

Deep Recurrent Neural Network for Mobile Human Activity Recognition with High Throughput

Masaya Inoue^{*} Sozo Inoue Takeshi Nishida
 Kyushu Institute of Technology Kyushu Institute of Technology Kyushu Institute of Technology
 1-1 Sensui ,Tobata 1-1 Sensui ,Tobata 1-1 Sensui ,Tobata
 Kitakyushu ,Fukuoka ,Japan Kitakyushu ,Fukuoka ,Japan Kitakyushu ,Fukuoka ,Japan
 q344203m@mail.kyutech.jp sozo@mns.kyutech.ac.jp nishida@cntl.kyutech.ac.jp

ABSTRACT

In this paper, we propose a method of human activity recognition with high throughput from raw accelerometer data applying a deep recurrent neural network (DRNN), and investigate various architectures and its combination to find the best parameter values. The “high throughput” refers to short time at a time of recognition. We investigated various parameters and architectures of the DRNN by using the training dataset of 432 trials with 6 activity classes from 7 people. The maximum recognition rate was 95.42% and 83.43% against the test data of 108 segmented trials each of which has single activity class and 18 multiple sequential trials, respectively. Here, the maximum recognition rates by traditional methods were 71.65% and 54.97% for each. In addition, the efficiency of the found parameters was evaluated by using additional dataset. Further, as for throughput of the recognition per unit time, the constructed DRNN was requiring only 1.347 [ms], while the best traditional method required 11.031 [ms] which includes 11.027 [ms] for feature calculation. These advantages are caused by the compact and small architecture of the constructed real time oriented DRNN.

Keywords

Human activity recognition; deep recurrent neural network; acceleration sensors

1. INTRODUCTION

The recognition of human activity is a task that is applicable to various domains, such as health care, preventive medicine, and elderly care. In addition, with the rapid spread of devices with built-in sensors such as smartphones recently, the cost of sensing devices has fallen significantly. As a result, researches on mobile activity recognition have been actively conducted [2].

In traditional activity recognition schemes, researchers have frequently used a machine learning method, such as decision tree, k -

^{*}He is the corresponding author, and executed theory construction, simulation, and verification of the results.

nearest neighborhood, naive Bayes, support vector machine, and random forest, to recognize activities from a feature vector extracted from signals in a time window by using statistic values or Fourier transformation.

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.

Compared to traditional activity recognition methods which are input feature vectors, in deep learning, the original data can be directly input. This allows the calculation of feature vectors to be skipped at the time of training and recognition, so that a speed-up can be expected, especially in the recognition. At the same time, we can also expect the recognition result to be highly accurate by virtue of the deep learning.

In this paper, we propose a method of human activity recognition from raw accelerometer data applying a RNN, and investigate various architectures and its combination to find the best parameter values.

By using a human activity sensing consortium (HASC) open dataset, the recognition ability of the constructed RNN was evaluated. We used the training dataset of 432 segmented trials with 6 activity classes from 7 people, and it was confirmed that the maximum recognition rate was 95.42% against the test data of 108 segmented trials each of which has single activity class. While the recognition rate of traditional method was 71.65%. Moreover, the maximum recognition rate was 83.43% against the test data of 18 multiple sequential trials, and while the recognition rate of the traditional method was 54.97%. Where “a trial” means one sequential data sample such as a segmented data or a sequence data. Moreover, a network reconstructed with the parameters investigated by using HASC dataset by using the human activity recognition (HAR) open dataset, 95.03% recognition rate was achieved. Further, for the throughput of the recognition per unit time, the proposed method was fast requiring only 1.347 [ms], while the existing method required 11.031 [ms] which includes 11.027 [ms] for feature extraction. Notice that this calculation time achieved by only using CPU. The fast response advantage is caused by the number of weights less than 10 % of the traditional method [23].

The contribution of this study includes the following three points:

1. In order to construction of a fast response classifier oriented

real time execution, we adopted a RNN architecture and evaluated its advantages compared with traditional methods.

2. To improve the accuracy of the RNN, various parameters were explored to investigate the factors that affect the accuracy. We used the two types of dataset.
3. The throughput for recognition with the RNN was evaluated, and it was shown to be faster than the existing method which includes feature calculation.

2. BACKGROUND AND RELATED WORK

Recently, many studies on the mobile activity recognition are carried out [5, 20]. For activity recognition technology, techniques for various applications, such as sports [27, 18], skills assessment [19], detection and evaluation of walking [22], medical analyses and nursing activity analysis [13] were proposed. In these techniques, a machine learning method to recognize the activity, such as decision tree, k -nearest neighborhood, naive Bayes, support vector machine, and random forest, is often employed as a basic technique, after the feature vector have been extracted from the signals by statistics or Fourier transformation by taking a time window [3].

Since activity recognition handles sequential data, techniques for sequential data, such as the hidden Markov model (HMM) [15] and conditional random fields [32], which are used in speech recognition and natural language processing, were proposed. In addition, while studies on completing the required activity recognition in real time have been conducted in a few work [17], these studies focused mainly on how to reduce the feature calculation by shifting the feature vectors. Similarly, many studies have been focused on reducing the resources required for executing the feature processing, e.g. [26, 4].

Moreover, in recent years, several activity recognition methods which use deep CNN have been proposed, and they have been confirmed that they can achieve high accuracy recognition than the traditional methods [31]. However these method require the time window to generate certain length segmentation of time series signal. Moreover, in general, CNNs have huge number of connection between inner layers. These features of CNNs are not suitable for real time execution of mobile devices.

An RNN can be used as a learning method and an estimator, and is suitable for handling time-series data, such as audio and video signals, and natural language. Early RNNs included the fully recurrent network developed in 1980, an interconnected type network such as the Hopfield network announced by John J. Hopfield in 1982, and so on. Then, hierarchical RNNs, such as the Elman network and the Jordan network were developed in the early 1990s [9]. The Elman network has a state feedback, and the Jordan network has a recurrent connection for the output feedback. This feedback contributes in order to extract features of dynamics of input signal. The RNN executes calculation processing of large network that led to the time direction at the training phase, and executes fast sequential calculation processing at the recognition phase.

