Deep Prototypical Networks for Imbalanced Time Series Classification under Data Scarcity

Chao Huang
University of Notre Dame
chuang7@nd.edu

Xian Wu
University of Notre Dame
xwu9@nd.edu

Xuchao Zhang
Virginia Tech
xuczhang@vt.edu

Suwen Lin
University of Notre Dame
slin4@nd.edu

Nitesh V. Chawla
University of Notre Dame
nchawla@nd.edu

ABSTRACT

With the increase of temporal data availability, time series classification has drawn a lot of attention in the literature because of its wide spectrum of applications in diverse domains (*e.g.*, healthcare, bioinformatics and finance), ranging from human activity recognition to financial pattern identification. While significant progress has been made to solve time series classification problem, the success of such methods relies on data sufficiency, and may not well capture the quality embeddings when training triple instances are scarce and highly imbalance across classes. To address these challenges, we propose a prototype embedding framework–Deep Prototypical Networks (DPN), which leverages a main embedding space to capture the discrepancies of difference time series classes for alleviating data scarcity. In addition, we further augment DPN framework with a relationship-dependent masking module to automatically fuse relevant information with a distance metric learning process, which addresses the data imbalance issue and performs robust time series classification. Experimental results show significant and consistent improvements compared to state-of-the-art techniques.

CCS CONCEPTS

• Mathematics of computing → Time series analysis; **• Computing methodologies → Learning latent representations**.

KEYWORDS

Time Series Classification; Deep Neural Network; Data Scarcity

ACM Reference Format:
Chao Huang, Xian Wu, Xuchao Zhang, Suwen Lin, and Nitesh V. Chawla. 2019. Deep Prototypical Networks for Imbalanced Time Series Classification under Data Scarcity. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM'19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3358162

CIKM'19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358162

1 INTRODUCTION

With the increase of temporal data availability, time series classification has drawn a lot of attention in the literature because of its wide spectrum of applications in diverse domains (*e.g.*, healthcare [17], bioinformatics [4] and finance [11]). In general, the goal of time series classification is to assign time-ordered sequences into specific categories based on various time series patterns [10]. To solve this problem, a significant line of previous research has focused on the exploration of various features and ensemble techniques. In particular, feature-based methods aims to identify a set of features that could represent the global/local time series patterns and then given to the classifiers [3, 12]. Given the extracted various features, ensemble based approaches incorporate different features into the integrated ensemble paradigms [1].

These above feature-based approaches require a set of informative and discriminating domain-specific features, which involves hand-engineering effort based on expert knowledge. To mitigate this limitation, several convolutional neural network based methods have been proposed and show promising performance for various applications [5, 9, 15]. Their key idea is to unify feature learning and classification parts with different levels of abstraction, based on multiple layers of processing units (*e.g.*, convolution and pooling). The success of such learning approaches largely relies on a sufficient amount of training instances, such that the learned embeddings could well preserve latent structures of time series data. However, many practical scenarios involve a limited amount of labeled data. For example, in enterprise or healthcare scenarios, labelling samples is often very expensive or even impossible [7, 14], yet it is important to build effective time series classification models given only a handful of labeled examples per class.

There exist two key challenges in order to address the problem of time series classification under data scarcity. *First*, In time series classification, performing pattern learning under data scarcity, will have the undesirable tendency to extremely overfit the highly imbalanced time series data [17]. Hence, the model requires a sophisticated strategy to transfer knowledge across different time series classes and guide the representation learning process, in order to alleviate data scarcity issue. *Second*, the temporal patterns of time series data often involve underlying factors which are dynamically evolving over time [16]. Despite the effectiveness of applying neural network techniques to capture the non-linearity in dynamic time series, their classification performance can be hindered by the data scarcity issue in the imbalanced time series data. It is also challenging to perform imbalance class learning with temporal dynamics from a limited amount of labeled time series data.

