involve approximating the value by running many experiments, these methods are time consuming. GA and PSO are heuristic algorithms, and they are limited to dealing with large-scale network. We use grid search, which involved dividing hyper-parameter values into several steps to create a grid of a certain range and then traversing all points of the grid to find the best values for these parameters.

Our LSTM's inner representation is always reset to 0 between series. As shown in Figure 6, for n channels' data, a new sample consisted of a window length series of T. Then, T is moved with a step size to form the next sample, using 50% overlap, which adds some redundancy during training and testing. Repeating the operation above yields a dataset suitable for training. Missing values between labels refer to the Null class. The last moment of the time window is chosen as a label for each sample's classification. Thus, if the class changes throughout the series, only the last time step will account for the classification for training and testing, and the 50% overlap account for learning some of the label changes. The label of the last moment is better at reflecting the intent as well as the possibly current behavior. Through a slide window process, the shape of the dataset $SequenceNum \times channels$ is converted to $sampleNum \times windowSize \times channels$ with overlap.

4.3. Results. MultiClass Hardware Friendly SVM (MC-HF-SVM) was proposed by Davide Anguita [8]. It allows better preservation of the life of a smart phone battery than the conventional floating-point-based formulation while maintaining comparable system accuracy levels. The performances of Bidir-LSTM and Res-LSTM are almost the same, both better than baseline LSTM. A brief explanation is that Bidir-LSTM can get information in both forward and backward passes, and Res-LSTM uses shortcut to transmit information directly. Among the algorithms in Table 1, Res-Bidir-LSTM achieves the best F1 score, 93.5%, because of residual and bidirectional connections. The accuracy and F1 score are almost the same for each model. It seems that class imbalance makes little difference with the dataset. Res-Bidir-LSTM is

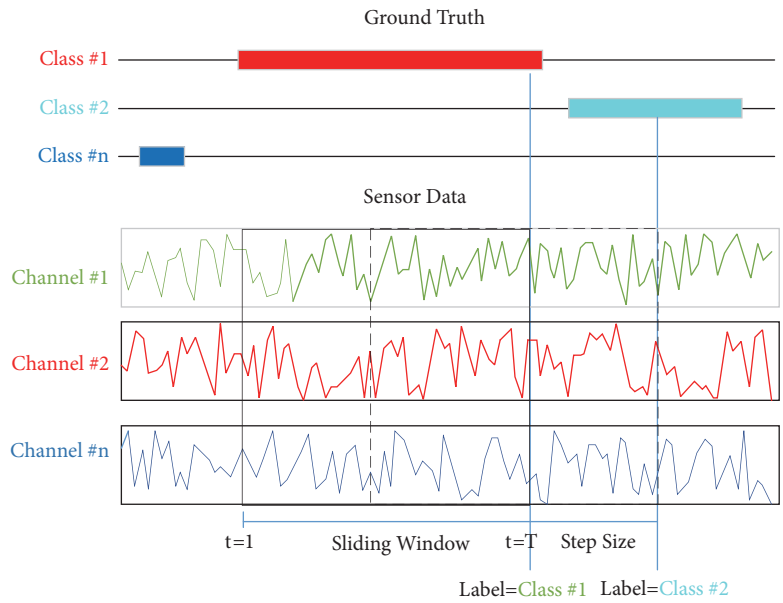

Figure 6: Sliding window. The ground truth represents labels for the classification.

Table 1: Accuracy and F1 score for each algorithm with the public domain UCI dataset. The best result in each column is highlighted in bold.

Algorithm	Accuracy	F1 score
MC-HF-SVM	89.3%	-
Baseline LSTM	90.8%	90.8%
Bidir-LSTM	91.1%	91.1%
Res-LSTM	91.6%	91.5%
Res-Bidir-LSTM	**93.6%**	**93.5%**

Table 2: Matrix confusion on test dataset using Dee-Res-Bidir-LSTM. WK, WU, WD, ST, SD, and LD for the public domain UCI dataset (representing WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, and LAYING_DOWN, respectively).

