Multi-Sampling Classifiers for the Cooking Activity Recognition Challenge

Ninnart Fuengfusin and Hakaru Tamukoh

Abstract We propose multi-sampling classifiers (MSC), a collection of multi-class and binary classifiers. To address the cooking activity recognition challenge (CARC), CARC consists of macro and micro labels. To deal with these labels, MSC uses a multi-class classifier to recognize the macro labels and 10 binary classifiers to examine whether each micro label exists. To shield the MSC model from sampling noise, we generate three distinct re-sampling rates of the temporal sensor data. All predictions of the three data sampling rates are gathered together using a soft-majority voting ensemble.

1 Introduction

The increasing demands of smart mobile phones, smart watches, and internet-of-things (IoT) [23] show a growing trend with respect to the availability of cheap temporal sensor data. This huge amount of sensor data can provide patterns that can be used to gain insight into user activities. Such insights can permit a device to have a broader prior knowledge of a user and be able to smartly interact with the user's preferences.

Activity recognition (AR) aims to map time series sensor signals into certain predefined activities. Machine learning (ML) algorithms are one of the methods used to achieve AR. However, ML might require domain specific feature extraction, which

Ninnart Fuengfusin
Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, 2-4 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka, 808-0196, Japan, e-mail: fuengfusin.ninnart553@mail.kyutech.jp

Hakaru Tamukoh
Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, 2-4 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka, 808-0196, Japan, e-mail: tamukoh@brain.kyutech.jp

consumes a large amount of user time. Recently, the deep-learning [18] (DL) model, or deep neural networks, has become one of dominant algorithms in ML, particularly in the field of image recognition [24], object detection [25] and image segmentation [12]. DL gains an advantage from its ability to automatically re-combine given features into new representations. However, the tradeoff is that DL requires a large amount of feature-rich data to operate efficiently with its ability to scale with the model [15, 13, 14].

Given the merits and demerits of each approach, a benchmark is necessary to measure the relative abilities of different approaches. The cooking activity recognition challenge (CARC) [3] corresponds to time series sensor data recorded from when users performed cooking activities [16, 17]. The data were recorded from smartphones, smartwatches, and motion capture markers. Each signal is segmented into 30 s intervals. The challenge is to recognize each signal segment given the provided macro activity and multi-micro activity labels. The macro label indicates the coarse activity, for example, *fruitsalad*, *sandwich*, or *cereal*. The micro label is a fine-grained activity that might look at a section of time within a segment, for example, *Put*, *Wash*, and *other*.

The CARC dataset consists of 288 training and 180 testing segments. The DL approach is not suitable for the CARC task due to the amount of data, which is not large enough to surpass the performance of conventional ML. Therefore, our considered classifiers use conventional ML approaches. The macro label can be solved directly using a multi-class classifier algorithm. However, the micro labels are different for each segment of data and might contain multi-labels; further, the number of labels is not fixed.

To solve this challenge, we propose multi-sampling classifiers (MSC). MSC solves the micro-label problem by treating each micro label as a binary classification problem. In other words, each binary-class classifier corresponds to a micro label. This classifier considers whether the micro label exists or not. Therefore, our backbone MSC model consists of a multi-class classifier for the macro labels and 10 binary classifiers for the 10 unique micro labels.

We pre-process the given data using three different sampling rates. The hypothesis of our pre-processing is that the varied sampling rates may allow our classifiers to detect certain patterns that might not be recognized at other sampling rates. With three different sampling rates, the algorithm can view three perceptions of the data trends from simple to complex views. All of the results of the classification of each sampling are used via a soft-voting ensemble.

Our main contributions are as follows.

- To identify the macro and micro labels, we propose MSC, that is a combination of a multi-class classifier and multiple binary classifiers.
- We propose a training method with pre-processing at three different rates. We show that this might improve the performance of a multi-sampling classifier.

2 Related Work

In this section, we describe a related work that has a similar research problem to MSC.