Motivated by the aforementioned challenges, this work proposes a general and flexible learning framework–Deep Prototypical Networks (DPN)–to exploit deep neural networks for end-to-end imbalanced time series classification under data scarcity. In particular, we develop a prototypical embedding module which maps time series input into an latent embedding space to serve as the prototype representation for each class, with a meta-learning framework. This prototype embedding space bridges the labeled set of examples (the support set) with unlabeled points (the query set), such that the knowledge from other classes can be leveraged to guide the embedding process across time series classes for alleviating data scarcity. In the mean time, we develop a relationship-dependent masking module to learn the distance metric between embeddings from deep neural network models, and automatically assign different weights to their different attributes with a high-level of non-linearities. Such augmented robust embedding space enables us to reduce noise or redundant information in the given imbalanced time series classes and perform time series classification.

The main contributions can be summarized as follows:

- We study the problem of classifying time series data from a limited amount of labeled training samples under data imbalance.
- We design a prototype embedding framework–Deep Prototypical Networks (DPN), which uses a main embedding space to capture the discrepancies of difference time series classes under the scenario of data scarcity.
- We further endow DPN with a relationship-dependent masking module to fuse relevant information into the automatically distance metric learning process, which alleviate data imbalance issue and perform robust time series classification.
- Through experiments performed on 49 time series classification benchmark datasets, we show that DPN consistently surpasses several state-of-the-art baselines.

2 PRELIMINARIES

In our work, we consider a time series classification scenario with a total number of C classes which is indexed by c.

Definition 1. ***Time Series.*** *A time series is a sequential set of measurements collected at equally spaced intervals over a period of time, i.e., a vector* $X = [x_1, \ldots, x_t, \ldots, x_T]$ $(t \in [1, \ldots, T])$ *in chronological order, where* x_t *is the value (i.e., continuous or discrete value) collected from the* t*-th time interval.*

Time Series Classification. The objective of time series classification task is to predict a class label c for a given fixed-length time series X whose label is unknown. Specifically, we aim to learn a classification function from a set of labeled time series (the training instances), to take an unlabeled time series (the testing instances) as input and outputs a label c.

3 METHODOLOGY

3.1 Prototypical Embedding Framework

We propose to leverage the knowledge across different time series classes to address the challenge of data scarcity. Inspired by [13], we first develop a prototype embedding framework which uncovers temporal dynamics under data scarcity. With a limited number of training time series instances, our designed prototype embedding framework embeds time series data into low-dimensional representations by modeling the time-evolving temporal patterns manifested in the time series topological structure. We define the support set S ($s_i \in S$ indexed by i) to represent the set of labeled instances and the query set Q ($q_j \in Q$ indexed by j) to indicate the set of unlabeled time series for making predictions. In general, our module leverages the support set S to extract a prototype vector representation from each time series class, and performs classification on the inputs from the query set based on their distance to the generated prototype representation of each class.

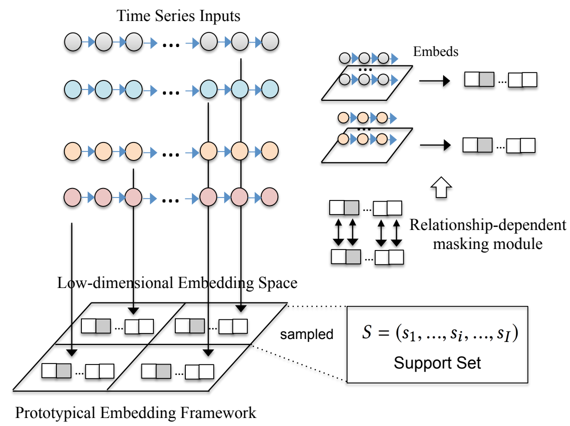


Figure 1: The Deep Prototypical Networks Framework.

Our model takes different time series as input and learns an embedding function to make similar examples (from the same class) close to each other and makes dissimilar examples far apart from each other. Specifically, we first perform a non-linear mapping of the time series inputs X into low-dimensional latent spaces via the embedding function $h(X)$. Then, we perform classification on a given query point in Q by discovering its nearest class prototype representation. Hence, our framework enables us to effectively transfer knowledge across different classes and guide the representation learning process. To calculate the prototype p_c of each class c, we take p_c to the mean of its support embedding space as:

$$S_c = \frac{\sum_i h(s_i) z_{i,c}}{\sum_i z_{i,c}} \tag{1}$$

where $z_{i,c} = \mathbb{I}[y_i = c]$. These prototypes of classes are considered as classifier metric to classify any query sample q_j from Q. Our model assigns a probability over any class c from class set C based on the distance which is formally defined as follows:

$$\hat{y}_{q,c} = \frac{exp(-||h(q) - S_c||)}{\sum_{c'} exp(-||h(q) - S_{c'}||)} \tag{2}$$

where $\hat{y}_{q,c}$ is the probability that the time series q is label as c class.