	WL	WU	WD	ST	SD	LD	Recall
WL	**476**	1	20	0	0	0	95.8%
WU	20	**429**	21	0	0	0	91.3%
WD	14	8	**398**	0	0	0	94.8%
ST	0	5	0	**426**	33	4	91.1%
SD	0	0	0	62	**473**	0	88.4%
LD	0	0	0	0	0	**537**	100%
Precision	93.3%	96.8%	90.7%	87.3%	93.5%	99.5%	93.6%

fine-tuned with 3 layers, 28 units/layer, and dropout is set as 0.8. The best hyper-parameter combination is chosen by grid search. We also test Res-Bidir-LSTM without dropout. It shows that the best F1 score is 92.8% and always gets at early epoch.

Table 2 shows the confusion matrix of Res-Bidir-LSTM with testing data. The values of prediction in six classes are in the range of 87.3% to 99.3%, and the values of recall are in the range of 88.4% to 100%. The integral accuracy reached 93.6%. An intuitive confusion matrix is shown in Figure 7. The color from blue to red represents the increasing percentage. It can be seen that the LAYING_DOWN class is recognized best, likely because triaxial acceleration and triaxial angular velocity are quite different from the values in other classes. Standing and sitting are sometimes misrecognized as each other; both involve static behavior. In fact, they are seemingly almost the same from the point of view of a device placed on the belly, which is how the dataset is gathered. Similarly, WALKING, WALKING_UPSTAIRS, and WALKING_DOWNSTAIRS all involve dynamic behavior, and there is some confusion among them. However, it is still possible to see a little clustering among these three classes in the matrix.

In Table 3, we compare proposed model with previous model and variants of LSTM. The best result is written in bold,

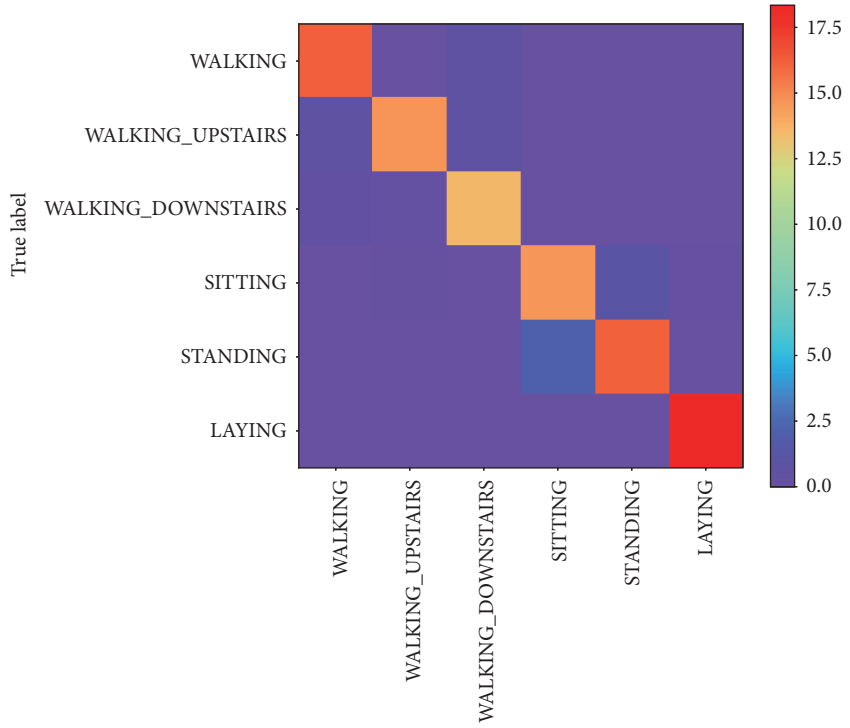

Figure 7: Normalized-to-percent matrix confusion on the test dataset using Dee-Res-Bidir-LSTM for the public domain UCI dataset. The columns represent the predicted classes, and the rows represent the actual classes.

Table 3: F1 score with the NULL class of each algorithm with the Opportunity dataset.

Algorithm	F1 score
1NN [7]	87%
CNN [9]	85.1%
Baseline LSTM	88.2%
Bidir-LSTM	89.2%
Res-LSTM	90.2%
Res-Bidir-LSTM	**90.5%**

which achieves 90.5%. Generally speaking, LSTM models outperform others in the experiment. It is because LSTM is good at dealing with time series. It keeps the relation of input sequence while the other models do not. Among LSTM models, both Res-LSTM and Bidir-LSTM are better than the vanilla one. As for the Res-Bidir-LSTM, it performs like an ensemble method lead to the best result. The model has 3 layers, 128 units/layer, and dropout is set as 0.6. We find that dropout seems to benefit little improvement, but the number of layer plays an import role. No matter layers increased or decreased, the F1 score will sharply decrease.