2.1 Learning subclass representations for visually-varied image classification.

Learning subclass representations for visually varied image classification (LSR) [19] was proposed for the 2013 Yahoo! large-scale flick-tag classification grand challenge [1]. This challenge consisted of 10 classes that were defined as top-class labels; each class consisted of 150,000 training and 50,000 test images. Each of the 10 classes was a top-rank tag generated by the users. The main challenge was the over-generalized top-class labels, for example, *nature*, which can be represented in various contexts with images such as *flower*, *bird*, and *forest*. This results in a model learning little from a given task.

This diverse definition of labels causes the model to be less robust. To address this problem, LSR can be trained with other provided tags that are not top-class labels; such labels are defined as sub-classes. LSR statistically finds the co-occurrence between subclass and top-class labels; if the co-occurrence ratio of the subclass to the sum of all the top-classes is higher than a pre-defined threshold, then the sub-classes produced by support vector machine (SVM) [8] will be used as the features and put into SVM to classify the top class. LSR concludes that, with the subclass approach, such a model out performs a model directly trained with top-level labels.

In our opinion, applying the LSR approach requires a statically significant amount of data, which the Yahoo! challenge provides. However, in CARC, there are only 288 training segments, with 3 macro labels and 10 different micro labels; therefore, the correlation between each macro label and the micro labels might not be sufficient.

3 Multi-Sampling Classifiers

In this section, we describe two main MSC principles: multi-classifiers and multi-sampling.

3.1 Multi-Classifiers

MSC is proposed to solve CARC; therefore, MSC directly focuses on solving for the macro and micro labels provided by CARC. The macro labels consist of *cereal*, *sandwich*, and *fruitsalad*. With three classes, the macro label can be solved for using

a multi-class classifier. The major challenge in CARC is the micro labels. MSC converts the micro label problem into multi-binary classification problems. This method is possible for CARC because there are no duplicated micro labels in the segments except for in one of the training segments. MSC uses the multi-binary classifiers to recognize the micro labels.

CARC consists of 10 unique micro labels: *Pour*, *Peel*, *Put*, *Take*, *Cut*, *Wash*, *Add*, *Mix*, *Open* and *other*. MSC matches each binary classifier to each unique micro label. Each binary classifier predicts whether its assigned micro label exists or not. Our backbone model is shown in Fig. 1. In this figure, the backbone model is called as a multi-classifier. This model contains a multi-class classifier to deal with the macro labels and 10 binary classifiers to deal with the micro labels.

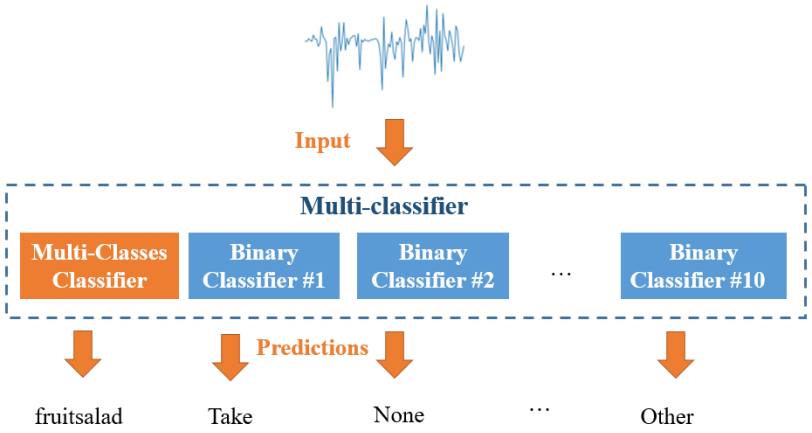

Fig. 1 Multi-classifier receives an input signal and produce the macro and micro label predictions.

3.2 Multi-Sampling

Multi-sampling is used to generate three different perspectives of the signals. Sampling with short intervals allows the ML algorithm to focus on every detail of the time series data. Conversely, sampling with long intervals allows the ML algorithm to view the trend in the time series data easier. Nonetheless, because since each sensor dataset contains a different rate, a re-sampling of all the temporal data is necessary to extend or reduce the length of all the signals to a certain length.

