2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)
Beijing, China, November 6-9, 2018

Multi-Sensor Fusion based Robot Self-Activity Recognition

Dingsheng Luo, Yang Ma, Xiangqi Zhang, Xihong Wu

Key Lab of Machine Perception (Ministry of Education), Speech and Hearing Research Center
Department of Machine Intelligence, School of EECS, Peking University, Beijing 100871, China
Email: {dsluo, mayang0001, zhangxiangqi, xhwu}@pku.edu.cn

Abstract— **Robots play more and more important roles in our daily life. To better complete assigned tasks, it is necessary for the robots to have the ability to recognize their self-activities in real time. To perceive the environment, robots usually equipped with rich sensors, which can be used to recognize their self-activities. However, the intrinsics of the sensors such as accelerometer, servomotor and gyroscope may have significant differences, individual sensor usually exhibits weak performance in perceiving the environment. Therefore, multi-sensor fusion becomes a promising technique so that to achieve better performance. In this paper, facing the issue of robot self-activity recognition, we propose a framework to fuse information from multiple sensory streams. Our framework takes Recurrent Neural Network(RNN) that uses Long Short-Term Memory(LSTM) units to model temporal information conveyed in multiple sensory streams. In the architecture, a hierarchy structure is used to learn the sensor-specific features, a shared layer is used to fuse the features extracted from multiple sensory streams. We collect a dataset on PKU-HR6.0 robot to evaluate the proposed framework. The experiment results demonstrate the effectiveness of the proposed framework.**

I. INTRODUCTION

Robots play more and more important roles in our daily life, e.g. robots are used to serve human society in daily life by making foods, doing cleaning and delivering medications in hospital,etc. When they execute the assigned tasks they need to handle tough environment like uneven terrains which can lead them to lose balance. Besides, the accumulation of the machine execution error may also lead robots to get out of control. As an extreme example, a walking robot accidentally fell, however, robot may keep moving its legs under a lying down status and may subjectively and wrongly think it still in walking status, since the current executing controller is for walking. To face the external harsh environments and internal execution error, the capability of self-activity recognition becomes significantly important, which can help robots to know their current status and ongoing activities so that to output the judgement about whether everything goes smoothly. Thereafter, robots can be free from occasional situations like violating constraints, running some already failed tasks or executing disorder activities.

To achieve such a goal, among various methodologies on activity recognition, a sliding window based method [1] is firstly investigated. In this kind of method, usually, a sliding window is firstly adopted for feature extraction and a distinguish classifier like Support Vector Machine(SVM), Naive Bayes(NB) or Decision Tree(DT) is followed for further activity classification. However, there are several drawbacks in the sliding window based method. In feature extraction process, typical features we select include time domain features like Standard Deviation and frequency domain features like Direct Current. But these features may not be representative as they are limited to heuristic knowledge. It is also time consuming to gain the features within the sliding window like frequency domain feature extraction via Fast Fourier Transformation (FFT). The discriminant classifiers like SVM, NB and DT can only use part of the context information, they can not model long sensor signal sequence that represent activity well. Because deep learning has achieved great success in the field of computer vision, speech recognition and many other, more and more researchers have used deep learning for activity recognition [2], [3], [4], [5]. One great power of deep learning models is their capacity of learning high level representation from raw data. In [2], Convolutional Neural Network(CNN) was used to automatically learn high level abstraction from raw time series signals. To further handle the temporal dependencies, Recurrent Neural Network(RNN) which can model the history information is a suitable choice. However, RNN is hard to train when the sequence is long because of the vanishing gradient problem [6]. To handle the long sequence problem in robot self-activity recognition, we use Long Short Term Memory(LSTM) to replace RNN. LSTM is introduced to solve the vanishing gradient problem by extending RNN with memory cell[7], which can be used to store, output and ease information. In [3], [4], [5], LSTM showed capacity for learning long temporal dependencies.