Figure 8 shows the F1 score trend with the training data and testing data for each model. Generally, when training is finished, both the training and testing results oscillate around a fixed value. Moreover, the results in the four groups are convergent. The amplitude of baseline LSTM is significantly higher than those of the others. Res-Bidir-LSTM achieves the best F1 score, ~0.9. Convergence rate can be arranged from slow to fast as baseline LSTM, Bidir-LSTM, Res-Bidir-LSTM, and Res-LSTM. However, the difference between Res-Bidir-LSTM and Res-LSTM is very small, and both are obviously different from the others. The results also show that residual connection is outstanding in convergence.

5. Conclusions

In this paper, the significance of HAR research is analyzed, and an overview of emerging methods in the field is provided. LSTM networks have been used in many innovations in natural language processing, speech recognition, and weather prediction. The technology is adapted to the HAR task. We propose the novel framework of the Res-Bidir-LSTM network. The deep network can enhance learning ability for faster learning in early training. It also guarantees the validity of information transmission through residual connections (on the depth dimension) and bidirectional cells (on the temporal dimension). In our experiments, the proposed network is able to improve the accuracy, by 4.8%, for the

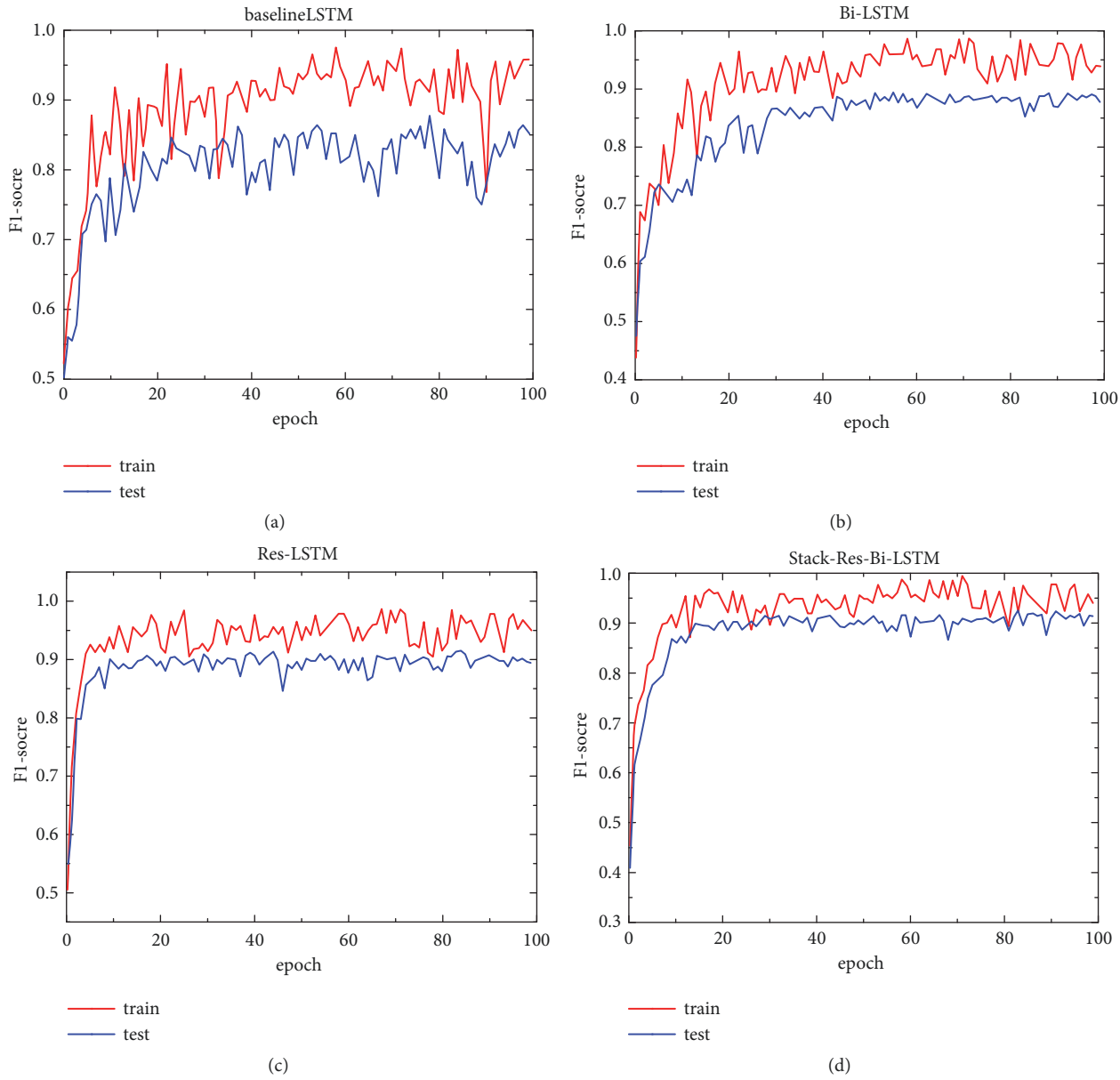

Figure 8: F1 score trends of algorithms for the Opportunity dataset. The blue line shows test data, and the red one indicates training data. (a)–(d) show baseline LSTM, Bidir-LSTM, Res-LSTM, and Res-Bidir-LSTM, respectively.

public domain UCI dataset and increase the F1 score, by 3.7%, for the Opportunity dataset in comparison with previous work.

We also find that window size is a key parameter. Too small window does not guarantee continuity of information, and too large window causes classification errors. Usually, 500 ms to 5000 ms will be appropriate for the window size. During model training, the architecture of the network, such as the layers and the cells in each layer, should be determined first, followed by the optimization of hyper-parameters, such as learning rate and the L2 weight decay multiplier. The values of hyper-parameters should be determined according to the specific architecture. For example, 28 cells are sufficient for the public domain UCI dataset, but 128 cells are better for the Opportunity dataset because it has more features and labels and, thus, increased overall complexity.

