Empirical Evaluation of Data Augmentations for Biobehavioral Time Series Data with Deep Learning

Huiyuan Yang, Han Yu and Akane Sano
Department of Electrical Computer Engineering
Rice University
Houston TX 77005, USA
{hy48, hy29, Akane.Sano}@rice.edu

Abstract

Deep learning has performed remarkably well on many tasks recently. However, the superior performance of deep models relies heavily on the availability of a large number of training data, which limits the wide adaptation of deep models on various clinical and affective computing tasks, as the labeled data are usually very limited. As an effective technique to increase the data variability and thus train deep models with better generalization, data augmentation (DA) is a critical step for the success of deep learning models on biobehavioral time series data. However, the effectiveness of various DAs for different datasets with different tasks and deep models is understudied for biobehavioral time series data. In this paper, we first systematically review eight basic DA methods for biobehavioral time series data, and evaluate the effects on seven datasets with three backbones. Next, we explore adapting more recent DA techniques (*i.e., automatic augmentation, random augmentation*) to biobehavioral time series data by designing a new policy architecture applicable to time series data. Lastly, we try to answer the question of why a DA is effective (*or not*) by first summarizing two desired attributes for augmentations (*challenging* and *faithful*), then utilizing two metrics to quantitatively measure them. The measurement can therefore guide us in the search for more effective DA for biobehavioral time series data by designing more challenging but still faithful transformations. Our code and results are available at this Link.

1 Introduction

Deep learning performs remarkably well in many fields, including computer vision (CV), natural language processing (NLP), and recently time series-related tasks [11, 32, 7]. Those successful applications increasingly inspire researchers to embrace deep learning for solving issues in human-centered applications that use physiological and behavioral time series data. However, the superior performance of deep models relies heavily on the availability of a large number of training data, but unfortunately, many human centered applications (*i.e., healthcare tasks*) usually do not have enough labeled samples, which may limit the wide adaptation of deep models to various computing tasks.

As an effective technique to increase the data variability and thus train deep models with better generalization, data augmentation (DA) is a critical step for the successful applications of deep learning models. While DA can yield considerable performance improvements, they do require domain knowledge and are task- and domain-dependent. For example, image rotation, a likely class-preserving behavior, is designed to rotate the input by some number of degrees. The image's class can still be recognized by humans, thus allowing the model to generalize in a way humans expect it to generalize. However, such an effective random angle-based rotation operation may not be applicable to other domains, *i.e., wearable data*. In addition, searching for the most effective DA methods for a new dataset is very time-consuming, and this motivated the proposal of several automatic DA search algorithms [5, 16, 6, 15, 17].

Workshop on Learning from Time Series for Health, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

The existing DA literature mainly focuses on computer vision, but its application to other domains, *i.e, biobehavioral time series data*, is understudied. A few works investigated the effectiveness of basic data augmentation methods for time series and wearable data [30, 12, 32, 1]. However, those works only investigated the very basic DAs, leaving the more recent DA techniques (*i.e., automatic DA*) unexplored. More importantly, it is still an open question why a data augmentation method works, and how to quantify its effectiveness. Therefore, in this paper, we first systematically review various basic DA methods for biobehavioral time series data, evaluating the effects on different datasets with varied backbones and tasks. Next, we validate the effectiveness of adapting more recent DA techniques (*i.e., automatic DA*) to biobehavioral time series data. Following the DADA[15], we designed a different policy architecture where the operations are differentiable with respect to different time series DA methods. Therefore, the model can be applied to biobehavioral time series data and the DA parameters and deep model weights can be jointly optimized. Lastly, we try to answer the open question of why a DA works(*or not*), by first summarizing two desired attributes (*challenging* and *faithful*) for an effective DA, and then utilizing two metrics to quantitatively measure the two attributes. We find that an effective DA needs to generate challenging but still faithful transformations, which can guide us for the search of more effective DA for biobehavioral time series data.

2 Related Works

Augmentations for Biobehavioral Time Series Data Most of the basic DA methods are borrowed from image or time series data augmentation, such as flipping, cropping and noise addition. These augmentation methods rely on adding random transformations to the training data. Um et al. [30] systematically evaluated six DA methods for wearable sensor data based Parkinson's disease monitoring, and found that the combination of rotational and permutational data augmentation methods improve the baseline performance the most. Ohashi et al. [21] proposed a rotation based DA method for wearable data. Alawneh et al. [1] investigated the benefits of adopting time series DA methods to biomedical time series data. Besides, DA methods have been also used to balance the dataset. For example, Cao et al. [4] used DA methods to balance the number of samples among different categories for automated heart disease detection. However, those related works only investigated the very basic DAs, and the effectiveness of adapting more advanced DAs is not explored yet for biobehavioral time series data. More importantly, the previous works did not explore the question of why a DA is effective, and vice versa.

Automatic Data Augmentation DA can be very useful for the training of deep models, but the success relies heavily on domain knowledge and also the extensive experiments to select the effective DA policies for a target dataset. Otherwise, a model may be negatively impacted by some DA policies [12, 32]. Therefore, it is nontrivial and desired to select the effective DA policy for a new dataset automatically. The pioneering work, AutoAugment [5], formats the process of searching DA as an optimization problem to search for the parameters of augmentation, and follwing work [10, 16, 15, 17, 6] were later proposed to improve the efficiency. Despite the success of automatic DA for computer vision tasks, the adaption to biobehavioral time series data is understudied. As long as we know, [25] in the only work which investigated the automatic differentiable data augmentation for EEG signals. However, our work is different with [25], as we target more diverse types of data and DA methods, and more importantly, we explore to explain why a DA works or not.

Quantitative Effectiveness Measurement for DA Although the effectiveness of DA is well acknowledged, a quantitative evaluation of the effectiveness of DA is still an open question. Currently, the most well-known hypothesis is that effective DA can produce samples from an "overlapping but different" distribution [2, 18], therefore improving generalization by training with the diverse samples. However, the role of distribution shift in training remains unclear. A more recent work [9] studied to quantify how DA improves model generalization, and introduced two measures: *Affinity* and *Diversity*, to predict the performance of an augmentation method. During our experiments, we adopt those two metrics to jointly evaluate the performance of different augmentations.

3 Methods and Experiments

Through the experiments, we aim to answer the following research questions:

R1: *What is the most effective DA method for a given backbone and dataset?*

R2: *What are the factors that impact the selection of DA?*

R3: *What are the general conclusion for DA methods in biobehavioral time series data?*

R4: *Why are some DA methods more effective than the others?*

3.1 Methods

To answer those questions, we first performed extensive experiments with various DA methods on different datasets with different tasks and backbones. Let $\mathcal{T}_\alpha$ denote an augmentation operations parameterized by α, given input data x, this procedure outputs augmented data $\hat{x} = \mathcal{T}_\alpha(x)$, where α controls the magnitude of the operation $\mathcal{T}$. Our augmentation set consists of eight operations: *Jittering, Scaling, Rotation, Permutation, Magnitude Warping, Time Warping, Window Slicing, and Window Warping*, which are widely used DAs drawn from the traditional time series processing literature.

