Joint-Label Learning by Dual Augmentation for Time Series Classification

Qianli Ma[1,2], **Zhenjing Zheng**[1], **Jiawei Zheng**[1], **Sen Li**[1], **Wanqing Zhuang**[1], **Garrison W. Cottrell**[3]
[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou
[2]Key Laboratory of Big Data and Intelligent Robot (South China University of Technology), Ministry of Education
[3]Department of Computer Science and Engineering, University of California, San Diego, CA, USA
qianlima@scut.edu.cn, 982360227@qq.com

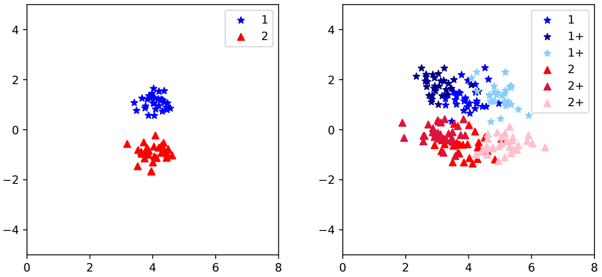
Abstract

Recently, deep neural networks (DNNs) have achieved excellent performance on time series classification. However, DNNs require large amounts of labeled data for supervised training. Although data augmentation can alleviate this problem, the standard approach assigns the same label to all augmented samples from the same source. This leads to the expansion of the data distribution such that the classification boundaries may be even harder to determine. In this paper, we propose Joint-label learning by Dual Augmentation (JobDA), which can enrich the training samples without expanding the distribution of the original data. Instead, we apply simple transformations to the time series and give these modified time series new labels, so that the model has to distinguish between these and the original data, as well as separating the original classes. This approach sharpens the boundaries around the original time series, and results in superior classification performance. We use Time Series Warping for our transformations: We shrink and stretch different regions of the original time series, like a fun-house mirror. Experiments conducted on extensive time-series datasets show that JobDA can improve the model performance on small datasets. Moreover, we verify that JobDA has better generalization ability compared with conventional data augmentation, and the visualization analysis further demonstrates that JobDA can learn more compact clusters.

Introduction

Time series classification (TSC) is a task that learns to recognize unlabeled time series given a set of time series from different categories. Such tasks are ubiquitous in daily life, such as auxiliary medical diagnosis (Dai et al. 2018; Perslev et al. 2019), speech analysis (Trentin, Scherer, and Schwenker 2015), action recognition (Yang et al. 2015; Tanfous, Drira, and Amor 2019; Ma et al. 2019), and so on.

In recent years, deep neural networks (DNNs) have been applied to a wide variety of tasks and achieved great success. Naturally, they have also been applied to TSC. Wang et al. (Wang, Yan, and Oates 2017) tested three different DNN architectures, multilayer perceptrons (MLP), residual networks (ResNet), and fully convolutional networks (FCN), and verified the effectiveness of these three models. Further, Karim et al. (Karim et al. 2018) proposed long short term memory (LSTM) fully convolutional networks (LSTM-FCN) that augment an FCN with either an LSTM network, or an LSTM network with attention. Although the DNN-based methods have reached the state-of-the-art performance in TSC, it is still a challenge to apply them to datasets with only a small amount of labeled data, since large amounts of labeled data are required for supervised training.



(a) original feature map (b) with data augmentation

Figure 1: Schematic diagram of data distribution (a) w/o data augmentation (b) w/ data augmentation.

One simple yet effective solution to tackle this problem is the use of data augmentation. Data augmentation increases the size of the training set by using synthesized samples. Some data augmentation methods have been proposed for TSC tasks to improve the generalization performance of the classifier. For example, simple transformations in the time domain (Le Guennec, Malinowski, and Tavenard 2016; Um et al. 2017) such as window slicing and warping, rotation, permutation, scaling, and jittering (adding noise), have been used for time-series data augmentation. Fawaz et al. (Fawaz et al. 2018) proposed a pattern mixing method that uses a weighted version of the DTW Barycentric Averaging algorithm to generate the new time-series samples. In pattern mixing, the new time-series samples are synthesized from multiple original samples from the same category instead of using transformations.

However, the common practice in data augmentation is to assign the same label to all augmented samples from the same source, which may cause some adverse effects on the learning of the model. We use a schematic diagram to illustrate what we mean. Figure 1(a) is the original data distribution. Figure 1(b) is the data distribution that uses data augmentation to increase the size of the training set, assigning the same label to all augmented samples from the same source. We can see that the original data distribution is expanded, since it is affected by the augmented samples with the same labels. Therefore, the data distribution of different categories can lead to category overlap, so that the classification boundary may not be well determined.