Future work will explore a more efficient way to tune parameters. Although the grid search is workable, the searching range must be changed manually, and the values are always fixed. It will be important to find an adaptive way to automatically adjust the searching process and also make the neural network's architecture evolve, such as by automatically reshaping, adding, and removing various layers. Also, exploring the effect of mixing 1D time-based convolution at one or some points in the LSTM cells might improve results. Finally, applying the Res-Bidir-LSTM network to other fields could be revealing. A good model should have outstanding generalization. Indeed, focusing on time series prediction

problems has value. Problems such as stock prediction and trajectory prediction may be explored.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

We thank Guillaume Chevalier, who supports this work by offering guidance and by contributing to building the neural network architecture. The Github repository is located at the following address: https://github.com/guillaume-chevalier/HAR-stacked-residual-bidir-LSTMs. The code is open source, and we sought to make it readily adaptable to new problems using a new configuration file. We also thank Maoguo Gong, who reviews this paper and puts forward constructive suggestions.

References

[1] H. Kalantarian, C. Sideris, B. Mortazavi, N. Alshurafa, and M. Sarrafzadeh, "Dynamic Computation Offloading for Low-Power Wearable Health Monitoring Systems," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 3, pp. 621–628, 2017.

[2] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 1, pp. 1–12, 2010.

[3] S. H. Ahmed and D. Kim, "Named data networking-based smart home," *ICT Express*, vol. 2, no. 3, pp. 130–134, 2016.

[4] S. Kumar, "Ubiquitous smart home system using android application," *International Journal of Computer Networks & Communications*, vol. 6, no. 1, pp. 33–43, 2014.

[5] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2015.

[6] H.-S. Yeo, B.-G. Lee, and H. Lim, "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware," *Multimedia Tools and Applications*, vol. 74, no. 8, pp. 2687–2715, 2015.

[7] H. Sagha, S. T. Digumarti, J. D. R. Millán et al., "Benchmarking classification techniques using the opportunity human activity dataset," in *IEEE International Conference on Systems, Man, and Cybernetics, Anchorage*, pp. 36–40, Anchorage, Alaska, USA, October 2011.