Further, we explored recent DA techniques to automatically search for effective DAs. DADA[15] was originally designed to improve the optimization efficiency for visual data through relaxing the optimization process as differentiable. To make it applicable to biobehavioral time series data and the DA parameters and deep model weights can be jointly optimized, we designed a different policy architecture where the operations are differentiable with respect to different time series DA methods. To avoid the complicated searching procedure of automatic DA a random DA [6] procedure was also adapted in our experiments.

Lastly, we also explored to answer the open questions of why some DA methods are more effective, and vice versa. Following previous work[29], we first summarized the desired attributes (*challenging* and *faithful*) for effective DA, and then we adopted a recent work [9] to quantitatively measure the two attributes respectively. Augmentations are designed to prevent models from over-fitting by increasing the number of samples in the training set. Based on the intuition that more diverse data should be more difficult for a model to fit, the augmented data should be complex and strong enough that a model must learn useful representation to perform the task. We use *challenging* to measure the intuition, which is defined as the ratio of final training loss of a model trained with a given augmentation, and the loss of the model trained on original data. Formally, let τ be an augmentation and $\mathcal{D}_{train}$ and $\mathcal{D}^{'}_{train}$ be the training data and augmented training data respectively. Further, let $\mathcal{L}(\theta|\mathcal{D})$ be the training loss of a model with parameter θ on the training data $\mathcal{D}$. Then, we can define Challenging as:

$$\mathcal{C}(\tau) = \frac{\mathcal{L}(\theta|\mathcal{D}^{'}_{train})}{\mathcal{L}(\theta|\mathcal{D}_{train})}, \tag{1}$$

$$\mathcal{D}^{'}_{train} = \left\{(\tau(x), y)|\forall(x, y) \in \mathcal{D}_{train}\right\} \tag{2}$$

The augmentation should generate challenging augmented data, but must not too challenging that make tasks impossible. For example, the random noise added to the signal are so strong that all the useful information might been destroyed. We use *faithful* to measure the requirement, which is defined as the ratio between the validation accuracy of a model trained on clean data and tested on an augmented validation set, and the accuracy of the same model tested on clean data. More formally, let $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ be the training and validation datasets, and let $\mathcal{D}^{'}_{val}$ be an augmented dataset derived from $\mathcal{D}_{val}$. Further, let $\mathcal{M}(\cdot)$ be a model trained on $\mathcal{D}_{train}$, and $\mathcal{A}_{cc}(\mathcal{M}, \mathcal{D})$ denote the accuracy of the model when evaluated on dataset $\mathcal{D}$. Then, for an augmentation τ, Faithful is defined as:

$$\mathcal{F}(\tau) = \frac{\mathcal{A}_{cc}(\mathcal{M}, \mathcal{D}^{'}_{val})}{\mathcal{A}_{cc}(\mathcal{M}, \mathcal{D}_{val})}, \tag{3}$$

$$\mathcal{D}^{'}_{val} = \left\{(\tau(x), y)|\forall(x, y) \in \mathcal{D}_{val}\right\} \tag{4}$$

3.2 Datasets

We conducted extensive experiments on **seven** biomedical time series datasets, including electrocardiogram (ECG) data: PTB-XL [31] and Apnea-ECG [22]; electroencephalogram (EEG) data: Sleep-EDF (expanded) [13] and MMIDB-EEG [27]; electrodermal activity (EDA) data: CLAS [19]; inertial measurement unit (IMU) data: PAMAP2 [23] and UCI-HAR [24]. Note that due to page limits, we listed more details in the supplementary materials, including the details of the individual dataset and tasks, magnitude range and default value for augmentations, model structures, etc.

Table 1: Performance in terms of 8 different DA methods and three backbones (*MLP, Conv-1D and ResNet-1D*) are reported on 7 datasets. Bold numbers indicate the best performance. The top three most effective DA methods (*if exists*) are colored in gray with different grayscale. Besides, the performance of more advanced techniques, including the automatic DA (*auto aug*) and random DA (*rand aug*), are also reported in the last two columns.

Dataset	Backbone	No Aug	Jittering	Scaling	Rotation	Permutation	Magnitude Warp	Time Warp	Window Slice	Window Warp	Auto Aug	Rand Aug
PTB-XL	MLP	60.29	58.66	61.14	55.45	64.04	60.23	60.41	**66.34**	63.80	62.77	61.86
	Conv-1D	75.00	74.88	**77.91**	71.55	76.63	76.57	64.41	75.91	77.72	78.75	77.91
	ResNet-1D	77.24	77.60	**78.03**	63.98	76.57	76.51	67.07	75.48	75.67	78.21	79.60
Apnea-ECG	MLP	53.57	53.46	54.76	54.63	55.26	54.42	**57.90**	56.19	56.98	57.42	57.84
	Conv-1D	80.14	80.00	73.19	**82.67**	81.27	71.76	78.54	75.94	79.20	79.1	81.33
	ResNet-1D	76.64	76.44	70.64	**77.74**	77.26	70.19	76.28	74.32	76.34	73.52	77.64
Sleep-EDFE	MLP	49.85	50.59	50.69	52.36	**56.27**	50.66	47.76	53.78	53.01	53.68	55.38
	Conv-1D	83.62	83.59	84.31	84.56	**85.31**	85.00	82.80	81.12	84.30	84.44	85.30
	ResNet-1D	83.61	81.55	83.40	84.10	**85.01**	83.09	82.09	82.80	83.98	85.26	85.16
MMIDB-EEG	MLP	77.24	78.10	74.76	50.92	68.39	73.14	78.21	74.43	**78.68**	79.29	78.53
	Conv-1D	79.29	**79.72**	77.13	48.98	75.62	77.67	77.67	74.97	77.99	81.23	81.66
	ResNet-1D	76.16	79.07	**80.15**	49.30	69.58	74.54	79.61	70.98	77.99	79.61	78.53
CLAS	MLP	76.71	75.37	74.51	77.02	**79.06**	76.16	78.59	77.88	77.49	74.93	74.91
	Conv-1D	62.83	66.12	70.59	76.54	73.80	71.37	**77.96**	75.37	73.64	73.26	68.73
	ResNet-1D	**76.54**	71.29	66.82	70.04	71.84	69.10	74.47	66.59	70.18	74.65	69.36
PAMAP2	MLP	61.15	61.45	60.86	13.88	**66.17**	61.74	59.82	62.92	65.14	60.71	54.51
	Conv-1D	89.22	91.58	91.14	67.06	91.58	**92.02**	88.33		91.88	91.88	88.18
	ResNet-1D	89.81	86.85	88.18	65.14	85.82	**92.02**	88.63	88.33	87.59	91.14	87.00
UCI-HAR	MLP	87.89	87.82	86.33	19.41	**90.80**	85.85	82.86	89.96	89.68	86.77	87.48
	Conv-1D	91.92	89.11	91.79	73.91	**93.55**	90.19	89.41	91.75	92.20	92.5	93.86
	ResNet-1D	89.79	87.95	88.63	65.12	91.89	89.62	88.39	90.91	**92.94**	91.72	93.38