There are some studies on recognition capability for robots. Robots usually carry on many sensors which can be used for their self-activity recognition. Sensor signals like accelerometer data, gyroscope data, joint angular positions, sonar data, lidar data and vision information from first-view camera can help a lot. In [8] acoustic signals are used for a legged robot to percept terrain types. In [9], vibration data is used for an outdoor robot percept different ground types. However, these works may not fully used the sensors worn on robot body. In order to avoid possible drawbacks derived from each individual sensor, multi-sensor fusion technology has received attention in recent decades [10]. In our research, to better recognize robot status multi-sensor fusion based method is considered. This is also inspired by the fact that humans combine signals from the five body senses (sight,

978-1-5386-7282-2/18/$31.00 ©2018 IEEE

sound, smell, taste, and touch) to better perceive the world including the inner status and the outer environments [11].

To overcome the drawbacks of sliding window based methods mentioned above, we use an end-to-end LSTM based framework as [12] which just takes a sample of raw sensor signal sequence as input. With this end-to-end LSTM based framework, our approach can work just in time, such that the classification can be done online at every time step. To learn sensor-specific features we use a hierarchy structure as [13] which feed different sensor signal sequences to different sub-models and fuse the extracted features in intermediate layers of the network. To fuse the extracted features better we add a shared layer which shared by different sub-models into the architecture describe before.

The remainder of this paper is organized as follows. In Section 2, we introduce the related work on activity recognition and sensor fusion. In Section 3, we first review the RNN and LSTM, then illustrate our end-to-end Hierarchy LSTM framework with shared layer. Activity dataset, experimental settings and results are given in Section 4. Finally, we conclude the paper in Section 5.

II. RELATED WORK

Investigation from literatures shows that few works have been done on robot self-activity recognition, but we can refer experience from the works on human activity recognition, for which a number of classifiers and various feature extraction methods were investigated. These methods can be integrated into a typical chain that contains four stages [1], firstly acquiring data from multiple sensors with some preprocessing, then segmenting the preprocessed data, afterward representative features are extracted from the segments, finally classifiers are trained with the extracted features and their corresponding labels.

Sliding window based method is a dominant framework for activity recognition. The procedures of this method is that, firstly using sliding window for sequence segmentation and feature extraction, then followed by discriminative classifiers like SVM or NB. The sliding window moves over the sequence with a fixed window length and step size to do data segmentation, window length and step size are parameters that require our prior knowledge. The parameters can be seen as a trade-off between accuracy and efficiency [14], the larger the step size and window length are the more accuracy we can get, but the time we need to go over the sequence and extract the designed features increases at meanwhile. In this framework, the accuracy may also limited by classifiers like SVM or NB, for they can only use partial context information.

With the great success of deep learning in other fields like Natural Language Processing, Computer Vision and Speech Recognition, more and more researchers have adopted deep learning for activity recognition [3], [2], [4], [5]. Two main problems need to be solved in activity recognition, i.e. how to get proper representation from the raw sensor signals and how to capture the temporal dependencies lay under sensor sequence. So deep learning models which can learn high level representation and model temporal dependencies are suitable choices.

In [2], an architecture employed CNN to do automatically feature extraction for activity recognition is proposed, and it got state-of-art result in Opportunity Activity Recognition dataset and Hand Gesture Dataset. One advantage of this proposed architecture is feature extraction with no need for hand-crafted manners and the other main advantage is feature extraction process and classification process are unified into one process to mutually enhanced performances.

Activities are composed by a series of complex relative movements, LSTM which can model the long term contextual information of sequential inputs is a fundamental choice. LSTM is usually stacked within or between layers to do features extraction, features fusion and sequence modeling [3], [5], [9], [13]. As for robot self-activity recognition, [15] which used Long-term Recurrent Convolutional Network(LRCN), may be one of the early attempts. It combined first-view vision features which extracted by CNN together with joint positions of arm as a representation for robot activity, then used LSTM to learn the pattern underlay the sequential features.