All of the data were re-sampled using an approach from [2]. The time series data were first down-sampled to 1 ms and then, up-sampled to the preferred periods. We selected three sampling rates, 100 ms, 300 ms, and 1000 ms. These sampling rates differed from each other by roughly three times. With a maximum recorded time length of 30 s, this resulted in time series data with lengths of 300, 100, and 30, respectively. We selected these ranges from the sampling to cause the shapes of

the waveforms to differ. The sampling length should be lower than 1000 because MSC requires an extensive training time for numbered models. With three different sampling rates, the three predictions can be realized. To reduce these predictions into one, we applied the same soft-weight voting. The multi-sampling is summarized in Fig. 2

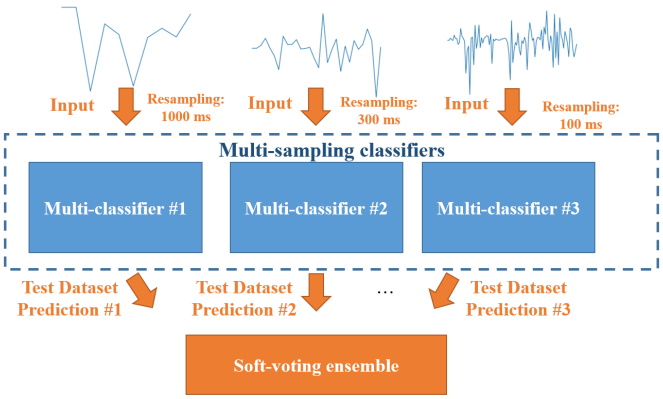

Fig. 2 The three different samplings are input into each multi-classifier. All of the test dataset predictions from each multi-classifier are voted on using soft-voting ensemble. The collection of multi-classifiers is called as the multi-sampling classifiers.

4 Experimental Results and Discussion

We described the overall setting of the experiment. We used all of the provided sensor data without excluding any data. CARC contains a number of missing feature segments, for example, the x-axis of the left wrist smartwatch data might be missing for the whole segment. We could not impute with the segment mean when all data in some of the feature segments is missing. Therefore, we imputed with the signal mean instead. All segments from the same feature was averaged to find the signal means. All of the missing data were replaced with the signal mean from the same feature. Each signal was scaled to the unit mean and variance. We randomly selected 33% of the training segments and split them into the validation dataset. Each given segment contains the non-uniform timestamps. The conventional resampling with an unevenly spaced signal might cause a shift between the original signal to the resampling signal. Therefore, we applied with two steps resampling from [2]. All sensor data was up-sampled to 1 ms and down-sampled to 100 ms, 300 ms, and 1000 ms.

To evaluate the models, the CARC metric uses the average accuracy, which is defined in Eq. 1, where a is the average accuracy, m_a is the macro accuracy and m_i is the micro accuracy. The micro accuracy metric or Hamming score [9] is defined

as in Eq. 2, where n_t is the number of correct predictions, and n_c is the cardinality of the union between labels and predictions.

$$a = \frac{m_a + m_i}{2} \tag{1}$$

$$m_i = \frac{n_t}{n_c} \tag{2}$$

MSC has an advantage of allowing the conventional binary classifiers to solve the multi-label classification problem. Therefore, we only needed to identify the suitable classifier instead of inventing a novel model. To identify the best classifier to apply in MSC, we tried several Python time series ML packages: `tslearn` [26],`sktime` [20], `seglearn` [4] and `tsfresh` [7]. From these packages, we did not select `tsfresh` because `tsfresh` consumed a considerable amount of time to perform the feature engineering with this dataset. `seglearn` also was not chosen because `seglearn` did not provide with a function that converts the element-wise to segment-wise preditions. Therefore, with the process of elimination, we selected `tslearn` and `sktime`.

Time series SVM (`TimeSeriesSVC`) and K-nearest neighbor (`KNeighbors-TimeSeriesClassifier`) are included in the `tslearn` modules. Time series random forest (`TimeSeriesForestClassifier`) is included in `sktime`. We evaluated all of these algorithms using the default settings with the macro label accuracy as our metric. Each input signal was sampled at 100 ms. The validation scores are listed in Table 1. In Table 1, the macro accuracy of the soft-voting ensemble of the multi-sampling is also displayed as well as the "Soft voting" section, which indicates the accuracy after each classifier was soft voted with three different samplings. Random forest from the `sktime` package was selected to be the main classifier in MSC because it out-performed the other models. Soft voting of the different samplings showed an improvement in the accuracy for `TimeSeriesSVC` and `TimeSeriesForestClassifier`. Nevertheless, the `KNeighborsTimeSeriesClassifier` had the worse macro accuracy.

