Team Rit's cooking: Cooking Activity Recognition Challenge at ABC2020

No Author Given

Abstract This paper reports the Cooking Activity Recognition Challenge by team Rit's cooking held in the International Conference on Activity and Behavior Computing (ABC 2020). Our approach leverages convolution layer and LSTM to recognize macro activities (recipe), and micro activities (body motion).

1 Introduction

This paper reports the solution of our team "Rits's cooking" to Cooking Activity Recognition Challenge held at International Conference on Activity and Behavior Computing (ABC2020).

This paper is organized as follows: Section 2 introduces the challenge, section 3 explain our model, section 4 evaluates our model, and section 5 conclude this paper.

2 Challenge

This section introduces the challenge goal, dataset, and evaluation criteria.

2.1 Challenge Goal

The goal of the Cooking Activity Recognition Challenge is to recognize both the macro activity (recipe) and the micro activities took place in a 30-second window based on acceleration data and motion capture data.

The training dataset contains data about 3 subjects and contains all activity labels. The test dataset contains data about the other subject and is not labeled. Participants

must submit their predicted macro and micro activities on the test dataset using their models.

2.2 Dataset

2.2.1 Sensors and subjects

The data has been collected from four subjects who had attached two smartphones on the right arm and left hip, two smartwatches on both wrists, and one motion capture system with 29 markers. The subjects cooked three recipes (sandwich, fruit salad, cereal) five times each by following a script for each recipe, but acted as naturally as possible.

2.2.2 Data structure

Training data contains data from three subjects (subject 1, 2, 3) out of the four subjects and test data contains the data from the fourth subject (subject 4).

Each recording has been segmented into 30-second segments. Each segment was assigned a random identifier, so the order of the segments is unknown. Each sensor data segment is stored in a separate file with the segment-id used to identify related files. Segments of the four sensors at same time frame were assigned same identifier.

Groundtruth for all the segments are stored in one file. This file contains one row per file, and each row contains the file name, the macro activity and the micro activities all separated by commas; e.g., `[subject1_file_939,fruitsalad,Take,Peel,]`, which means that in segment 939 the subject 1 took something and peel something while making fruit salad. The micro activity is multi-label recognition task.

The macro activity is three classes: sandwich, fruitsalad, and cereal and the micro activity is ten classes: Cut, Peel, Open, Take, Put, Pour, Wash, Add, Mix, other.

2.2.3 Statistics

Table 1 shows the number of segments for each subject, the number of annotated classes of macro activity (one in this challenge), max, mean, and min number of annotated classes of micro activities, max, mean, and min length of the segments.

Subject	Body part	# of segments	# of macro	# of micro			Length		
				max	mean	min	max	mean	min
1	left hip	80	1	5	2.09	1	159	131.9	1
	left wrist						8191	2945	0
	right arm						1470	1309	8
	right wrist						8257	4484	0
2	left hip	105	1	6	2.26	1	505	428.3	10
	left wrist						5986	2171	0
	right arm						1500	1272	8
	right wrist						2992	2465	0
3	left hip	103	1	6	2.30	1	519	429.3	32
	left wrist						5529	774.6	0
	right arm						1594	1182	164
	right wrist						5938	3559	0
4	left hip	180	1	Unknown	Unknown	Unknown	534	406.7	46
	left wrist						7143	1126	0
	right arm						1479	1233	86
	right wrist						8761	2080	0

Table 1 Statistics of the dataset.

2.3 Evaluation criteria

Submissions will be evaluated by the average of the accuracy of macro activity classification (ma) and the average accuracy of micro-activity classification (mi). That is (ma+mi)/2.

The average accuracy of micro-activity classification is based on the multi-label accuracy formula. The accuracy of one sample is given by the following equation; the number of correct labels predicted (logical product of prediction set P and groundtruth set G) divided by the number of total true and predicted labels (logical sum of P and G).

$$accuracy = \frac{P \cap G}{P \cup G} \tag{1}$$

3 Method

This section explains our method that recognizes macro and micro activities from acceleration data for each segment. Note that our method does not use motion capture data.

3.1 Overview

Figure 1 shows the structure of our model. The detailed process in each layer is explained in the later subsections.