Recently, many of the methods using CNNs and RNNs aims to recognize by using raw signal directly without extraction of the feature vectors in advance.

In the method using a combination of CNN and RNN [23], the network achieves further high accuracy recognition by the feature extraction ability of dynamics in the RNN. On the other hand, the adoption of the CNN architecture causes the increasing of recognition rate, the increasing of the computational cost, and the utilization of the time window.

The RNN is a high throughput network architecture that can deal with raw sensor data without feature extraction and can recognize

by thorough fast sequential processing. In this paper, we propose a method to execute training and recognition of the RNN (i.e. deep RNN) which has multi internal layer by using raw acceleration data without feature extraction aiming at a high-precision activity recognition with high throughput.

3. RECURRENT NEURAL NETWORK

In the following, the basic processing methods for execution of training and recognition of a RNN are explained.

3.1 Deep recurrent neural 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, $\mathbf{u}^{(l),k} = [u_1^{(l),k} \ u_2^{(l),k} \ \dots \ u_j^{(l),k} \ \dots \ u_J^{(l),k}]^T$ is the input vector of the l -th layer at time k and $\mathbf{z}^{(l),k} = [z_1^{(l),k} \ z_2^{(l),k} \ \dots \ z_j^{(l),k} \ \dots \ z_J^{(l),k}]^T$ 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 $\mathbf{x}^k = \mathbf{z}^{(1),k}$ in the input layer, and $\mathbf{v}^k = \mathbf{u}^{(L),k}$ and $\mathbf{y}^k = \mathbf{z}^{(L),k}$ in the output layer. In addition, in the following, the arbitrary unit numbers (and the total numbers of units) of 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 $\mathbf{W}^{(l)} (\in \mathbb{R}^{J \times I})$ and $\mathbf{R}^{(l)} (\in \mathbb{R}^{J \times J})$ is the recurrent weight in the l -th layer ($l = 2, \dots, L-1$), where j' is an arbitrary unit number of the l -th layer before one time unit. At this time, the components of $\mathbf{u}^{(l),k}$ are given by

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

Here $w_{ji}^{(l)}$ and $r_{jj'}^{(l)}$ represent the element of $\mathbf{W}^{(l)}$ and $\mathbf{R}^{(l)}$, respectively. 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

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

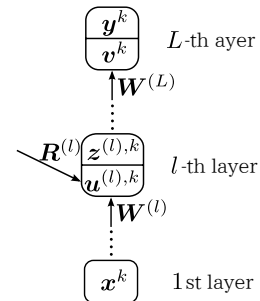


Figure 1: Schematic representation of the DRNN

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

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

3.2 Learning method

3.2.1 Error function

When performing multi-class classification into class $C_1, \dots, C_h, \dots, C_H$, by using the softmax function, let the output of the h -th unit of the output layer be the following equation. Further, the output y_h of the individual unit means a probability belonging to class C_h .

$$y_h \equiv z_h^{(L)} = \frac{\exp(u_h)}{\sum_{q=1}^H \exp(u_q)} = p(C_h | \mathbf{x}). \quad (4)$$

When an input \mathbf{x} is given, this probability y_h is classified into the largest class, and

$$E(\mathbf{w}) = - \sum_{n=1}^N \sum_{h=1}^H d_{nh} \log y_h(\mathbf{x}_n; \mathbf{w}) \quad (5)$$

is defined as the error function, and updating the variable vector \mathbf{w} to minimize this becomes the learning policy. The \mathbf{d}_n represents n -th supervised vector and the d_{nh} represents h -th elements of \mathbf{d}_n . The value of element is set as 1 if the h -th element corresponds to the class of \mathbf{d}_n , and otherwise it is set as 0. Eqn. (5) is called the cross entropy function.

3.2.2 Mini-batch stochastic gradient descent

It is possible to use the gradient descent method to minimize the error function. Let D be the number of elements of \mathbf{w} ; the gradient of the error function is expressed by

$$\nabla E \equiv \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \left[\frac{\partial E(\mathbf{w})}{\partial w_1} \cdots \frac{\partial E(\mathbf{w})}{\partial w_D} \right]^T.$$

The gradient descent method searches the local minimum value in the neighborhood by repeating many times to change \mathbf{w} in the negative gradient direction by a very small amount. Let the weight in the t -th time of repeat be \mathbf{w}^t ; then, it becomes

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \epsilon \nabla E, \quad (6)$$

where ϵ is a parameter called the learning rate.

The mini-batch stochastic gradient descent method collects a small number of sample sets B^t (called a *mini-batch*) in each repeat to calculate the gradient using the average

$$E^t(\mathbf{w}) = \frac{1}{||B^t||} \sum_{n \in B^t} E_n(\mathbf{w}) \quad (7)$$

of error for each sample n among them. It is known that the local solution avoidance performance is high because the calculation converges quickly with this method.

As the learning rate ϵ is high, learning becomes faster; however, if it is too high, since it vibrates near the local minimum value of the error function, an adjustment method of the learning rate, called Adaptive moment estimation (Adam), is employed [16].

