TSA on AutoPilot: Self-tuning Self-supervised Time Series Anomaly Detection

Boje Deforce*
KU Leuven
Leuven, Belgium

Meng-Chieh Lee
Carnegie Mellon University
Pittsburgh, USA

Bart Baesens
KU Leuven - University of Southampton
Leuven, Belgium - Southampton, UK

Estefanía Serral Asensio
KU Leuven
Leuven, Belgium

Jaemin Yoo
KAIST
Seoul, South Korea

Leman Akoglu
Carnegie Mellon University
Pittsburgh, USA

Abstract

Time series anomaly detection (TSAD) finds many applications such as monitoring environmental sensors, industry KPIs, patient biomarkers, etc. A two-fold challenge for TSAD is a versatile and unsupervised model that can detect various *different types* of time series anomalies (spikes, discontinuities, trend shifts, etc.) *without any labeled data*. Self-supervised models in particular tackle unsupervised TSAD by transforming the input via various augmentations to create pseudo anomalies for training. However, their performance is sensitive to the choice of augmentation, which is hard to choose in practice, while there exists no effort in the literature on data augmentation tuning for TSAD without labels. Our work aims to fill this gap. We introduce TSAP for *TSA "on autoPilot"*, which can *(self-)tune* augmentation hyperparameters end-to-end. It stands on two key components: a differentiable augmentation architecture and an unsupervised validation loss to effectively assess the alignment between augmentation type and anomaly type. Case studies show TSAP's ability to select the augmentation type and associated hyperparameters.

1 Introduction

Anomaly detection (AD) is a critical task in various domains such as cybersecurity, healthcare, and finance. AD is especially important in time series to ensure system safety and reliability. Thus, there exists a large body of work on time series AD as presented in various surveys [1, 2]. Recent advances in self-supervised learning (SSL) have significantly impacted AD by surpassing traditional unsupervised (or one-class) learning approaches. SSL's key advantage lies in its ability to self-generate labeled samples, or *pseudo* anomalies, enabling a more focused exploration of a plausible subspace rather than an exhaustive, impractical search of the entire space. Central to SSL-based AD are data augmentation functions, which create such pseudo labels. These are then used for the (self-)supervised training of an anomaly detector, such as by predicting whether the input is augmented or not [3], which augmentation function is used [4], or through contrastive learning [5]. The success of these approaches highly depends on how well the augmented data mimics the true anomalies [6].

In this paper we introduce TSAP, a novel approach for SSL-based time series anomaly detection (TSAD) "on autoPilot" equipped with end-to-end augmentation hyperparameter tuning. Tuning both discrete and continuous hyperparameters of augmentation enables TSAP to be an anomaly detector that is most suitable for a given task. We summarize our main contributions as follows:

1. **Problem:** Our work is the first attempt to tune both discrete and continuous hyperparameters of data augmentation in SSL-based TSAD, without labels at training time.

*boje.deforce@kuleuven.be

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

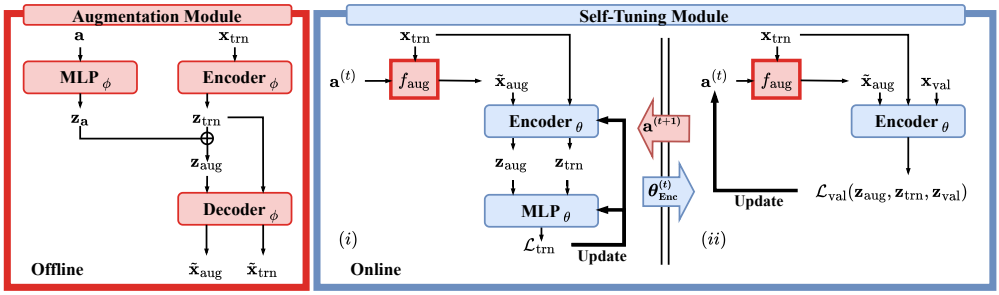

Figure 1: Our TSAP framework for end-to-end self-tuning TSAD. **Left:** Offline trained, differentiable augmentation model $f_{\text{aug}}(\cdot;\boldsymbol{\phi})$ takes as input the normal data and augmentation hyperparameter(s) $\mathbf{a}$, and outputs pseudo-anomalies $\tilde{\mathbf{x}}_{\text{aug}}$. **Right:** Self-tuning engine incorporates the pre-trained f_{aug} (with parameters $\boldsymbol{\phi}$ frozen), alternating between two phases as outlined in Algo. 1.