Signals which can help robots recognize their self-activities are often observed using multiple sensors. Sensor fusion which means to get a richer representation compared to use the sensor signals individually is needed [11]. Sensor fusion is usually classified into three categories, early fusion or data-level fusion, intermediate fusion or feature-level fusion and late fusion or decision-level fusion [16].

Data-level fusion means fusing the raw or preprocessed data from different sensors. In [3], the authors use signals collected from ten sensors worn on ten different body positions to classify the activity performed. In their work, authors concatenated the ten separate sensor signals and then used the concatenated sensor signals as the input of a Deep LSTM model. However, the raw data-level fusion may not consider the sensor-specific intrinsics, especially for sensors that vary significantly.

Feature-level fusion means the fusion of the hidden layer units of different sub-models, this often leads to a hierarchy structure in neural network. In [5], a hierarchical LSTM is used for human activity classification. In their work, according to the human body structure, they decomposed the body into five parts including two arms, two legs and one trunk. The lower level of the network is used to model the Separate five parts describe above, with the network going deeper it focus more on the combination of individual parts, e.g. combine left arm and trunk into left upper body and then combine left upper body and right upper body into upper body and finally combine upper body and lower body into whole body. In a driver activity participation task [13] three parallel LSTMs are used in first layer to extract features from three different sensors including a camera facing the car driver, a camera facing the road and a GPS recording vehicle's dynamics. From these works, we can see that the hierarchy structure compared to the deep model without hierarchy structure can help to learn sensor-specific features

and the common features underly in the separate extracted features.

Decision-level fusion means the aggregation(e.g. averaging score or voting) of decisions from classifiers trained separately on different sensor signals. Late fusion performs bad unless the input sensor signals are significantly uncorrelated [16].

The structure we propose combines the two structures of Data-level and Feature-level. We use separate LSTM layer to learn sensor-specific features, and a shared layer to let the network better integrate features from different sensors.

Among these different kinds of fusion methods, feature-level fusion is used mostly widely, because of the facility provided by the neural network architecture. However, when and how to integrate hidden units requires research.

III. Multi-Sensor Fusion based Model

In this part ,we first give an overview of recurrent neural network(RNN) and Long-Short Term Memory(LSTM) to let this chapter be self-contained. Continually, we introduce the LSTM based end-to-end architecture which can help robots identify in real time. Then introduce the hierarchy structure we used and the shared layer we proposed to fuse the sensor-specific features.

A. Review of RNN and LSTM

The main difference between RNN and other feedforward neural networks is the inputs of RNN not only contains the input of current time step but also the hidden state of the previous time step. Due to this recurrent structure, RNN is able capture the temporal dependencies in sequential inputs. Given an input sequence $x = (x_1, x_2, ..., x_t)$, the hidden states of a recurrent layer $h = (h_1, h_1, ..., h_t)$ and the outputs of RNN $y = (y_1, y_2, ..., y_t)$ can be calculated as follows

$$h_t = f\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right) \tag{1}$$

$$y_t = Softmax\left(W_{oh}h_t + b_o\right) \tag{2}$$

where W_{hx}, W_{hh} and W_{oh} denote the connection weights from input layer to hidden layer, hidden layer to hidden layer and hidden layer to output layer respectively. b_h and b_o represent bias vectors for hidden layer and output layer. f is activation function which used $tanh$ or $sigmoid$ commonly and $Softmax$ is activation function that maps a vector to different class with different probability.

When facing long sequence, RNN is hard to train due to the exploding gradient and vanishing gradient problem [6], [17]. As for the exploding gradient problem gradient clipping technique can be used [17], and for vanishing gradient problem LSTM was designed [7]. Fig. 1 illustrates a LSTM unit. It contains a memory cell which convey information through out the time sequence and three different gates including a input gate, a forget gate and a output gate. The three gates control three operations to the information stored in memory cell, add(input gate), remove(forget gate), output(output gate).