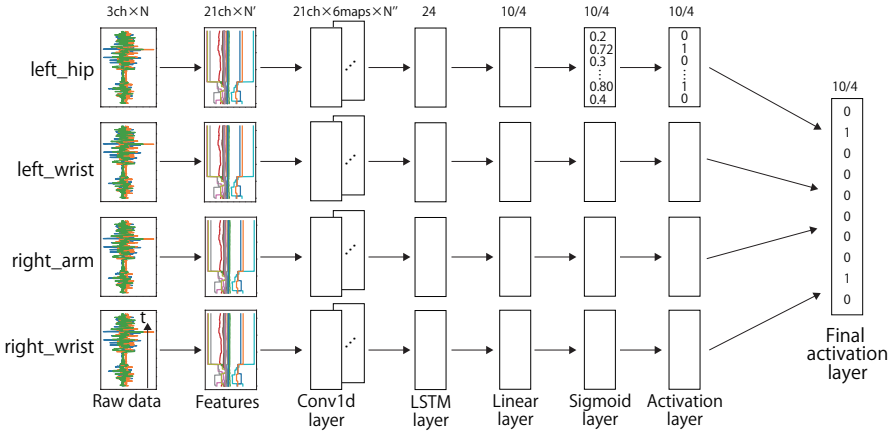

Fig. 1 Our mode. The first layer is raw data provided as it is. The second layer is hand crafted features consisting of 21 channels. Conv1d layer is one dimensional convolutional layer consisting of 6 maps for 21 channels, 126 maps in total. The 126-channel time-series data will be fed into the next LSTM layer consisting of 24 hidden layer. Then 24 dimensional tensor is shrunk to 10 dimensional tensor, then sigmoid function is applied, one-hot encoded with predetermined threshold. At last, four one-hot encoded vectors are merged and final multi-label prediction is obtained.

3.2 Preprocessing

Hand crafted feature values are extracted from the raw data $[[x_1, \ldots, x_N], [y_1, \ldots, y_N], [z_1, \ldots, z_N]]$, where x, y, z are raw data of x, y, z axis, and N is the number of samples in a 30-second segment. The features are mean, variance, max, min, root mean square (RMS), interquartile range (IQR), and zero crossing rate (ZCR) for x, y, and z axis, respectively. These features are calculated over a 50ms-window slid in steps of 3 seconds. From the preprocessing, 7 features $\times$ 3 axes = 21 dimensions feature time series are obtained for one sensor. The dataset includes the data obtained at four body parts, therefore this process is conducted for each sensor.

3.3 Model

The 21-dimensional feature time-series data is fed into our model consisting of 1d convolutional layer, LSTM layer, linear layer, and Sigmold layer. The process at each layer is as follows:

- **1d convolutional layer** has an input of sequence length $N' \times 21$ channels and an output of sequence length $N'' \times$ map size M. N' is the length of time-series after the feature extraction, which is smaller than the original raw data. N'' is the length of time-series after the convolution, which is $N' - K + 1$, where K is the kernel size. N' and N'' are variable length because the dataset has deficit values and sampling frequency is different to the sensors. Segments whose length is less than 10 is discarded and not fed in to the model. Kernel size K is set to 10. Map size M is the number of filters and set to $6 \times 21 = 126$. Here, the convolution is depthwise, i.e., the convolution is conducted for each channel and there are 6 filters for each channel.
- **LSTM layer** has an input of sequence length $\times$ 126 channels and an output of 24 dimensional tensors. The LSTM is many to one. The number of hidden layer is 24 and the last output of the LSTM layers are obtained. At this moment, the output is no longer time-series, but one tensor.
- **Linear layer** has an input of 24 dimensional flattened tensor and an output of 10/4 dimensional tensors. For macro (recipe) recognition, output is 4 dimensions, for micro activities, output is 10 dimensions.
- **Sigmoid layer** applies the sigmoid activation function to the 10/4 dimensional tensors, which represents the likelihood of the classes.
- **Activation layer** has an 10/4 dimensional tensor and an output of 10/4 dimensional one-hot vector. This layer activates the prediction classes whose values are more than the threshold Th. For micro activity recognition, the output one-hot vector has multiple 1 elements since the data is multi-labeled, e.g., [0,1,0,0,0,0,0,0,0,0] or [0,0,1,0,1,0,0,0,0,0]. The threshold Th is determined in the training phase by finding the best accuracy by changing the threshold from 0 to 1. For recipe recognition, the vectors in the sigmoid layer are used, not one-hot encoded.

3.4 Loss Function and Optimizer

The models for four sensors are trained separately. The model is trained on BCEWithLogistsLoss in PyTorch for micro activities and its weight was set to one for all classes. For macro activity, CrossEntropyLoss in PyTorch is used as loss function. Adam was used for optimizer for macro and micro activities.