2. **New TSAD Method:** We propose TSAP[2], which accommodates various time series anomaly types and enables automatic tuning of related hyperparameters (e.g., magnitude, duration) with a differentiable validation loss quantifying alignment between augmented and unlabeled test data.
3. **Effectiveness:** By carefully selecting augmentation type and its (continuous) hyperparameters, TSAP outperforms existing unsupervised and self-supervised approaches, including the SOTA NeuTraL-AD [7] which also employs learnable augmentations.

2 TSAP: Time Series Anomalies on AutoPilot

There are two notable challenges that need to be addressed for automatic selection of both discrete and continuous hyperparameters for SSL-based TSAD:

C1. Differentiable Augmentation: Developing an augmentation function that is differentiable with respect to its hyperparameters, enabling gradient-based optimization.

C2. Comparable Validation Loss: Formulating a validation loss that quantifies alignment between $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{test}}$ while being comparable *across* different hyperparameter initializations.

To address **C1** and **C2**, we center our framework around two main ideas. First, TSAP employs a differentiable augmentation module implemented as an Encoder-Decoder neural network, f_{aug}, parameterized by $\boldsymbol{\phi}$. This module approximates the anomaly-generating mechanism, conditioned on $\mathbf{a} \in \mathcal{A}^P$, $P \geq 1$, where $\mathcal{A}^P$ represents the domain of all possible hyperparameter values (e.g., magnitude, duration, ...). Importantly, this module is pre-trained independently, establishing it as an *offline* component of the framework. Second, at test time *online*, TSAP iteratively refines the detector f_{det}'s parameters $\boldsymbol{\theta}$ as well as augmentation hyperparameters $\mathbf{a}$, through alternating detection and alignment phases. Alignment is performed on part of the unlabeled $\mathcal{D}_{\text{test}}$, referred to as $\mathcal{D}_{\text{val}}$. Based on this, we propose TSAP, a self-tuning self-supervised TSAD framework illustrated in Fig. 1.

2.1 Differentiable Augmentation Module

Anomaly Injection Scheme We carefully consider accommodating six types of anomalies that are common in real-world time series; namely, trend, extremum, amplitude, mean shift, frequency shift, and platform. We provide a detailed description with visual examples of the anomalies in Appx. A. Each type of anomaly has three hyperparameters; including its starting position (`location`), duration (`length`), and severity (`level`). Extremum captures a spike and only has two hyperparameters as its duration is always 1. Based on $\mathcal{D}_{\text{trn}}$, the anomaly generation scheme g creates an augmented dataset $\mathcal{D}_{\text{aug}} = \{g(\mathbf{x}_{\text{trn}};\mathbf{a}) \mid \mathbf{x}_{\text{trn}} \in \mathcal{D}_{\text{trn}}, \mathbf{a} \sim \mathcal{A}^P\}$, where $\mathbf{a}$ is the vector of augmentation hyperparameters that are uniformly randomly sampled from the hyperparameter domain $\mathcal{A}^P$.

Model Design To build an augmentation model f_{aug} for time series, we use a convolutional neural network (CNN) to encode the input time series $\mathbf{x}_{\text{trn}}$ into the feature map $\mathbf{z}_{\text{trn}}$. We then encode the augmentation hyperparameters $\mathbf{a}$ into $\mathbf{z}_{\text{a}}$, which has the same shape as $\mathbf{z}_{\text{trn}}$, with a multilayer perceptron (MLP). Since the feature map $\mathbf{z}_{\text{trn}}$ generated by the CNN encoder keeps the positional information of the original time series, adding $\mathbf{z}_{\text{a}}$ to $\mathbf{z}_{\text{trn}}$ ensures that only the part with the desired `location` and `length` in $\mathbf{z}_{\text{aug}}$ is manipulated. To ensure that the feature maps $\mathbf{z}_{\text{aug}}$ and $\mathbf{z}_{\text{trn}}$ are in the same embedding space, they share the same decoder to reconstruct back to the time series $\mathbf{x}_{\text{aug}} = g(\mathbf{x}_{\text{trn}};\mathbf{a})$

