HUMAN ACTIVITY RECOGNITION

A PREPRINT

Liangzhu Li
Institute of Signal Processing and System Theory
University of Stuttgart
st165534@stud.uni-stuttgart.de

Siying Xu
Institute of Signal Processing and System Theory
University of Stuttgart
st165695@stud.uni-stuttgart.de

February 11, 2020

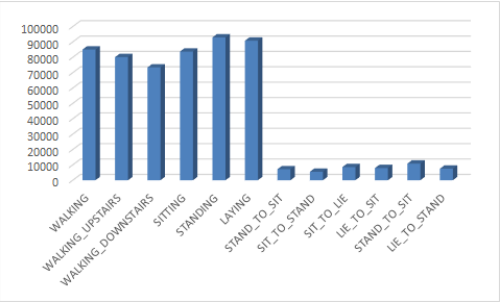
ABSTRACT

Long short term memory (LSTM) is a variant of recurrent neural network (RNN). Under the premise of inheriting recurrent neural network's outstanding memory ability in solving time series problems, it overcomes the problem that the historical signal cannot be transmitted to the current time under RNN network structure. In this paper, we apply the LSTM to human activity recognition. By using the data collected by the accelerometer and gyroscope, human activities are classified into 12 classes. The final classification accuracy is over 0.92.

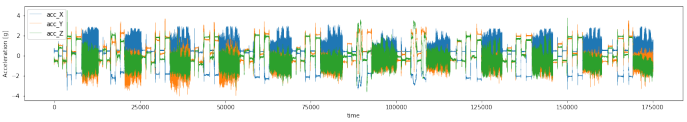
1 Dataset

The raw data file of Human Activities and Postural Transition Dataset (HAPT) was used in our project. The HAPT dataset is recorded by 30 participants, and it is split into 21 users for training, 3 users for validation and 6 users for test. In HAPT dataset, the accelerometer and gyroscope individually record the acceleration and angular velocity in x,y,z directions, so the input data has 6 channels. Specific activities and postural are divided into 12 classes, containing six basic activities and six postural changes.

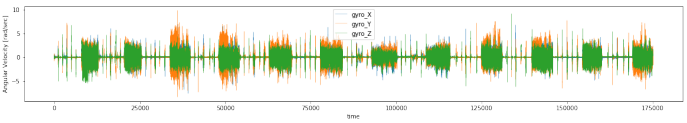
As can be seen from figure 1(a), the data of basic activities is much more postural changes. So we also used random oversampling to balance the dataset. Figure 1(b) and (c) individually show the acceleration and angular velocity of the test dataset.



(a) Data distribution of HAPT



(b) Acceleration of test dataset



(c) Angular velocity of test dataset

Figure 1: Information of HAPT

2 Data preprocessing and input pipeline

2.1 Data preprocessing

To prevent large offsets and variances in the input data from affecting the final training performance, Z-Score normalization is used to normalize the input data by removing mean and then divided by variance. Normalization is performed for each channel and each experiment independently.

Unlabeled data occupies a large proportion in the dataset, which not only increases the data processing time, but also reduces the accuracy of classification because of noise. To solve this problem, we removed the datapairs without label and created new datasets.

The data preprocessing will consume a lot of time if it is re-run each time, so the preprocessed data was saved to 6 csv files, that is, the data and labels of the training set, validation set, and test set. During model training, these 6 files were directly loaded to save time.

2.2 Input pipeline

To create fixed-length input sequences, sliding window was employed in this project. Window length was chosen as 250 samples, and window shift was chosen as 125 samples. This implies that the overlap of each window is 0.5.

3 Model and training

3.1 LSTM network

Recurrent neural network (RNN) is proposed for time-domain sequence data[1]. The output of a neuron can directly act as an input to itself at the next time, so that the output of the network is the joint affection of current input and all previous inputs. However, it is difficult for RNN to learn and save long-term information. The reason is gradient vanish, that is, the gradient generated at current time can only propagate in limited layers. It cannot affect the historical layer beyond a certain time. As a result, RNN does not perform well on long sequence data.