3.2 Relationship-Dependent Masking Module

In the distance estimation process of our DPN framework, each element of the prototype embedding is equally weighted. However, this may not be realistic, due to the fact that each element in the learned prototype representation has different importance weights to characterize the dependent relation between two embedding vectors. To mitigate this limitation, we further develop a relationship-dependent masking module to adaptively capture the implicit element-wise correlations among prototype embeddings, and let the learning algorithm focus on the most informative parts in estimating the distance. In specific, our masking module aims to free the main embedding space from the fixed-length internal representation by introducing a vector to model the dependence degrees of each element in the prototype embedding p_c. Formally, the importance computation formulations in DPN are given as follows:

$$\begin{aligned} H_c &= \tanh(W_c v + b_c), \\ M_c &= \mathrm{sigmoid}(W_i H_c + b_i) \end{aligned} \tag{3}$$

where $M \in \mathbb{R}^{d_v}$ is embedding mask, $W_c \in \mathbb{R}^{k \times d_v}$, $W_i \in \mathbb{R}^{d_v \times k}$ are transformation matrices, $b_c \in \mathbb{R}^k$, $b_i \in \mathbb{R}^{d_v}$ are bias terms. We estimate the probability $\hat{y}_{q,c}$ by incorporating mask vectors into the distance calculation process as follows: $\hat{y}_{q,c} = \frac{exp(-||(h(q)-S_c)M_c||)}{\sum_{c'} exp(-||(h(q)-S_{c'})M_{c'}||)}$.

3.3 The Learning Process of DPN Framework

We utilize cross entropy as the metric in our loss function which is defined as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{i}^{N}\sum_{c}^{C} y_{i,c}\log(\hat{y}_{i,c}) \tag{4}$$

where $y_{i,c}$ and $\hat{y}_{i,c}$ represent the actual and estimated probability that the time series i is label c. The model parameters can be derived by minimizing the average loss, iterating over training episodes and performing a gradient descent update for each. Here, episode represents sampled mini-batches during training and each episode is designed to mimic the data scarcity scenario by subsampling classes and data instances. The utilization of episodes improves the generalization of training process under data scarcity and the performance is evaluated on test set episodes.

4 EVALUATION

4.1 Data Description

We evaluate all methods thoroughly on the UCR time series classification archive[1], which consists of 49 datasets selected from various real-world domains. We measure the data balance in training set by Shannon Entropy, formally defined as follows:

$$\beta = -\frac{\sum_{i=1}^{k}\frac{c_i}{n}\log\frac{c_i}{n}}{\log k} \tag{5}$$

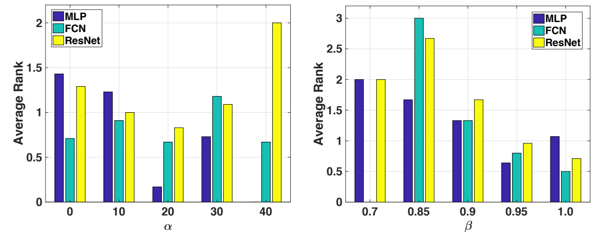
where c_i represents the number of instance in class i, k is the number of classes. The lower β indicates the higher imbalance degree.

4.2 Experimental Settings

4.2.1 **Baselines.** We consider the following baselines:

- Synthetic Minority Over-sampling Technique (SMOTE: short for SMT) [6]: it is a representative over-sampling approach for the class imbalance problem, in which the minority class is over-sampled by creating "synthetic" examples.

[1]http://www.timeseriesclassification.com/dataset.php



(a) Data Scarcity Degree α (b) Data Imbalance Degree β

Figure 2: Effect of data scarcity and imbalance degree.