Recently, in the visual domain, Lee et al. (Lee, Hwang, and Shin 2020) proposed augmenting the data by rotation and color, and learning the joint distribution of the original labels and the self-generated labels (e.g., (dog, 90°) is a label). Inspired by this work, we propose Joint-label learning by Dual Augmentation (JobDA) for TSC, which augments original data using self-supervision from two aspects: sample augmentation and label augmentation. Specifically, we first propose a novel time-series sample augmentation method called time-series warping (TSW) that can simulate time-shifting or deformations of the local patterns while keeping the length of the time series unchanged. Then we assign the self-supervised label to each sample according to the particular TSW transform applied, performing label augmentation. Finally, combining the original and self-supervised labels, a novel joint-label learning method is utilized to learn multiple compact clusters using the original time series and augmented ones. In this way, JobDA can increase the size of the training set while learning multiple compact clusters of time series data instead of expanding the original data distribution. Our contributions can be summarized as follows:

- We propose a dual-augmentation mechanism that augments the original time series from two aspects: sample augmentation and label augmentation. We use time-series warping (TSW) which simulates deformations of the local patterns. All data transformed the same way receives a new label.
- Combining original labels and self-supervised ones, we propose a novel joint-label learning method to learn multiple compact clusters for time series classification. This enhances generalization by avoiding expansion of the original data distribution.
- Experiments conducted on extensive time-series datasets show that JobDA can improve the model performance on small datasets. Moreover, we verify that JobDA has better generalization ability than conventional data augmentation, and the visualization analysis further demonstrates that JobDA can learn more compact clusters.

Related Work

Time-Series Data Augmentation

In recent years, deep neural networks (DNNs) have achieved excellent performance on TSC. The superior performance of deep learning methods relies heavily on a large amount of labeled data to avoid overfitting. As a simple yet effective method to increase the size of the training set, data augmentation plays a crucial role in the application of DNNs in TSC.

Time-series data augmentation (TSDA) in the time domain is the most common method, manipulating the original time series directly. Guennec et al. (Le Guennec, Malinowski, and Tavenard 2016) proposed window slicing and window warping for TSDA. Similar to the cropping of the image, window slicing randomly selects continuous slices of a given time series and assigns them the same label. Window warping warps a randomly selected slice of time series by upsampling or downsampling, changing the length of time series. Um et al. (Um et al. 2017) used a variety of augmentations such as rotation, permutation, scaling, magnitude warping, jittering (adding noise), and cropping for CNN-based classification. Fawaz et al. (Fawaz et al. 2018) used a weighted version of the DTW Barycentric Averaging algorithm to generate the new time series. In addition to TSDA in the time domain, some studies investigate data augmentation from the perspective of the frequency domain. For example, Eyobu et al. (Steven Eyobu and Han 2018) conducted two data augmentations (local averaging and shuffling of feature vectors) on the time-frequency features that are generated by the short Fourier transform (STFT). In addition, generative adversarial networks (GANs) (Nikolaidis et al. 2019; Yoon, Jarrett, and Der Schaar 2019) can be used to generate time-series samples from the same domain.

However, common practice in data augmentation is to assign the same label to all augmented samples from the same source, which may bring some adverse effects on model learning. The original data distribution may be expanded, since it is affected by the augmented samples with the same labels. Therefore, the data distribution of different categories can lead to overlapping distributions so that the classification boundary may not be well determined.

Self-Supervised Auxiliary Learning

Self-supervised learning (Doersch, Gupta, and Efros 2015; Dosovitskiy et al. 2016; Gidaris, Singh, and Komodakis 2018) was proposed for unsupervised learning originally. Some recent studies used self-supervised learning as an auxiliary task to help the primary task learn better by training the auxiliary task alongside the primary task. In the auxiliary task, the model predicts which transformation is applied to the input given the transformed samples. For example, Gidaris et al. (Gidaris et al. 2019) used self-supervision as an auxiliary task that predicted the rotations of images to boost few-shot visual learning. Lee et al. (Lee, Hwang, and Shin 2020) learned the joint distribution of the original labels and self-supervised labels that are generated based on the rotations of images to improve image classification. Inspired by this work, we introduce the idea of self-supervision to learn multiple compact clusters for time series classification.

Proposed Method

In Figure 2(a), we make a comparison between the proposed Joint-label learning by Dual Augmentation (JobDA) and previous approaches (w/o data augmentation and w/ data augmentation) at the training phase. We show the proposed method at test time in Figure 2(b). JobDA consists of two modules: dual augmentation and joint-label learning. The

