Multi-Task Classifier Sharing for Cross-Dataset Time-Series Classification

Yujing Wang[1,2*], Pingping Lin[1*], Zhuo Li[1,2], Congrui Huang[1], Bixiong Xu[1], Yunhai Tong[2]
[1]Microsoft, [2]Peking University
{yujwang,pinlin}@microsoft.com, lizhuo@stu.pku.edu.cn, {conhua,bix}@microsoft.com, yhtong@pku.edu.cn

ABSTRACT

Multi-task learning has shown its advantages in NLP and CV domains but has not been widely applied for time-series classification. In this paper, we propose a novel multi-task learning approach for time-series classification based on intelligent classifier sharing. Based on the proposed method, we demonstrate that a time-series classification task can benefit from semantically relevant ones automatically without human intervene. Experiments are carried out on the 15 largest datasets in UCR repository. Overall, the average accuracy of 15 datasets has been enhanced from 86.8% to 88.2% compared to the best state-of-the-art model in the literature. Analyses also verify that the task relations captured by our Multi-Task Classifier Sharing (MTCS) method present good interpretability.

ACM Reference Format:
Yujing Wang[1,2*], Pingping Lin[1*], Zhuo Li[1,2], Congrui Huang[1], Bixiong Xu[1], Yunhai Tong[2]. 2022. Multi-Task Classifier Sharing for Cross-Dataset Time-Series Classification. In *Proceedings of AMLTS22*. ACM, Atlanta, GA, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Time-series classification is a ubiquitous task in various business scenarios. For example, oil companies monitor the status of their oil rigs by collecting time series from thousands of sensors and getting alerts before incidents happen. In real scenarios, it is often hard to acquire sufficient training labels for a specific task, which hinders the accuracy of the corresponding time-series classifier. Therefore, it is a natural requirement to train a unified time-series classification model, where multiple tasks can collaborate with each other to improve the performance of individual ones.

Existing research works on time-series classification mainly focus on the improvement of representation learning. For example, T-Loss [10] leverages the triplet loss to minimize the representation distances between similar samples while maximizing them between dissimilar samples. TS2Vec [23] provides a universal framework for learning time-series representations in different granularities and semantic levels through hierarchical contrastive loss. However, to the best of our knowledge, all these models are trained by individual datasets separately while ignoring the useful information in other datasets. In this paper, we aim to study if multiple datasets of time-series can benefit from each other. A straightforward solution is to pre-train a self-supervised representation model on all datasets, and then perform time-series classification based on the unified representations. However, the performance of cross-dataset classification becomes even worse when representing all time series jointly. We conjecture that the severe negative transfer phenomenon is caused by the unawareness of task-oriented semantics in each downstream task. Presumably, a multi-task learning framework can incorporate the semantics of different tasks with diverse time-series distributions, which would improve the effectiveness of knowledge sharing among different datasets.

Multi-task learning has achieved tremendous success in many fields including natural language understanding and computer vision, yet few successful stories have been carried out in the time-series domain, partially due to the diverse semantics of time series. Following common practices in other domains, like MT-DNN [13] for NLP, a straightforward way is to leverage a shared encoder for all time series and adopt a separate classifier for each task. But this design has two limitations. (1) All classifiers are trained separately without sharing information with each other; (2) It is hard for the shared encoder to incorporate all kinds of features for a large number of classification tasks with diverse semantics and data distributions. Thus, we argue that only sharing encoders is insufficient, while sharing the information between classifiers is also crucial to the success of cross-dataset time-series classification.

To this end, we propose a novel Multi-Task Classifier Sharing (MTCS) framework for cross-dataset time-series classification. Unlike a traditional multi-task learning (MTL) architecture where each task corresponds to a separate classifier, the classifiers in the MTCS framework share knowledge collaboratively. To achieve this goal, we calculate an affinity matrix between predictive label logits of the target task and all rest ones. Based on the affinity matrix, MTCS model smartly predicts which pairs of labels are semantically relevant and establishes a soft weight-sharing loss to bridge corresponding parameters in multiple classifiers. Extensive experiments demonstrate the effectiveness of our proposed model. We show that MTCS improves an average of 1.6% accuracy on the largest 15 UCR datasets compared to the best state-of-the-art model.