[8] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Energy efficient smartphone-based activity recognition using fixed-point arithmetic," *Journal of Universal Computer Science*, vol. 19, no. 9, pp. 1295–1314, 2013.

[9] J. B. Yang, M. N. Nguyen, P. P. San et al., "Deep convolutional neural networks on multichannel time series for human activity recognition in," in *Proceedings of the International Conference on Artificial Intelligence*, AAAI Press, 2015.

[10] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, pp. 115–140, 2016.

[11] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Proceedings of the 27th Annual Conference on Neural Information Processing Systems, NIPS 2013*, USA, December 2013.

[12] C. Dong, C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Computer Vision-ECCV 2014*, vol. 8692 of *Lecture Notes in Computer Science*, pp. 184–199, Springer, New York, NY, USA, 2014.

[13] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 6645–6649, May 2013.

[14] A. Hannun, C. Case, J. Casper, B. Catanzaro et al., "Deep speech: Scaling up end-to-end speech recognition," https://arxiv.org/abs/1412.5567.

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

[16] A. Kumar, O. Irsoy, P. Ondruska et al., "Ask me anything: Dynamic memory networks for natural language processing," *CoRR*, 2015.

[17] K. Cho, B. van Merrienboer, C. Gulcehre et al., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 5, no. 2, pp. 157–166, 1994.

[21] M. Wöllmer, F. Eyben, J. Keshet, A. Graves, B. Schuller, and G. Rigoll, "Robust discriminative keyword spotting for emotionally colored spontaneous speech using bidirectional LSTM networks," in *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2009*, pp. 3949–3952, Taiwan, April 2009.

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

[23] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," https://arxiv.org/abs/1505.00387.

[24] R. Pascanu, T. Mikolov, and Y. Bengio, *Understanding the exploding gradient problem*, 2012.

[25] D. Kingma and B. Jimmy, "Adam: A method for stochastic optimization," https://arxiv.org/abs/1412.6980.

[26] I. Sergey and S. Christian, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," https://arxiv.org/abs/1502.03167.

[27] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proceedings of the 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013*, pp. 437–442, Belgium, April 2013.

[28] J. W. Maclean, *Spatial Coherence for Visual Motion Analysis*, Springer, Berlin, Germany, 2006.

[29] M. Zeng, L. T. Nguyen, B. Yu et al., "Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors," in *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services*, pp. 197–205, Austin, TX, USA, November 2014.

[30] T. Plötz, N. Y. Hammerla, and P. Olivier, "Feature Learning for Activity Recognition in Ubiquitous Computing," in *In Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1729–1734, Barcelona, Spain, 2011.

[31] D. Gordon, J. Czerny, and M. Beigl, "Activity recognition for creatures of habit: Energy-efficient embedded classification using prediction," *Personal and Ubiquitous Computing*, vol. 18, no. 1, pp. 205–221, 2014.

[32] S. Wiesler, A. Richard, R. Schlüter, and H. Ney, "Mean-normalized stochastic gradient for large-scale deep learning," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '14)*, pp. 180–184, IEEE, Florence, Italy, May 2014.

[33] Y. Yao, L. Zhang, Y. Liu et al., "An improved grid search algorithm and its application in PCA and SVM based face recognition," *Journal of Computational Information Systems*, vol. 10, no. 3, pp. 1219–1229, 2014.

[34] E. Levy, O. E. David, and N. S. Netanyahu, "Genetic algorithms and deep learning for automatic painter classification," in *Proceedings of the 16th Genetic and Evolutionary Computation Conference, GECCO 2014*, pp. 1143–1150, Canada, July 2014.

[35] G. Fornarelli and A. Giaquinto, "Adaptive particle swarm optimization for CNN associative memories design," *Neurocomputing*, vol. 72, no. 16-18, pp. 3851–3862, 2009.

[36] A. R. Syulistyo, D. M. J. Purnomo, M. F. Rachmadi et al., "Particle Swarm Optimization (Pso) for Training Optimization on Convolutional Neural Network (Cnn)," *Jurnal Ilmu Komputer dan Informasi*, vol. 9, no. 1, pp. 52–58, 2016.