Table 1 Macro label accuracy at the baseline setting for the time series ML models.

Algorithm	Package	Macro accuracy	Soft voting
`KNeighborsTimeSeriesClassifier`	`tslearn`	0.7396	0.7292
`TimeSeriesSVC`	`tslearn`	0.6458	0.6563
`TimeSeriesForestClassifier`	`sktime`	0.8438	0.8542

We found out that the macro and micro label require a distinct set of hyper parameters due to the different complexity of the tasks. In general, the high complexity model performed well with the macro label. However, the high complexity model over-fitted the micro label. Therefore, we selected `TimeSeriesForestClassifier` with 1000 estimators as the multi-class classifier and `TimeSeriesForestClassifier` with

50 estimators as the binary classifier. The micro, macro, and average accuracy of these models are shown in Table 2. Our model displayed its best average accuracy at the 300 ms sampling rate. The last row of Table 2 indicates the accuracies after the soft voting. These accuracies are shown the improvement in accuracy for 100 ms and 300 ms sampling rates. However, the soft voting accuracy cannot exceed the accuracy from 300 ms sampling rate. We expected that the soft voting accuracy might be better with a greater number of samplings. However, by increasing a sampling, the extra 11 classifiers are needed to be trained. This affects the training time of MSC to increase exponentially. Therefore, we push this hypothesis to further investigate in future work.

Table 2 Validation results with the different sampling rates and soft voting.

Sampling rate	Macro accuracy	Micro accuracy	Average accuracy
1000 ms	0.875	0.7927	0.8339
300 ms	0.8854	0.8255	0.8555
100 ms	0.8542	0.816	0.8351
Soft voting	0.875	0.8191	0.847

Within the test dataset, we found that MSC did not produce micro label predictions for 26 test segments. On the other hand, this problem did not occur with the validation dataset. We investigated this problem by looking for the difference between the training, test, and validation dataset. The major difference is the training and validation dataset were recorded from human subjects 1, 2, and 3. However, the test dataset was recorded from subject 4. We hypothesized that the distribution of validation and test dataset might vary, and our MSC was not robust enough to handle this problem. Another hypothesis is the converting process from a micro label to binary labels might cause high sparsity within binary labels. The binary classifier might assume the output is most likely zero and does not produce the micro label. We considered exploring this sparsity hypothesis in future work. If this hypothesis is correct, we might solve this problem by applying techniques for dealing with an imbalance dataset, such as SMOTE [5], Borderline-SMOTE [10], and ADASYN [11].

To fix this missing label problem with the available method, no output should be changed to the most frequent micro label in the training data instead, in this case, the label was *Take*, as shown in Fig. 3.

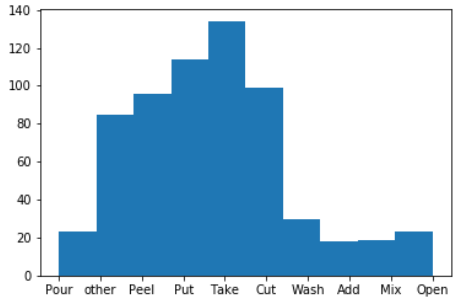

Fig. 3 Distribution of the training micro labels after partitioning the sequence of micro labels.

5 Conclusion

We proposed MSC for CARC. MSC provides a simple alternative to solve a multi-label classification problem by converting the problem into a multi-binary classification problem. This allows conventional binary classifiers to be deployed to solve the multi-label classification problem. On the other hand, this method requires a number of binary classifiers. This affects the increase in computation time. To deal with CARC, our MSC model consisted of a multi-class classifier and 10 binary classifiers to deal with the macro and micro labels. To allow the model to overcome noise from re-sampling, our models were trained with three different sampling rates, and all predictions of the test data from the different sampling rates were soft voted.