- Adaptive Synthetic Sampling Approach for Imbalanced Learning (ADASYN: short for ASY) [8]: this method is an adaptive over-sampling learning developed for imbalanced data classification task by generating synthetic data samples for the minority class.
- Synthetic Minority Over-sampling Technique with Edited Nearest Neighbor Rule (SMOTE-ENN: short for SMT-E) [2]: it integrates the over- and under-sampling imbalanced class learning framework, by jointly balance training instances and removing noisy examples on the wrong side of the decision border.
- Prototype Networks (PN) [13]: this approach is proposed to address the data scarcity challenge in classification task based on inductive bias learning.

The parameter settings of SMT and SMT-E is consistent with that in [6] and [2]. In MLP transformation module, we set the number of hidden layers as 3 and 500 hidden units in each layer. Following the setting in [15], we set the architecture of FCN as (8*64, 5*128, 3*128) and architecture of ResNet as ((8*64, 5*64, 3*64), (8*128, 5*128, 3*128), (8*256, 5*256, 3*256)), where $a*b$ represents the kernel size a and the number of filters b in each layer. In *DPN*, we set the support size and mask hidden dimensionality as 5 and 64, respectively. The batch size and learning rate is set as 16 and 0.005.

4.3 Evaluation on Benchmark Dataset

We normalize the datasets by following the preprocessing step in [11]. All the experiments use the default training and testing set splits provided by UCR, and the results are rounded to three decimal places. We use error rate as an evaluation metric following the same experimental setting in [15], defined as: error $= 1 - \frac{1}{N}\sum_i \mathbb{I}[\hat{y}_i = y_i]$, where N and i are the number and index of testing instances, respectively. We report the test error rate from the best performed model trained with the minimum cross-entropy loss.

4.4 Effect of Data Scarcity and Imbalance Degree

We investigate the effectiveness of *DPN* with respect to different data scarcity degrees α and imbalance degrees β. As shown in Figure 2. We can observe that our *DPN* is not strictly sensitive to data scarcity and imbalance degrees, which suggests the robustness of our developed *DPN* in time series classification task.

5 CONCLUSION

This paper develops an effective learning framework, Deep Prototypical Networks (DPN), for imbalanced time series classification under data scarcity. In DPN, we first propose a prototypical embedding framework to embeds time series data into low-dimensional

Table 1: Performance comparison of different methods in time series classification. α and β represents average instances per class and balance density degree of time series in training data, respectively.