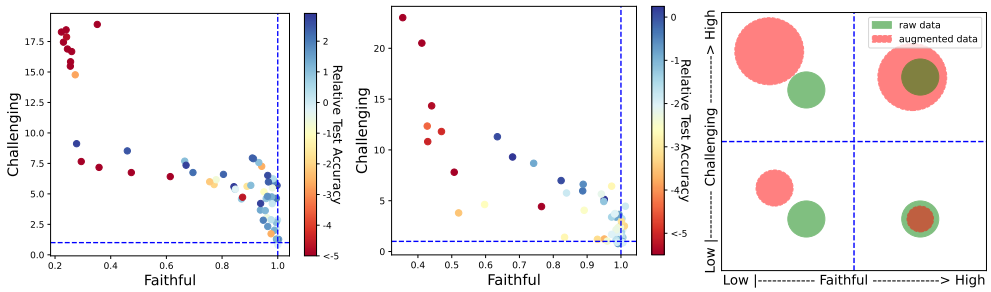
3.3 Experimental Results

Performance evaluation of eight basic DA methods. Experiments were first conducted on seven datasets with three different backbones and eight basic DA methods, and the results are shown in Table.1. We may find that: I). *All the Tasks can benefit from DAs.* (**R1,R3**) Comparing with the models trained *w/o* augmentation, training with proper DA usually can achieve higher performance, and the improvement ranges from 0.5% to 15% over different datasets. II). *The effectiveness of DA depends on many factors.* (**R2**) From the grey colored areas, our first impression is that there is no such a single augmentation method works equally well for all the different datasets. The dataset varies in terms of data type (*e.g., ECG, EEG, EDA and IMU*) and tasks (*sleep quality, heart disease, human activities, etc*). We can find that the effectiveness of an augmentation not only depends on the dataset itself (*data type, task*), but also the selection of backbones. For example, with the same backbone (*i.e., ResNet-1D*) and data type (*ECG*), the top three augmentations for PTB-XL and Apnea-ECG are totally different. *scaling* and *jittering* work the best for PTB-XL dataset, while *rotation* and *permutation* show the highest improvement for the Apnea-ECG dataset. Besides, we also observe that *permutation* appears 13 times in the top three augmentations, which counts for around half of the rows. Therefore, it is suggested to try *permutation* first for the related applications.

Performance evaluation of automatic and practical DA methods. The performance evaluation of automatic and random DA is displayed in the last two columns of Table.1, and we can conclude that I). *Random DA is both effective and efficient* (**R3**). As we can see from Table.1, the performance trained with random DA generally outperforms the baselines (*No Aug*), and are comparable with the best performance across different datasets and backbones. Therefore, random DA is an efficient, effective, and practical automated data augmentation method for biobehavioral time series data. II). *Automatic DA outperforms most comparable baselines* (**R1**). From the various possible DA combinations and their corresponding magnitudes, the automatic DA method helps us search for the more effective DA policies (operations and magnitudes). As we may find that the automatic DA method (*Auto*) generally outperforms the baselines across different datasets and backbones, and achieves comparable or higher performance than the best performance, which is achieved by manually selected DAs.

Quantitative measurement of the effectiveness for different DA methods. (**R4**) We tried to answer the question of why some DA methods are more effective than others, and how can we quantitatively measure the effectiveness of different DA methods? First, we summarized the desired attributes for effective DAs:

1) *challenging*: the augmented data should be complex and strong enough that a deep model must learn useful representations to perform the task.



(a) PTB-XL (b) PAMAP2 (c) Illustration

Figure 1: Augmentation performance is determined by both *faithful* and *challenging*. Test accuracy plotted against each of *faithful* and *challenging* in the PTB-XL dataset (a) and PAMAP2 dataset (b), where each point represents a different DA (65 augmentations in total). Color shows the final test accuracy relative to the baseline model trained without augmentation. (c) Illustration of how raw data and augmented data are associated in terms of the two metrics. A larger circle represents higher diversity while the distributional similarity is depicted trough the overlap of circles. Test accuracy generally improves with both high *faithful* and high *challenging* (*upper right space*).

2) *faithful*: the DA must not make the task impossible, being so strong that they destroy all features of the input. For example, the random noise added to jittering should not be so strong that makes learning impossible from the augmented data.

Next, we use the metrics: $\mathcal{C}(\tau)$ and $\mathcal{F}(\tau)$, to quantify how DA improves model's performance by quantitatively measuring *challenging* and *faithful* for different DAs. As shown in Fig.1 (c), the horizontal axis is used to measure how faithful of a DA, while vertical axis measures the level of challenge resulting from applying an augmentation. Fig.1 (a) and (b) measure both *faithful* and *challenging* across 65 different augmentations for PTB-XL and PAMAP2 dataset. We find that many augmentations that dramatically decrease the performance have low value in *faithful* measurement and high value in *challenging* measurement (upper left). On the other hand, many successful augmentations (*blue points*) lay in the area with high value in *faithful*, and for fixed (high) value of *faithful*, test accuracy generally increases with the increase of *challenging*. In summary, *challenging* and *faithful* together provide an explanation of an augmentation policy's benefit to a model's performance.

In other words, effective DA is a trade-off between generating more diverse data that should be more difficult for a model to fit, while remaining faithful that the learning task is still possible from the augmented data.

4 Conclusion

In this work, we first conducted a comprehensive and systematic evaluation of various basic DAs on different biobehavioral datasets, finding that all the datasets can benefit from DA if using them carefully. The effectiveness of an augmentation depended on both the dataset itself (*data type, task*) and the used backbone. Further, we explored to adopt more recent DA techniques for biobehavioral signals by designing a different policy architecture. The results demonstrated that automatic DA can help learn effective policies but at a cost of high computational complexity, while random DA was demonstrated to be both effective and efficient during our experiments. At last, we attempted to answer the question of why data augmentation is effective (*or not*), by first summarizing two desired attributes (*challenging* and *faithful*), and then two we used two quantitative metrics to measure the corresponding attributes of DA. This can help understand why a DA works (*or not*), and guide us for the search of more effective DA methods in the future.