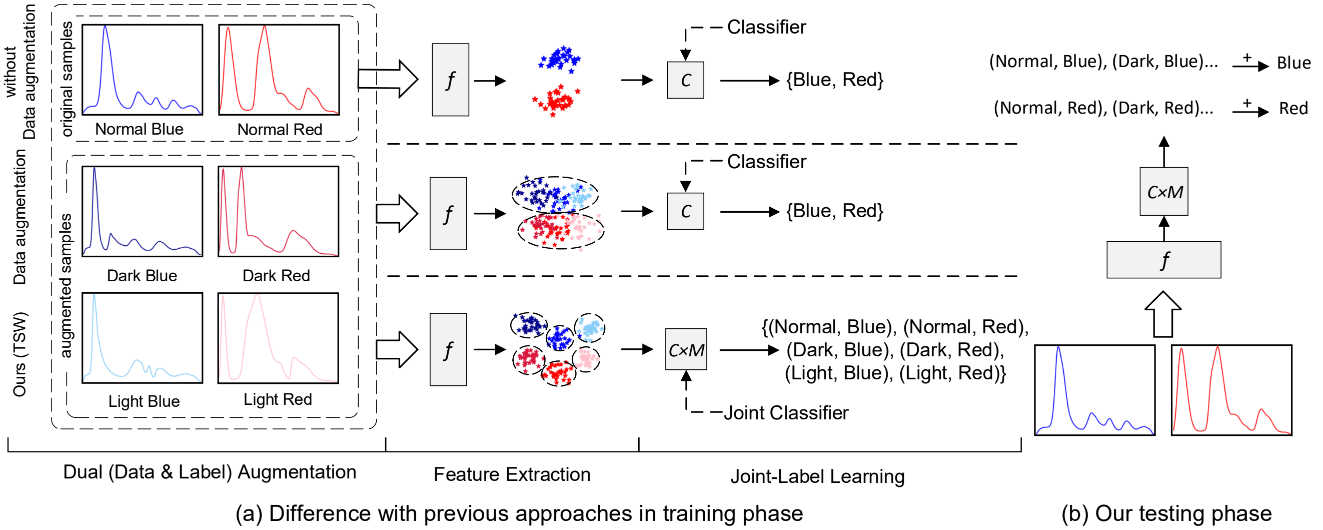

without Data augmentation
Data augmentation
Ours (TSW)
original samples
augmented samples
Normal Blue
Normal Red
Dark Blue
Dark Red
Light Blue
Light Red
f
Classifier
C
{Blue, Red}
C×M
{(Normal, Blue), (Normal, Red), (Dark, Blue), (Dark, Red), (Light, Blue), (Light, Red)}
Joint Classifier
(Normal, Blue), (Dark, Blue)... + Blue
(Normal, Red), (Dark, Red)... + Red
Dual (Data & Label) Augmentation
Feature Extraction
Joint-Label Learning
(a) Difference with previous approaches in training phase
(b) Our testing phase

Figure 2: (a) Comparison of our Joint-label learning by Dual Augmentation (JobDA) and previous approaches (w/o data augmentation and w/ data augmentation) in the training phase. (b) Our inference method in the testing phase.

dual augmentation is first used to perform sample augmentation and label augmentation on the training set, which increases the size of the training set and helps the classifier learn more compact clusters. The joint classifier with softmax output is performed on the time series of the training set and augmented set to predict the joint-labels consisting of original labels and self-supervised labels. In the testing phase, we use joint classifier to obtain the final distribution for each joint-label. We only need to consider the original category in the testing phase. Hence, we predict an original label by summing the probabilities of all joint-labels that come from this original category.

Dual Augmentation

Sample Augmentation with Time-Series Warping. The purpose of sample augmentation is to improve the generalization performance of model by increasing the size of training set. Namely, given a training set with n time series $\boldsymbol{T} = \{\boldsymbol{t}_1, \cdots, \boldsymbol{t}_i, \cdots, \boldsymbol{t}_n\}$, each time series $\boldsymbol{t}_i$ contains m ordered real values. $y_i \in \{1, 2, \cdots, C\}$ denotes the label of the i-th sample, and C is the number of categories. The goal is to create augmented set $\boldsymbol{T}^{'}$ by applying transformations to the original time series such that the classifier is trained on $\boldsymbol{T}^{\text{aug}} = \boldsymbol{T} \cup \boldsymbol{T}^{'}$ to improve its generalization performance.

To improve the generalization performance of the time-series classifier, we propose a novel time-series sample augmentation called time-series warping (TSW). TSW alternately compresses and expands different subsequences of the time series by using downsampling and upsampling operation while keeping the length of the time series unchanged.

For each time series $\boldsymbol{t}_i \in \mathbb{R}^{m \times 1}$, we first divide it into N continuous subsequences with equal length $L = \lfloor \frac{m}{N} \rfloor$. Then we perform downsampling or upsampling on each subsequence of the time series. Average pooling with a stride of 2 is used for downsampling. For upsampling, we insert the average value between every two values with a stride of 2. We perform downsampling and upsampling operations on the N subsequences of the time series alternately, and concatenate them together, an approach that could be fancifully called the “fun-house mirror” transformation. To produce different transformations, we vary N. TSW is a simple transformation on the original time series, which is an efficient operation. Obviously, many other transformations are possible, and we use this one for convenience. It is worth investigating what other transformations can be used, as some will undoubtedly work better than others. We discuss this question further in section B of the supplementary material.

Label Augmentation. The common practice of data augmentation is assigning the same label to all augmented time series from the same source, and so the number of categories is unchanged. In this case, the patterns of the augmented time series should be similar to the ones in the original time series so as not to introduce too much noise. However, this is not always possible. Consider a binary classification task, if the augmented samples of class 1 are close to the samples of class 2, assigning these augmented samples to class 1 is clearly inappropriate, as they may now overlap with class 2. In this case, the classifier will still try to make these augmented and original samples of class 1 close to each other in the feature space, even though the examples are in class 2. This process is likely to expand the distribution of the original data, and the classification boundary may not be well determined. Inspired by recent work in self-supervised learning, we assign a self-supervised label to each time series according to the different transformations. Hence, if there are M transformations (including the identity transformation for the original categories), there will now be MC categories.

As an example, in Figure 2(a), there are two categories (Blue and Red) in the original training set. After data and label augmentation, each time series has two joint labels. One is the original label (Blue and Red), and the other is the self-supervised label (Normal (original time series), Dark, and Light). Using the cross-product of these labels gives rise to six categories.

Joint-Label Learning

After augmentation, we train the classifier on the new training set $\boldsymbol{T}^{\text{aug}}$. Again, our approach differs from previous ones because we learn a separate category for each transformation. This approach is introduced to learn multiple compact clusters instead of expanding the distribution of the original time series.