The contributions of this paper are three-fold. First, we propose a Multi-Task Classifier Sharing (MTCS) framework for cross-dataset time-series classification. The result establishes a new state-of-the-art accuracy on the UCR top-15 datasets. Second, it is non-trivial to share knowledge between two distinct classifiers with disjoint label spaces. In this paper, we propose a novel method that calculates the affinity score between each pair of labels and leverages an MSE loss to facilitate soft weight sharing between corresponding classifiers. Third, Visualization analysis illustrates the effectiveness of derived affinity matrices. To the best of our knowledge, this is the first work that proposes parameter sharing among different classifiers with disjoint label spaces. The essence is generic, which may be beneficial to other multi-task classification applications.

*Equal contribution.

AMLTS22, October 24, 2022, Atlanta, GA, USA
2022. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORKS

Time-series classification. Time-series classification is a basic yet important task with numerous real-life applications. Over the last decades, many works have been proposed to handle this task. Among them, One-nearest neighbor classifier with Dynamic Time Wrapping (DTW) distance metric is a common baseline algorithm [16–18]. Deep neural networks have also been investigated for time-series classification in recent years and achieve competitive performance [9, 12]. Besides, ensembling the outputs of several independently trained models into a single prediction is an effective technique to get a higher accuracy score, such as COTE [2] and HIVE-COTE [1].

Time-series representation learning. The performance of time-series classification mainly depends on the effectiveness of time-series representations. There are various attempts for improving time-series representation in recent years. T-Loss [10] exposes an encoder-only architecture and designs a triplet loss to make the encoder unsupervised. TNC [19] assumes that close time steps in a time series have similar representations and proposes a temporal neighborhood sampling method. TST [24] applies a transformer model to learn time-series representations, while TS-TCC [8] proposes a novel temporal contrasting module to learn robust temporal representations. TS2Vec [23] provides a universal framework for learning time-series representations in different granularities and semantic levels through hierarchical contrastive loss. These models generate good representations of time-series and improve the performance of downstream tasks like clustering or classification. However, they are trained separately for each individual dataset, ignoring the abundant information of common characteristics of time series from different sources.

Multi-task learning and classifier sharing. Multi-task learning (MTL) has been applied across many applications of machine learning, including natural language processing [4], speech recognition [6] and computer vision [11]. MTL aims to learn an inductive transfer, which is provided by auxiliary tasks and losses. As a result, models can generalize better to multiple tasks or data distributions through multi-task learning. Existing MTL methods usually utilize hard or soft parameter sharing techniques to handle multiple tasks in one model. Hard parameter sharing is the most commonly used approach [3]. In this setting, most parts of the model are shared between all tasks, while a task-specific output layer is held out for each task. On the other hand, in soft parameter sharing, each task has its own model and parameters. Meanwhile, a regularization objective is applied to pull the parameters of relevant tasks to be closer, for instance, using an l_2 norm [7] or trace norm [21]. To the best of our knowledge, most existing works study the setting of sharing encoders instead of sharing classifiers.

3 METHODOLOGY

3.1 Overall Pipeline

The overall pipeline of Multi-Task Classifier Sharing (MTCS) for cross-dataset time-series classification is illustrated in Figure 1, which consists of two major stages, namely *encoding stage* and *classification stage*. In the encoding stage, time-series from multiple datasets are fed into a shared encoding layer. Next, in the classification stage, the model predicts classification labels for each task based on a task-specific MLP classifier, and the corresponding supervised signals of downstream tasks are leveraged to train both classifiers and encoders from end to end. Unlike the traditional Multi-Task Learning (MTL) networks which leverage an isolate MLP classifier for each task, MTCS introduces a classifier sharing mechanism that performs weight sharing across multiple classifiers. To enable weight sharing among different classifiers with disjoint label spaces, we propose a soft weight-sharing strategy based on label relevance, which encourages the corresponding MLP parameters for relevant labels to be similar. To estimate the relevance score between a pair of labels, each time series is fed into all MLP classifiers, while the cosine similarity of corresponding logits is taken as a proxy of label relevance. Then, a Mean Squared Error (MSE) loss can be added to bridge the classifier parameters of relevant labels. Finally, the whole MTCS network can be optimized by the cross-entropy and MSE losses jointly.