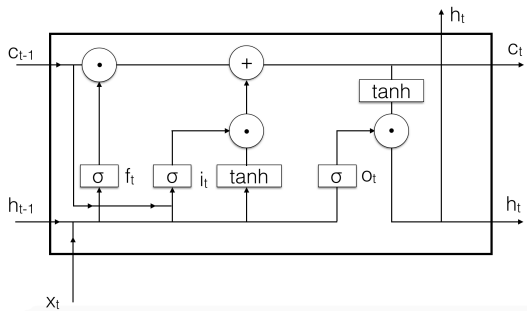


Fig. 1. An illustration of Long Short Term Memory cell architecture.

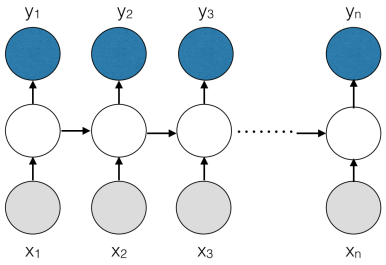


Fig. 2. Sequence-to-sequence training procedure for RNN, which maps input sequence X = (x_1, x_2, x_3, ... , x_n) to target sequence Y = (y_1, y_2, y_3, ... , y_n).

The activations of the memory cell and three gates can be calculated as follows

$$f_t = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + W_{fc}c_{t-1} + b_f\right) \tag{3}$$

$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + W_{ic}c_{t-1} + b_i\right) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh\left(W_{cx}x_t + W_{ch}h_{t-1}\right) \tag{5}$$

$$o_t = \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + W_{oc}c_{t-1} + b_o\right) \tag{6}$$

$$h_t = o_t \odot tanh\left(c_t\right) \tag{7}$$

where h is hidden states. f, i, o and c are the forget gate, input gate, output gate and cell states respectively, they are all the same size with the hidden states. σ is logistic function, $tanh$ is activation function using $tanh$, $\odot$ is product, they are all applied element-wise. W_{fx}, W_{fh}, W_{fc}, W_{ix}, W_{ih}, W_{ic}, W_{cx}, W_{ch}, W_{ox}, W_{oh}, W_{oc} are weight matrixes mapping from input to output(e.g. W_{ox} maps input x to output gate o).

B. End-to-End Framework

We use an end-to-end LSTM based framework that take each sample of sensor signal sequence as input. Instead of use the label data at the last time step of input sequence, we use the label to serve as supervised data at every time step. Now suppose the data is in pair of input sequence $x = (x_1, x_2, ..., x_T)$ and class label y, we train our model with sequence-to-sequence LSTM which maps $(x_1, x_2, ..., x_T)$ to $(y_1, y_2, ..., y_T)$ where $y_i = y$ for every i in range T. This framework is showed in Fig. 2. This sequence-to-sequence

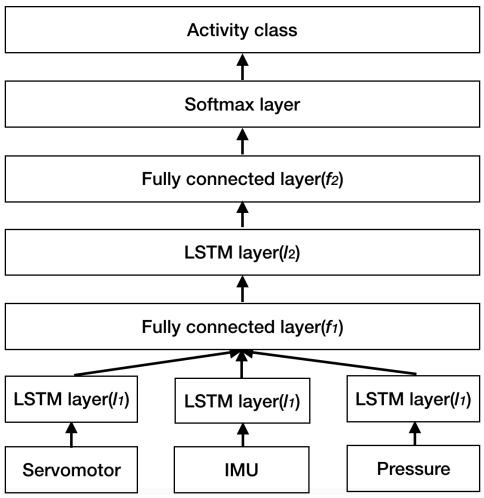


Fig. 3. Architecture of Hierarchy LSTM.

