Structure Investigation of Deep Recurrent Neural Network for Human Activity Recognition

Masaya Inoue[†1,a)] Sozo Inoue[†1,b)] Takeshi Nishida[†1,c)]

Abstract: A deep recurrent neural network (DRNN) for human activity recognition is constructed. The DRNN processes directly time series data from acceleration sensor put on by a person. Moreover, the DRNN has deep layers with recurrent connections and long short-term memory (LTSM) structure. The recognition targets are six type activity and the DRNN is trained by using Human Activity Sensing Consortium (HASC) open data sets. In this paper, investigation results of structure parameters of the DRNN such as the number of the units and the number of layers to improve recognition rate are described.

Keywords: Human activity recognition, deep recurrent neural network, acceleration sensors, long short-term memory

1. Introduction

Recurrent neural networks (RNN) is the name of neural networks that include a directed closed cycle. The RNN is suitable for handling time-series data, such as audio and video signals, and natural language. In recent years, the hierarchical multi-layered convolutional neural network (CNN) has achieved noteworthy results in areas such as image processing, and is drawing attention to the method called deep learning. In this trend, because the RNN also has a deep layer for temporal direction, it has come to be captured as a deep learning method.

In this paper, for the purpose of estimating the state of human activity using an Deep RNN (DRNN) with multi internal layer RNN. we introduce a method of estimating the activity from the data of acceleration sensors in a portable terminal.

2. Structue of DRNN

Structure of the developed DRNN is described in this section. We used a framework "chainer" provided by Preferred Networks inc. for the implementation.

2.1 Mathematical model

Let us assume a Deep RNN (DRNN) with L layers, as shown in Fig. 1. This network is an Elman-type network in which internal layers are completely connected at the same hierarchy in the time direction. Here, $\boldsymbol{u}^{(l),k} = [\ u_1^{(l),k}\ u_2^{(l),k}\ \cdots u_j^{(l),k} \cdots u_J^{(l),k}\ ]^T$ is the input vector of the l-th layer at time k and $\boldsymbol{z}^{(l),k} = [\ z_1^{(l),k}\ z_2^{(l),k}\ \cdots z_j^{(l),k}\ \cdots z_J^{(l),k}\ ]^T$ is the output vector of the l-th layer at time k. A pair of each elements of the input and output vectors is called a unit. j is an arbitrary unit number of the l-th layer and J is the total number of units. We assume $\boldsymbol{x}^k = \boldsymbol{z}^{(1),k}$ in the input layer, and $\boldsymbol{v}^k = \boldsymbol{u}^{(L),k}$ and $\boldsymbol{y}^k = \boldsymbol{z}^{(L),k}$ in the output layer. In addition, in the following, the arbitrary unit numbers (and the total numbers of units) of the the $(l-1)$-th layer are represented by i (and I, respectively). At this time, the input propagation weight from the $(l-1)$-th layer to the l-th layer is represented by

$$\boldsymbol{W}^{(l)} = \begin{bmatrix} w_{11}^{(l)} & \cdots & w_{1i}^{(l)} & \cdots & w_{1I}^{(l)} \\ \vdots & \ddots & & & \vdots \\ w_{j1}^{(l)} & & w_{ji}^{(l)} & & w_{jI}^{(l)} \\ \vdots & & & \ddots & \vdots \\ w_{J1}^{(l)} & \cdots & w_{Ji}^{(l)} & \cdots & w_{JI}^{(l)} \end{bmatrix} \in \mathbb{R}^{J\times I}$$

and

$$\boldsymbol{R}^{(l)} = \begin{bmatrix} r_{11}^{(l)} & \cdots & r_{1j'}^{(l)} & \cdots & r_{1J}^{(l)} \\ \vdots & \ddots & & & \vdots \\ r_{j1}^{(l)} & & r_{jj'}^{(l)} & & r_{jJ}^{(l)} \\ \vdots & & & \ddots & \vdots \\ r_{J1}^{(l)} & \cdots & r_{Jj'}^{(l)} & \cdots & r_{JJ}^{(l)} \end{bmatrix} \in \mathbb{R}^{J\times J}$$

is the recurrent weight in the l-th layer ($l = 2, \cdots, L-1$), where j' is an arbitrary unit number of the l-th layer before one time unit. At this time, the components of $\boldsymbol{u}^{(l),k}$ are given by

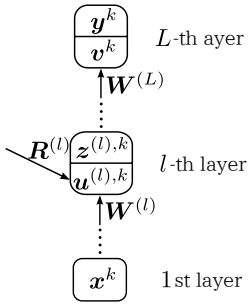

Fig. 1 Schematic representation of the RNN.