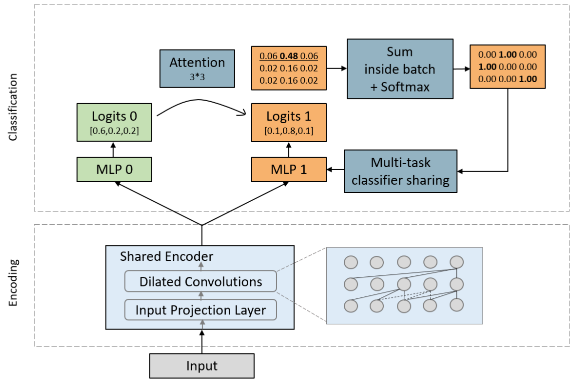

Figure 1: Overall framework of Multi-Task Classifier Sharing (MTCS) for cross-dataset time-series classification.

3.2 TS2Vec Encoder

We adopt TS2Vec [23], a state-of-the-art model for single-dataset time-series classification as the backbone architecture of encoding layer. As shown in Figure 1, the TS2Vec encoder consists of an input projection module and a dilated Convolutional Neural Network (CNN) [22] with L sequentially connected residual blocks, where the l-th block has a dilation factor 2^l. First, the input projection module maps each input timestamp into a high-dimensional vector; Second, the L-layers dilated CNN architecture is applied to each time series input, resulting in a contextual representation for each timestamp. At last, the representations for all timestamps are fed into a max pooling layer, which generates an instance-level representation for the entire time series.

3.3 Multi-Task Classifier Sharing

On the basis of time-series representations derived by the encoder, we stack a task-specific MLP (Multi-Layer Perceptron) classifier to predict the classification labels for each task. Without loss of generality, the number of layers in the MLP classifier is set as 1, and the methodology can be extended to multi-layer MLP classifiers with little modifications. For the k-th task with N possible labels, the label probabilities of instance x_i can be calculated by

$$p_k(\mathbf{c}|x_i) = \text{softmax}(\mathbf{W}_k f(x_i)), \tag{1}$$

where $\mathbf{c} = \{c_1, c_2, ..., c_N\}$ is the set of possible labels; $f(x_i) \in \mathbb{R}^D$ is the encoding representation for instance x_i; and $W_k \in \mathbb{R}^{D\times N}$ is the learnable matrix for the classifier.

Given ground truth labels, the probabilities for each class can be learned by the cross-entropy loss function, where $\mathbb{1}(x_i, c_j)$ indicates if x_i falls into the class c_j.

$$\ell^k_{cls}(x_i) = -\sum_{c_j\in\mathbf{c}} \mathbb{1}(x_i, c_j)\log(p_k(c_j|x_i)), \tag{2}$$

It is non-trivial to share knowledge effectively among multiple classifiers, as different classifiers hold disjoint label spaces. To address this challenge, we propose a soft weight-sharing strategy by making corresponding MLP parameters similar between relevant labels. Specifically, we assume $W_k \in \mathbb{R}^{D\times N}$ and $W_j \in \mathbb{R}^{D\times M}$ are weight matrices for the classifiers of the k-th and j-th tasks respectively. $W_{k,q}$ denotes the q-th component of W_k, which corresponds to the weight of q-th label for the k-th task. Similarly, $W_{j,p}$ corresponds to the p-th label for the j-th task. If the labels p and q are semantically relevant, we utilize the loss function below to facilitate soft parameter sharing between them.

$$\ell^{(k,j)}_{ws} = \frac{1}{MN}\sum_{p\in P}\sum_{q\in Q} \mathbb{1}(q,p)(W_{k,q} - W_{j,p})^2, \tag{3}$$

where P is the label space of the j-th task, and Q is the label space of the k-th task; M and N are the possible label counts for the j-th and k-th tasks, respectively; $\mathbb{1}(q,p)$ indicates whether two labels q and p are relevant in semantics.

The rest challenge is how to estimate the semantic relevance between two disjoint labels automatically. In this paper, we propose a data-driven method based on the classification logits. In other words, the relevance of two labels can be calculated by their corresponding distributions of label probabilities for the same input. Suppose the input time series be x_i, the predicted probabilities for the j-th task be $\mathbf{A}_j(x_i) = \{a_{j,p}|p \in P\} \in \mathbb{R}^M$, and those for the k-th task be $\mathbf{A}_k(x_i) = \{a_{k,q}|q \in Q\} \in \mathbb{R}^N$. We first calculate the outer product of these two vectors, and then sum them for all time-series instances inside a given batch,

$$\mathbf{A} = \frac{1}{|X|}\sum_{x_i\in X} \mathbf{A}_j(x_i) \otimes \mathbf{A}_k(x_i) = \{A_{p,q}\} \in \mathbb{R}^{M\times N}, \tag{4}$$

where X contains all instances in the current batch, and $|X|$ is the batch size; $\mathbf{A}_{p,q}$ denotes the relevance score between label p in the j-th task and label q in the k-th task. Finally, we adopt a pre-defined threshold τ to determine if p and q are relevant.