We hope our experimental results could shed some light on DA for biobehavioral time series data, as a carefully selected DA could outperform many claimed improvements in the literature. Our next plan is to investigate domain knowledge inspired DA, and also develop more efficient way to quantitatively measure the effectiveness of a DA method.

Acknowledgments

This work is supported by NSF #2047296 and #1840167.

References

[1] L. Alawneh, T. Alsarhan, M. Al-Zinati, M. Al-Ayyoub, Y. Jararweh, and H. Lu. Enhancing human activity recognition using deep learning and time series augmented data. *Journal of Ambient Intelligence and Humanized Computing*, 12(12):10565–10580, 2021.

[2] Y. Bengio, F. Bastien, A. Bergeron, N. Boulanger-Lewandowski, T. Breuel, Y. Chherawala, M. Cisse, M. Côté, D. Erhan, J. Eustache, et al. Deep learners benefit more from out-of-distribution examples. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 164–172. JMLR Workshop and Conference Proceedings, 2011.

[3] R. B. Berry, R. Budhiraja, D. J. Gottlieb, D. Gozal, C. Iber, V. K. Kapur, C. L. Marcus, R. Mehra, S. Parthasarathy, S. F. Quan, et al. Rules for scoring respiratory events in sleep: update of the 2007 aasm manual for the scoring of sleep and associated events: deliberations of the sleep apnea definitions task force of the american academy of sleep medicine. *Journal of clinical sleep medicine*, 8(5):597–619, 2012.

[4] P. Cao, X. Li, K. Mao, F. Lu, G. Ning, L. Fang, and Q. Pan. A novel data augmentation method to enhance deep neural networks for detection of atrial fibrillation. *Biomedical Signal Processing and Control*, 56:101675, 2020.

[5] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

[6] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[7] J. C. B. Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

[8] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[9] R. Gontijo-Lopes, S. Smullin, E. D. Cubuk, and E. Dyer. Tradeoffs in data augmentation: An empirical study. In *International Conference on Learning Representations*, 2021.

[10] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pages 2731–2741. PMLR, 2019.

[11] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[12] B. K. Iwana and S. Uchida. An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):e0254841, 2021.

[13] B. Kemp, A. Zwinderman, B. Tuk, H. Kamphuisen, and J. Oberyé. Sleep-edf database expanded. 2018.

[14] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. Kamphuisen, and J. J. Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg. *IEEE Transactions on Biomedical Engineering*, 47(9):1185–1194, 2000.

[15] Y. Li, G. Hu, Y. Wang, T. Hospedales, N. M. Robertson, and Y. Yang. Dada: differentiable automatic data augmentation. *arXiv preprint arXiv:2003.03780*, 2020.

[16] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim. Fast autoaugment. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] A. Liu, Z. Huang, Z. Huang, and N. Wang. Direct differentiable augmentation search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12219–12228, 2021.

[18] Z. Liu, H. Jin, T.-H. Wang, K. Zhou, and X. Hu. Divaug: Plug-in automated data augmentation with explicit diversity maximization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4762–4770, 2021.

[19] V. Markova, T. Ganchev, and K. Kalinkov. Clas: A database for cognitive load, affect and stress recognition. In *2019 International Conference on Biomedical Innovations and Applications (BIA)*, pages 1–4. IEEE, 2019.

[20] F. Moya Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst, and M. Ten Hompel. Convolutional neural networks for human activity recognition using body-worn sensors. In *Informatics*, volume 5, page 26. Multidisciplinary Digital Publishing Institute, 2018.

[21] H. Ohashi, M. Al-Nasser, S. Ahmed, T. Akiyama, T. Sato, P. Nguyen, K. Nakamura, and A. Dengel. Augmenting wearable sensor data with physical constraint for dnn-based human-action recognition. In *ICML 2017 times series workshop*, pages 6–11, 2017.

[22] T. Penzel, G. B. Moody, R. G. Mark, A. L. Goldberger, and J. H. Peter. The apnea-ecg database. In *Computers in Cardiology 2000. Vol. 27 (Cat. 00CH37163)*, pages 255–258. IEEE, 2000.

[23] A. Reiss and D. Stricker. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th international symposium on wearable computers*, pages 108–109. IEEE, 2012.

[24] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767, 2016.

[25] C. Rommel, T. Moreau, J. Paillard, and A. Gramfort. Cadda: Class-wise automatic differentiable data augmentation for eeg signals. *arXiv preprint arXiv:2106.13695*, 2021.

[26] K. Roots, Y. Muhammad, and N. Muhammad. Fusion convolutional neural network for cross-subject eeg motor imagery classification. *Computers*, 9(3):72, 2020.

[27] G. Schalk, D. McFarland, T. Hinterberger, N. Birbaumer, and J. Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *IEEE Transactions on Biomedical Engineering*, 51(6):1034–1043, 2004.

[28] A. Supratak and Y. Guo. Tinysleepnet: An efficient deep learning model for sleep stage scoring based on raw single-channel eeg. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 641–644. IEEE, 2020.

[29] A. Tamkin, M. Wu, and N. Goodman. Viewmaker networks: Learning views for unsupervised representation learning. *arXiv preprint arXiv:2010.07432*, 2020.

[30] T. T. Um, F. M. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić. 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*, pages 216–220, 2017.

[31] P. Wagner, N. Strodthoff, R.-D. Bousseljot, D. Kreiseler, F. I. Lunze, W. Samek, and T. Schaeffter. Ptb-xl, a large publicly available electrocardiography dataset. *Scientific data*, 7(1):1–15, 2020.

[32] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478*, 2020.

Appendix

A Basic Data Augmentation Methods

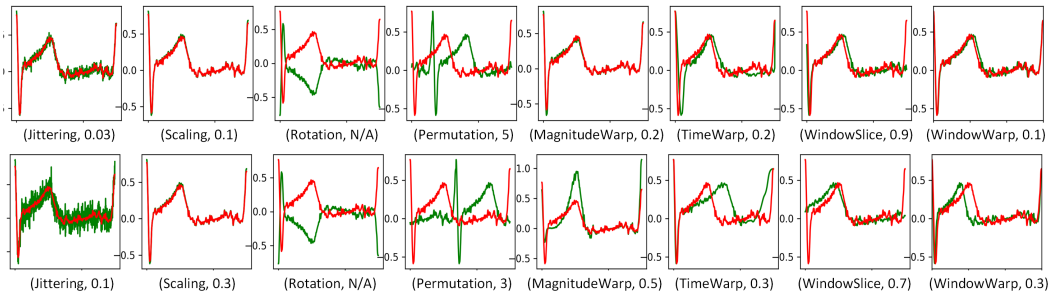


Figure 2: Examples of different data augmentation methods used in the experiments. The red lines indicate the input data and the green lines are the augmented data based on the eight data augmentation operations including *Jittering, Scaling, Rotation, Permutation, Magnitude Warping, Time Warping, and Window Warping* with two different magnitudes.