[†1] Presently with Kyushu Institute of Technology
[a)] m104016m@mail.kyutech.jp
[b)] sozo@mns.kyutech.ac.jp
[c)] nishida@cntl.kyutech.ac.jp

$$u_j^{(l),k} = \sum_i^I w_{ji}^{(l)} z_i^{(l-1),k} + \sum_{j'}^J r_{jj'}^{(l)} z_{j'}^{(l),k-1}. \quad (1)$$

The elements of the output vector of the l-th layer are expressed as

$$z_j^{(l),k} = f^{(l)}(u_j^{(l),k}),$$

where $f^{(l)}(\cdot)$ is called the *activation function*, and functions such as the sigmoid function $f(u) = \tanh(u)$, logistic sigmoid function $f(u) = 1/(1+e^{-u})$, and rectified linear unit (ReLU) function $f(u) = \max(u, 0)$ are frequently used.

Here, for simplicity, by introducing the 0-th weight $w_{j0}^{(l)}$ and the 0-th unit $z_0^{(l-1),k} = 1$, biases can be collectively described as

$$\boldsymbol{z}^{(l),k} = \boldsymbol{f}^{(l)}(\boldsymbol{W}^{(l)}\boldsymbol{z}^{(l-1),k} + \boldsymbol{R}^{(l)}\boldsymbol{z}^{(l),k-1}), \quad (2)$$

where $\boldsymbol{f}(\boldsymbol{a}) = [\, f(a_1)\ f(a_2) \cdots f(a_N)\ ]^T$. From this equation, it is possible to obtain the output of an arbitrary time by shifting of k. However, since elements of $\boldsymbol{v}^k$ has no recurrent connection, it is the same as the first term of Eqn. (1). Therefore, the final output vector $\boldsymbol{y}$ is derived by

$$\boldsymbol{y}^k = \boldsymbol{f}^{(L)}(\boldsymbol{v}^k) = \boldsymbol{f}^{(L)}(\boldsymbol{W}^{(L)}\boldsymbol{z}^{(L-1),k}). \quad (3)$$

The classification is achieved by equating the number of the dimensions of the output layer and the number of the classes. Then, the activation function of output layer was employed softmax function. The modification of weights were executed by using gradient descent method and the gradients were derived by using truncated BPTT method.

Truncated BPTT method is a kind of BTTP method which expands the network in accordance with the time series and executes the back propagation. truncated BPTT is a method to apply the BPTT separated time series by a appropriate constant[1]. In the following, this constant is called truncated time.

2.2 LSTM

Long short-term memory (LSTM) is a type of NN model for time series data. It is utilized mainly to replace some units of the RNN to solve the problems of an input/output weight conflict [2] which is the conflicts between the input from the previous layer and the recurrent value, and vanishing gradient problem [3] where a delta vanishes or explodes by the deep backward propagation.

A structural diagram of the LSTM is shown in Fig. 2. A structure for storing the internal state, called a memory cell, is provided in the LSTM, allowing it to perform the controls, such as whether to write the information to the cell, read the information from the cell, or delete the information of the cell.

The structure horizontally propagating straight in the top portion of Fig. 2 is for solving the vanishing gradient problem and is called the Constant Error Carousel (CEC). It is possible to solve the vanishing gradient problem by introducing the CEC when back-propagating the error in the recurrent direction [2]. In addition, with this structure, the state C inside the internal layer unit is transmitted at the next time. Although a memory cell is also clearly shown in Fig. 2, in fact the state is preserved through the entire structure of the CEC.

The input gate and output gate are for eliminating input and

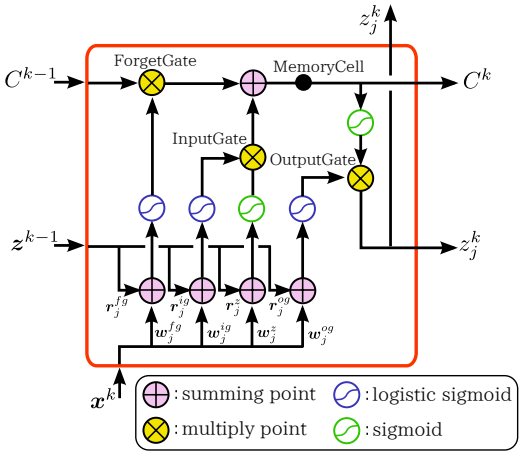

Fig. 2 Structural drawing of LSTM.