$$\mathbb{1}(p,q) = \begin{cases} 1, & \text{if } \frac{MN\cdot \mathbf{A}_{p,q}}{\sum_{i,j}\mathbf{A}_{i,j}} > \tau, \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where M and N correspond to the x-th and y-th dimension of matrix $\mathbf{A}$, respectively. In the equation above, we consider two labels relevant if their relevance score is τ times larger than the average relevance between all pairs of labels. By default, we set τ as 2 in all experiments. Notably, if no labels are relevant, the corresponding relevance matrix $\mathbf{A}$ should hold an approximately uniform distribution after summarizing all instances in a batch. In this case, the classifier sharing loss becomes zero and does not affect the final performance.

3.4 Optimization

Loss Function. Overall, the loss function in the multi-task training stage can be formulated as

$$\ell = \sum_{i\in Z}\ell^i_{cls} + \alpha\sum_{k\in Z}\sum_{j\neq k}\ell^{(k,j)}_{ws}, \tag{6}$$

where Z is the set of all tasks, ℓ_{cls} and ℓ_{ws} stands for the classification and weight sharing losses defined in Equation 2 and 3, respectively; The hyper-parameter α is used to balance the importance of two losses. Note that when the joint label space for all tasks is extremely large, we need a prior of task relationship to avoid exhaustive enumeration, which will be explored in future works.

Fine-tuning. After multi-task training, a fine-tuning stage is often necessary to achieve a superior performance. In our implementation, we do not fine-tune the encoders. Following existing works [10, 23], we only fine-tune the classifiers to demonstrate the generality of time-series representations learned in the multi-task training stage. We leverage SVM classifiers in the fine-tuning stage as they are adopted by existing state-of-the-art approaches (T-Loss [10] and TS2Vec [23]).

4 EXPERIMENTS

4.1 Datasets and Preprocessing

The UCR Archive [5] is commonly used to evaluate the performance of different approaches in time-series classification. We evaluate the results of our methods compared to other SOTAs on *top 15 largest datasets* in the UCR archive. During preprocessing, we normalize datasets using z-score [23] so that the set of observations for each dataset has zero mean and unit variance. Our framework is able to process multiple tasks without aligning their length, while for a variable-length dataset, we pad all the series to the same length.

4.2 Settings

The dimension of output representation vector is set as 320 in MTCS and all baseline models. In baseline methods (TS2Vec, T-Loss, TNC, TS-TCC, TST and DTW), time-series from different datasets are not learned together. They train one model for each dataset and evaluate it on the corresponding test set. MTL performs multi-task training using TS2Vec as the shared encoder, and MTCS further introduces a classifier sharing strategy among multiple classifiers. For MTCS, MTL and TS2Vec, we set the number of dilated convolution layer as $L = 10$ and hidden dimension as $D = 64$. The hyper-parameter α in MTCS is set as 0.1. First, we shuffle all the time-series data in UCR archive for all datasets and sample a mini-batch from one of the dataset in each iteration. Then, the model is updated with an AdamW [14] optimizer with an initial learning rate of 0.001. The models are implemented with PyTorch and run on one NVIDIA V100 Tensor Core GPU with batch size 32 for 5 epochs.

4.3 Comparison with State-of-the-art Methods

We compare our approach with several state-of-the-art works of time-series representation learning. T-Loss [10] adopts a triplet loss