For the original training set $\boldsymbol{T}$ with n time series and C original categories, we perform M transformations on each time series, so the number of self-supervised categories is M. The key of the joint-label learning method is learning the joint-label composed of the original and self-supervised label of each time series so that the original data distribution will not be affected by the augmented samples with the same original labels. Therefore, the joint classifier is trained on the new training set $\boldsymbol{T}^{\text{aug}}$ with Mn time series and MC categories. Let $g(\cdot;\boldsymbol{\omega})$ denotes the joint classifier, where $\boldsymbol{\omega}$ is the weights of joint classifier. For a training time series $\boldsymbol{t}$ on $\boldsymbol{T}^{\text{aug}}$, the conditional distribution over each joint-label can be defined by

$$\boldsymbol{z} = g(\boldsymbol{t};\boldsymbol{\omega}), \tag{1}$$

$$P(\boldsymbol{MC}|\boldsymbol{t}) = \text{softmax}(\boldsymbol{z}), \tag{2}$$

where $\boldsymbol{z} \in \mathbb{R}^{MC\times 1}$ denotes output vector generated by the joint classifier. $P(\boldsymbol{MC}|\boldsymbol{t})$ denotes the conditional label distribution of the input time series $\boldsymbol{t}$.

A potential limitation of our approach is if C is large, and the categories are close to one another, we may proliferate categories and end up with overlap between the new set. In the supplementary material, we indeed do perform worse on a 14-way classification on the `FacesUCR` dataset. This is a set of "time series" generated by converting the outline of the profiles of different graduate student's heads, using multiple profile views of each person, into one-d time series. This dataset is likely to have similar sequences across people.

Inference

During the testing phase, we only need to consider how to classify the original time series into the original C categories, given that the classifier is trained to classify time series into MC categories. We predict original label by summing the probabilities of all joint-labels that come from the same original category. Given a testing time series $\boldsymbol{x}$, we can obtain the conditional distribution of $\boldsymbol{x}$ on $C \times M$ categories as follows

$$\boldsymbol{p} = \text{softmax}(g(\boldsymbol{x};\boldsymbol{\omega})), \tag{3}$$

where $\boldsymbol{p} = \{p_{1,1}, \cdots, p_{1,M}, \cdots, p_{i,1}, \cdots, p_{i,M}, \cdots, p_{C,1}, \cdots, p_{C,M}\}$ denotes the probability distribution of the MC joint labels and $g(\boldsymbol{x};\boldsymbol{\omega})$ is the classifier parameterized by $\boldsymbol{\omega}$. The conditional distribution over each original category label can be defined by

$$P(\boldsymbol{C}|\boldsymbol{x}) = \{p_1, \cdots, p_i, \cdots, p_C\} \quad \text{where} \quad p_i = \sum_{k=1}^{M} p_{i,k}. \tag{4}$$

Experiments

Experimental Setup

We describe the settings of our experiments in this section.

Datasets. We conduct experiments on the UCR time series classification archive[1] (Chen et al. 2015) to compare the proposed method with other methods. The UCR time series classification archive contains 85 publicly available time-series datasets, and each dataset was split into training and testing set using the standard split. To maintain the integrity of the experiments, we conducted experiments on 85 UCR datasets. The statistics of these 85 datasets are shown in section A of the supplementary material.

Baselines. The proposed method is compared with three SOTA deep learning-based time series classification methods (Wang, Yan, and Oates 2017): Multilayer Perceptron (MLP), Fully Convolutional Network (FCN), and Residual Network (ResNet). The introduction to these three baselines are shown in section A of the supplementary material. In addition, we compare the proposed method with 1-Nearest Neighbor with Dynamic Time Warping (1NN-DTW) (Berndt and Clifford 1994), which achieved very good performance on small UCR time series datasets.

Implementation Details. Keras 2.2.4[2] is used to implement all our experiments, which run on an Intel Core i7-6850K 3.60GHz CPU, 64GB RAM, and a GeForce GTX 1080-Ti 11G GPU. We perform four TSW-based transformations (including the original time series) on each time series of the training set for sample augmentation. In addition to the original time series, the number of subsequences N used in the other three transformations are 2, 4, and 8, respectively. The loss function is categorical cross-entropy. We choose the model architecture that achieves the lowest training loss and report its performance on the test set (the UCR time series archive does not have holdout set splits). The classification accuracy is used to evaluate the performance of the model, and the macro-F1 score (Yang 1999) is used for class-imbalanced classification. To reduce the impact of random initialization, we run each experiment five times and report the mean and standard deviation.

Comparison with State-of-the-art Methods

The proposed method is compared with three SOTA deep learning-based time series classification methods: MLP, FCN, and ResNet. For a fair comparison, we also construct three methods using the same augmented dataset without the joint labels (i.e., the traditional approach to data augmentation). We call these: MLP with single-label learning (MLP_SL), FCN with single-label learning (FCN_SL), and

[1] https://www.cs.ucr.edu/~eamonn/time_series_data/

[2] https://github.com/fchollet/keras

	MLP	FCN	ResNet	MLP_SL	FCN_SL	ResNet_SL	ResNet_JL
#Best	4	14	21	5	13	17	**37**
Avg_rank	5.853	4.000	3.053	5.494	3.771	3.176	**2.653**

Table 1: Statistical results of ResNet_JL and 6 classifiers on 85 UCR datasets. The best result is indicated as bold.

