Multi-Channels LSTM Networks for Fence Activity Classification

Kelu Hu1,∗†, Student Member, Chunlei Zheng1, Wei He1, Xinghe Bao1††, and Yingguan Wang1, Nonmembers

SUMMARY We propose a novel neural networks model based on LSTM which is used to solve the task of classifying inertial sensor data attached to a fence with the goal of detecting security relevant incidents. To evaluate it we deployed an experimental fence surveillance system. By comparing experimental data of different approaches we find out that the neural network outperforms the baseline approach.

key words: LSTM, neural network, fence, inertial sensors

1. Introduction

A fence is built to protect important spots/buildings/sites (such as airports, warehouses, etc.). But a fence is easy to be crossed with brutal force or cheap tools. Hence a fence surveillance system becomes a necessity. Usually a fence surveillance system comprises cameras, optical fibers, and various sensors (infrared sensors, inertial sensors, etc.). Once these devices detected an object near the fence an alarm is triggered by an algorithm which usually compares signal with thresholds. In order to recognize different activities on the fence, sophisticated algorithms are needed. Conventional approaches do the task by extracting pre-designed features from sensor raw data, then use classifiers such as SVM, GMM et al. [1]–[3]. An example of pre-designed features are: amplitude, kurtosis, root mean square, variance, mean, standard deviation, etc.

In recent years, the neural network technology based on deep learning has made great progress, especially in the field of image processing [4]–[7], audio processing [8]. Inspired by these applications, we try to apply neural network technology to the task of fence activity classification. The signal generated by inertial sensors is a time series sequence which can be processed by Long Short Time Memory (LSTM) networks which is a RNN network that prevents the vanishing gradient problem. LSTM technology was first proposed by S. Hochreiter in [9] and is now widely used [10].

We present a novel neural network model to deal with the task of recognizing signal of inertial sensors on the fence. With this neural network, it is no longer necessary to extract the pre-defined features from raw data, instead the neural network makes it possible to learn features from raw signal. Each channel of the model takes a single dimension of multivariate time series as input of LSTM and learns features individually. Then the model merges the features of the three channels into the next LSTM for further processing, and finally a MLP neural network is used as the classifier.

To evaluate the performance of our model we carry out an experiment around the task which is to classify activities that may occur on a fence. The experimental results show that our model outperforms the baseline method.

The rest of the paper is organized as follows. In Sect. 2, we provide a review of classification researches concerning fence surveillance systems and/or inertial sensors. In Sect. 3 we present the architecture of our model and describe how to implement and train it. In Sect. 4 we conduct experiments on our fence and evaluate the performance. Finally we conclude our paper and point out directions for future research in Sect. 5.

2. Related Work

Inertial sensors have long been used in different application domains. In some cases they are deployed on fences to recognize security incidents, while in others, they are attached to human bodies to classify human activity. Different researches are based on inertial sensor data.

Both Wittenburg et al. [11] and Dziengel et al. [12] carry out their experiment on a fence. Wittenburg et al. [11] define a scalar quantity as the intensity of an event, then aggregate these intensity values, finally these aggregated values are compared to pre-designed patterns to be recognized as 6 events. Dziengel et al. [12] extract features from sensor raw data, then do assessment with an Euclidean-based prototype classifier. Yousefi et al. [13] define several features: signal variation, relative energy and use a Bayesian classifier. Kim et al. [14] also build a fence surveillance system. Instead of attaching inertial sensor node on the fence, they place their inertial sensors on the ground near the fence. They propose an algorithm of adaptive threshold which make decision every 0.5 seconds. No other methods are used to classify incidents.

Zheng et al. [6] propose a novel deep learning framework for multivariate time series classification, each channel of which takes a single dimension of multivariate time
series as input and learns features with Convolutional Neural Networks (CNN) individually, then combines the learnt features of each channel and feeds them into a MLP to perform classification finally. This model outperforms the competing baseline methods on two data sets.

Hannink et al. [15] utilize wearable inertial sensors to measure stride-related biomechanical parameters. They present 2 models which both extract features with deep convolutional neural networks and classify data with densely connected layers. The ensemble approach outperforms the combined model.

Neverova et al. [16] propose an optimized shift-invariant dense convolutional mechanism (DCWRNN) to extract features. GMM is used for the recognition task.