Data augmentation methods rely on adding random transformations to the training data, and the transformations can be generally classified into three categories: magnitude-based, time-based, and frequency-based transformations. Magnitude-based transformations are applied to the wearable data along the variate or value axes. Time-based transformations change the time steps, and frequency-based transformations warp the frequencies, respectively. During our experiments, we will mainly focus on the magnitude and time-based transformations.

Let $\mathcal{T}_\alpha$ denote an augmentation operations parameterized by α, given input data x, this procedure outputs augmented data $\hat{x} = \mathcal{T}_\alpha(x)$, where α controls the magnitude of the operation $\mathcal{T}$.

We consider a set of data augmentation methods drawn from the traditional time series processing literature. Specifically, our augmentation set consists of **eight** operations: *Jittering, Scaling, Rotation, Permutation, Magnitude Warping, Time Warping, Window Slicing, and Window Warping*.

Given an input data $\mathbf{x} = [x_1, x_2, \ldots, x_T]$ with T, the number of time steps, and each element x_t can be univariate or multivariate.

Jittering. A random noise is added to the input data:

$$\hat{\mathbf{x}} = \mathcal{T}_\alpha(\mathbf{x}) = [x_1 + \epsilon_1, x_2 + \epsilon_2, \ldots, x_T + \epsilon_T] \tag{5}$$

where $\hat{\mathbf{x}}$ is the augmented data, ϵ is a random noise (*i.e.,* Gaussian noise) added to each time step t. Assume $\epsilon \sim \mathcal{N}(0, \sigma^2)$, and the standard deviation $\alpha = \{\sigma\}$ of the added noise is a hyper-parameter that needs to be pre-determined.

Scaling. The magnitude of the data in a window is changed by multiplying a random scalar.

$$\hat{\mathbf{x}} = \mathcal{T}_\alpha(\mathbf{x}) = [\epsilon x_1, \epsilon x_2, \ldots, \epsilon x_T] \tag{6}$$

where ϵ can be determined by a Gaussian distribution $\epsilon \sim \mathcal{N}(1, \sigma^2)$ with $\alpha = \{\sigma\}$ as a hyperparameter.

Rotation. Rotate each element by a random rotation matrix. Although rotating data by a random angle can create plausible patterns for images, it might not be suitable for time series data. A widely used alternative is flipping, which is defined as:

$$\hat{\mathbf{x}} = \mathcal{T}_\alpha(\mathbf{x}) = [-x_1, -x_2, \ldots, -x_T] \tag{7}$$

Permutation. Perturb the location of the data in a single window. It should be noted that permutation operation does not preserve time dependencies. Permutation can be performed in two ways: equal sized segments and variable sized segments. First, the data $\mathbf{x}$ is split into N segments, each of the segment has a length of $\frac{T}{N}$ (*assuming equal sized segments*); then the location of those segments are randomly permuted.

$$\hat{\mathbf{x}} = \mathcal{T}_\alpha(\mathbf{x}) = [segment_{1^*}, \ldots, segment_{N^*}] \tag{8}$$

where $segment_{i^*}$ is the i^*-th segment of the input data $\mathbf{x}$, and $i^* \in [1, N]$. The number of segments $\alpha = \{N\}$ is a hyperparameter to be pre-determined.

Magnitude Warping. Change the magnitude of each sample by convolving the data window with a smooth curve.

$$\hat{\mathbf{x}} = \mathcal{T}_\alpha(\mathbf{x}) = [\gamma_1 x_1, \gamma_2 x_2, \ldots, \gamma_T x_T] \tag{9}$$

where $\gamma_1, \gamma_2, \ldots \gamma_T$ is a sequence created by interpolating a cubic spline function $\mathcal{S}(\mathbf{u})$ with knots $\mathbf{u} = u_1, u_2, \ldots u_I$. I is the number of knots and each knot u_i is sampled from a Gaussian distribution $\mathcal{N}(1, \sigma^2)$, therefore, the operation parameters $\alpha = \{\sigma, I\}$.

Time Warping. Perturb the temporal location by smoothly distorting the time intervals between samples, which is similar to the magnitude warping operation and defined as:

$$\hat{\mathbf{x}} = \mathcal{T}_\alpha(\mathbf{x}) = [x_{\gamma_{(1)}}, x_{\gamma_{(2)}}, \ldots, x_{\gamma_{(T)}}] \tag{10}$$

where $\gamma(\cdot) : i \rightarrow j, and\ i, j \in [1, T]$ is a warping function that warps the time steps based on a smooth curve. The smooth curve is defined by a cubic spline $\mathcal{S}(\mathbf{u})$, which is exactly the same as used in the magnitude warping operation, and the operation parameters $\alpha = \{\sigma, I\}$.

Window Slicing. Slice time steps off the ends of the pattern, which is equivalent to cropping for image data augmentation. The operation is defined as:

$$\hat{\mathbf{x}} = \mathcal{T}_\alpha(\mathbf{x}) = [x_\delta, \ldots, x_t, \ldots, x_{W+\delta}] \tag{11}$$

where $0 \leq W \leq T$ is the size of a window that needs to be pre-determined, and δ is a random integer such that $0 \leq \delta \leq T - W$, and the operation parameters $\alpha = \{W\}$.

Window Warping. Randomly select a window with the length of W from the time series and stretch it by K or contract it by $\frac{1}{K}$. Linear interpolation is used for other part of the time series, so that the output will have equal length to the input. Note that, in this paper we only consider $K = 2$, but other ratios could be used as well. The length of window is a hyper-parameter to be pre-determined and $\alpha = \{W\}$

With different operations and their corresponding magnitude parameters α, the generated augmented data are also different. As shown in Fig.2, the two rows represent the examples of the augmented data where the eight basic data augmentation methods are applied to the same input with two different magnitude parameters α for each operation.

B Datasets

PTB-XL: The PTB-XL [31] dataset is a large dataset containing 21,837 clinical 12-lead electrocardiogram (ECG) records from 18,885 patients of 10 second length, where 52% are male and 48% are female with ages range from 0 to 95 years (median 62 and interquantile range of 22). There are two sampling rates: 100 Hz and 500 Hz, available in the dataset, but in our experiments, only data sampled at 100 Hz are used. The raw ECG data are annotated by two cardiologists into five major categories, including normal ECG (NORM), myocardial infarction (MI), ST/T Change (STTC), Conduction Disturbance (CD) and Hypertrophy (HYP). The dataset contains a comprehensive collection of various co-occurring pathologies and a large proportion of healthy control samples. We experimented classifying all 5 cardiac conditions as learning tasks. Further, to ensure a fair comparison of machine learning algorithms trained on the dataset, we follow the recommended splits of training and test sets, which results in a training/testing ratio of 8/1.