3.2.3 Back Propagation Through Time (BPTT)

The back propagation through time (BPTT) method can be applied the error back-propagation method to the developed DRNN

regarded as a large NN expanded in the time direction. The technique is described below. First, we introduce here a quantity called *delta* for the unit j of layer l

$$\delta_j^{(l),k} \equiv \frac{\partial E_n(\mathbf{w})}{\partial u_j^{(l),k}}, \quad (8)$$

This value can also be derived from the values of the $(l+1)$ -th layer of the same time and the time $k+1$ of the same layer. Let the derivative of the function $f(u)$ be $f'(u)$; then

$$\delta_j^{(l),k} = \left(\sum_h w_{hj}^{(l+1)} \delta_h^{(l+1),k} + \sum_{j'} r_{jj'}^{(l)} \delta_{j'}^{(l),k+1} \right) f'(u_j^{(l),k}) \quad (9)$$

holds. The gradient can be calculated based on these equations. The weights $w_{ji}^{(l)}, r_{jj'}^{(l)}$ to be updated become

$$\frac{\partial E(\mathbf{w})}{\partial w_{ji}^{(l)}} = \sum_{k=1}^K \frac{\partial E(\mathbf{w})}{\partial u_j^{(l),k}} \frac{\partial u_j^{(l),k}}{\partial w_{ji}^{(l)}} = \sum_{k=1}^K \delta_j^{(l),k} z_i^{(l-1),k}, \quad (10)$$

$$\frac{\partial E(\mathbf{w})}{\partial r_{jj'}^{(l)}} = \sum_{k=1}^K \frac{\partial E(\mathbf{w})}{\partial u_j^{(l),k}} \frac{\partial u_j^{(l),k}}{\partial r_{jj'}^{(l)}} = \sum_{k=1}^K \delta_j^{(l),k} \delta_{j'}^{(l),k-1}. \quad (11)$$

Using these relations, the gradients can be calculate by propagating the deltas inversely from output to input, as shown in Fig. 2.

Here, the calculation amount becomes too enormous for practical use in the normal BPTT to calculate the gradient by dating back all the times. Therefore, the truncated BPTT method [28], which sets the time to date back to an appropriate constant to perform BPTT for each the time, is used.

3.3 Long short-term memory (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 [11] which is the conflicts between the input from the previous layer and the recurrent value, and vanishing/exploding gradient problem [25] where a delta vanishes or explodes by the deep backward propagation. In this section, we describe the structure of LSTM which solves input/output weight conflicts and vanishing gradients, and gradient clipping method to avoid exploding gradients.

3.3.1 Structure of LSTM

A structural diagram of the LSTM is shown in Fig. 3. 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

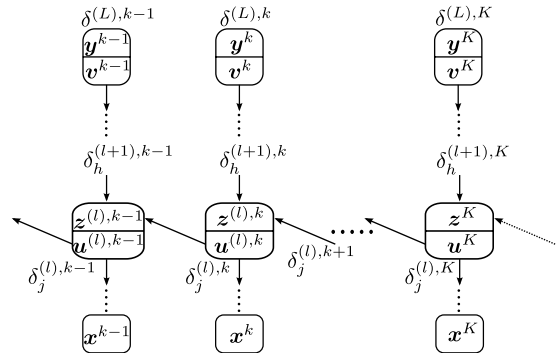


Figure 2: Back propagation in DRNN

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. 3 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 [11]. 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. 3, in fact the state is preserved through the entire structure of the CEC.

The input gate and output gate are for eliminating input and output weight conflicts [11]. Let us consider the input gate as an example. First, the output vector $z^{k-1} (\in \mathbb{R}^{J \times 1})$ before 1 time and the input vector $x^k (\in \mathbb{R}^{I \times 1})$ of the present time multiplied by the transmission weight $r_j^{ig} (\in \mathbb{R}^{1 \times J})$, $w_j^{ig} (\in \mathbb{R}^{1 \times I})$ 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} :

$$\phi^k = \sigma(w_j^{ig} x^k + r_j^{ig} z^{k-1} + b^{ig}). \quad (12)$$

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 [6]. 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 r^{ig} , w^{ig} for the transmission to the input gate, r^{og} , w^{og} for the transmission to the output gate, and r^{fg} , w^{fg} for the transmission to the forget gate

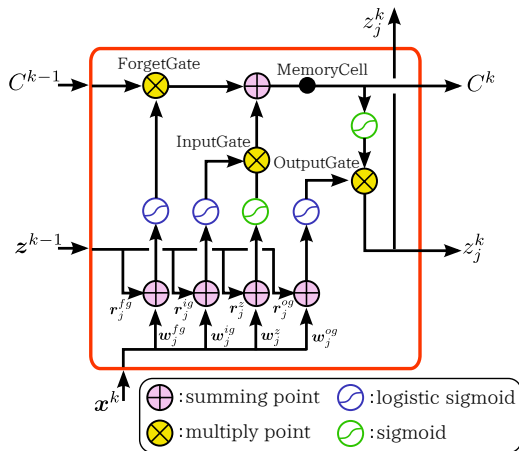


Figure 3: Structural drawing of LSTM

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.

3.3.2 Gradient clipping

The exploding gradient problem is solved with a technique called gradient clipping.

Gradient clipping is a method of correcting the $L2$ norm of the gradient so that it does not exceed the threshold value [25]. Specifically, when letting the threshold value be c ,

$$\|\nabla E\| \geq c \quad (13)$$

is met, and a new value is assigned to the gradient as

$$\nabla E \leftarrow \frac{c}{\|\nabla E\|} \nabla E. \quad (14)$$

3.4 Avoiding overfitting

Learning, as in NNs, by adopting the error function as an optimization function may cause overfitting, in which a model that captures too much peculiarities of the training data and does not fit to the new test data is generated. As methods to avoid overfitting, regularization and dropout are available, as described in the following.

3.4.1 Regularization

Overfitting is liable to occur when the degree of freedom of the network is too high for the training data. However, in many cases, the training data cannot be easily increased, and the degree of freedom of the network should not be easily reduced, because it is deeply involved in the expressive power of recognition. Therefore, a regularization method to mitigate overfitting by providing some type of constraint on the learning parameter is required. The gradient clipping described in Section 3.3.2 is an exploding gradient problem-solving technique as well as a regularization technique.

3.4.2 Dropout

Dropout, which was developed recently, is a widely used overfitting avoidance technique. At the time of training, the units of the internal and output layers are disabled by selecting them at a constant rate p . That is, learning is performed as if they did not exist from the beginning. On this occasion, selection of the unit to be disabled is performed at every time to update the weight. At the recognition, all the units are used to perform the forward propagation calculation. However, the outputs of the units belonging to the target of disabled layers are uniformly multiplied by p at the time of training [10].