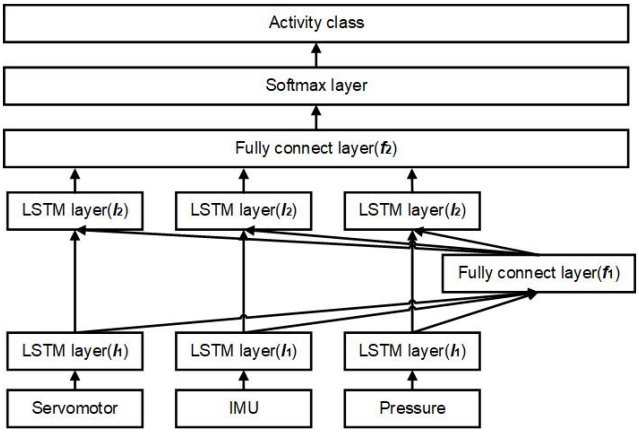


Fig. 4. Architecture Hierarchy LSTM with a shared layer.

procedure can achieve our goal of real-time recognition and can force the classifier to classify correctly as early as possible at the same time.

To use our model in a classification task, Softmax layer is taken as the last layer of the neural network architecture. Softmax function can map the hidden states to a real-valued vector, the vector can be used to represent a categorical distribution, what is a probability distribution over K different possible classes. The probability for class c given the hidden states has the form that

$$P(y = c|h) = \frac{exp(W_c h + b_c)}{\sum_{i \in K} exp(W_i h + b_i)} \tag{8}$$

where h is the hidden states, W_i and b_i represent the weight and bias for class c respectively and K is the total number of activity categories.

We use a loss layer with cross entropy loss, for a sequence of length T, the loss function has the form that

$$loss = \sum_{t \in T} \sum_{k \in K} -z_{tk} log(y_{tk}) \tag{9}$$

where z is the one-hot encoding of class label and z_{tk} is equal to 1 if the label at time step t equals to k and otherwise 0, y_{tk} is the probability of k we predict at time step t.

C. Hierarchy Architecture with a Shared Layer

To better introduce the shared layer we use, we first review the base hierarchy architecture which our architecture derived from. The architecture is shown in Fig. 3. In the first LSTM layer l_1, three sensor signals are fed into three different LSTM layer l_1. A fully connected layer f_1 is used to fuse the representations of the three sub-models. Then LSTM layer l_2, fully connected layer f_2 and Softmax layer are sequentially applied to get the prediction of behavior. The network architecture we use is illustrated in Fig. 4. Compare to the hierarchy structure in Fig. 3, we keep separate LSTM layer l_1 to learn sensor-specific features, and we add the shared layer f_1 to let the network better integrate features from different sensors.

Now we have three sensory streams, S = $\{s_1, s_2, ..., s_t\}$, I = $\{i_1, i_2, ..., i_t\}$, P = $\{p_1, p_2, ..., p_t\}$, where S denotes the signal sequence of servomotors, s_t denotes the servomotor signal at timestamp t. I and i_t, P and p_t for IMU and pressure transducer correspondingly. For the first LSTM layer l1, we can compute the hidden state for LSTM layer l1 by,

$$(h^t_{1_s}, c^t_{1_s}) = LSTM^1_s(s_t, h^{t-1}_{1_s}, c^{t-1}_{1_s}) \tag{10}$$

$$(h^t_{1_i}, c^t_{1_i}) = LSTM^1_i(i_t, h^{t-1}_{1_i}, c^{t-1}_{1_i}) \tag{11}$$

$$(h^t_{1_p}, c^t_{1_p}) = LSTM^1_p(p_t, h^{t-1}_{1_p}, c^{t-1}_{1_p}) \tag{12}$$

then we use a fully connected layer the get the shared information from three different sensors

$$f^t_1 = W_f([h^t_{1_s}; h^t_{1_i}; h^t_{1_p}] + b_f) \tag{13}$$

where ; means concatenation, after we get the shared representation of three different sensors, we combine the representation with sensor-specific features extracted from first LSTM layer, so for the states of the second LSTM layer,

$$(h^t_{2_s}, c^t_{2_s}) = LSTM^2_s([h^t_{1_s}; f^t_1], h^{t-1}_{1_s}, c^{t-1}_{1_s}) \tag{14}$$

$$(h^t_{2_i}, c^t_{2_i}) = LSTM^2_i([h^t_{1_i}; f^t_1], h^{t-1}_{1_i}, c^{t-1}_{1_i}) \tag{15}$$

$$(h^t_{2_p}, c^t_{2_p}) = LSTM^2_p([h^t_{1_p}; f^t_1], h^{t-1}_{1_p}, c^{t-1}_{1_p}) \tag{16}$$