Apnea-ECG: The Apnea-ECG [22] dataset studies the relationship between human sleep apnea symptoms and heart activities (monitored by ECG). This database can be accessed through Physionet[8]. This dataset contains 70 records with a sampling rate of 100 Hz, from where 35 records were divided into training, and the other 35 were divided into the test set. The duration of the records varies from slightly less than 7 hours to nearly 10 hours. The labels were the annotation of each minute of each recording indicating the presence or absence of sleep apnea. Thus, we split the ECG recording into each minute, which was a total of 6000 data points for each separation. We extracted 17233 samples for the training set and 17010 samples for the test set. And the ratio of non-apnea and apnea samples in the training set was 61.49% to 38.51%.

Sleep-EDFE: The Sleep-EDF (expanded) [13] dataset contains whole-night sleep recordings from 822 subjects with physiological signals and sleep stages that were annotated manually by well-trained technicians. In this dataset, the physiological signals, including Fpz-Cz/Pz-Oz electroencephalogram (EEG), electrooculogram (EOG), and chin electromyogram (EMG), were sampled at 100 Hz. We targeted to detect 5 sleep stages. including wakefulness, stage N1, N2, N3, and REM [3]. To model the relationship between the sleep patterns and physiological data, we split the whole-night recordings into 30-second Fpz-Cz ECG segments as in [28], which resulted in a total of 42308 ECG and sleep pattern pairs. We divided 25% of the samples into a testing set according to the order of the subject IDs.

MMIDB-EEG: The MMIDB-EEG dataset [27] studies the relationship between physiological EEG and human body physical/imaginary movement. This dataset contains over 1,500 EEG recordings in 1-2 minutes with a sampling rate of 160Hz from 109 subjects. Each subject performed baseline (eyes open and close) and four tasks, including open and close left or right fist, imagine opening and closing left or right fist, open and close both fists or feet, and imagine opening and closing both fists or both feet. Following [26], we omit the data from 6 subjects due to the incorrect annotations and split the remaining data into 4s segments, which results in 4635 segments in total. Our task focuses on the classification between baseline and physical hand movement. Further, data from 22 subjects, based on the order of subject ids, is split into the test set, and the rest samples are employed as the training set.

CLAS: The CLAS dataset [19] aims to support research on the automated assessment of certain states of mind and emotional conditions using physiological data. The dataset consists of synchronized

Table 2: List of basic data augmentation methods discussed in this paper. Additionally, the range of magnitude for individual operations is also reported. Some operations do not use the magnitude information (e.g. Rotation).

Operation Name	Description	Range of Magnitude	Default Value
Jittering	Add noise to the inputs	[0, 0.2]	0.03
Scaling	changes the magnitude of the data in a window by multiplying a random scalar.	[0, 0.5]	0.1
Rotation	Rotate each element by a random rotation matrix. A widely used alternative is flipping.	N/A	N/A
Permutation	Perturb the location of the data in a single window	[0, 8]	5
Magnitude Warping	Changes the magnitude of each sample by convolving the data window with a smooth curve.	[0, 0.5]	0.2
Time Warping	Perturb the temporal location by smoothly distorting the time intervals between samples.	[0, 0.5]	0.2
Window Slicing	Slice time steps off the ends of the pattern, which is equivalent to cropping for image data augmentation.	[0.5, 1.]	0.9
Window Warping	takes a random window of the time series and stretches it by 2 or contracts it by 1/2	[0, 0.3]	0.1

recordings of ECG, photopletysmogram (PPG), electrodermal activity (EDA), and acceleration (ACC) signals. There are 62 healthy subjects who participated and were involved in three interactive tasks and two perceptive tasks. The perceptive tasks, which leveraged the images and audio-video stimuli, were purposely selected to evoke emotions in the four quadrants of arousal-valence space. In this study, our goal was to use the EDA signal to detect binary high/low stress states that are annotated in arousal-valence space. We processed the raw EDA data with a lowpass Butterworth filter with a cutoff frequency of 0.2 Hz, then split the sequences into 10-second segments. We divided the train/test set in a subject-independent manner and utilized the data from 17 subjects as the test set according to subject ids (> 45).

PAMAP2: The PAMAP2 [23] physical activity monitoring dataset consists data of 18 different physical activities, including household activities (sitting, walking, standing, vacuum cleaning, ironing, etc) and a variety of exercise activities (Nordic walking, playing soccer, rope jumping, etc), performed by 9 participants wearing three inertial measurement units (IMU) and a heart rate monitor. Accelerometer, gyroscope, magnetometer and temperature data are recorded from the 3 IMUs placed on three different locations (1 IMU over the wrist on the dominat arm, 1 IMU on the chest and 1 IMU on the dominant side's ankle) with sampling frequency 100Hz. Heart rate data are recorded from the heart rate monitor with sampling frequency 9Hz. The resulting dataset has 52 dimensions (3 x 17 (IMU) + 1 (heart rate) = 52). Following the same setting as used in [20, 29], we linearly interpolated the missing data (upsampling the sampling frequency from 9Hz to 100Hz for heart rate), then took random 10s windows from subject recordings with an overlap of 7s, using the same train/validation/test splits. As mentioned in [20, 29], 12 of the total 18 different physical activities are used in the experiments.

UCI-HAR: The UCI-HAR [24] dataset was collected from a group of 30 volunteers with an age range from 19 to 48 years. During the data collection, all the subjects wore a smartphone (Samsung Galaxy S II) with embedded inertial sensors around their waist and were instructed to follow an activity protocol performing six basic activities, including three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs). The dataset also included postural transitions that occurred between the static postures, but they were discarded in our experiments (also in related works), due to a much smaller size for those transitions. 3-axial linear acceleration and 3-axial angular velocity were captured using the embedded accelerometer and gyroscope of the smartphone at a constant rate of 50Hz. Following the authors' suggestion, the sensor signals were sampled in a fixed-width sliding windows of 2.56 second and 50% overlap, resulting in 128 readings per window.

Table 3: Details of datasets used in the experiments.