output weight conflicts [2]. Let us consider the input gate as an example. First, the output vector $\boldsymbol{z}^{k-1}(\in \mathbb{R}^{J\times 1})$ before 1 time and the input vector $\boldsymbol{x}^k(\in \mathbb{R}^{I\times 1})$ of the present time multiplied by the transmission weight $\boldsymbol{r}_j^{ig}(\in \mathbb{R}^{1\times J})$, $\boldsymbol{w}_j^{ig}(\in \mathbb{R}^{1\times I})$ are summed to pass through the logistic function. This is expressed by the following equation, where the logistic function is expressed by σ and the input gate bias by b^{ig}:

$$\phi^k = \sigma(\boldsymbol{w}_j^{ig}\boldsymbol{x}^k + \boldsymbol{r}_j^{ig}\boldsymbol{z}^{k-1} + b^{ig}). \quad (4)$$

Because the logistic function returns a value in the range of 0 to 1, when multiplied by the original transmission input at the next input gate, it controls “how much of the input to pass”. When ϕ^k is 0, it does not pass the input completely, and it passes all the inputs when it is 1. The same operation is performed in the output gate. By providing a gate that performs such an operation, it is possible to determine whether to memorize a state or to read the memorized state in accordance with the input value or the output value to the internal layer unit.

The purpose of the forget gate is to determine whether or not to forget the memorized state [4]. However, the memory forgetting mentioned here does not refer to inheriting the value of the memory cell before one time. The mechanism of the forget gate is the same as that of the input and output gates, as described above. The forget gate operates, for example, to perform efficient learning even in cases, such as that where the pattern of the time series data is changed suddenly to a pattern having no correlation with the previous context.

By introducing the LSTM, the weights to be updated will increase. More specifically, the bias weight in addition to $\boldsymbol{r}^{ig}, \boldsymbol{w}^{ig}$ for the transmission to the input gate, $\boldsymbol{r}^{og}, \boldsymbol{w}^{og}$ for the transmission to the output gate, and $\boldsymbol{r}^{fg}, \boldsymbol{w}^{fg}$ for the transmission to the forget gate will increase by each gate weight. These weights can also be updated by transmitting the delta inside the LSTM block using the back propagation method.

3. Experiment

3.1 Dataset

The HASC corpus is a data set for machine learning gathered and distributed by the Human Activity Sensing Consortium (HASC). This dataset contains six types of activity, “stay”,

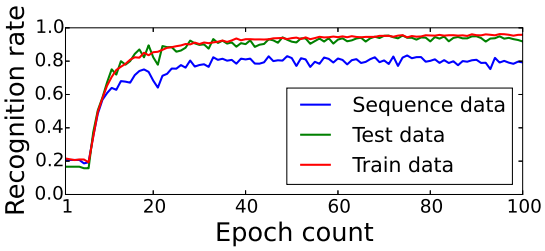

Fig. 3 Accuracy transition of the best model.

"walk", "jog", "skip", "stairup" and "stair down". In this study, we used a part of the three axis acceleration signals of the HASC corpus as a data set. This data set is largely divided into "segmented data" when only a single activity was performed in one trial and "sequence data" when multiple activities were performed consecutively.

In the evaluation, we divided the segmented data into the training data of 432 trials and the test data of 108 trials, maintaining that the number of samples in each activity class balance each other.

Based on the data, we evaluated three types of accuracy:

Training accuracy Perform training with the training data, and estimate with the training data.

Test accuracy Perform training with the training data, and estimate with the segmented test data.

Sequence accuracy Perform training with the training data, and estimate with the sequence data.

Note that, because of the design of HASC dataset, for both sequence data and test data, the same subjects could be included. In this study, the number of subjects is a seven people.

3.2 Result

3.2.1 State of learning

For state of learning of the best model is shown in Fig. 3. Best model is a model derived by adjusting several parameters. Fig. 3 shows the transition of the correct solution rate in each epoch. In the best model, the test accuracy was 95.42% at maximum. The accuracy for the sequence data was 83.43% at maximum.

In Fig. 3, it can be seen that the accuracy increases as the epochs proceed. According to the results in this figure, we judged that it is reasonable to stop the learning at about epoch 80, and for the subsequent evaluations, we extracted the average accuracy from epoch 71 to epoch 80 for comparison.

