UDC 004.81

HUMAN PHYSICAL ACTIVITY RECOGNITION ALGORITHM BASED ON SMARTPHONE DATA CONVOLUTIONAL NERUAL NETWOEK AND LONG SHORT TIME MEMORY

Z.X. YANG, Z.Y. CHEN

Belarusian State University of Informatics and Radioelectronics, Republic of Belarus

Received March 19, 2023

Abstract. A deep learning framework for activity recognition based on smartphone acceleration sensor data, convolutional neural network (CNN) and long short-term memory (LSTM) is proposed in the paper. The proposed framework aims to improve the accuracy of human activity recognition (HAR) by combining the strengths of CNN and LSTM. The CNN is used to extract features from the acceleration data and the LSTM is used to model the temporal dependencies of the data. The proposed framework is evaluated on the publicly available dataset, it includes 6 different actions: walking, walking upstairs, walking downstairs, sitting, standing, and laying. The recognition accuracy has reached 94 %.

Keywords: HAR, CNN, LSTM, acceleration sensor.

Introduction

With the rapid popularity of smartphones and the rapid development of micro sensors, various MEMS sensor devices are embedded in people's smartphone. The main advantages of MEMS sensors are small size, light weight, low power consumption, high reliability, high sensitivity, easy integration, etc. In addition, the research on human behavior recognition based on sensor-based intelligent devices has become an emerging research topic in recent years, traditional machine learning algorithms extract feature vectors from data to distinguish between classes of activities, and researchers have done a lot of research work in this area. FGD Silva [1] used two methods, FDR and PCA, to extract 19 features from the data and used SVM method to classify the activities, VNT Sang [2] applied KNN, ANN and SVM algorithms for classification and recognition of human behavior, R Singh [3] et al. proposed an algorithm for human state recognition activity using decision tree C4,5 by data mining algorithm. Since traditional machine learning algorithms require manual extraction of features in the data, and it is difficult for non-professionals to extract effective feature sets, manual extraction is also subject to human error and time-consuming, all of these methods which will reduce the accuracy of classification and recognition, but neural networks greatly compensate for the lack of manual feature extraction in traditional machine learning by building a multi-level automatic feature extraction architecture.

Data acquisition for human behavior recognition is basically divided into two categories, video image-based data acquisition and wearable sensor-based data acquisition. In this paper, we focus on human activity recognition based on wearable sensing data. This paper uses the built-in acceleration sensor of a smartphone for data collection.

The network model consists of three convolutional layer and LSTM network layers, fully connected layer, and one Softmax layer, and predicts the corresponding human actions from the data set. The proposed algorithm's accuracy achieved 94 % and the loss is about 0,02 at the test set.

Dataset

We selected the data set from the UCI Human Activity Recognition Using Smartphones Data Set. The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities wearing a smartphone on the waist. Using its embedded accelerometer, they captured 3-axial linear acceleration at a constant rate of 50 Hz. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets.

The sensor signals (accelerometer) were pre-processed by applying noise filters and then sampled in fixed width sliding windows of 2,56 sec and 50 % overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0,3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.

After the data set is obtained, the maximum and minimum values of the data are normalized using Formula (1). The data in the training set is 7352×561, and 7350×560 data are selected as the data set required for the experiment. It was found that many data with the same output in the data set were connected, which did not meet the assumption that the training data were independent and identically distributed. randon.seed = 314 was selected to generate a random sequence of length 7350, which was used as the index of the data:

$$X_{scale} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}, \tag{1}$$

where $X_{\max}$ – denotes the maximum value in the feature, $X_{\min}$ – denotes the minimum value in the feature.

Framework of CNN and LSTM algorithm

The structure diagram of the algorithm is shown in Figure 1. Based on the cell phone tri-axis acceleration data collected from the public dataset, the algorithm predicts six different human behaviors through convolutional layer and LSTM, fully connected layer, and Softmax layer, and finally outputs the behavior type with the highest probability as the output result.