Method			MLP						FCN						ResNet					
Data	α	β	BSC	SMT	ASY	SMT-E	PN	DPN	BSC	SMT	ASY	SMT-E	PN	DPN	BSC	SMT	ASY	SMT-E	PN	DPN
50words	9	0.90	0.88	0.88	0.88	0.88	**0.32**	0.36	0.52	0.67	0.55	0.98	0.46	**0.41**	0.44	0.53	0.51	0.52	**0.42**	**0.42**
Adiac	10	0.99	0.98	0.98	0.98	0.98	**0.80**	0.82	0.98	0.98	0.98	0.98	**0.25**	0.27	0.98	0.98	0.98	0.98	0.29	**0.25**
ArrowHead	12	1.00	0.61	0.61	0.61	0.61	0.29	**0.03**	0.70	0.70	0.86	0.70	**0.32**	0.35	0.70	0.70	0.70	0.70	0.53	**0.41**
Beef	6	1.00	0.80	0.80	0.80	0.80	0.43	**0.06**	0.88	0.80	0.80	0.47	**0.30**	0.37	0.88	0.60	0.80	0.80	**0.40**	0.50
BeetleFly	10	1.00	0.50	0.50	0.50	0.50	**0.20**	**0.20**	0.50	0.50	0.30	0.50	0.20	**0.05**	0.50	0.50	0.50	0.50	**0.05**	**0.05**
BirdChick	10	1.00	0.50	0.50	0.50	0.50	**0.35**	**0.35**	0.50	0.50	0.50	0.50	0.30	**0.20**	0.50	0.50	0.50	0.50	0.25	**0.05**
CBF	10	0.99	0.67	0.66	0.66	0.67	0.38	**0.36**	0.03	0.67	0.05	0.04	0.02	**0.01**	0.04	0.03	0.02	0.01	0.02	**0.00**
Car	15	0.99	0.78	0.78	0.78	0.78	0.33	**0.27**	0.78	0.78	0.78	0.78	0.23	**0.13**	0.78	0.68	0.57	0.78	**0.15**	**0.15**
CinCECG	10	0.96	0.75	0.75	0.75	0.75	**0.14**	0.17	0.46	0.53	0.49	0.55	0.33	**0.28**	0.54	0.49	0.47	0.44	**0.32**	0.40
Coffee	14	1.00	**0.46**	**0.46**	**0.46**	**0.46**	**0.46**	**0.46**	0.54	0.54	0.54	0.54	0.04	**0.00**	0.46	0.46	0.46	0.46	0.04	**0.00**
CricketX	32	1.00	0.93	0.92	0.93	0.93	**0.46**	0.52	0.91	0.35	0.32	0.33	0.32	**0.27**	0.36	0.38	0.35	0.98	**0.31**	0.34
CricketY	32	1.00	0.93	0.93	0.93	0.93	0.52	**0.51**	0.37	0.37	0.40	0.39	0.36	**0.33**	0.40	0.44	0.41	0.60	0.33	**0.31**
CricketZ	32	1.00	0.94	0.92	0.94	0.94	**0.43**	0.55	0.34	0.37	0.30	0.33	**0.27**	0.29	**0.30**	0.90	0.39	0.38	0.31	0.35
eReduction	4	0.90	**0.70**	**0.70**	**0.70**	**0.70**	0.89	0.89	0.70	0.70	0.70	0.70	0.10	**0.08**	0.70	0.88	0.70	0.70	**0.05**	0.06
ECG5Days	11	0.97	0.50	0.50	0.53	0.50	**0.38**	0.50	0.25	0.04	0.07	0.18	0.01	**0.01**	0.12	0.50	0.28	0.50	**0.00**	**0.00**
CBF	10	1.00	0.87	0.87	0.87	0.87	0.51	**0.45**	0.87	0.87	0.87	0.87	0.11	**0.06**	0.87	0.87	0.87	0.87	**0.10**	0.14
FaceAll	40	1.00	0.99	0.91	0.99	0.91	0.39	**0.36**	0.20	0.32	0.28	0.31	0.20	**0.14**	0.98	0.96	**0.17**	0.18	0.21	0.19
FaceFour	6	0.95	0.70	0.70	0.70	0.70	**0.16**	**0.16**	0.20	0.29	0.18	**0.16**	**0.16**	0.17	0.23	0.27	0.70	0.28	0.23	**0.22**
FacesUCR	14	0.95	0.86	0.86	0.93	0.86	0.36	**0.34**	0.21	0.22	0.27	0.23	0.13	**0.10**	0.21	0.30	0.23	0.23	0.17	**0.12**
GunPoint	25	1.00	0.51	0.51	0.51	0.51	0.28	**0.20**	0.06	0.51	0.07	0.51	0.03	**0.01**	0.51	0.51	0.23	0.51	**0.03**	0.49
Haptics	31	0.98	0.79	0.78	0.71	0.79	0.60	**0.59**	0.68	0.68	0.66	0.64	**0.59**	0.60	0.68	0.74	0.70	0.70	0.61	**0.57**
Herring	32	0.96	**0.41**	0.59	0.59	**0.41**	**0.41**	**0.41**	**0.41**	0.59	0.59	**0.41**	0.50	0.42	0.41	0.41	0.59	0.41	0.39	**0.33**
InlineSk	14	0.99	0.84	0.91	0.84	0.84	**0.66**	0.80	0.77	0.91	0.79	0.76	**0.70**	0.80	**0.71**	0.91	0.78	0.76	0.81	0.81
InsectWS	20	1.00	0.91	0.91	0.91	0.91	0.41	**0.38**	0.91	0.67	0.75	0.61	0.63	**0.61**	0.91	**0.55**	0.91	0.91	0.57	0.60
PowerDem	33	1.00	0.50	0.50	0.50	0.50	0.06	**0.04**	0.04	0.04	0.04	**0.03**	0.04	0.04	**0.03**	0.04	0.04	0.05	0.44	0.04