As mentioned previously, there are many parameters in the RNN; even if we just use the RNN, trial and error for setting the parameters depending on the problem will be required. Therefore, in the task of activity recognition, an examination of the parameters, such as the numbers of layers and units, truncated time, and dropout rate, should contribute to the study of the activity recognition using the RNN.

4. ACTIVITY RECOGNITION USING RNN

We applied the DRNN described in Section 3 to human activity recognition to verify its accuracy and performance.

The items to be verified were as follows:

1. Does the recognition accuracy increase as compared to that of other algorithms?

2. Is any influence exerted on the accuracy when some parameters are changed?
3. How long is the throughput time of the recognition as compared with other algorithms?

4.1 Dataset

The HASC corpus is a dataset for machine learning gathered and distributed by HASC [14], distributed at a state with a detailed label attached to the data measured by sensors mounted on a mobile device. In this study, we used a part of the acceleration signals of the HASC corpus as a dataset. The dataset are divided into “segmented data” and “sequence data”, the former includes single activity in one trial and the latter includes multiple consecutive activities. The details of these two types of data are shown in Table 1. The segmented data are suitable for use as training data because they are able to label easily. On the other hand, since the sequence data are constructed by seamless measurement of human activities, these are resemble actual human activities.

4.1.1 Cross validation

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

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

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

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

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

Note that, because of the design of HASC dataset, for both sequence data and test data, the same person could be included.

As a measure of accuracy, the proportion of samples successfully recognized in the evaluation samples was used.

4.1.2 Additional dataset

As an additional dataset to examine the generality of our method, we adopted the Human Activity Recognition using Smartphones dataset (*HAR dataset*)[1] in the UCI Machine Learning Repository, and applied the best parameters found in the HASC dataset. The sensor data were collected using smartphones equipped with a three-axis accelerometer and a gyroscope. The smartphones were attached on the waists of the 30 persons. They have six types of activity classes, which are “Standing”, “Sitting”, “Laying”, “Walking”, “Walking downstairs”, and “Walking upstairs”, and compiled

as sequential data. For cross validation, we used the first three fourth samples as training data, and last one fourth as test data.

4.2 DRNN-based activity recognition

In order to perform high throughput activity recognition for each time by using the three-axis acceleration of a smartphone as the direct input, we constructed a DRNN such that the three-axis acceleration data of each time corresponded to the three-dimensional input layer, and six activity classes to the six-dimensional output layer. Each unit of each internal layer was an LSTM unit. The activation function of the output layer and the error function were defined by a softmax function and a cross entropy function, respectively. The truncated BPTT under the mini-batch stochastic gradient descent method was used to update the weights at the time of training. The number of internal layers, the number of units inside the internal layer, the number of times dating back was performed by truncated BPTT (called truncated time), the maximum gradient c , and also the dropout probability p were set to be variable in order to search the most appropriate value experimentally. The details of this DRNN are summarized in Table 2.

This network outputs an activity class, which corresponds to an element having the largest value among the elements of the output vector obtained when an input vector is input, as the recognized result.

A flow of the process of training and evaluation will be described below. The outline is also shown in Fig. 4.

- (0) Shuffle all the trials of training data and divide them into mini-batch sets of 20 trials.
- (1) For the first mini-batch,
 1. Take the time k at random.
 2. Let the truncated time be T , and divide the time range for truncated BPTT into $[k, k + T - 1]$, $[k + T, k + 2T - 1]$, \dots , $[k + K' - T, k + K']$, where the final value of the range was set as $K' = 1200$.
 3. For each range, obtain an error function from the input and output to update the weights by performing the error back-propagation.
- (2) Perform the same processing as (1) for the subsequent mini-batch.
- (3) Call a period until the processing for all mini-batches is complete *1 epoch*.
- (4) Obtain the accuracy for the test data to valuate the generalization performance.

Table 1: Details of HASC dataset

	Segmented data	Sequence data
Signal in one measurement	time [s], X axis [G], Y axis [G], Z axis [G]	time [s], X axis [G], Y axis [G], Z axis [G]
Frequency	100 [Hz]	100 [Hz]
Targeted activity	“stay”, “walk”, “jog” “skip” “stair up”, “stair down”	“stay”, “walk”, “jog” “skip” “stair up”, “stair down”
Measurement time	20 [s]	120 [s]
Type of Activity in one measurement	1	6
Number of person	7	7
Number of trials	540	18
Type	Single activity	Multiple activity

Table 2: Details of DRNN

Setting items	Detail
Activation function of output layer	Softmax
Error function	Cross entropy
Type of internal layer unit	LSTM
Mini-batch size	20
No. of time stamps in a mini-batch	$K' = 1200$
Initial Weights	random $[-0.1, 0.1]$
Initial bias	None
Learning rate adjustment	Adam
Input dimension	3
Output dimension	6

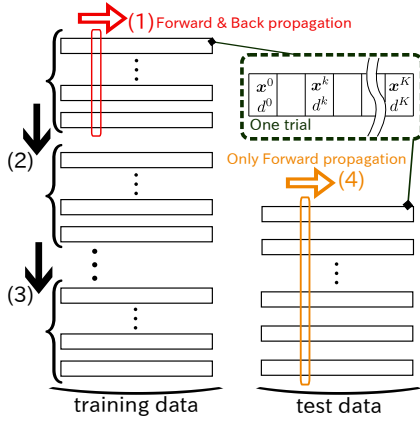


Figure 4: Flow of an epoch

Repeat (1) to (4), and stop the training after repeating a sufficient number of epochs. training phase ends in up here. After the training phase, the activity recognition of the sequence data are executed by using only the forward propagation of the learned model. Record the accuracies of training data, test data, and sequence data in each epoch to plot the changes.

Chainer [29], provided by Preferred Networks, Inc., was used to implement the DRNN. Chainer is a framework (FW) for NN. In Chainer, various NN models can be flexibly written in the Python language. The experiment was conducted in an environment as shown in Table 3.