3. Multi-Channels LSTM Networks

3.1 Architecture

We propose a novel deep neural network framework for activity recognition based on inertial sensor data, which we refer to as Multi-Channels LSTM (MCLSTM). The architecture is based on several observations. First, for conventional approaches, pre-designed features are necessary. Since features can be learned for image classification [4], [5], and Zhang et al. [17] learn features from raw inertial data with CNN, we think it is also possible to extract features from inertial data with other neural networks. Second, data from an inertial sensor is a multivariate time series which are multiple 1D subsequences, we can simplify the architecture by separating multivariate time series based on axes. Finally, LSTM is suitable for modeling time series as they operate on input information as well as a trace of previously acquired information, which allows it to learn the temporal dynamics of sequential data.

We separate multivariate time series into univariate ones and perform feature learning on each univariate series individually. Then these features are concatenated and further processed by an LSTM. Finally a dense layer is used to do the classification. To be understood easily, we illustrate the architecture of MC-DCNN in Fig. 1. Next, we describe how each layer works.

Reshape layer. The input of each reshape block is an inertial data sequence of one axis, which is denoted as \( X = \{x_1, x_2, \ldots, x_n\}, \) \( 1 \leq i \leq n, \) \( x_i \in \mathbb{R}, \) where \( i \) denotes the timestep of the sequence. The layer reshapes it and outputs \( X', \) which contains \( m \) timesteps and \( k \) values per timestep:

\[
X' = \{x'_1, x'_2, \ldots, x'_m\}, \quad m = \frac{n}{k}, \quad 1 \leq j \leq m, \quad x'_j \in \mathbb{R}^k
\]

Since an input sequence is reshaped to \( m \) different vectors which may be largely different if the sequence starts at different time. Simply put, our model should be shift-invariant. In order to acheive shift-invarince, we employ sliding window as a data augment method which extracts more than one segments from the same activity with different time-shifts, Fig. 3.

LSTM layer 1. This layer extract features from the raw data reshaped by reshape layers. It outputs not only the last sequence but the full sequence. We adopt \( \tanh(\cdot) \) as activation function, and \( \text{sigmoid}(\cdot) \) as recurrent activation function. The design that this layer comprises three LSTM blocks instead of one is based on 3 reasons: (a) 3 axes of our sensors have different physical directions, the plane of axis X and axis Y corresponds to the fence plane, and direction of axis Z is perpendicular to the fence plane. Thus dynamics of time series of 3 axes are different. (b) Network parameters are reduced. Take the network configuration of Sect. 3.2 for example, each LSTM block has 365K parameters, layer 1 totally has 1.1M parameters. A single LSTM with the same size as layer 1 has 3.2M parameters which is roughly 3 times the parameters of our layer 1. (c) It is possible for this layer to be computed parallelly when it is implemented in a real-world application.

LSTM layer 2 concatenates every output sequence of LSTM layer 1. It only outputs the last sequence. We adopt \( \tanh(\cdot) \) as activation function, and \( \text{sigmoid}(\cdot) \) as recurrent activation function.

MLP layer 3 is a fully-connected MLP, e.g. a dense layer, which performs a non-linear transformation. We adopt \( \text{sigmoid}(\cdot) \) as activation function.

MLP layer 4 is the final layer which yields the classification outcome with a softmax regression output.

3.2 Model Implementation and Training

We implement our model in keras. The architecture can be depicted with following pseudocode. For this example, every sample of each axis comprise 1000 timesteps, reshape layer transforms them into 10 timesteps.

1: input_x = Input(shape=(1000,))
2: input_y = Input(shape=(1000,))
3: input_z = Input(shape=(1000,))
4:
5: input_x = Reshape((10,100))(input_x);
6: input_y = Reshape((10,100))(input_y);
7: input_z = Reshape((10,100))(input_z);
8:
9: layer1x = LSTM(256)(input_x);
10: layer1y = LSTM(256)(input_y);
11: layer1z = LSTM(256)(input_z);
12: x1 = Dense(128)(layer1x);
13: x2 = Dense(128)(layer1y);
14: x3 = Dense(128)(layer1z);
15:融合1 = concatenate([x1, x2, x3])
16: layer2 = Dense(128)(融合1)
17: output = Dense(1)(layer2)
18: output = Activation('softmax')(output)
19: model = Model(inputs=[input_x, input_y, input_z], outputs=output)
20: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
21: model.fit([input_x, input_y, input_z], labels, epochs=100, batch_size=32, validation_data=validation_data)
Fig. 2  Experiment setup. a) is our fence. b) is a sensor node on the fence. c) is a man climbing the fence. d) is our board.

Fig. 3  Sliding window example. Normalized ‘shake’ data of x-axis is segmented by sliding window. Four sliding windows W1, W2, W3, W4 are illustrated, 3 of which are labelled ‘shake’, the rest one is labelled ‘null’. By extracting more than one segments from a single activity, the shift-invariance of our model is improved.