The long-short-term memory network (LSTM) proposed by Hochreiter S solves this problem through a unique gate unit. LSTMs are more and more applied to process time-domain sequence data[2].

3.2 Model architecture

The data of human activity is time sequence data. The information at past moments has a strong influence on the current moment. The use of long-short-term memory (LSTM) can not only effectively transfer past information to the current calculations, but also overcome the problem that the RNN network cannot transfer information which is far away.[3].

So LSTM layers are used in our model to learn temporal dependencies between samples. The model has totally 6 layers, including 1 LSTM layers, 2 dropout layers and 2 dense layers. The output sequence contains the same samples as input sequence, that is, sequence-to-sequence classification.

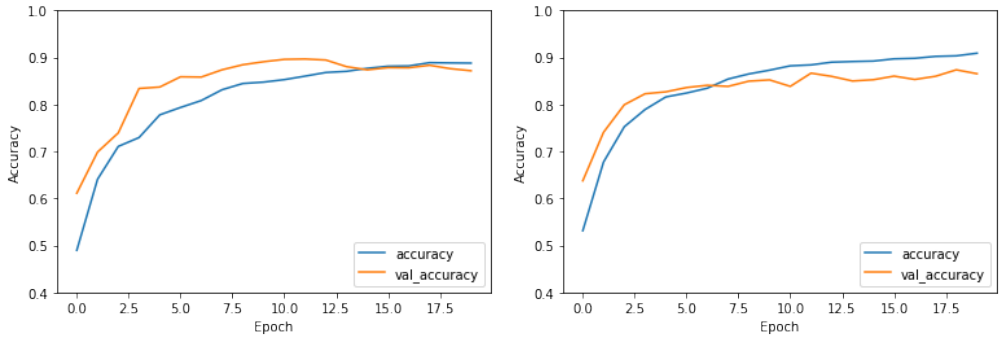
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 250, 6)]          0
_________________________________________________________________
lstm (LSTM)                  (None, 250, 128)          69120
_________________________________________________________________
dropout (Dropout)            (None, 250, 128)          0
_________________________________________________________________
dense (Dense)                (None, 250, 64)           8256
_________________________________________________________________
dropout_1 (Dropout)          (None, 250, 64)           0
_________________________________________________________________
dense_1 (Dense)              (None, 250, 12)           780
=================================================================
Total params: 78,156
Trainable params: 78,156
Non-trainable params: 0
```

Figure 2: Model architecture

3.3 Training routine

We used model.fit() from keras to train the model and the optimizer was Adam. The batchsize of input data was chosen as 32, and the training process contains totally 20 epochs.

Figure 3 compares the training and validation accuracy of the model with one LSTM layer and the model with two LSTM layers during training. As can be seen from the figure, the final accuracies of two models are similar. However, a model containing two LSTM layers will greatly increase parameters and result in more time for training, so we ultimately chose the model containing only one LSTM layer.



(a) Model with 1 LSTM layer (b) Model with 2 LSTM layers

Figure 3: Training and validation accuracy

4 Results and analysis

4.1 results