Here, the training time was reduced by parallel processing using the GPU, and the CPU was used for evaluating of the throughput of the constructed DRNN. This strategy is based on a following policy; the RNNs is trained by a large-scale computer architecture, and its execution is done by a standard type mobile terminal.

4.3 Comparative methods

As comparative methods, decision tree, support vector machine (SVM), and random forest were used.

The comparative methods, rather than the raw sensor data, and require time windows to calculate the feature vectors. Therefore, we extracted feature vectors from the three-axis accelerometer data. For the sensor data, time windows of 5 s were extracted, shifting every 2.5 [s], as in Bao et al. [3].

For each time window, we calculated 27 feature values, following the studies in [33, 34, 12]. The number of feature variables used in each time window was 27, including: (1-3) mean value of each axis, (4-6) variance of each axis, (7) mean sum of the absolute values of each axis, (8-9) first and second eigenvalue of the covariance matrix between the axes, (10) sum of the vertical component ratios for the intensity, (11-13) covariance ratio in the x and y -direction for the z -component variance of each axis, (14-16) variance ratio of the back and forth difference in the x and y -direction for the

variance of the back and forth difference in the z -direction of each axis, (17-19) mean FFT-domain energy of each axis, (20) mean FFT-domain energy of the intensity, (21-23) FFT-domain entropy of each axis, (24) FFT-domain entropy of the intensity, (25) number of mean crosses of the mean intensity, (26) number of crosses of the zone of the mean intensity $\pm 0.1[G]$, and (27) number of samples outside the zone of the mean intensity $\pm 0.1[G]$.

We reduced these 27 feature variables to 13 by applying stepwise-feature selection [8] using logistic regression. As a result, 13 feature variables, 1, 2, 6, 7, 9, 11, 12, 13, 15, 20, 21, 24, and 26, were adopted.

For these selected feature vectors, machine learning methods by decision tree, SVM, and random forest was applied, and in each of these a grid search were conducted over the training data to choose the best model.

4.4 Throughput evaluation

For evaluating the throughput of the recognition, the time required for the recognition of the entire sequence data was divided by the number of samples of the sequence data to derive the mean value of the recognition throughput per time unit. For the comparative methods, we calculated the computation time of a feature vector in one time window, the time taken to recognize the activity from the feature vector in one time window, and the sums of these values.

5. RESULTS

In the following, we compare the results of evaluating the accuracy of the activity recognition with the DRNN by searching various parameters with those of the accuracy of the existing technique. First, after showing the best model that yielded the highest accuracy, we show the accuracy when the parameters are varied. Furthermore, for the throughput of the recognition, we compare the time required for feature calculation and recognition in the existing technique and the time required until the result is output after inputting one sample in the DRNN.

5.1 Best model

The best model is the model that showed the best recognition result for the sequence data during the experiment. The parameters selected in the best model are shown in Table 4.

Fig. 5 shows the transition of the correct recognition rate in each epoch. The “recognition rate” is derived by the ratio of the correct recognition time against total time for each trial, and it is also referred to simply as accuracy. In the best model, the test recognition rate was 95.42% at maximum. The recognition rate for the sequence data was 83.43% at maximum.

In Fig. 5, it can be seen that the recognition rate increases as the epochs proceed. According to the results in this figure, we judged that it is reasonable to stop the training at about epoch 80, and for the subsequent evaluations, we extracted the average recognition rate from epoch 71 to epoch 80 for comparison.

Table 3: Computing environment

OS	Ubuntu14.04LTS (64-bit)
CPU	Intel Corei5-4590 3.3 GHz
RAM	DDR3-1600 24 GB
GPU	NVIDIA Quadro K2200
FW	Chainer 1.5.1
Python Ver.	2.7.6
CUDA Ver.	7.0

Table 4: Best model parameters

Parameters	Best value
Number of internal layers	3
Number of units in one layer	60
Truncated time	$T = 30$
Gradient clipping parameter	$c = 5$
Dropout rate	$p = 0.5$

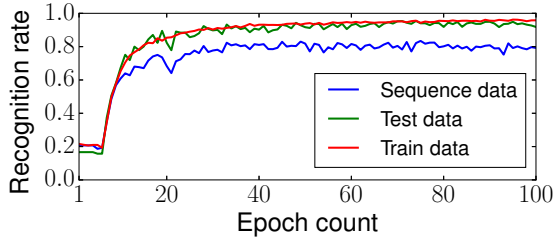


Figure 5: Accuracy transition of the best model

5.2 Comparison with existing methods

Fig. 6 shows a graph comparing the accuracy of the existing method with that of the proposed method. The Bar represents the mean recognition rate from epoch 71 to epoch 80. In the test data, the DRNN shows results that are 35.18%, 27.76% and 22.35% better than those of decision tree, SVM, and random forest, respectively. In the sequence data, it shows results that are 28.03%, 26.04% and 26.74% better than those of decision tree, SVM, and random forest, respectively.

5.3 Varying parameters

The results obtained by changing the number of internal layers are shown in Fig. 7. The thin line at the top of bar 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 recognition rate is highest in the case of three layers for any of the training, test, and sequence data. In particular, for the sequence data, the recognition rate 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. 8. We changed only the number of internal 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 recognition rate is highest in the case of 60 units in test and sequence data. In particular, in the sequence data the recognition rate 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. 9. 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.