Lightning2	30	0.92	0.46	0.46	0.46	0.46	0.23	**0.20**	0.15	0.20	0.16	**0.12**	0.38	0.25	0.46	**0.16**	0.18	0.18	0.28	0.29
Lightning7	10	0.96	0.74	0.74	0.86	0.74	0.29	**0.27**	**0.22**	0.29	0.40	0.40	0.29	**0.22**	0.34	0.27	0.36	0.98	**0.25**	0.27
Meat	20	1.00	0.67	0.67	0.67	0.67	**0.15**	0.35	0.67	0.67	0.67	0.67	0.08	**0.05**	0.67	0.67	0.67	0.67	0.07	**0.02**
MedicalI	38	0.71	**0.49**	0.96	0.96	**0.49**	0.82	0.73	0.26	0.26	0.28	0.29	0.30	**0.24**	0.28	0.25	**0.25**	0.31	0.28	0.26
MoteStrain	10	1.00	0.46	0.46	0.46	0.46	0.34	**0.15**	0.54	0.15	0.24	0.17	0.12	**0.10**	0.20	0.21	0.46	0.46	0.13	**0.12**
ECGThorax1	42	1.00	0.98	0.98	0.98	0.98	0.74	**0.42**	0.19	0.21	0.98	0.20	**0.09**	0.11	0.17	0.15	0.17	0.98	**0.07**	0.22
ECGThorax2	42	1.00	0.98	0.97	0.98	0.88	0.37	**0.34**	0.16	0.16	0.98	0.14	**0.09**	0.11	0.16	0.14	0.98	0.98	0.08	**0.08**
OSULeaf	33	0.97	0.82	0.87	0.91	0.82	**0.46**	**0.46**	0.27	0.91	0.91	0.22	0.15	**0.07**	0.24	0.77	0.24	0.28	0.08	**0.04**
OliveOil	7	0.93	0.60	0.60	0.60	0.60	0.83	**0.27**	0.60	0.60	0.60	0.60	0.23	**0.17**	0.60	0.60	0.60	0.60	**0.17**	**0.17**
Phoneme	5	0.90	**0.89**	**0.89**	0.98	**0.89**	0.94	0.93	0.75	**0.75**	0.78	0.77	0.77	0.76	0.76	**0.74**	0.76	0.77	0.78	0.78
Plane	15	0.99	0.91	0.80	0.91	0.91	**0.04**	**0.04**	0.03	0.03	0.04	0.04	0.01	**0.00**	0.02	0.80	0.02	0.01	0.01	**0.00**
hapeletSim	10	1.00	**0.50**	**0.50**	**0.50**	**0.50**	**0.50**	0.51	0.50	0.50	0.50	**0.00**	**0.00**	**0.00**	0.50	0.50	0.50	0.50	**0.03**	0.06
ShapesAll	10	1.00	0.98	0.98	0.98	0.98	**0.29**	0.32	0.98	0.98	0.98	0.98	0.17	**0.17**	0.98	0.98	0.98	0.98	0.18	**0.12**
SonyAIBO1	10	0.88	0.57	0.43	0.43	0.57	0.36	**0.34**	0.12	**0.06**	0.07	0.06	0.18	0.15	0.11	0.57	0.11	**0.07**	0.17	0.16
SonyAIBO2	10	0.97	0.38	0.62	0.38	0.38	**0.19**	0.20	**0.06**	0.16	0.61	0.08	0.12	0.07	0.08	0.09	**0.04**	0.89	0.09	0.09
SwedLeaf	33	1.00	0.95	0.94	0.95	0.95	0.51	**0.42**	0.11	0.94	0.08	0.10	0.07	**0.06**	0.12	0.93	0.11	0.11	**0.07**	0.08
Symbols	4	0.95	0.83	0.83	0.83	0.83	0.18	**0.17**	0.41	0.83	0.25	0.20	0.11	**0.09**	0.30	0.83	0.24	0.83	**0.13**	0.14
ToeSeg1	20	1.00	0.47	0.47	0.47	0.47	0.35	**0.33**	0.53	0.53	0.22	0.53	0.08	**0.04**	0.17	0.53	0.53	0.15	0.41	**0.08**
ToeSeg2	18	1.00	0.81	0.81	0.81	0.81	**0.15**	0.20	0.41	0.81	0.22	0.23	0.17	**0.10**	0.18	**0.12**	0.37	0.81	0.31	0.13
Trace	25	0.99	0.81	0.77	0.81	0.81	0.49	**0.40**	**0.00**	**0.00**	0.31	0.30	**0.00**	**0.00**	0.01	0.25	0.01	**0.00**	0.01	**0.00**
TwoLeadECG	11	1.00	0.50	0.50	0.50	0.50	0.28	**0.23**	0.50	0.50	0.50	**0.01**	0.10	0.08	0.50	0.50	0.50	0.28	0.12	**0.11**
Wine	28	1.00	0.50	0.50	0.50	0.50	0.50	**0.43**	0.50	0.50	0.50	0.50	**0.24**	0.26	0.50	0.50	0.50	0.50	0.26	**0.22**
WordSyns	10	0.87	0.78	0.78	0.98	0.78	**0.41**	0.82	0.62	**0.58**	0.60	0.62	0.60	0.81	0.57	**0.55**	0.97	0.58	0.55	0.66
Worms	36	0.91	0.58	0.58	0.82	0.58	0.54	**0.15**	0.49	**0.41**	0.45	0.43	0.44	0.44	0.50	0.39	0.49	0.49	0.44	**0.37**
Win			6	4	3	6	20	**33**	4	5	0	6	12	**30**	3	5	3	2	16	**26**
Avg Rank			2.04	2.71	3.90	4.29	1.12	**0.94**	2.63	3.16	3.61	3.22	1.47	**0.90**	2.43	3.12	3.14	3.84	1.37	**1.10**