11: \[ \text{layer1}z = \text{LSTM}(256)(\text{input}_z); \]
12: \[ \text{layer1} = \text{Concatenate()}(\text{[layer1}x, \text{layer1}y, \text{layer1}z]) \]
14: \[ \text{layer2} = \text{LSTM}(256)(\text{layer1}); \]
16: \[ \text{layer3} = \text{Dense}(128)(\text{layer2}); \]
17: \[ \text{layer4} = \text{Dense}(6,\text{activation} = \text{‘softmax’})(\text{layer3}); \]

For the sake of efficiency, when training and testing, data are segmented on mini-batches of a size of 100 data segments. The model is trained offline on a standard workstation with 2× Intel Xeon E5 CPU, 64GB DDR3 RAM.

4. Experiment

4.1 Experiment Setup

We deployed a chain link fence in our institute, Fig. 2 a), with 5 sensor nodes, Fig. 2 b). 5 subjects performed 5 activities: kick, climb, lean, shake, hammer. Here ‘hammer’ means a subject is hammering a fence post. Each subject waited 10 seconds before a repetition. Our sensor node operates using a STM32 microcontroller. A MEMS inertial sensor LIS344 is equipped. We set the sampling rate to 1 kHz. A program running on our workstation acquires all sampled sensor data. Our datasets are recorded continuously.

All data were preprocessed to fill in missing values using linear interpolation, and to do a per channel normalization to interval \([-1, 1]\) with a zero mean. A sliding window of fixed length is used to segment the data. The length of the window is 1000ms with a step size of 250ms, Fig. 3.

4.2 Experiment Results & Discussion

Normalized data of 6 classes of x-axis are showed in Fig. 4. The number of segments obtained with the sliding window method is detailed in Table 1. All these segments comprise our dataset, 85% of which is used for training, 15% for testing.

To demonstrate the relative merit of the proposed
method, we have compared the performance of our method against another 2 methods. First, we have considered a baseline approach for evaluation that is based on pre-designed features. 17 features are extracted from every data segment, 13 of which are extracted from raw signal: interquartile range, amplitude, kurtosis, root mean square, variance, mean, standard deviation, skewness, min, mean-cross, median, max, zero-cross, 4 of which are from first derivative: root mean square, variance, mean, standard deviation. So we can get 51 features from segments of 3 axes of one sliding window. These features are classified with a SVM. Second, we prepare a model referred as SCLSTM which is the same with MCLSTM except that the reshape layer contains one single reshape block and layer 1 of SCLSTM is one single LSTM network instead of three. The LSTM block of layer 1 of SCLSTM has the same size of the whole layer 1 of MCLSTM.

From the results in Table 2, we can see that MCLSTM outperforms the baseline approach. It improves the accuracy by 3.5%, F1 score by 0.026. For some classes (hammer, null), it is noticed that results of baseline approach are better than MCLSTM. As Fig. 4 shows that amplitudes of these activities are obviously different from others, we suppose that baseline method may benefit from adopting amplitude as one of pre-designed features. For other classes (kick, climb, lean, shake), MCLSTM leads to better accuracy. These findings support the hypothesis that the LSTM-based model takes advantage of learning the temporal feature activation dynamics. Results of MCLSTM and SCLSTM are roughly the same, but SCLSTM takes more time for training. It implies that separating input sequence by axis does not harm effectiveness and improves efficiency.

We also tried different reshape layer configuration. As we mentioned, input sequence of each channel $X$ is reshaped to $m$ timesteps which has $k$ features individually. We tried network with different $k$. Notice that input size of each block of layer 1 is also changed according to $k$. MCLSTM network is trained and tested with different $k$, Fig. 5. We can see that too small or too big values of $k$ produce worse F1 scores. When $k = 1$ each LSTM block of layer 1 only take one value per timestep while the block has a hidden layer with many nodes. When $k = 1000$, there is only one timestep for each LSTM block of layer 1. Both of these configuration just do not take advantage of the ability of LSTM.

An appropriate $k$ is around 100~250.

5. Conclusion

Inertial sensors are widely used in real world applications, such as mobile phones, wearable devices, and fence surveillance system, etc. Conventional approaches of classifying time series of inertial sensors usually depends on pre-designed features. We present a novel neural network model which can be used to classify raw data of inertial sensors of a fence. The model learns features from individual time series of each channel automatically, and combines features together with a dense layer to perform classification. With the help of this framework we need not design features manually. We evaluate our MCLSTM model on a data set of an experimental fence surveillance system. Results show that our model outperforms the baseline method.