Dataset	Class	N_{max}/N_{min}	N_{spc} before DA	N_{spc} after DA	ResNet	ResNet_SL	ResNet_JL
Earthquakes	2	2.97	[104,35]	[104,140]	0.506	0.537	**0.542**
DistPhaxAgeGrp	3	4.33	[15, 59, 65]	[60, 59, 65]	0.697	**0.719**	0.711
ProxPhaxTW	6	36.00	[2,67,40,10,14,72]	[8,67,40,40,64,72]	0.486	0.523	**0.542**
ECG5000	5	146.00	[292,177,10,19,2]	[292,177,40,76,8]	0.592	0.582	**0.598**

Table 2: Macro-F1 score on four imbalance datasets. N_{max}/N_{min} is the ratio between the numbers of samples of most and least frequent classes. N_{spc} is the number of samples for each category. The best result is indicated as bold.

ResNet with single-label learning (ResNet_SL). Their training set is identical to ours, using the same four TSW-based transformations, so they only perform sample augmentation on the training set without label augmentation. We use ResNet with joint-label learning (ResNet_JL) to evaluate the performance of our proposed method since ResNet performs better than the other two models (Ismail Fawaz et al. 2019). ResNet_JL is trained on the training set after dual augmentation. The results of MLP, FCN, and ResNet are collected from (Ismail Fawaz et al. 2019). The full results of these methods on 85 UCR datasets are shown in section D of the supplementary material.

As shown in Table 1, ResNet_JL achieves the best results on 37 of the 85 datasets and also the best average rank of 2.653. We see that single-label learning with the dataset augmented by TSW can improve model performance. For example, MLP_SL and FCN_SL are numerically superior to MLP and FCN in average rank, respectively. However, the improvement of single-label learning is relatively small, and it may even reduce model performance. For example, ResNet is slightly better numerically than ResNet_SL in average rank. In addition, ResNet_JL and ResNet_SL achieve higher accuracy than ResNet on 45 and 36 datasets, respectively, which shows that our method can be adapted to a wider range of tasks. To further analyze the performance, we also conduct the Nemenyi non-parametric statistical test (Demšar 2006) and plot the critical difference diagram in Figure 8 of the supplementary material. The Nemenyi test shows that ResNet_JL is significantly superior to MLP-based and FCN-based methods at $p < 0.05$ level, and slightly superior to ResNet-based methods. Therefore, JobDA can improve model performance effectively.

For smaller UCR time series datasets, 1NN-DTW (Berndt and Clifford 1994) achieved very good performance. To further verify the effectiveness of the proposed method, we compare our method with 1NN-DTW on 44 datasets with training set sizes of 200 or less. The results are shown in Table 9 of the supplementary material. We see that our method achieves higher classification accuracy in the vast majority of datasets than 1NN-DTW.

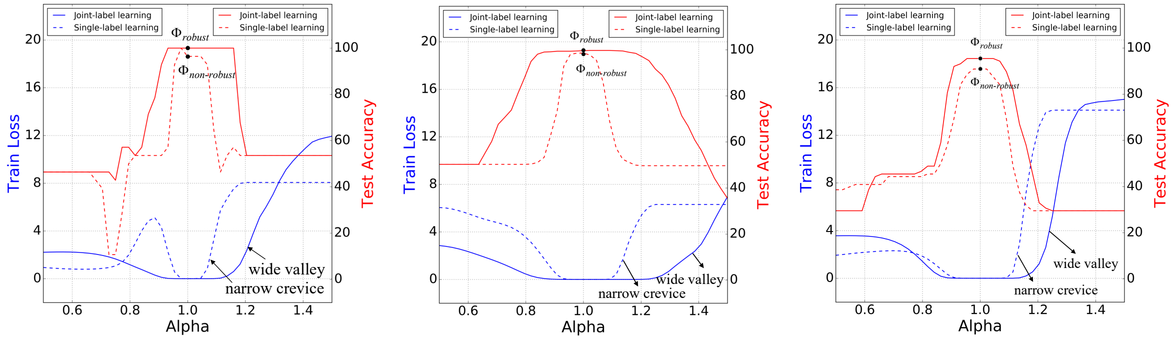
Class-imbalanced Classification

In classification tasks, class imbalance usually reduces the performance of the classifier since the classifier cannot identify data from minority classes easily. One way to solve the problem of class imbalance is oversampling the minority classes in the training set so that the number of samples in each category is closer. To explore whether our method can improve model performance on imbalanced classification similarly to conventional data augmentation, we use single-label learning and joint-label learning to oversample four UCR datasets with different imbalance ratios and test their performance. Here we use the Macro-F1 score as the evaluation metric of model performance on imbalanced datasets. As shown in Table 2, both single-label learning, and joint label-learning can improve the model performance, and ResNet_JL achieves the best performance on 3 of the 4 datasets, which shows that joint-label learning can further improve the model performance on imbalanced datasets.