Acknowledgements This research was supported by JSPS KAKENHI Grant Numbers 17K20010.

Appendix

As a requirement of CARC, our experimental settings are given in Table 3.

Table 3 Description of our experimental settings.

	Description
Sensor modalities	We used all provided sensors.
Features used	We applied all provided features.
Programming language and libraries	Python with `keras` [6], `sklearn` [22], `sktime`, `pandas` [21], `tslearn`, and `numpy` [27]
Window size and post processing	30,000 ms
Training and testing time	1h 49min 12s
Specifications	CPU Intel Xeon E5-1620v3, RAM DDR4 64 GB, GPU GeForce GTX 1080

References

1. Yahoo! large-scale flickr-tag image classification grand challenge. http://www.sigmm.org/archive/MM/mm13/acmmm13.org/submissions/call-for-multimedia-grand-challenge-solutions/yahoo-large-scale-flickr-tag-image-classification-challenge/index.html (2013). Cited 18 May 2020
2. Rafael schultze-kraft - building smart iot applications with python and spark. https://www.youtube.com/watch?v=XkBbAymUDEo (2017). Cited 18 May 2020
3. Alia, S.S., Lago, P., Takeda, S., Adachi, K., Benaissa, B., Ahad, M.A.R., Inoue, S.: Summary of the cooking activity recognition challenge. In: Human Activity Recognition Challenge, Smart Innovation, Systems and Technologies. Springer (2020)
4. Burns, D.M., Whyne, C.M.: Seglearn: A python package for learning sequences and time series. The Journal of Machine Learning Research **19**(1), 3238–3244 (2018)
5. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)
6. Chollet, F., et al.: Keras. https://github.com/fchollet/keras (2015)
7. Christ, M., Braun, N., Neuffer, J., Kempa-Liehr, A.W.: Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). Neurocomputing **307**, 72–77 (2018)
8. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)
9. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Pacific-Asia conference on knowledge discovery and data mining, pp. 22–30. Springer (2004)
10. Han, H., Wang, W.Y., Mao, B.H.: Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: International conference on intelligent computing, pp. 878–887. Springer (2005)
11. He, H., Bai, Y., Garcia, E.A., Li, S.: Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), pp. 1322–1328. IEEE (2008)
12. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp. 2961–2969 (2017)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
14. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European conference on computer vision, pp. 646–661. Springer (2016)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
16. Lago, P., Takeda, S., Adachi, K., Alia, S.S., Matsuki, M., Benai, B., Inoue, S., Charpillet, F.: Cooking activity dataset with macro and micro activities (2020). DOI 10.21227/hyzg-9m49. URL http://dx.doi.org/10.21227/hyzg-9m49

17. Lago, P., Takeda, S., Alia, S.S., Adachi, K., Bennai, B., Charpillet, F., Inoue, S.: A dataset for complex activity recognition withmicro and macro activities in a cooking scenario. arXiv preprint arXiv:2006.10681 (2020)
18. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
19. Li, X., Xu, P., Shi, Y., Larson, M., Hanjalic, A.: Learning subclass representations for visually-varied image classification. arXiv preprint arXiv:1601.02913 (2016)
20. Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., Király, F.J.: sktime: A unified interface for machine learning with time series. arXiv preprint arXiv:1909.07872 (2019)
21. McKinney, W., et al.: pandas: a foundational python library for data analysis and statistics. Python for High Performance and Scientific Computing **14**(9) (2011)
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
23. Perera, C., Liu, C.H., Jayawardena, S., Chen, M.: A survey on internet of things from industrial market perspective. IEEE Access **2**, 1660–1679 (2014)
24. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)
25. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. arXiv preprint arXiv:1911.09070 (2019)
26. Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., Woods, E.: tslearn: A machine learning toolkit dedicated to time-series data (2017). https://github.com/tslearn-team/tslearn
27. Walt, S.v.d., Colbert, S.C., Varoquaux, G.: The numpy array: a structure for efficient numerical computation. Computing in Science & Engineering **13**(2), 22–30 (2011)