Dataset	Data type	# Subjects	# Channels	# Length	# Train	# Test	# Classes	Task
PTB-XL [31]	ECG	18885	1	1000	14618	1652	5	cardiac condition classification
Apnea-ECG [22]	ECG	32	1	6000	17233	17010	2	sleep apnea detection
Sleep-EDF [14]	EEG	22	1	3000	31731	10577	5	sleep stage recognition
MMIDB-EEG [27]	EEG	109	64	640	3708	927	2	movement recognition
CLAS [19]	EDA	62	1	960	993	359	2	stress and affect detection
PAMAP2 [23]	IMU	9	52	1000	4775	677	12	human activity recognition
UCI-HAR [24]	IMU	30	9	128	7352	2947	6	human activity recognition

C Details of Data Augmentation Methods Used in This Paper

C.1 Random Augmentation

The final algorithm is defined as: $randaugment(J, M)$, where J is the number of augmentation transformations to be selected from the K operation pool, M is the magnitude for all the transformations, then the J operations with magnitude M will be sequentially applied to the input data.

C.2 Differentiable Data Augmentation

A collection of DA policies contains K sub-policies $\mathbf{P}_k, k \in [1, K]$, and each sub-policy $\mathbf{P}_k$ includes N basic data augmentation operations (*i.e., jittering, scaling, rotation*) that are applied to input signal sequentially. Each operation can be represented as $\mathbf{O}_k^i(\mathbf{x}; p_k^i, m_k^i)$, $k \in [1, K]$, $i \in [0, J]$, where K is the total number of sub-policies, J is the operations included in each sub-policy, p_k^n is the probability of applying the operation and m_k^n represents the magnitude of the operation. The objective is to jointly optimize a model's parameter θ and also the $\alpha = \{p, m\}$, thus the trained deep models can generalize well on the testing dataset. Due to the page limits, readers are encouraged to check more details in the original paper of DADA [15]. The details of the training steps for the automatic differentiable data augmentation method is illustrated in Algorithm.1, where $\mathcal{L}(\theta|\mathcal{D})$ represents the cross-entropy loss on the data $\mathcal{D}$.

D Implementation Details

We use three deep learning architectures, including MLP, Conv-1d, and ResNet-1d. These networks were chosen due to being effective in time series data and also being used in a wide range of biomedical applications.

We use an Adam optimizer with initial learning rate of 1×10^{-3}, and the learning rate is decayed by 0.9 after every 5 epoches. The batch size is 100, and we train the model for 50 epochs. Our model is implemented in the PyTorch deep learning framework, and is trained and tested on the NVIDIA GeForce 3090Ti GPU. To evaluate the performance, average accuracy of three runs is reported. For fair comparison, we use the default value as magnitude for different basic augmentations on different datasets, therefore we can guarantee that the varied performance is caused by augmentation rather than fine-tuning. For random data augmentation, we set $J = 2$, $K = 8$ and magnitude $M = 12$ during our experiments. For automatic data augmentation, $K = 14$ sub-policies are randomly generated from the 8 basic data augmentation methods, and each sub-policy contains $J = 2$ operations.

E Model Architecture

The detailed information of the Conv-1D and ResNet-1D are illustrated in Table. 4 and Table. 4.

Algorithm 1 The training steps for the automatic differentiable data augmentation method

Input: Training set $\mathcal{D}_{train} = \{x_i, y_i\}_{i=1}^{N}$, $\mathcal{D}_{valid} = \{x_i, y_i\}_{i=1}^{M}$, training epochs E, batch size B, Model parameters θ, DA policy parameters $\mathcal{T}_\alpha$, learning rate ξ_1, ξ_2 and ϵ

Output: Model Parameters θ and DA parameters α

for *epoch* $\leftarrow$ *1 to* E **do**
  for *batch* $\leftarrow$ *1 to max_iter* **do**
    //Randomly sample two batches
    Randomly sample $D_{train}^{batch}, D_{valid}^{batch}$ from training set $\mathcal{D}_{train}, \mathcal{D}_{valid}$ respectively.
    //Compute loss and gradient on training data with DA
    Compute training loss of $\mathcal{L}\big(\theta|\mathcal{T}_\alpha(D_{train}^{batch})\big)$;
    $g_\theta \leftarrow \mathcal{L}\big(\theta|\mathcal{T}_\alpha(D_{train}^{batch})\big).backward(\theta)$;
    $\theta^{'} := \theta - \xi_1 g_\theta$;
    //Compute loss and gradient on validation data without DA.
    Compute validation loss of $\mathcal{L}(\theta^{'}|D_{valid}^{batch})$;
    $g_\theta^{'} \leftarrow \mathcal{L}(\theta^{'}|D_{valid}^{batch}).backward(\theta)$;
    //Calculate gradient g_α with respect to α
    $g_\alpha^{+} \leftarrow \mathcal{L}\big(\theta + \epsilon g_\theta^{'}|\mathcal{T}_\alpha(D_{train}^{batch})\big).backward(\alpha)$
    $g_\alpha^{-} \leftarrow \mathcal{L}\big(\theta - \epsilon g_\theta^{'}|\mathcal{T}_\alpha(D_{train}^{batch})\big).backward(\alpha)$
    $g_\alpha = \frac{(g_\alpha^{+} - g_\alpha^{-})}{2\epsilon}$
    //Update policy parameters α.
    $\alpha := \alpha - \xi_2 g_\alpha$
    //Update model parameters θ.
    Compute training loss of $\mathcal{L}\big(\theta|\mathcal{T}_\alpha(D_{train}^{batch})\big)$;
    $g_\theta \leftarrow \mathcal{L}\big(\theta|\mathcal{T}_\alpha(D_{train}^{batch})\big).backward(\theta)$;
    $\theta := \theta - \xi_1 g_\theta$;

Table 4: Structure of the Conv-1D and ResNet-1D model. *B*: batch size; *L*: length of sequence; *C*: number of channels.

(a) Structure of the Conv-1D model.

Layer Name	Input Shape	Output Shape	Parameter
Reshape	[B, L, C]	[B, C, L]	-
Conv1d-1	[B, C, L]	[B, 32, L]	[1x5, 32]
	BatchNorm, ReLU, Max Pooling		
Conv1d-2	[B, 32, L/3]	[B, 64, L/3]	[1x5, 64]
	BatchNorm, ReLU, Max Pooling		
Conv1d-3	[B, 64, L/9]	[B, 128, L/9]	[1x5, 128]
	BatchNorm, ReLU, Max Pooling		
Conv1d-4	[B, 128, L/27]	[B, 256, L/27]	[1x5, 256]
	BatchNorm, ReLU		
Average Pool	[B, 256, L/27]	[B, 256, 1]	-
fc-1	[B, 256]	[B, Classes]	[256xClasses]

(b) Structure of the ResNet-1D model.