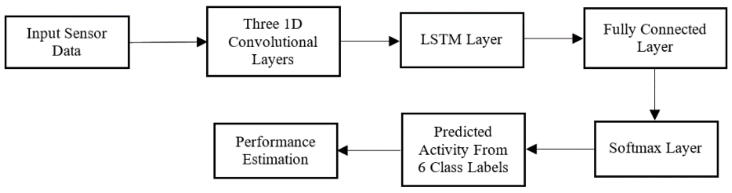


Figure 1. The network structure of CNN-LSTM

1D convolutional layer 1

For the convolutional 1, we got the input tensor is (100,1,560), the batch size is 100, and input channel is 1, length of signal sequence is 560, the hyperparameter of the convolutional layer is shown as Table 1.

In the 1D convolutional layer, it has two kinds of parameters: weights and biases. The total number of parameters is just the sum of all weights and biases:

$$W_c = K \times C \times N, \tag{2}$$

where the W_c – is the number of weights of the convolutional layer, C – is the number of channels of the input, N – is the number of kernels, K – is the size of kernels used in the convolutional layer, so we can get the weight of the convolutional layer is 320:

$$P_c = W_c + B_c, \tag{3}$$

where P_c – is the number of parameters of the convolutional layer, B_c – is the number of biases of the convolutional layer, W_c – is the number of weights of the convolutional layer, so we can get the number of parameters of the convolution layer 1 is 384.

The calculation method of size (O) of the output is shown in formula (4):

$$O = \frac{I - K + 2P}{S} + 1, \tag{4}$$

where the I – is the size (width) of input, K – is the size (width) of kernels used in the convolutional layer, S – is the stride of the convolution operation, P – is padding value. Moreover, we can get the size of the output is 278.

Table 1. **The hyperparameter of the 1D convolution layer 1**

Hyperparameter	Value
Input channel	1
Output channel	64
Kernel size	5
Stride	2
Padding	0

1D convolutional layer 2

For the convolutional 2, we got the input tensor is (100,64,278), the batch size is 100, and input channel is 64, length of signal sequence is 278, the size of the output is 137. The parameter and hyperparameter of the convolutional layer are shown as Table 2 and Table 3.

Table 2. **The parameter value of the 1D convolution layer 2**

Parameter	Value
Weight	20480
Bias	64
Total	20544

Table 3. **The hyperparameter of the 1D convolution layer 2**

Hyperparameter	Value
Input channel	64
Output channel	64
Kernel size	5
Stride	2
Padding	0

1D convolutional layer 3

For the convolutional 3, we got the input tensor is (100,64,137), the batch size is 100, and input channel is 64, length of signal sequence is 137, the size of the output is 67. The parameter and hyperparameter of the convolutional layer are shown as Table 4 and Table 5.

Table 4. **The parameter value of the 1D convolution layer 3**

Parameter	Value
Weight	20480
Bias	64
Total	20544

Table 5. **The hyperparameter of the 1D convolution layer 3**

Hyperparameter	Value
Input channel	64
Output channel	64
Kernel size	5
Stride	2
Padding	0

LSTM layer

In the LSTM layer, input size is taken as 64 because the feature used is the 3rd convolutional output feature, which is the dimension 64, using convolutional features for temporal recognition, and the number of hidden unit size is 128. For the input tensor shape (100,64), after processing through the LSTM layer, the output shape become (100,128). The parameters are shown in Table 6. The hyperparameter of the layer are shown in Table 7, each LSTM has 128 hidden units, and that is, the number of neurons in the LSTM unit is 128.

Table 6. **The parameter value of the LSTM layer**

Parameter	Value
Input tensor shape	(100,64)
LSTM units	128
Output tensor shape	(100,128)

Table 7. **The hyperparameter of the LSTM layer**

Hyperparameter	Value
Hidden layer	3
Hidden unit size	128
Learning rate	0.001
Dropout	0.9
Batch size	100
Epoch	50

Fully connected layer

In the fully connected layer, we get the input tensor derived from the output of the last hidden layer state of the LSTM layer, the input shape is (100,128).

After we got the input, we can multiply the input by the weight matrix [128,6] and add a bias [6], then we get the output of the fully connected layer (100,6), the relevant parameters are shown in Table 8 and the calculation process is shown in formula (5):

$$y = xA + b, \tag{5}$$

where x – is the input, A – is the weight matrix, b – is the bias.