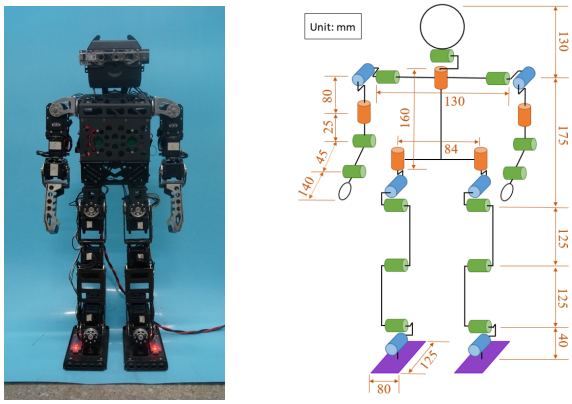
the features combined from three sensors can be represented by,

$$f^t_2 = W_f([h^t_{2_s}; h^t_{2_i}; h^t_{2_p}] + b_f) \tag{17}$$

finally, we get the prediction with a Softmax layer,

$$y_t = Softmax(f^t_2) \tag{18}$$

As for training, the loss of proposed architecture is be computed in the way described in formula (9). And Back Propagation Through Time (BPTT) is used [18] to update all the weights in this architecture.



(a) (b)

Fig. 5. The humanoid robot PKU-HR6.0. (a) The appearance (b) The schematic diagram of the joints' positions and orientations

TABLE I

ACTION LIST IN THE DATASET OF PKU-HR6.0

Action	Description	Duration(*s*)
Walk on marble	Robot walk on marble	12
Walk on carpet	Robot walk on carpet	12
Lay back	Robot lie on the ground with a back up gesture	5
Lay face	Robot lie on the ground with a face up gesture	5
Sit	Robot keep sitting posture	5
Crouch state	Robot keep crouching posture	5
Crouch up	The robot changes from kneeling to standing	4
Crouch down	The robot changes from standing to kneeling	4
Stand	Robot keep standing posture	5
Wave hand	Robot wave to people	5
Shake hand	Robot shake hand with people	5
Nod head	Robot nod its head	5
Brace floor	Robot support itself on the floor	5

IV. EXPERIMENTS AND RESULTS

A. PKU-HR6.0 Activity Dataset

Robots need to accomplish tasks in different situations.To let them know what they are doing, we need to equip robots with the ability of self-activity recognition. Due to this goal, we investigate the activity datasets about robots. In [19], the authors collected a dataset using a NAO robot. The dataset including 6 activities, such as left arm ring, right arm ring, etc. These activities are performed by two arms involving ten joints. In [15], the authors collected both arm joint positions in 3D space and first-view vision of Baxter Robot. 12 daily tasks like sweeping, pushing button were performed by the Baxter Robot. To our knowledge, there exists no dataset about humanoid robots for whole body activities, so in this research,we collect a dataset on PKU-HR6.0 robot.