Table 1: Accuracy comparison on 15 largest datasets in UCR

Dataset	MTCS	MTL	TS2Vec	T-Loss	TNC	TS-TCC	TST	DTW
Crop	**0.766**	0.765	0.756	0.722	0.738	0.742	0.710	0.665
ElectricDevices	0.707	0.702	**0.721**	0.707	0.700	0.686	0.676	0.602
StarLightCurves	**0.978**	0.969	0.969	0.964	0.968	0.967	0.949	0.907
Wafer	0.997	0.996	**0.998**	0.992	0.994	0.994	0.991	0.980
TwoPatterns	**1.0**	0.999	**1.0**	0.999	0.993	0.976	0.871	0.905
ECG5000	**0.941**	0.938	0.935	0.933	0.937	**0.941**	0.928	0.924
FordA	**0.945**	0.944	0.936	0.928	0.894	0.930	0.568	0.555
UWaveGestureLibraryAll	**0.968**	**0.968**	0.930	0.896	0.903	0.692	0.475	0.892
UWaveGestureLibraryX	**0.828**	**0.828**	0.795	0.785	0.781	0.733	0.569	0.728
UWaveGestureLibraryY	**0.764**	0.762	0.719	0.71	0.697	0.641	0.348	0.634
UWaveGestureLibraryZ	0.782	0.778	0.770	0.757	0.721	0.692	0.475	**0.892**
FordB	**0.815**	0.812	0.794	0.793	0.733	**0.815**	0.507	0.620
ChlorineConcentration	0.827	0.768	**0.832**	0.750	0.760	0.753	0.562	0.648
NonInvasiveFetalECGThorax1	**0.961**	0.950	0.930	0.878	0.898	0.898	0.471	0.790
NonInvasiveFetalECGThorax2	0.951	**0.954**	0.938	0.919	0.912	0.913	0.832	0.865
15 datasets Avg.	**0.882**	0.876	0.868	0.849	0.842	0.825	0.662	0.774

Table 2: Accuracy comparison on 10 selected datasets in UCR

Dataset	MTCS	MTL	TS2Vec
Lightning2	**0.934**	0.918	0.869
Lightning7	0.849	0.849	**0.863**
EOGHorizontalSignal	**0.616**	0.588	0.539
EOGVerticalSignal	**0.530**	0.475	0.503
Chinatown	0.977	**0.980**	0.965
MelbournePedestrian	0.957	0.944	**0.959**
SmallKitchenAppliances	0.731	**0.741**	0.731
LargeKitchenAppliances	**0.885**	**0.885**	0.861
Computers	**0.676**	0.672	0.660
HouseTwenty	**0.933**	0.916	0.916
10 datasets Avg.	**0.809**	0.797	0.787

to distinguish different time-series. TNC [19] retrieves positive samples from time-series signals within a neighborhood. TS-TCC [8] transforms raw time-series data by using weak and strong augmentations before they are fed into encoders. TST [24] learns time-series representations based on the transformer encoder architecture. We also compare to DTW, a one-nearest-neighbor classifier with DTW distance measurement. The classification accuracy scores on the 15 largest dataset in UCR are summarized in Table 1. All existing SOTAs train each dataset separately, while TS2Vec achieves the best performance among them by proposing a hierarchical contrasting loss. MTL trains all 15 datasets jointly by a shared encoder and separate classifiers, which improves the average accuracy of TS2Vec by relatively 0.92%. Moreover, MTCS further improves the average accuracy by 0.68% by the proposed classifier sharing method. As we can see in Table 1, MTCS is globally the best among all state-of-the-arts models on 15 datasets, demonstrating the effectiveness of both cross-dataset training and classifier sharing.

4.4 Analysis

4.4.1 Self-supervised pre-training v.s. Multi-task training. According to TS2Vec [23], the encoding layer can be optimized through a self-supervised hierarchical contrastive loss. For a comparison, we pre-train the time-series representation on 15 datasets together using the self-supervised loss, and then fine-tune the classifier for each task based on the learned representations. All settings follow the open source version of TS2Vec except that the representation model is learned jointly on all 15 datasets. As a result, we only get an 81.2% average accuracy, which shows a huge performance degradation compared to original TS2Vec that learns the representation of each task separately. Instead, we can boost the average accuracy to 88.2% by leveraging the MTCS framwork. This indicates the superiority of multi-task training mechanism for time-series classification tasks, perhaps owing to its ability to intelligently incorporate the semantics of different downstream tasks and suppress negative transfer from irrelevant or conflicting tasks.

4.4.2 Attempt on MMoE. We replace the encoder in MTCS by a network composed of multiple experts like MMoE [15], but do not see any improvement on the 15 largest UCR datasets (0.882 v.s. 0.881). We guess that the capacity of encoder network is already sufficient and multiple experts are unnecessary for this setting.