Generalization Ability Analysis

In this section, we explore the generalization ability of our proposed method. Recent research has explored the issue of why deep networks generalize despite being overparameterized. They provide the insight that although many solutions can achieve zero training error, some can generalize better since they are converging to flat minima rather than deep, sharp minima (Chaudhari et al. 2017; Keskar et al. 2017). A flat minima is a large connected region in weight space where the error remains approximately constant. In contrast, a sharp minima is a connected region in weight space where the error changes rapidly (Hochreiter and Schmidhuber 1997). Therefore, the small perturbations do not cause significant performance degradation for flat minima.

To investigate whether our method has converged to a flat minimum, we need to qualitatively analyze the objective function. However, the objective function of DNNs is complicated and high-dimensional. It is difficult to analyze the objective function in a two-dimensional visualization. Goodfellow et al. (Goodfellow, Vinyals, and Saxe 2015) provided a simple technique to qualitatively analyze the objec-



(a) Coffee (b) ECGFiveDays (c) FaceFour

Figure 3: Linear parametric plots. The left vertical axis corresponds to training loss, the right vertical axis corresponds to test accuracy. The solid line indicates training ResNet with joint-label learning and dashed line indicates training ResNet with single-label learning. Red: accuracy; Blue: error)

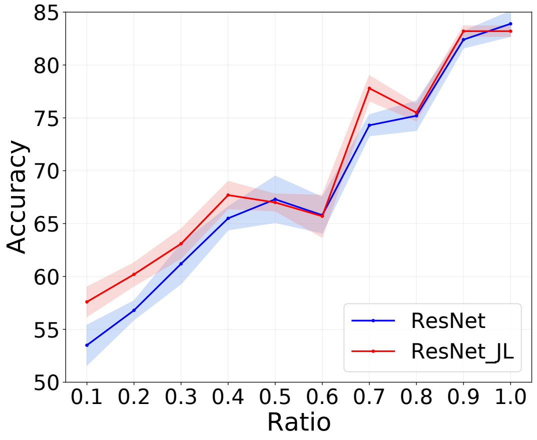
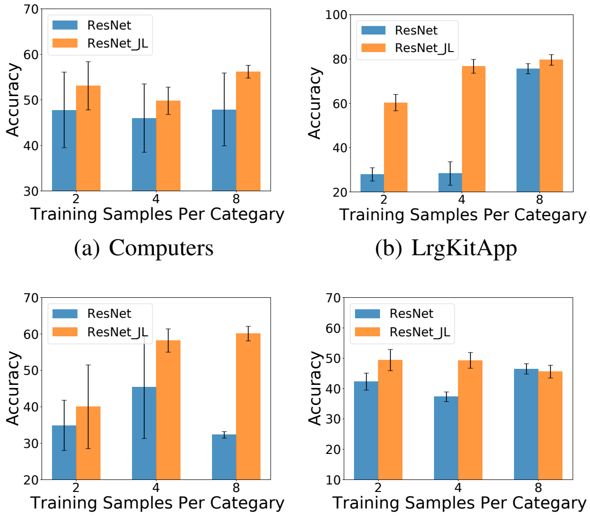


Figure 4: The training set size vs accuracy on the PhalangesOutlinesCorrect dataset.

tive function, which makes the cross-section of the object function observable. Specifically, let $\mathbf{\Phi}_{\text{init}}$ and $\mathbf{\Phi}_{\text{best}}$ denote the initial solution and the final solution that achieves the lowest training loss. The training loss and testing accuracy are evaluated at a series of points $\mathbf{\Phi} = (1-\alpha)\mathbf{\Phi}_{\text{init}} + \alpha\mathbf{\Phi}_{\text{best}}$ for varying values of α. We set $\alpha \in [0.5, 1.5]$, where $\alpha = 1.0$ denotes the results that achieve the lowest training loss. As shown in Figure 3, we present the linear parametric plots of ResNet_JL and ResNet_SL on three datasets. The solid and dashed line are used to indicate ResNet_JL and ResNet_SL, respectively, while red lines correspond to accuracy, and blue lines correspond to the cross-section of the error surface. We can see that the final solution of ResNet_JL falls in a wide valley, while the final solution of ResNet_SL falls in a narrow crevice. In addition, the final solution of ResNet_JL has better robustness since the model has a higher and smoother testing accuracy near the final solution. Therefore, our method has found a flat minimum, which leads to higher generalization performance.



(a) Computers (b) LrgKitApp

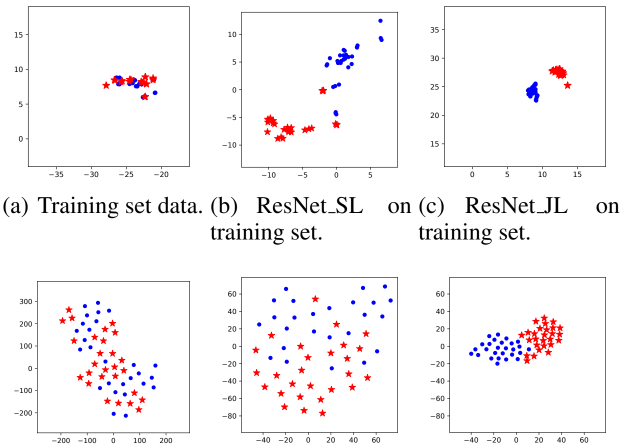
(c) ProxPhxCorr (d) RefrigerationDevices

Figure 5: The accuracy vs training set size (only a few training samples per category) on four UCR datasets.