The robot we use, as shown in Fig. 5, is named as PKU-HR6.0. PKU-HR6.0 is the 6th generation kid-size humanoid robot (3.93kg weight and 59.50cm heights) designed and developed by our lab. An Intel mini PC NUC 5I7RYH (2 cores, 4 threads) serves as the central controller. It works on Ubuntu 14.04 LTS and Robot Operating System (ROS) Indigo. It has 22 degrees of freedom (DOFs) driven by Dynamixel RX-28 and RX-64 servomotors, four servomotors in each arm, six servomotors in each leg and two servomotors in neck. Further the robot is equipped with a 3-axes IMU and one pressure transducer under each foot. With 22 servomotors we record angular position for each joint, such we have a signal with 22 dimensions. With IMU we record 3 axes linear acceleration, 3 axes angular velocity and 3 axes angular acceleration, such we have a signal with 9 dimensions. With pressure transducers we record 4 physical pressure per foot, such we have a signal with 8 dimensions. The sensor signal at each time-step has totally 39 dimensions.

This dataset involves 14 kinds of activities. We record the data with a sample rate 30HZ, the activity sequences last from 3 seconds to 15 seconds, so each sequence has 150 to 450 sensor signal samples. Each activity was performed 12 times. The activities we collects are listed in TABLE I.

B. Experiment Settings

We validate our proposed architecture on our PKU-HR6.0 dataset. To study the effectiveness of the proposed architecture, we conduct experiments under different network architectures derived from the proposed architecture:

- Deep LSTM only uses servomotor signal sequence as input
- Deep LSTM only uses inertial measurement unit(IMU) signal sequence as input
- Deep LSTM only uses pressure transducers signal sequence as input
- Deep LSTM uses concatenated servomotor, IMU and pressure transducer signal sequence as input
- Hierarchy LSTM feeds different sensor signal sequence to sensor relative sub-model and concatenates extracted features in intermediate layer
- Hierarchy LSTM feeds different sensor signal sequence to sensor relative sub-model and use a shared layer to fuse extracted features

We implement all the architectures mentioned above by using PyTorch which is a deep learning software package open sourced by Facebook [20]. The number of units in input layer is set according to the dimension of input sensor signal. The number of units in output layer is set to 14, for we have 14 kinds of activities to recognize. As for hidden layer, for different architectures we cannot use the same number if hidden units, so we keep the number of parameters for different architectures in the same level.

In order to get the best performance, we first normalized the data, such that input data have zero-mean and a standard deviation of 1. Then we use the method mentioned in [5] to smooth the samples in the sensor signal sequence, which has the form

$$f_i = (-3x_{i-2} + 12x_{i-1} + 17x_i + 12x_{i+1} - 3x_{i+2}) / 35 \quad (19)$$

where x_i denotes the data samples in the i frame, and f_i denotes the smoothed result. We use the smoothed signal sequence as input.

TABLE II
ACCURACY FOR SELF-ACTIVITY RECOGNITION

Method	Accuracy(%)
Deep LSTM with servomotors	48.98
Deep LSTM with IMU	70.15
Deep LSTM with pressure transducers	69.52
Deep LSTM with concatenated input	74.80
Hierarchy LSTM	83.50
Hierarchy LSTM with Shared Layer	90.50

In training phase, LSTM is trained using Stochastic Gradient Descent(SGD) with BPTT, the back propagation step length is set to 10. The learning rate is set to 0.00015 at beginning and is decreased during the training process. All weights of the network are initialized randomly from the interval [-0.1, 0.1]. To prevent the exploding gradient we clip the gradient when its norm is larger than 5.

C. Results

Recognition accuracy for different models are listed in TABLE II. When we use servomotor, IMU and pressure transducer signal individually the performances are really poor, they have accuracy of 48.98%, 70.15% and 69.52% respectively. With the concatenated sensor signal as input, we get a accuracy of 74.80% which is higher than the best accuracy achieved by individual sensors. This improvement shows, though the sensor intrinsics vary significantly, they have complementary information that can promote each other. After we use hierarchy structure shown in Fig. 3, which feed different sensor signals to different sub-models, the accuracy increases 9.31% compared to the accuracy using concatenated input. This increment shows that sub-models in hierarchy structure can help to learn sensor-specific features. Finally, after we add the shared layer previously proposed as illustrated in Fig. 4, we get the best performance, which means the shared layer previously proposed helps to fuse features extracted from different sub-models better than just concatenated them into LSTM layer.