3.5 Final Prediction Classes Activation

Through the process above, up to four predictions are obtained. At last, our method merges the predictions and output the final prediction. In detail, for micro activity recognition, the four one-hot vectors are summed up then the final prediction is done as follows. Note that segments whose length is less than 10 is not fed into the system and does not output prediction, therefore the cases when the number of predictions are one, two, three are also considered as shown in Table 2. From the table, the number of predictions is one or two, i.e., segments of three or two sensors are too short to be fed into the system, index which is greater than or equal to 1 is activated as final prediction. The number of prediction is three or four, index which is greater than or equal to 2 is activated as final prediction.

For example, suppose that micro activities [“Cut”, “Peel”, “Open”, “Take”, “Put”, “Pour”, “Wash”, “Add”, “Mix”, “other”] are one-hot encoded and predictions of the four sensors are [1,0,0,0,0,0,0,0,0,0] for left hip, [1,0,1,0,0,0,0,0,0,0] for left wrist, [0,0,1,0,0,0,0,0,0,0] for right arm, and [0,1,0,0,0,0,0,0,0,0] for right wrist. The summed up one-hot vectors is [2,1,2,0,0,0,0,0,0,0] and the number of predictions is four in this case. Indices whose values are greater than or equal to 2, i.e., index 0 and 2, are activated, and our method outputs Cut and Open as a prediction of micro activities for the segment. Index 1 (Peel) is not activated.

For macro activity recognition, the four vectors in sigmoid layer are summed up then the index showing the maximal value is activated since macro activity is single label. For example, suppose macro activities [“sandwich”, “fruitsalad”, “cereal”] and the four vectors in sigmoid layer are [0.1, 0.5, 0.9] for left hip, [0.1, 0.2, 0.6] for left wrist, [0.1, 0.6, 0.8] for right arm, and [0.3, 0.2, 0.7] for right wrist. The summed up vector is [0.6, 1.5, 3.0]. Index 2, which is showing the greatest value, is activated and our method outputs cereal as a prediction of macro activity for the segment. Threshold is not used for macro activity recognition since macro activity is single label.

Number of prediction	Activated classes
1	≥ 1
2	≥ 1
3	≥ 2
4	≥ 2

Table 2 Final activation algorithm for micro activities.

4 Evaluation

This section evaluates loss and accuracy in training phase and processing time in training and testing phases.

4.1 Environment

We implemented the program in Python 3.6.7, PyTorch 1.4.0, CUDA 10.0, and cuDNN 7402. The specification of the computer used for the evaluation is as follows: OS is Windows 10 Pro. CPU is Intel Core i7-8700K 3.7KHz. RAM is DDR4 64GB. GPU is NVIDIA GeForce RTX 2080Ti GDDR6 11GB. All the data were stored on local HDD. In the training phase, all data of subject 1, 2, and 3 (288 segments) were used for training in one epoch, which was iterated 1,000 epochs.

4.2 Result

Figure 2 shows accuracy of micro activities for each body part in 1,000 epochs. The accuracy was calculated using the one-hot vectors in the activation layer in Fig.1. Figure 3 shows loss of micro activities for each body part in 1,000 epochs. The loss was calculated using the vectors in the sigmoid layer in Fig.1. Please note that our model was trained separately for the body parts and the model at 1,000 epoch was used for testing the data of subject 4.

Figure 4 shows accuracy of macro activities for each body part in 1,000 epochs. The accuracy was calculated using the one-hot vectors in the activation layer in Fig.1. Figure 5 shows loss of macro activities for each body part in 1,000 epochs. The loss was calculated using the vectors in the sigmoid layer in Fig.1.

Table 3 shows memory usage on CPU and GPU, and processing time taken in training phase and testing phase.

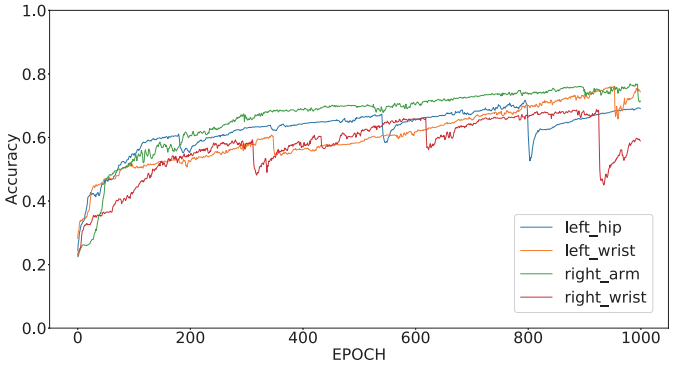

Fig. 2 Accuracy of micro activities for training data (subject 1, 2, 3).