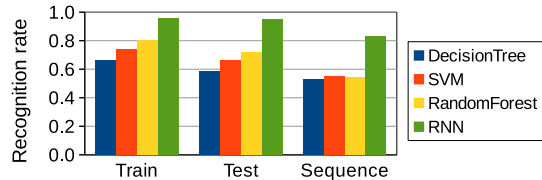


Figure 6: Accuracy of comparative methods and best model

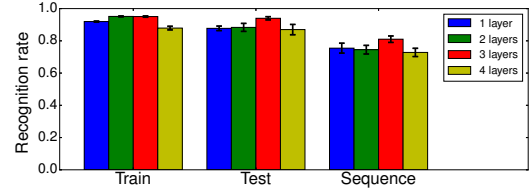


Figure 7: Comparison of accuracy according to the number of internal layers

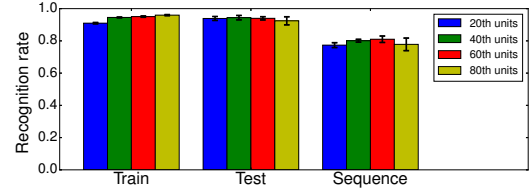


Figure 8: Comparison of accuracy according to the number of units

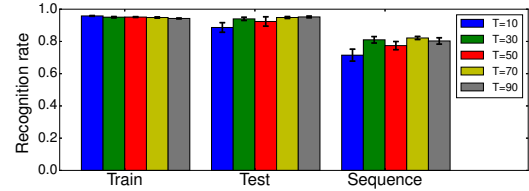


Figure 9: Comparison of accuracy according to the truncated times

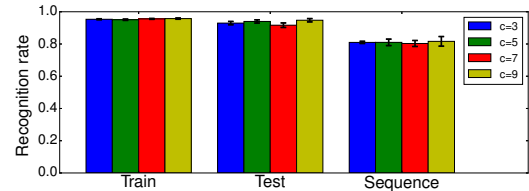


Figure 10: Comparison of accuracy according to the gradient clipping parameters

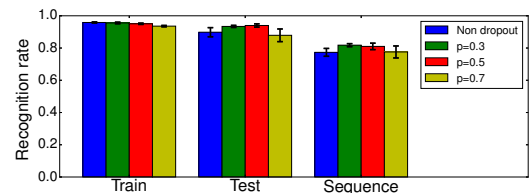


Figure 11: Comparison of accuracy according to dropout rates

The results of experiments where the gradient clipping parameter was changed are shown in Fig. 10. We changed only the gradient clipping parameter, using 3, 5, 7, and 9, among the parameters of

the best model. The figure shows that no significant recognition rate difference due to the variation in the gradient clipping parameter can be observed.

The results of experiments where the dropout probability p was changed are shown in Fig. 11. We changed only the dropout probability p , using 0, 0.3, 0.5, and 0.7, among the parameters of the best model. The $p = 0.3$ model showed the highest recognition rate for the test and sequence data. In particular, for the sequence data, the recognition rate of this model was about 4.6% higher than that of the lowest recognition rate model without dropout.

The above results show that, in the task of activity recognition in this time, a large difference appeared in the recognition rate for five parameters. In particular, for the parameter of truncated time, there is a difference of 10.7% between the maximum and minimum recognition rate, revealing that the parameter adjustment is effective.

5.4 Throughput of activity recognition

We summarize the throughputs of the proposed method and the existing method in Table 5. When compared according to only the calculation time at the time of recognition, the existing method is 1.342 [ms] faster, but if compared according to the substantial time, the proposed method is 9.671 [ms] faster, because the existing method requires that the feature vector extracted as a pre-processing.

5.5 Result with additional dataset

As a result of applying our method and parameters for HAR dataset, the recognition rate was 95.03% at 45th epoch. with the cross validation with first three fourth samples as training data and last one fourth as test data. Fig. 12 shows the transition of the correct recognition rate in each epoch.

6. DISCUSSION

We consider the proposed method in the light of the experimental results obtained. In addition, in the following, we give more importance to the accuracy for the test and sequence data than the accuracy for the training data as a basic evaluation criterion.

6.1 Activity recognition with the best model

Using the best model, it was possible to perform the recognition with a higher recognition rate and faster response speed than those of the traditional methods. In particular, an recognition rate of 95.42% at maximum was obtained for the segmented test data.

On the other hand, for the sequence data, the recognition rate dropped to 83.43% at maximum. The recognized results for the sequence data shown in Fig. 13 verify this. The horizontal axis represents 10 [ms] per 1 time in time number and the vertical axis represents acceleration in the gravitational acceleration unit [G]. A color chart displayed as "True" represents the correct solution label, and a color chart displayed thereunder as "Es" represents the estimation label. In Fig. 13(a), it can be observed that in general the recognition performance was good, but it caused similar activities to be erroneously recognized at the time 6000 – 7000. In addition,

it caused a delay in recognition near the time 2300, as seen in Fig. 13(b). This phenomenon is considered to have been caused by the fact that unlearned signals that cannot be classified into any activity during the transition of activity were input.

6.2 What are the optimal parameters?

6.2.1 Number of layers and units

Because NNs can handle higher order feature vectors, as the number of layers is increased, the goodness of fit to the training data is increased. It can be seen in Fig. 7 that, in the present experiment, the training accuracy increased as the number of layers was increased up to three in the internal layer. However, the accuracy decreased in the four-layer model. This phenomenon can be interpreted to have occurred as a result of increasing in the learning difficulty by the increase in the excessive number of layers. Further, for the generalization performance, it is known that the accuracy is liable to decrease because of the overfitting when the freedom of the model becomes too high. This fact was also demonstrated in this experiment by the results for the test and sequence data shown in Fig. 7. According to the above, an unnecessary increase in the layers in the model design should be avoided, because it may lead to a reduction in the generalization performance.

Furthermore, when the number of internal layers is increased, the computation time and memory usage are increased. In this experiment, the computation times taken per epoch were 58.89 [s] with the single layer, 89.59 [s] with two layers, 116.39 [s] with three layers, and 144.83 [s] with 4 layers of the internal layer, respectively. In addition, the throughputs at the time of recognition were 0.512 [ms] with single layer, 0.909 [ms] with two layers, 1.347 [ms] with three layers, and 1.720 [ms] with four layers of the internal layer, respectively. This time, we chose the best model based on the accuracy, but if we prefer to obtain a high throughput at the expense of accuracy, simplifying the calculation by reducing the layers should be considered.

Almost the same consideration may be also possible for the number of units as the number of layers. As can be seen in Fig. 8, in order to obtain a high generalization performance, the number of units should not be excessively increased. Further, when increasing the number of units, because the amount of computation time and memory usage increases, an adjustment will be required when a trade-off is implemented.