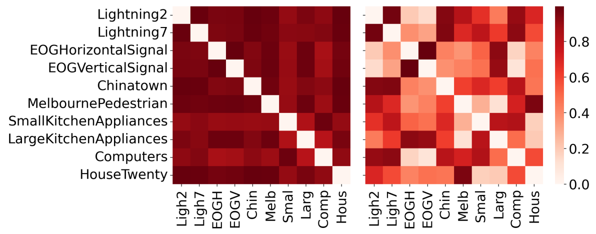

Figure 2: Cross-dataset covariance similarity score on the 10 selected datasets in UCR. The left is from self-supervised pre-training, while the right is from MTCS.

4.4.3 Visualization of task relations. Intuitively, semantically relevant tasks would get more similar time-series representations, and the proposed MTCS model should learn effective task correlations to ensure an effective classifier sharing. To verify this, we follow the covariance similarity proposed by [20] to visualize the learned correlations of each pair of tasks. We select 10 datasets in UCR Archive for analytic purpose, for which we know exact semantics of the original tasks. Results are visualized in Figure 2, where the left one is for self-supervised pre-training and the right one is for MTCS. The darker the cell is, the more relevant the task pair is. It is quite evident that MTCS does capture explainable relevance among these tasks. For instance, *Lightning2* and *Lightning7* are both associated with lightning, and they have higher relevance as predicted by the model. *EOGHorizontalSignal* and *EOGVerticalSignal* are two separate channels of the electrooculography signal, and they are the most relevant to each other as expected. *SmallKitchenAppliances* and *LargeKitchenAppliances* are electricity consumption data from the same households by small and large appliances, respectively, which also share high relevance score. Besides, there are other highly-relevant pairs, like *MelbournePedestrian* and *HouseTwenty*, maybe because they have implicitly similar data distributions. On the contrary, when we perform self-supervised training without supervised signals, the corresponding heatmap (left figure) does not capture reasonable task correlations. This verifies our motivation that task-oriented semantics in supervised labels are crucial for cross-dataset time-series understanding. Moreover, with the effective task relationship, MTCS achieves better performance than MTL by sharing parameters among relevant classifiers (Table 2).

5 CONCLUSION

In this paper, we propose a novel multi-task learning method with intelligent classifier sharing (MTCS) for cross-dataset time-series classification. Experimental results on the 15 largest datasets in UCR repository prove the effectiveness of MTCS, and extensive analyses also demonstrate the explainability of the proposed classifier sharing strategy. In the future work, we plan to explore better ways to capture label correlations, for example, leveraging the covariance similarity proposed in [20]. Moreover, we aim to introduce a constraint of task relationship to avoid exhaustive combination of a large number of tasks.

REFERENCES

[1] Anthony Bagnall, Michael Flynn, James Large, Jason Lines, and Matthew Middlehurst. 2020. On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (hive-cote v1. 0). In *International Workshop on Advanced Analytics and Learning on Temporal Data*. Springer, 3–18.

[2] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. 2015. Time-series classification with COTE: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering* 27, 9 (2015), 2522–2535.

[3] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[4] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. 160–167.

[5] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica* 6, 6 (2019), 1293–1305.

[6] Li Deng, Geoffrey Hinton, and Brian Kingsbury. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 8599–8603.

[7] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*. 845–850.

[8] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. 2021. Time-series representation learning via temporal and contextual contrasting. *arXiv preprint arXiv:2106.14112* (2021).

[9] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep neural network ensembles for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–6.

[10] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. 2019. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems* (2019).

[11] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.

[12] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962.

[13] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504* (2019).

[14] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[15] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1930–1939.

[16] Meinard Müller, Henning Mattes, and Frank Kurth. 2006. An efficient multiscale approach to audio synchronization.. In *ISMIR*, Vol. 546. Citeseer, 192–197.

[17] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (1978), 43–49.

[18] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.

[19] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. 2021. Unsupervised representation learning for time series with temporal neighborhood coding. *arXiv preprint arXiv:2106.00750* (2021).

[20] Sen Wu, Hongyang R Zhang, and Christopher Ré. 2020. Understanding and improving information transfer in multi-task learning. *arXiv preprint arXiv:2005.00944* (2020).

[21] Yongxin Yang and Timothy M Hospedales. 2016. Trace norm regularised deep multi-task learning. *arXiv preprint arXiv:1606.04038* (2016).

[22] Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* (2015).

[23] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. 2022. TS2Vec: Towards Universal Representation of Time Series. *AAAI* (2022).

[24] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2114–2124.