Table 8. **The input, condition, and output of fully connected layer**

Input shape	(100,128)
Weight matrix	[128,6]
Bias	[6]
Output shape	(100,6)

Softmax layer

After getting the output from the fully connected layer, we use the activation function Softmax to map the output of the neuron to the (0,1) interval, which allowed us to perform multiple classification:

$$\mathrm{Soft\,max}\left(Z_{\mathrm{i}}\right) = \frac{e^{Z_i}}{\sum_{c=1}^{C} e^{Z_c}}, \tag{3}$$

where Z_i – is the output value of the i class, C – is the number of output nodes, which also represents the number of categories in the classification. In this dataset, because there are 6 feature actions in this dataset, we set the C value to 6.

Evaluation indicators and results

Evaluation refers to the stage where the results of model testing are measured and evaluated. We measure the performance of the model using a confusion matrix, which is also used to determine accuracy. The confusion matrix which we used is shown in Table 9.

Table 9. **Confusion matrix of the six classifications tasks**

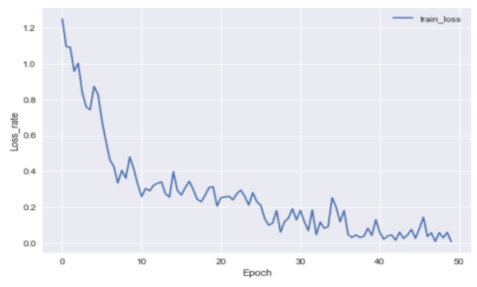
Six-class label		Predicted label						Total
		Laying	Walking	Upstairs	Downstairs	Sitting	Standing	
True label	Laying	A_{LL}	A_{LW}	A_{LU}	A_{LD}	A_{LST}	A_{LSD}	T_L
	Walking	A_{WL}	A_{WW}	A_{WU}	A_{WD}	A_{WST}	A_{WSD}	T_W
	Upstairs	A_{UL}	A_{UW}	A_{UU}	A_{UD}	A_{UST}	A_{USD}	T_U
	Downstairs	A_{DL}	A_{DW}	A_{DU}	A_{DD}	A_{DST}	A_{DSD}	T_D
	Sitting	A_{STL}	A_{STW}	A_{STU}	A_{STD}	A_{STST}	A_{STSD}	T_{ST}
	Standing	A_{SDL}	A_{SDW}	A_{SDU}	A_{SDD}	A_{SDST}	A_{SDSD}	T_{SD}

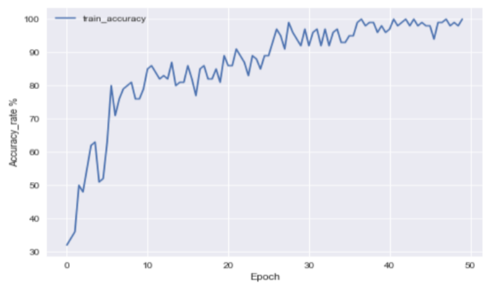
Each sample in the classification task has only one defined category, and a prediction of that category is a correct classification, and a failure to predict it is a classification error, so the most intuitive metric is Accuracy. The formula is as follows:

$$Accurancy = \frac{A_{LL} + A_{WW} + A_{UU} + A_{STST} + A_{SDSD}}{T_L + T_W + T_U + T_D + T_{ST} + T_{SD}}, \qquad (4)$$

where the A_{LL} – denotes the accuracy of correct prediction of laying, A_{WW} – denotes the accuracy of correct prediction of walking, A_{UU} – denotes the accuracy of correct prediction of upstairs, A_{DD} – denotes the accuracy of correct prediction of downstairs, A_{STST} denotes the accuracy of correct prediction of sitting, A_{SDSD} – denotes the accuracy of correct prediction of laying.

Our model is training and testing on the UCI HAR dataset, and 70 % of the volunteers was selected for generating the training data and the remaining 30 % is used to test the data. We set the training set to 100 and train on the training set for 50 epochs. The loss value and the accuracy values are shown in Figure 2.