6.2.2 Truncated time

For the truncated time, it can be seen in Fig. 9 that the performance decreased for the test and sequence data at $T = 10$, but the optimum value cannot be obtained stably. Here, the recognition rate, which was relatively high at $T = 30$ and $T = 70$, is about 200 and 470, respectively, if converted into beats per minutes (BPM). It is considered that one cycle of human walking and activities cap-

Table 5: Evaluation of throughput

	Feature [ms]	Recognition [ms]	total [ms]
Decision Tree	11.027	0.004	11.031
SVM	11.027	0.123	11.150
Random Forest	11.027	0.056	11.083
RNN	-	1.347	1.347

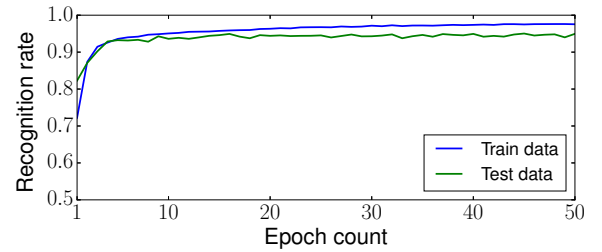
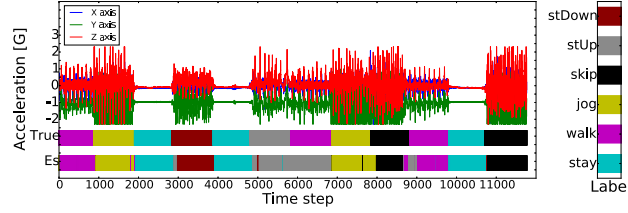
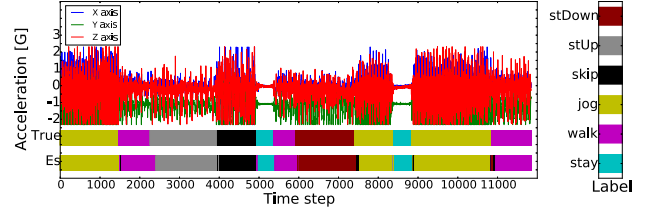


Figure 12: Result for HAR dataset



(a) Recognition example 1



(b) Recognition example 2

Figure 13: Visualization of the recognized result

tured by the network may fall within a range of this degree. Thus, in order to capture the features of input signals, it is considered effective to take the variation period of the signals being handled into account when determining the truncated time.

6.2.3 Gradient clipping parameter

In Fig. 10, no significant performance difference generated by the gradient clipping parameters among the data used in this experiment is seen. It is possible that gradient explosion did not occur to a great extent for the data in this time. In addition, in the DRNN, because the likelihood that gradient explosion will occur increases as the truncated time and the number of layers increase, it is necessary to suppress the gradient moderately by taking also the values of other parameters into consideration.

6.2.4 Dropout rate

First, as can be seen in Fig. 11 in the cases where the dropout was and was not applied, applying it leads to a tendency that the recognition rate is liable to become higher for the test and sequence data. For the dropout probability, approximately 50%, as frequently used in CNNs, is considered to be appropriate also in the RNN.

6.2.5 Validity of parameters

It is found from Fig. 12 that when the parameters adjusted for the recognition of HASC dataset were utilized for the training of the HAR dataset, a high recognition rate of 95.03% was obtained.

6.3 Throughput and training time

The throughput of the recognition was 1.347 [ms], an 8.19 times faster speed than that of the existing method. Considering that the data are currently acquired at 100 [Hz], this throughput is sufficient to allow real time processing. In addition, because the RNN basically performs only the product-sum operation by the number of times of the dimensions of R and W at the time of recognition, it is considered that implementation in low power devices such as smartphones may also be possible in the future.

On the other hand, the training time was 116.39 [s] per epoch on average. This is a very large and non-negligible amount of time; however, by calculating the training using a high-speed computer in advance, high throughput processing may be possible at the time of recognition. In addition, in principle, because the DRNN can perform the online learning on a mini-batch basis, by devising a method of feeding the training data, a high throughput can be expected also in the training.

The DRNN can achieve both high-speed response with high recognition rate by using CPU. These advantages are caused by compactly and small size of the DRNN. In concrete, the total size of the inner variables and architecture from the input layer to the output layer of the DRNN is 74166 pcs, and this is the size of less than 10 % of the conventional CNN+LSTM model [23]. Our approach

is shown that have a clear advantage for the miniaturization of the devices of recognition.

6.4 Future direction

In this experiment, the activity recognition results of the DRNN were good in terms of recognition rate and throughput. However, new techniques are constantly being developed also for RNNs. Many techniques can be utilized for the activity recognition, such as an approach [21] using the ReLU function instead of LSTM, a method [24] to automatically organize the feature vectors by applying the pre-training used in the CNN, and a method [30] to accurately estimate the behavior of things by using the physical laws of the real world and its simulation. The verification of these techniques is one of the challenges for future studies.

In addition, in this study the recognition rate of the sequence data decreased. However, to resolve this issue, it would be possible to apply the HMM as a post-processing, or to apply a method that takes into account the context of the label in the RNN, called connectionist temporal classification [7]. And, we develop a compact DRNN circuit to equip into a mobile device.

7. CONCLUSION

In this paper, the DRNN was constructed for human activity recognition using raw time series data of acceleration sensors mounted on a mobile device with high recognition rate and high throughput. The maximum recognition rate was 95.42% against the test dataset and was 83.43% against multiple sequential test dataset. Here, the maximum recognition rate by traditional methods was 71.65% and 54.97% respectively. Further, the efficiency of the tuned parameters was confirmed by using the sequential dataset. For the throughput of the recognition per unit time, the constructed DRNN requires only 1.347 [ms], while the traditional method requires 11.031 [ms] which includes 11.027 [ms] for feature extraction. In the future, many techniques, such as a forget-mechanism and pre-training, optimization methodology, and a method that takes the series into consideration, should be studied.

8. ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Number 26280041.

9. REFERENCES

- [1] Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. A public domain dataset for human activity recognition using smartphones. In *ESANN* (2013).
- [2] Avci, A., Bosch, S., Marin-Perianu, M., Marin-Perianu, R., and Havinga, P. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey.

- In *Architecture of computing systems (ARCS), 2010 23rd international conference on*, VDE (2010), 1–10.
- [3] Bao, L., and Intille, S. S. *Activity Recognition from User-Annotated Acceleration Data*. 2004.
 - [4] Bhattacharya, S., Nurmi, P., Hammerla, N., and Plötz, T. Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive and Mobile Computing* 15 (2014), 242–262.
 - [5] Bulling, A., Blanke, U., and Schiele, B. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys* 46 (2014), 1–33.
 - [6] Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: continual prediction with LSTM. *Neural computation* 12, 10 (2000), 2451–2471.
 - [7] Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, ACM (2006), 369–376.
 - [8] Guyon, I., and Elisseeff, A. An introduction to variable and feature selection. *Journal of Machine Learning Research* 3 (2003), 1157–1182.
 - [9] Haykin, S. *Neural networks-A comprehensive foundation*, 1994.
 - [10] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv: 1207.0580* (2012), 1–18.
 - [11] Hochreiter, S., Hochreiter, S., Schmidhuber, J., and Schmidhuber, J. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–80.
 - [12] Inoue, S., Ueda, N., Nohara, Y., and Nakashima, N. Mobile Activity Recognition for a Whole Day: Recognizing Real Nursing Activities with Big Dataset. In *ACM Int’l Conf. Pervasive and Ubiquitous Computing (Ubicomp)* (Osaka, 2015).
 - [13] Inouye, S. K., Foreman, M. D., Mion, L. C., Katz, K. H., and Cooney, L. M. Nurses’ recognition of delirium and its symptoms: comparison of nurse and researcher ratings. *Archives of internal medicine* 161 (2001), 2467–2473.
 - [14] Kawaguchi, N., Ogawa, N., Iwasaki, Y., Kaji, K., Terada, T., Murao, K., Inoue, S., Kawahara, Y., Sumi, Y., and Nishio, N. Hasc challenge: gathering large scale human activity corpus for the real-world activity understandings. In *Proceedings of the 2nd Augmented Human International Conference*, ACM (2011), 27.
 - [15] Kim, E., Helal, S., and Cook, D. Human Activity Recognition and Pattern Discovery. *Pervasive Computing, IEEE* 9 (2010), 48–53.
 - [16] Kingma, D. P., and Ba, J. L. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations* (2015), 1–13.
 - [17] Krishnan, N. C., and Cook, D. J. Activity recognition on streaming sensor data. *Pervasive and Mobile Computing* 10, PART B (2014), 138–154.
 - [18] Kunze, K., Barry, M., Heinz, E., Lukowicz, P., Majoe, D., and Gutknecht, J. Towards Recognizing Tai Chi {Å} An Initial Experiment Using Wearable Sensors. In *Applied Wearable Computing (IFAWC), 2006 3rd International Forum on*, VDE (2006), 1–6.
 - [19] Ladha, C., Hammerla, N. Y., Olivier, P., and Plötz, T. ClimbAX: skill assessment for climbing enthusiasts. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, ACM (2013), 235–244.
 - [20] Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., and Campbell, A. T. A survey of mobile phone sensing. *IEEE Communications Magazine* 48 (2010), 140–150.
 - [21] Le, Q. V., Jaitly, N., and Hinton, G. E. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *CoRR abs/1504.0* (2015).
 - [22] Mazilu, S., Blanke, U., Dorfman, M., Gazit, E., Mirelman, A., M Hausdorff, J., and Tröster, G. A Wearable Assistant for Gait Training for Parkinson’s Disease with Freezing of Gait in Out-of-the-Lab Environments. *ACM Transactions on Interactive Intelligent Systems (TiIS)* 5, 1 (2015), 5.
 - [23] Ordóñez, F. J., and Roggen, D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16, 1 (2016), 115.
 - [24] Pasa, L., and Sperduti, A. Pre-training of Recurrent Neural Networks via Linear Autoencoders. *Advances in Neural Information Processing Systems* 27 (2014), 3572–3580.
 - [25] Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. *Proceedings of The 30th International Conference on Machine Learning*, 2 (2012), 1310–1318.
 - [26] Saeedi, R., Schimert, B., and Ghasemzadeh, H. Cost-Sensitive Feature Selection for On-Body Sensor Localization. *2nd International Workshop on Human Activity Sensing Corpus and its Application (HASCA2014) held at UbiComp 2014* (2014), 833–842.
 - [27] Strohmman, C., Harms, H., and Tröster, G. What do sensors know about your running performance? In *Proceedings - International Symposium on Wearable Computers, ISWC* (2011), 101–104.
 - [28] Sutskever, I. Training Recurrent neural Networks. *PhD thesis* (2013), 101.
 - [29] Tokui, S., Oono, K., Hido, S., and Clayton, J. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)* (2015).
 - [30] Wu, J., Yildirim, I., Lim, J., Freeman, W., and Tenenbaum, J. Galileo : Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning. *Advances in Neural Information Processing Systems* 28 (NIPS 2015) (2015), 1–9.
 - [31] Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L., and Krishnaswamy, S. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, Buenos Aires, Argentina (2015), 25–31.
 - [32] Zhan, K., Faux, S., and Ramos, F. Multi-scale Conditional Random Fields for first-person activity recognition. *Pervasive Computing and ...* (2014).
 - [33] Zhang, M., and Sawchuk, A. Motion primitive-based human activity recognition using a bag-of-features approach. *Proceedings of the 2nd ACM SIGHIT ...*, 1 (2012), 631.
 - [34] Zhang, M., and Sawchuk, A. A feature selection-based framework for human activity recognition using wearable multimodal sensors. In *Int. Conf. Body Area Networks* (2011), 92–98.