[2]Code and datasets available at: `https://github.com/B-Deforce/TSA-on-autoPilot`.

and $\mathbf{x}_{\text{trn}}$. As such, the loss function of f_{aug} is based on the reconstruction of both $\mathcal{D}_{\text{trn}}$ and $\mathcal{D}_{\text{aug}}$:

$$\mathcal{L}_{\text{aug}} = \sum_{\substack{\mathbf{x}_{\text{trn}} \in \mathcal{D}_{\text{trn}} \\ \mathbf{a} \sim \mathcal{A}^P}} \left((\mathbf{x}_{\text{trn}} - \tilde{\mathbf{x}}_{\text{trn}})^2 + (g(\mathbf{x}_{\text{trn}}, \mathbf{a}) - \tilde{\mathbf{x}}_{\text{aug}})^2 \right) \tag{1}$$

where $\tilde{\mathbf{x}}_{\text{aug}}$ denotes the output of $f_{\text{aug}}(\mathbf{x}_{\text{trn}}; \mathbf{a})$ for the same $\mathbf{a}$ as sampled by g (see Fig. 1 left).

2.2 Self-Tuning Module

Central to TSAP is the self-tuning module, which operates by iteratively refining the detector's parameters, $\boldsymbol{\theta}$, and the augmentation hyperparameters, $\mathbf{a}$. The process is structured into two phases: detection and alignment (see Fig. 1 right). The overall algorithm is given in Algo. 1.

Algorithm 1 Self-Tuning Module of TSAP

Input: Train & val. data $\mathcal{D}_{\text{trn}}$ & $\mathcal{D}_{\text{val}}$, pre-trained aug. model $f_{\text{aug}}(\cdot; \boldsymbol{\phi})$, detector $f_{\text{det}}(\cdot; \boldsymbol{\theta})$, max epoch T, #inner-loops L, and step sizes α and β
Output: Optimized aug. hyperparameters $\mathbf{a}^*$ and optimized parameters $\boldsymbol{\theta}^*$ corresponding to the lowest $\mathcal{L}^*_{\text{val}}$

$\mathbf{a}^{(0)} \sim \text{Uniform}(\mathcal{A}^P)$
for $t = 0, 1, \ldots, T-1$ **do**
  $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{trn}}(\boldsymbol{\theta}, \mathbf{a}^{(t)})$ **for** l **in** L ▷ Phase (i)
  $\mathcal{Z}_{\text{trn}} \leftarrow \{f^{\text{enc}}_{\text{det}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{D}_{\text{trn}}\}$ ▷ Phase (ii) starts
  $\mathcal{Z}_{\text{aug}} \leftarrow \{f^{\text{enc}}_{\text{det}}(f_{\text{aug}}(\mathbf{x}, \mathbf{a}^{(t)})) \mid \mathbf{x} \in \mathcal{D}_{\text{trn}}\}$
  $\mathcal{Z}_{\text{val}} \leftarrow \{f^{\text{enc}}_{\text{det}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{D}_{\text{val}}\}$
  $\mathcal{Z}_{\text{trn}}, \mathcal{Z}_{\text{aug}}, \mathcal{Z}_{\text{val}} \leftarrow \text{norm.}(\mathcal{Z}_{\text{trn}}, \mathcal{Z}_{\text{aug}}, \mathcal{Z}_{\text{val}})$
  $\mathbf{a}^{(t+1)} \leftarrow \mathbf{a}^{(t)} - \beta \nabla_{\mathbf{a}} \mathcal{L}_{\text{val}}(\mathcal{Z}_{\text{trn}}, \mathcal{Z}_{\text{aug}}, \mathcal{Z}_{\text{val}})$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$ ▷ Phase (ii) ends
  if $\mathcal{L}^{(t)}_{\text{val}} < \mathcal{L}^*_{\text{val}}$ **then** $\mathbf{a}^* \leftarrow \mathbf{a}^{(t+1)}$; $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$ **end if**
end for
Return $\mathbf{a}^*$ and $\boldsymbol{\theta}^*$