Layer Name	Input Shape	Output Shape	Parameter
Reshape Conv1d	[B, L, C] [B, C, L]	[B, C, L] [B, 64, L]	- 1x3, 64, max pool
Layer1_x	[B, 64, L/2]	[B, 64, L/2]	$\begin{bmatrix} 1\times 3, & 64 \\ 1\times 3, & 64 \end{bmatrix} \times 2$
Layer2_x	[B, 64, L/2]	[B, 128, L/4]	$\begin{bmatrix} 1\times 3, & 128 \\ 1\times 3, & 128 \end{bmatrix} \times 2$
Layer3_x	[B, 128, L/4]	[B, 256, L/8]	$\begin{bmatrix} 1\times 3, & 256 \\ 1\times 3, & 256 \end{bmatrix} \times 2$
Layer4_x	[B, 256, L/8]	[B, 512, L/16]	$\begin{bmatrix} 1\times 3, & 512 \\ 1\times 3, & 512 \end{bmatrix} \times 2$
Average pool	[B, 512, L/16]	[B, 512, 1]	-
FC	[B, 512]	[B, Classes]	[512, Classes]

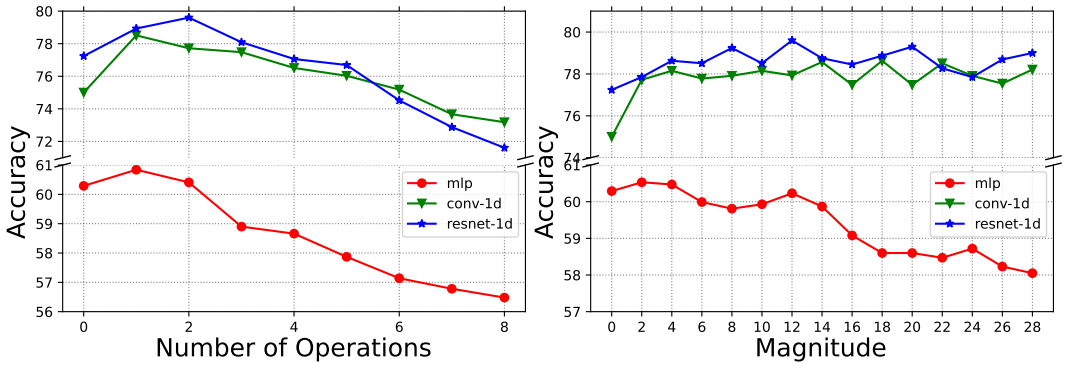
F Supplementary Results

F.1 The effect of different number of operations and different value of magnitude in random DA

The random DA method achieves improved or comparable performances across different tasks and datasets using the fixed number of transformations and fixed value of magnitude. To further study the sensitivity of random DA to the selection of transformations and magnitude, we run experiments of random DA with different number of operations and magnitude on the PTB-XL dataset. The results in Fig.3 (a) (*fixed M*) suggest that the random DA improves performance as the number of operations is increased, even with only 1 operation ($J = 1$). However, the performance is dropped

once $J > 4$, which is potentially caused by the fact that the augmented data is too challenging for the model to learn anything meaningful after J consecutive augmentations, therefore, leading to dropped performance.

A constant magnitude M sets the distortion magnitude to a constant number during training. To validate the impact of different magnitudes, we run experiments with fixed $J = 2$ and varied numbers in terms of magnitude. The results in Fig.3 (b) suggest that the model is not very sensitive to the change of magnitude (*except MLP, as it is a relatively weak backbone*). That is because first, the magnitude is limited in a small reasonable range for all the operations (*details in supplementary materials*); and second, the composition of J random consecutive operations could potentially alleviate the sensitivity to the different magnitudes.



(a) Number of operation (b) Magnitude

Figure 3: Performance when number of operation (a) and magnitude (b) is changed on the PTB-XL dataset.

F.2 Visualization of policies in automatic DA

An illustration of the development of probability (*normalized*) for individual DA policy over training epochs is shown in Fig.4, where all the 14 sub-policies start from the same probability, and change with the training epochs. For example, the probability is gradually increased for *sub-policy_1*, *sub-policy_5* and *sub-policy_14*, indicting those policies are more effective than others. Although automatic differentiable DA shows improved performance, it comes at a cost in terms of computational complexity, which is roughly four times than the corresponding baseline models.

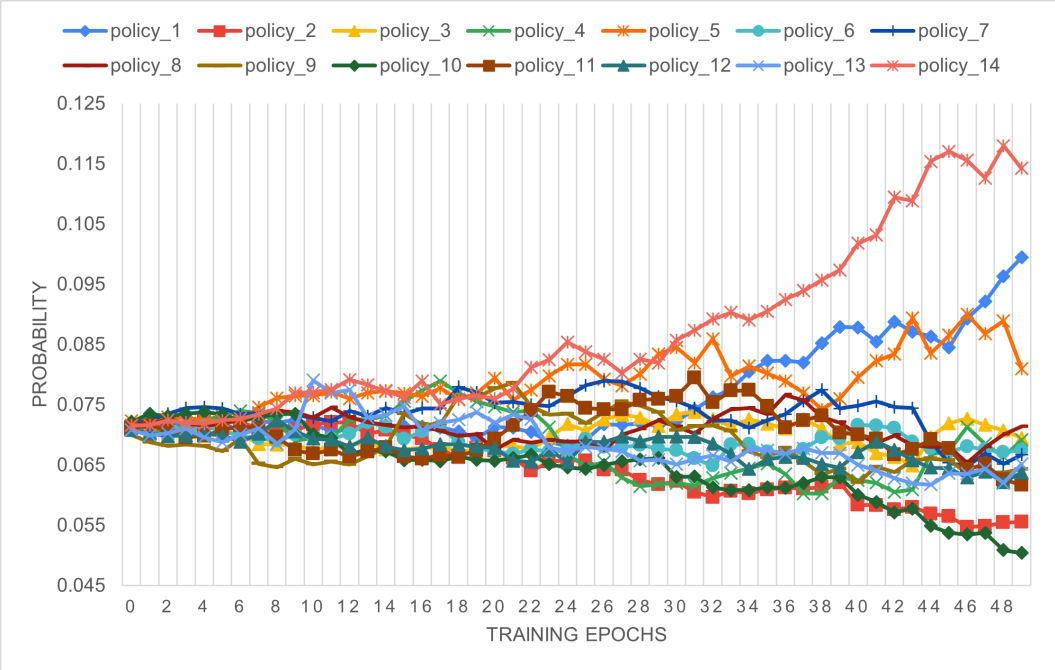


Figure 4: The development of probability parameters for each data augmentation policy in the PTB-XL dataset. All policies start with the same probability, and are increased or decreased in terms of probability during training. Higher probability means the policy has much higher chance to be selected, indicting the corresponding policy is more effective.