We use four accuracy indicators to evaluate the performance of the model, that is, test accuracy, recall, precision and F1 score.

- Test accuracy: It is the ratio of number of samples correctly classified by the classifier to the total number of samples. That is, the accuracy of the test dataset when the loss function is 0/1 loss.
- Recall: The formula for recall is
$$R = \frac{TP}{TP + FN} \tag{1}$$
which is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).
- Precision: The formula for precision is
$$P = \frac{TP}{TP + FP} \tag{2}$$
which is the number of correct positive results divided by the number of all positive results returned by the classifier.
- F1 score: It is the harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

After training, the test accuracy of unbalanced dataset can achieve 0.922, recall is 0.772, precision is 0.805 and f1 score is 0.783.

Compared to unbalanced dataset, the dataset after random oversampling has slight decrease in accuracy because of noise. It is test accuracy is 0.921, recall is 0.759, precision is 0.791 and f1 score is 0.770.

4.2 Analysis

Confusion matrix and visualization of test dataset were used in our project to evaluate the performance of the model.

4.2.1 Visualization

The visualization of the result for a sequence of test set is helpful to intuitively analyze model performance. Figure 4 shows the actual and predicted labels of test sample within one sliding window. It can be seen from the figure that the trends of two curves match with each other very well, which means that the model has good performance.

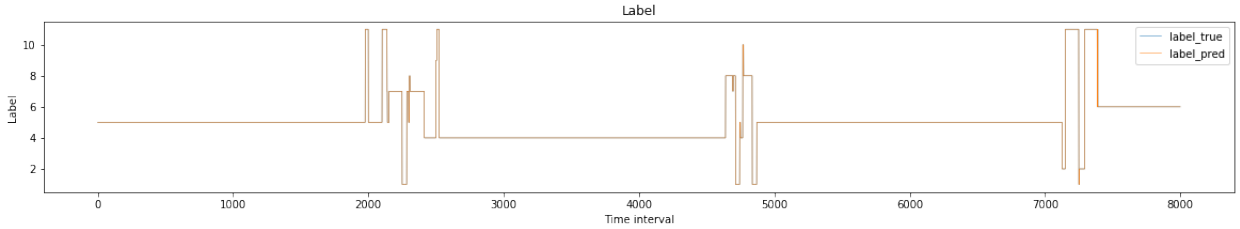


Figure 4: Visuaization of results

4.2.2 Confusion matrix

Through the confusion matrix, we can intuitively compare the correctness of different classes. Figure 5 shows the confusion matrix after oversampling. Due to imbalance of dataset, samples of basic activities are much more than postural transitions, so the first six squares, which represent the correct classification for basic activities, are darker than the last six squares. After randomly oversampling the unbalanced dataset, the color difference has been slightly improved, but the effect is not signifiant.

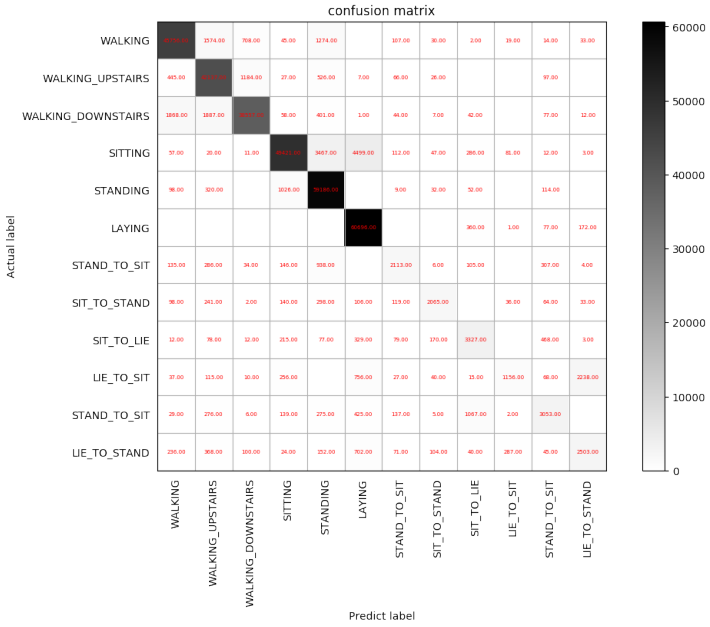


Figure 5: Confusion matrix after oversampling

References

[1] Medsker L R, Jain L C. Recurrent neural networks[J]. Design and Applications, 2001, 5.

[2] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[3] Gers F A, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM[J], 1999.