Phase (i) Detection This phase is focused on estimating the parameters $\boldsymbol{\theta}^{(t)}$ of the detector f_{det} (comprising of an encoder $f^{\text{enc}}_{\text{det}}$ and a discriminator $f^{\text{mlp}}_{\text{det}}$) by minimizing the cross-entropy loss $\mathcal{L}_{\text{trn}}$. This aims to classify between the normal samples $\mathbf{x}_{\text{trn}}$ and the augmented pseudo anomalies $\tilde{\mathbf{x}}_{\text{aug}}$ by their embeddings $\mathcal{Z}_{\text{trn}}$ and $\mathcal{Z}_{\text{aug}}$, where $\mathcal{Z}_{\text{trn}} = \{f^{\text{enc}}_{\text{det}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{D}_{\text{trn}}\}$ and $\mathcal{Z}_{\text{aug}} = \{f^{\text{enc}}_{\text{det}}(f_{\text{aug}}(\mathbf{x}; \mathbf{a}^{(t)})) \mid \mathbf{x} \in \mathcal{D}_{\text{trn}}\}$ denote the embeddings of the training data and augmented data, given the current *fixed* $\mathbf{a}^{(t)}$ at iteration t. Note that the parameters $\boldsymbol{\phi}$ of f_{aug} are frozen throughout this phase.

Phase (ii) Alignment Subsequently, the alignment phase adjusts $\mathbf{a}$ to optimize the unsupervised differentiable validation loss $\mathcal{L}_{\text{val}}$, computed based on the embeddings from the now-updated $f^{\text{enc}}_{\text{det}}$. $\mathcal{L}_{\text{val}}$'s objective is to measure the degree of alignment between $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{val}}$ in the embedding space, as expressed by the Wasserstein distance in Eq. (2), Appx. B.1. Note that the embeddings are normalized to ensure scale invariance before being passed to $\mathcal{L}_{\text{val}}$. This avoids the trivial solution of achieving alignment by setting all embeddings to $\mathbf{0}$ [8]. As the embeddings $\{\mathcal{Z}_{\text{trn}}, \mathcal{Z}_{\text{aug}}, \mathcal{Z}_{\text{val}}\}$ in $\mathcal{L}_{\text{val}}$ are obtained through the updated $f^{\text{enc}}_{\text{det}}$, the optimization needs to track the change in $\boldsymbol{\theta}$ caused by the update of $\mathbf{a}$. As such, TSAP employs a second-order optimization process, similar to [8].

Augmentation Type Selection While Algo. 1 describes continuous hyperparameter tuning, the discrete hyperparameter (i.e. augmentation/anomaly type) selection is done through grid search as the number of anomaly types is finite. Hence, we initialize TSAP for different augmentation types and compare $\mathcal{L}_{\text{val}}$ across types to select the one that yields the best alignment. The idea is that the wrong augmentation type will have poor alignment, while one that reflects the true anomalies in $\mathcal{D}_{\text{test}}$ will result in better alignment, granted proper tuning of the continuous hyperparameters through Algo. 1.

3 Experiments

We evaluate TSAP on six distinct TSAD tasks. Four of these are conducted in a *controlled* environment, using the 2017 ECG PhysioNet Challenge dataset [9]. The remaining two are *natural* anomalies, derived from the CMU Motion Capture (MoCap) dataset. In the former, the anomaly types are manually injected in $\mathcal{D}_{\text{test}}$ based on the types discussed in Sec. 2.1. For the latter, the anomaly types in $\mathcal{D}_{\text{test}}$ are unknown and it is the goal of TSAP to find the type that yields best alignment between $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{val}}$ (part of $\mathcal{D}_{\text{test}}$), expressed by $\mathcal{L}_{\text{val}}$. See Appx. C for further details.