V. CONCLUSION AND FUTURE WORK

To prevent robots from the damage of anomaly situations and ensure that the robot completes its task correctly, we present an architecture for multi-sensor fusion based robot self-activity recognition, which can help robots online perceiving their activities in real time. The architecture is composed by end-to-end manner which can let robots make their recognition in real time, a hierarchy structure which can help to learn sensor-specific features and a shared layer which can fuse sensor-specific features better. The experimental results on PKU-HR6.0 dataset demonstrate the effectiveness of the proposed architecture. With this architecture, robots can know their status every time such to be more intelligent.

In experiment we find similar activities are difficult to be distincted from each other. In the future, we would like to explore more activities to evaluate the proposed framework for multi-sensor based robot self-activity recognition, and may further explore the use of much more complicated sensors that can be used for robot self-activity recognition.In addition, we also plan to place the same sensor in the wearable device or exoskeleton, for we believe that the method applies not only to robots but also to humans.

ACKNOWLEDGMENT

The work is supported in part by the National Natural Science Foundation of China (No. U1713217, No. 11590773), the Key Program of National Social Science Foundation of China (No. 12 & ZD119).

REFERENCES

[1] Bulling A, Blanke U, Schiele B. A tutorial on human activity recognition using body-worn inertial sensors[J]. ACM Computing Surveys (CSUR), 2014, 46(3): 33.
[2] Yang J, Nguyen M N, San P P, et al. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition[C]//Ijcai. 2015, 15: 3995-4001.
[3] Ordóñez F J, Roggen D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition[J]. Sensors, 2016, 16(1): 115.
[4] Hammerla N Y, Halloran S, Ploetz T. Deep, convolutional, and recurrent models for human activity recognition using wearables[J]. arXiv preprint arXiv:1604.08880, 2016.
[5] Du Y, Wang W, Wang L. Hierarchical recurrent neural network for skeleton based action recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1110-1118.
[6] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult[J]. IEEE transactions on neural networks, 1994, 5(2): 157-166.
[7] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
[8] Christie J, Kottege N. Acoustics based terrain classification for legged robots[C]//Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016: 3596-3603.
[9] Otte S, Weiss C, Scherer T, et al. Recurrent Neural Networks for fast and robust vibration-based ground classification on mobile robots[C]//Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016: 5603-5608.
[10] Lenz I, Lee H, Saxena A. Deep learning for detecting robotic grasps[J]. The International Journal of Robotics Research, 2015, 34(4-5): 705-724.
[11] Elmenreich W. An introduction to sensor fusion[J]. Vienna University of Technology, Austria, 2002, 502.
[12] Song S, Lan C, Xing J, et al. An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data[C]//AAAI. 2017, 1(2): 4263-4270.
[13] Jain A, Singh A, Koppula H S, et al. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture[C]//Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016: 3118-3125.
[14] Huynh T, Schiele B. Analyzing features for activity recognition[C]//Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies. ACM, 2005: 159-163.
[15] Li Z, Au C W, Kakiuchi Y, et al. What am i doing? Robotic self-action recognition[C]//Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on. IEEE, 2016: 165-170.
[16] Lahat D, Adali T, Jutten C. Multimodal data fusion: an overview of methods, challenges, and prospects[J]. Proceedings of the IEEE, 2015, 103(9): 1449-1477.
[17] Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks[C]//International Conference on Machine Learning. 2013: 1310-1318.
[18] Kawakami K. Supervised Sequence Labelling with Recurrent Neural Networks[D]. PhD thesis. Ph. D. thesis, Technical University of Munich, 2008.
[19] Noda K, Arie H, Suga Y, et al. Multimodal integration learning of robot behavior using deep neural networks[J]. Robotics and Autonomous Systems, 2014, 62(6): 721-736.
[20] Paszke A, Gross S, Chintala S, et al. Automatic differentiation in pytorch[J]. 2017.