3.2.2 Search parameters

The results obtained by changing the number of internal layers are shown in Fig. 4. The thick bar represents the mean estimation accuracy from epoch 71 to epoch 80 and the thin line at the top represents the standard deviation. We changed only the number of internal layers among the parameters of the best model. As can be seen in the figure, the accuracy is highest in the case of three layers for any of the training, test, and sequence data. In particular, for the sequence data, the accuracy is about 8.2% higher than that of the worst four-layer model.

The results obtained by changing the number of internal layer units are shown in Fig. 5. We changed only the number of internal

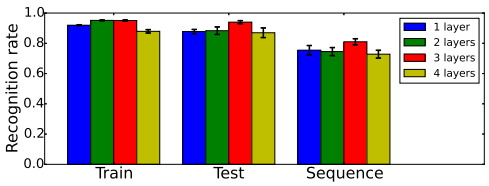

Fig. 4 Comparison of accuracy according to the number of internal layers.

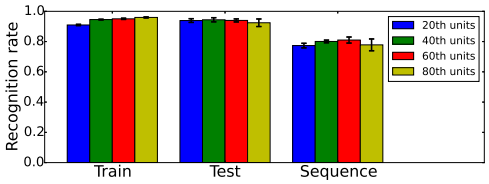

Fig. 5 Comparison of accuracy according to the number of units.

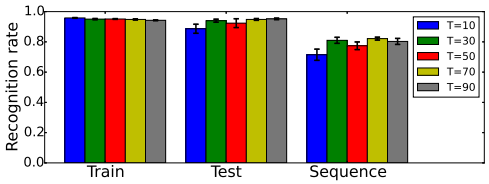

Fig. 6 Comparison of accuracy according to the truncated timess.

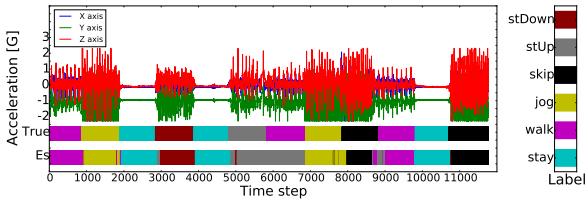

Fig. 7 Estimation example.

layer units, using 20, 40, 60, and 80 units, among the parameters of the best model. In the figure, it can be seen that the accuracy is highest in the case of 60 units in test and sequence data. In particular, in the sequence data the accuracy is about 3.7% higher than that of the lowest, 20-unit, model.

The results of the experiments where the Truncated time was changed are shown in Fig. 6. We changed only the Truncated time, using 10, 30, 50, 70, and 90, among the parameters of the best model. The figure shows that the performance is relatively good at $T = 30$ or $T = 70$, and worst at $T = 10$. For the sequence data, a difference of about 10.7% occurred between the most accurate $T = 70$ model and the $T = 10$ model.

It is found from experimental results that the best number of layers, units and truncated time are three, 60, and 30, respectively for human activity recognition. The serial estimation, such as shown in Fig. 7, can be executed by using this model. The horizontal axis represents 10 [ms] per 1 time in time number and the vertical axis represents acceleration in the gravitational acceleration unit [G].

4. Conclusion

We found a suit DRNN model for human activity recognition by using acceleration sensor signals. However, because the learning phase executed by using segmented action data sets, several

recognition errors and recognition delay were occurred during the time of action sequence transition. We think that such problems can be improved by devising the learning method and the usage of data sets. In addition, we adopted relatively simple data sets, i.e. the number of label is six, and then our methodology described in this paper calls for further investigation to be applied to more complex problems.

Acknowledgments This work was supported by Grant-in-Aid for Scientific Research(B) [No.26280041]. We gratefully appreciate this financial support.

References

[1] Sutskever, I.: Training Recurrent neural Networks, *PhD thesis*, p. 101 (2013).
[2] Hochreiter, S., Hochreiter, S., Schmidhuber, J. and Schmidhuber, J.: Long short-term memory., *Neural computation*, Vol. 9, No. 8, pp. 1735–80 (online), DOI: 10.1162/neco.1997.9.8.1735 (1997).
[3] Pascanu, R., Mikolov, T. and Bengio, Y.: On the difficulty of training recurrent neural networks, *Proceedings of The 30th International Conference on Machine Learning*, No. 2, pp. 1310–1318 (online), DOI: 10.1109/72.279181 (2012).
[4] Gers, F. A., Schmidhuber, J. and Cummins, F.: Learning to forget: continual prediction with LSTM., *Neural computation*, Vol. 12, No. 10, pp. 2451–2471 (online), DOI: 10.1162/089976600300015015 (2000).