Future work includes optimization of the model and implementation in real world system. We believe that this model could be improved, for example combine raw data and man-made features, [18]. In real world scenario, inertial sensors are used with Wireless Sensor Network (WSN), [11]–[13]. In a WSN environment, a wireless sensor node extracts features and transmits them to a base station which does the classification. We could try to improve our model and implement it in this scenario.

References

[1] L. Bao and S.S. Intille, “Activity recognition from user-annotated acceleration data,” Proc. Pervasive Computing, PERVASIVE 2004, Vienna, Austria, pp.1–17, 2004.
[2] J. Lan and Y. Shi, “Vehicle detection and recognition based on a MEMS magnetic sensor,” 4th IEEE International Conference on Nano/Micro Engineered and Molecular Systems, NEMS 2009, pp.404–408, Jan. 2009.
[3] K. Altun and B. Barshan, “Human activity recognition using inertial/magnetic sensor units,” International Conference on Human Behavior Understanding, pp.38–51, 2010.
[4] A. Krizhevsky, I. Sutskever, and G.E. Hinton, “ImageNet classification with deep convolutional neural networks,” NIPS, vol.1, p.4, 2012.
[5] Y. LeCun and Y. Bengio, “Convolutional networks for images,
speech, and time series,” The Handbook of Brain Theory and Neural Networks, MIT Press, pp.255–258, 1995.

[6] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” Computer Vision and Pattern Recognition, pp.3156–3164, 2015.

[7] M. Liwicki, A. Graves, H. Bunke, and J. Schmidhuber, “A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks,” 9th International Conference on Document Analysis and Recognition, 2007.

[8] A. Graves, A.-R. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” 2013 International Conference on Acoustics, Speech, and Signal Processing, pp.6645–6649, 2013.

[9] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol.9, no.8, pp.1735–1780, 1997.

[10] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM networks,” Proc. 2005 IEEE International Joint Conference on Neural Networks, IJCNN ’05, vol.4, pp.2047–2052, July 2005.

[11] G. Wittenburg, K. Terfloth, F.L. Villafuerte, T. Naumowicz, H. Ritter, and J. Schiller, “Fence monitoring: Experimental evaluation of a use case for wireless sensor networks,” Proc. 4th European Conference on Wireless Sensor Networks, pp.163–178, 2007.

[12] N. Dziengel, M. Ziegert, S. Adler, Z. Kasmi, S. Pfeiffer, and J. Schiller, “Energy-aware distributed fence surveillance for wireless sensor networks,” Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing, pp.353–358, 2011.

[13] A. Yousefi, A.A. Dibazar, and T.W. Berger, “Intelligent fence intrusion detection system: Detection of intentional fence breaching and recognition of fence climbing.” 2008 IEEE Conference on Technologies for Homeland Security, pp.620–625, 2008.

[14] Y. Kim, J. Kang, D. Kim, E. Kim, P.K. Chong, and S. Seo, “Design of a fence surveillance system based on wireless sensor networks,” Proc. 2nd International Conference on Autonomic Computing and Communication Systems, Turin, Italy, pp.4:1–4:7, 2008.

[15] J. Hannink, T. Kautz, C.F. Pasluosta, K.-G. Gallich, J. Klucken, and B.M. Eskofier, “Sensor-based gait parameter extraction with deep convolutional neural networks,” IEEE J. Biomed. Health Inform., vol.21, no.1, pp.85–93, 2017.

[16] N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, and G. Taylor, “Learning human identity from motion patterns,” IEEE Access, vol.4, pp.1810–1820, April 2016.

[17] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J.L. Zhao, “Time series classification using multi-channels deep convolutional neural networks,” Web-Age Information Management, F. Li, G. Li, S. Hwang, B. Yao, and Z. Zhang, eds. Lecture Notes in Computer Science, vol.8485, pp.298–310. Springer International Publishing, 2014.

[18] D. Ravi, C. Wong, B. Lo, and G.Z. Yang, “A deep learning approach to on-node sensor data analytics for mobile or wearable devices,” IEEE J. Biomed. Health Inform., vol.21, no.1, pp.56–64, 2017.

[19] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,” Proc. 23rd International Conference on Machine Learning, New York, NY, USA, pp.369–376, 2006.

[20] F.J. Ordóñez and D. Roggen, “Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition,” Sensors, vol.16, no.1, p.115, Jan. 2016.

[21] Z. Wang, W. Yan, and T. Oates, “Time series classification from scratch with deep neural networks: A strong baseline,” eprint arXiv:1611.06455, Nov. 2016.