representations. Then, we design a relationship-dependent masking module to augment the main embedding space of DPN by automatically learning the importance weights of prototype embeddings. One future direction is to incorporate general knowledge from external sources to further improve the DPN model.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments. This work is supported in part by the National Science Foundation (NSF) grants IIS-1447795 and CNS-1629914.

REFERENCES

[1] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. 2015. Time-series classification with COTE: the collective of transformation-based ensembles. *TKDE* 27, 9 (2015), 2522–2535.

[2] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations* 6, 1 (2004), 20–29.

[3] Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. 2013. A bag-of-features framework to classify time series. *TPAMI* 35, 11 (2013), 2796–2802.

[4] Victor Bisot, Romain Serizel, Slim Essid, and Gaël Richard. 2016. Acoustic scene classification with matrix factorization for unsupervised feature learning. In *ICASSP*. IEEE, 6445–6449.

[5] Zhicheng Cui, Wenlin Chen, et al. 2016. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995* (2016).

[6] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. 2018. SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *JAIR* 61 (2018), 863–905.

[7] Yanwei Fu, Timothy M Hospedales, Tao Xiang, and Shaogang Gong. 2015. Transductive multi-view zero-shot learning. *TPAMI* 37, 11 (2015), 2332–2345.

[8] Haibo He, Yang Bai, et al. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *IJCNN*. IEEE, 1322–1328.

[9] Fazle Karim, Somshubra Majumdar, et al. 2017. LSTM Fully Convolutional Networks for Time Series Classification. (2017). arXiv:arXiv:1709.05206

[10] Steven Cheng-Xian Li and Benjamin M Marlin. 2016. A scalable end-to-end gaussian process adapter for irregularly sampled time series classification. In *NIPS*. 1804–1812.

[11] John Paparrizos and Luis Gravano. 2015. k-shape: Efficient and accurate clustering of time series. In *SIGMOD*. ACM, 1855–1870.

[12] Patrick Schäfer. 2016. Scalable time series classification. *DMKD* 30, 5 (2016), 1273–1298.

[13] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *NIPS*. 4077–4087.

[14] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *NIPS*. 3630–3638.

[15] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *IJCNN*. IEEE, 1578–1585.

[16] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, Louis Faust, and Nitesh V Chawla. 2018. Restful: Resolution-aware forecasting of behavioral time series data. In *CIKM*. ACM, 1073–1082.

[17] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. 2015. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.. In *IJCAI*, Vol. 15. 3995–4001.