Analysis of Training Set Size

To verify that the proposed method can effectively solve the problem of the scarcity of labeled training data, we conducted experiments on the PhalCorr (1800 trainging samples) dataset. We only use part of the training samples (scale from 0.1 to 1.0) to train the model. Specifically, given a ratio r, we sample $r \times N_c$ samples from each category of samples to form the training set, where N_c denotes the number of samples in the c-th category of original training set.

The experimental results of training set size vs accuracy on the PhalCorr dataset are shown in Figure 4. Regardless of ResNet or ResNet_JL, the classification accuracy increases with the increase of training data. In addition, we see that ResNet_JL can improve the model performance better when the training set size is small. To further explore the capability of the proposed method to solve the problem of scarcity of



(a) Training set data. (b) ResNet_SL on training set. (c) ResNet_JL on training set.

(d) Testing set data. (e) ResNet_SL on testing set. (f) ResNet_JL on testing set.

Figure 6: The visualizations with t-SNE on the Wine dataset. The subfigures on each row from left to right are the original time series data, the feature maps of ResNet_SL and the feature maps of ResNet_JL, respectively.

	ResNet	M=2	M=3	M=4	M=5
#Better	-	32	32	35	31
p-value	-	0.030	0.006	0.016	0.005

Table 3: Statistical results of ResNet and ResNet_JL with different M on 44 UCR datasets.

labeled training data, we use extremely few original training samples, reducing the training sample size of each category to 2, 4, and 8, and conduct experiments on 4 UCR datasets. As shown in Figure 5, we see that ResNet_JL can improve the model performance overall. This shows that ResNet_JL can improve the model performance on small datasets. More examples can be found in the supplementary material.

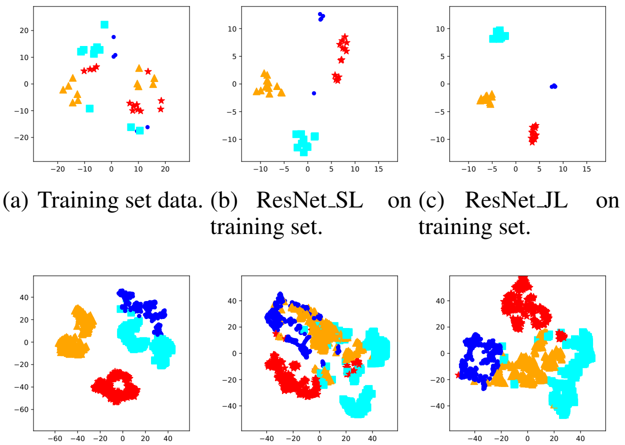
Analysis of Transformation Number M

To explore the effect of the transformation number M on the model performance, we evaluate the performance of ResNet_JL with different transformation number M ($M = 2, 3, 4, 5$, including the original time series) on 44 UCR time series datasets. The full results are shown in section E of the supplementary material.

As shown in Table 3, we see that ResNet_JL with M=2, 3, 4, and 5 achieved the better (or equal) results (p-value) of 32(0.030), 32(0.006), 35(0.016), and 31(0.005) than ResNet in 44 small UCR datasets, respectively. This shows that no matter which value M takes, ResNet_JL can achieve better performance than ResNet.

Visualization Analysis

To explore the effectiveness of joint-label learning, we take the feature maps of ResNet_SL and ResNet_JL on the Wine dataset and CinCECGtorso dataset, respectively, and use t-SNE (Maaten and Hinton 2008) to map the features into a 2-D space. The features of the Wine dataset after dimensionality reduction are shown in Figure 6. Comparing the feature maps of ResNet_SL (Figure 6(b)) and ResNet_JL (Figure 6(c)) on the training set, we find that ResNet_JL learns the more compact clusters, while ResNet_SL expands the original data distribution. In addition, comparing the feature maps of ResNet_SL (Figure 6(e)) and ResNet_JL (Figure 6(f)) on testing set, the classification boundary of ResNet_JL is well determined compared to ResNet_SL. Similarly, the same phenomenon also appears in the CinCECG-torso dataset (Figure 7). Therefore, joint-label learning doesn't just increase the size of the training set, but also doesn't expand the distribution of the original data, making the classification boundary better determined.



(a) Training set data. (b) ResNet_SL on training set. (c) ResNet_JL on training set.

(d) Testing set data. (e) ResNet_SL on testing set. (f) ResNet_JL on testing set.

Figure 7: The visualizations with t-SNE on the CinC_ECG_torso dataset.

Conclusion

In this paper, we propose a novel time-series data augmentation method called Joint-label learning by Dual Augmentation (JobDA), which can enrich the training samples and learn multiple compact clusters instead of expanding the distribution of the original data. Unlike conventional data augmentation, JobDA assigns a new label that combines original and self-supervised label for each sample. Experiments conducted on extensive time-series datasets show that JobDA can improve the model performance on small datasets. Moreover, we verify that JobDA has better generalization ability than conventional data augmentation, and the visualization analysis further demonstrates that JobDA can learn the more compact clusters. JobDA is a fully supervised method because the class and transformation of augmented samples are both known during training. In future work, we will consider extending it to the unsupervised learning.