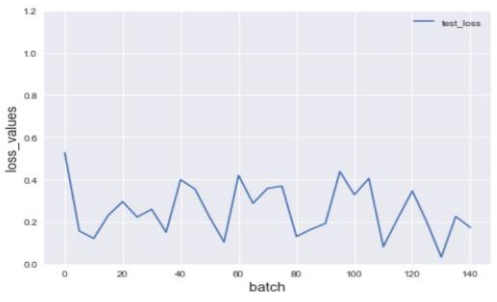


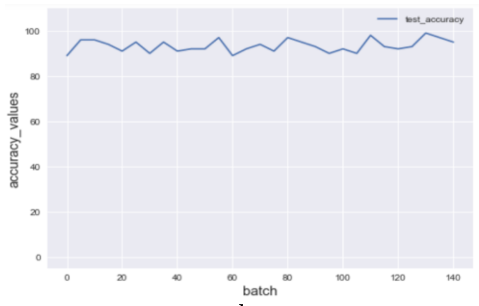


a *b*

Figure 2. Loss and accuracy rate of the CNN-LSTM model: *a* – loss rate during the training; *b* – accuracy rate during the training

Next, we will apply the trained model to the test set to see how well the model performs on the test set. Figure 3 visualizes the accuracy and loss rate of the model on the test set.





a *b*

Figure 3. Loss and accuracy rate of the CNN-LSTM model: *a* – loss rate during the testing; *b* – accuracy rate during the testing

It can be seen from the test set that there is some fluctuation in the loss rate of the model, since the batch size is only 100, it does not look smooth on the test set, but it still has smaller loss rate and higher accuracy on average. Our proposed model achieves an average recognition accuracy of 94 % on the training set, while the average value of loss is about 0,25.

Table 10. **Confusion matrix of the six-class dataset scheme on the test dataset**

Six-class label		Predicted label					
		Laying	Walking	Upstairs	Downstairs	Sitting	Standing
True label	Laying	1	0	0	0	0	0
	Walking	0	0,970	0,0061	0,022	0	0
	Upstairs	0	0,067	0,900	0,037	0	0
	Downstairs	0	0,044	0,093	0,860	0	0
	Sitting	0	0	0	0	0,940	0
	Standing	0	0	0	0	0,086	0,910

The confusion matrix in Table 10 is used to determine the performance of the trained model on the six human action categories. It displayed the accuracy of each class where laying is 100 %, Walking is 97 %, Walking upstairs is 90 %, Walking downstairs is 86 %, sitting is 94 %, and standing is 91 %. Next, normalize the confusion matrix in Figure 4.

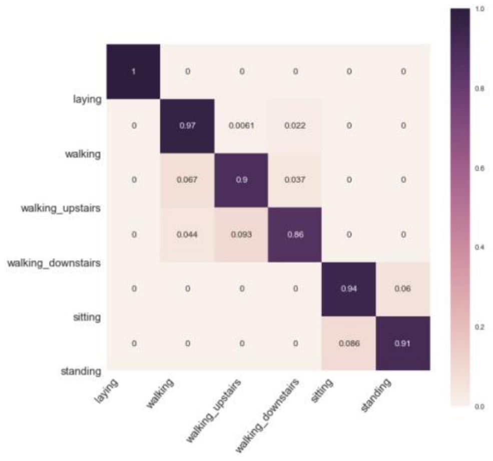


Figure 4. Normalized confusion matrix of six class dataset schemes

Conclusion

This paper is based on the UCI HAR dataset, the dataset was collected by using a smartphone placed around the waist of the tested volunteers, it includes six different human behavioral states: walking, walking upstairs, walking downstairs, sitting, standing, and laying. The proposed model includes three 1D convolutional layers, LSTM network layer, fully connected layer, and Softmax layer. The algorithm extracts data features from the input signal sequence using the three-layer convolutional neural network, and then uses the features as the input of the LSTM neural network. After obtaining the final output of the LSTM neural network, it is mapped to the fully connected layer. Finally, the output is transformed into the probability corresponding to the state through the Softmax layer. With the training of this neural network, our algorithm achieves an average accuracy of 94 % for the six feature activities.

References

1. Silva F.G., Galeazzo E. // Accelerometer based intelligent system for human movement recognition. 2013. P. 20–24.
2. Vnt Sang., Vu Ngoc Thanh. // Human activity recognition and monitoring using smartphones. 2015. P. 481–485.
3. Singh R., Kumar H., Singla R.K. // Analysis of Feature Selection Techniques for Network Traffic Dataset. 2014. P. 42–46.