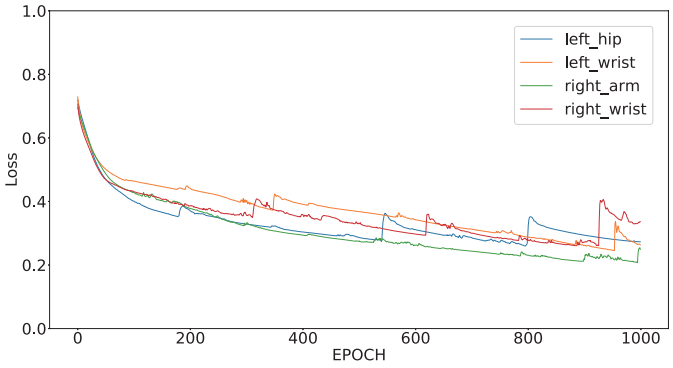

Fig. 3 Loss of micro activities for training data (subject 1, 2, 3).

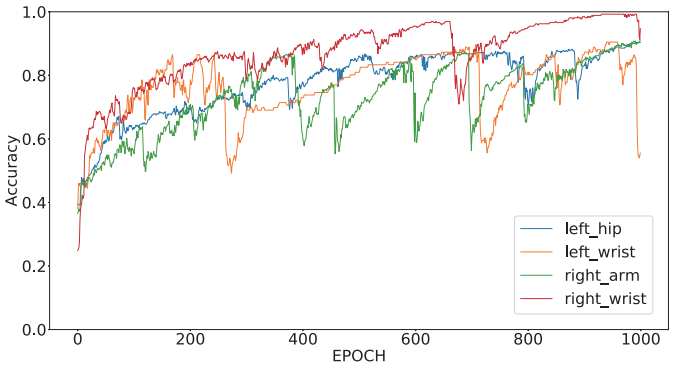

Fig. 4 Accuracy of macro activities for training data (subject 1, 2, 3).

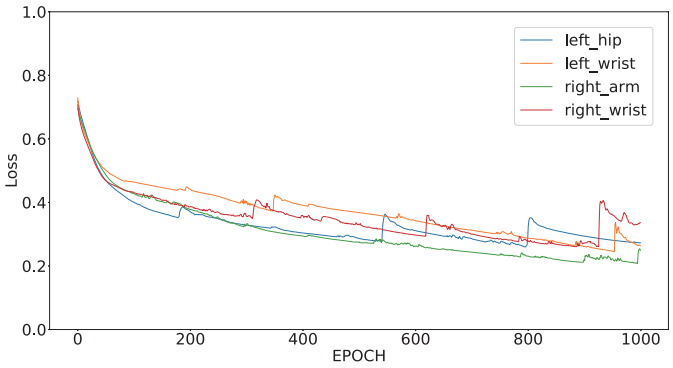

Fig. 5 Loss of macro activities for training data (subject 1, 2, 3).

5 Conclusion

This paper reported the solution of our team “Rits’s cooking” to Cooking Activity Recognition Challenge held at International Conference on Activity and Behavior Computing (ABC2020).

Resource	Macro	Micro
CPU memory	2391MB	2391MB
GPU memory	1.6GB	1.6GB
Training time (1,000 epoch)	21.554s	28.891s
Testing time (1,000 epoch)	58.042s	59.299s

Table 3 CPU and GPU memory usage and time taken in training and testing. These figures are when data of four body parts are processed at once.

Appendix

Used sensor modalities

Four acceleration sensors at lest hip, left wrist, right arm, and right wrist from three subjects were used. Mocap data was NOT used.

Features used

Seven kinds of features were used: Mean, variance, max, min, root mean square (RMS), interquartile range (IQR), and zero crossing rate (ZCR). These features are extracted for x, y, and z axis, respectively.

Programming language and libraries used

Python 3.6.7 was used. For network implementation, PyTorch 1.4.0 was used.

Window size and post processing

Window size is 500 ms and step size is one sample.

Training and testing time

Training time (1,000 epoch) was 21.554s for macro activity and 28.891s for micro activity. Testing time (1,000 epoch) was 58.042s for macro activity and 59.299s for micro activity.

Machine specification

OS: Windows 10 Pro. CPU: Intel Core i7-8700K 3.7KHz. RAM: DDR4 64GB. GPU: NVIDIA GeForce RTX 2080Ti GDDR6 11GB.