Acknowledgments

We thank the anonymous reviewers for their helpful feedbacks. The work described in this paper was partially funded by the National Natural Science Foundation of China (Grant Nos. 61502174, 61872148), the Natural Science Foundation of Guangdong Province (Grant Nos. 2017A030313355, 2019A1515010768), the Guangzhou Science and Technology Planning Project (Grant Nos. 201704030051, 201902010020), the Key R&D Program of Guangdong Province (No. 2018B010107002), and the Fundamental Research Funds for the Central Universities.

Ethics Statement

We have expanded the application of deep learning to time series data where there is a relatively small amount of data. This can be important in applications where data is difficult or expensive to come by. On the other hand, there is little social impact of this work.

References

Berndt, D. J.; and Clifford, J. 1994. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, 359–370. Seattle, WA.

Chaudhari, P.; Choromanska, A.; Soatto, S.; LeCun, Y.; Baldassi, C.; Borgs, C.; Chayes, J.; Sagun, L.; and Zecchina, R. 2017. Entropy-SGD: Biasing gradient descent into wide valleys. In *Proceedings of International Conference on Learning Representations*.

Chen, Y.; Keogh, E.; Hu, B.; Begum, N.; Bagnall, A.; Mueen, A.; and Batista, G. 2015. The UCR time series classification archive. www.cs.ucr.edu/∼eamonn/time_series_data/.

Dai, C.; Wu, J.; Pi, D.; and Cui, L. 2018. Brain EEG time series selection: A novel graph-based approach for classification. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, 558–566. SIAM.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7: 1–30.

Doersch, C.; Gupta, A.; and Efros, A. A. 2015. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 1422–1430.

Dosovitskiy, A.; Fischer, P.; Springenberg, J. T.; Riedmiller, M.; and Brox, T. 2016. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38(9): 1734–1747.

Fawaz, H. I.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2018. Data augmentation using synthetic data for time series classification with deep residual networks. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases Workshop on Advanced Analytics and Learning on Temporal Data*.

Gidaris, S.; Bursuc, A.; Komodakis, N.; Pérez, P.; and Cord, M. 2019. Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE International Conference on Computer Vision*, 8059–8068.

Gidaris, S.; Singh, P.; and Komodakis, N. 2018. Unsupervised representation learning by predicting image rotations. In *Proceedings of International Conference on Learning Representations*.

Goodfellow, I.; Vinyals, O.; and Saxe, A. M. 2015. Qualitatively characterizing neural network optimization problems. In *Proceedings of International Conference on Learning Representations*.

Hochreiter, S.; and Schmidhuber, J. 1997. Flat minima. *Neural Computation* 9(1): 1–42.

Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Deep learning for time series classification: A review. *Data Mining and Knowledge Discovery* 33(4): 917–963.

Karim, F.; Majumdar, S.; Darabi, H.; and Chen, S. 2018. LSTM fully convolutional networks for time series classification. *IEEE Access* 6: 1662–1669.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In *Proceedings of International Conference on Learning Representations*.

Le Guennec, A.; Malinowski, S.; and Tavenard, R. 2016. Data augmentation for time series classification using convolutional neural networks. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases Workshop on Advanced Analytics and Learning on Temporal Data*.

Lee, H.; Hwang, S. J.; and Shin, J. 2020. Self-supervised label augmentation via input transformations. In *International Conference on Machine Learning*, 5714–5724. PMLR.

Ma, H.; Li, W.; Zhang, X.; Gao, S.; and Lu, S. 2019. AttnSense: Multi-level attention mechanism for multimodal human activity recognition. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 3109–3115. AAAI Press.

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9: 2579–2605.

Nikolaidis, K.; Kristiansen, S.; Goebel, V.; Plagemann, T.; Liestøl, K.; and Kankanhalli, M. 2019. Augmenting physiological time series data: A case study for sleep apnea detection. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 376–399.

Perslev, M.; Jensen, M.; Darkner, S.; Jennum, P. J.; and Igel, C. 2019. U-Time: A fully convolutional network for time series segmentation applied to sleep staging. In *Advances in Neural Information Processing Systems*, 4417–4428.

Steven Eyobu, O.; and Han, D. S. 2018. Feature representation and data augmentation for human activity classification based on wearable IMU sensor data using a deep LSTM neural network. *Sensors* 18(9): 2892.

Tanfous, A. B.; Drira, H.; and Amor, B. B. 2019. Sparse coding of shape trajectories for facial expression and action recognition. *IEEE transactions on pattern analysis and machine intelligence* 42(10): 2594–2607.

Trentin, E.; Scherer, S.; and Schwenker, F. 2015. Emotion recognition from speech signals via a probabilistic echo-state network. *Pattern Recognition Letters* 66: 4–12.

Um, T. T.; Pfister, F. M.; Pichler, D.; Endo, S.; Lang, M.; Hirche, S.; Fietzek, U.; and Kulić, D. 2017. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, 216–220.

Wang, Z.; Yan, W.; and Oates, T. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 1578–1585. IEEE.

Yang, J.; Nguyen, M. N.; San, P. P.; Li, X. L.; and Krishnaswamy, S. 2015. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 3995–4001.

Yang, Y. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval* 1(1-2): 69–90.

Yoon, J.; Jarrett, D.; and Der Schaar, M. V. 2019. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, 5508–5518.