Table 1: Detection peformance of baselines w.r.t. F_1 and AUROC on test data across six TSAD tasks (cf. Appx. C.1, Table 3). TSAP has the lowest average rank w.r.t. F_1 and AUROC, with low std. dev.

Methods	PhysioNet A		PhysioNet B		PhysioNet C		PhysioNet D		MoCap A		MoCap B		Avg. Rank	
	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC
OC-SVM	0.182	0.468	0.182	0.472	0.373	0.803	0.393	0.806	**1.000**	**1.000**	0.546	0.806	7.2 (3.7)	6.8 (3.3)
LOF	0.999	**1.000**	0.999	**1.000**	0.354	0.738	0.358	0.725	0.196	0.506	0.221	0.577	6.8 (3.9)	6.5 (4.4)
ARIMA	0.885	0.960	0.829	0.965	**0.991**	**0.999**	**0.999**	**0.999**	0.870	0.955	0.225	0.537	4.3 (3.2)	4.7 (3.7)
IF	0.255	0.587	0.232	0.576	0.183	0.402	0.182	0.356	0.864	0.965	0.342	0.758	8.8 (1.7)	8.7 (1.9)
MP	0.812	0.743	0.812	0.744	0.280	0.712	0.284	0.734	**1.000**	**1.000**	**1.000**	**1.000**	5.0 (3.6)	4.8 (3.4)
EncDec-LSTM	0.190	0.508	0.190	0.508	0.415	0.812	0.442	0.819	0.980	0.999	0.909	0.996	6.5 (2.0)	6.7 (1.9)
SR-CNN	0.965	0.990	0.964	0.998	0.983	**0.999**	0.991	**0.999**	0.302	0.700	0.214	0.512	5.2 (4.2)	4.8 (4.5)
USAD	0.183	0.425	0.184	0.428	0.430	0.822	0.409	0.828	**1.000**	**1.000**	**1.000**	**1.000**	5.5 (4.0)	5.7 (4.5)
NeuTraL-AD	0.211	0.732	0.263	0.679	0.561	0.868	0.526	0.862	**1.000**	**1.000**	**1.000**	**1.000**	4.2 (2.9)	3.8 (2.5)
TimeGPT	0.327	0.714	0.318	0.711	0.218	0.580	0.217	0.525	0.348	0.743	0.385	0.683	8.0 (1.9)	8.3 (1.6)
TSAP (ours)	**1.000**	**1.000**	**1.000**	**1.000**	0.973	**0.999**	0.991	0.998	0.889	0.969	**1.000**	**1.000**	**2.3** (2.0)	**2.2** (2.0)

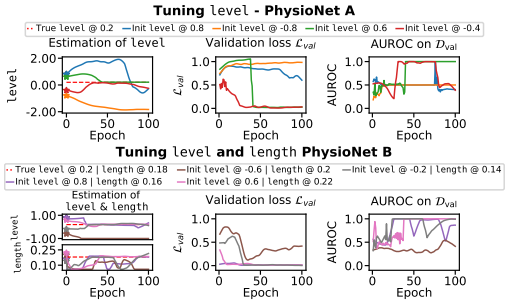

Figure 2: **Tuning continuous augmentation hyperparameter(s) with TSAP. Top:** For Platform anomalies at true `level` (red dashed line), various initializations converge near the true value (left) by minimizing val. loss (center), which leads to high detection performance (right). **Bottom:** Multiple continuous hyperparameters (`level` and `length`) converge near true values (left) through minimizing val. loss (center), resulting in high AUROC (right). Diverged results correspond to high val. loss, aiding in rejecting low-performance models.

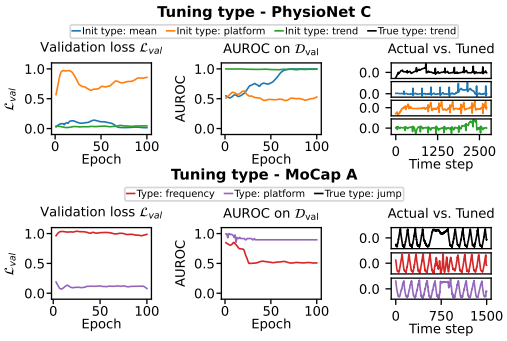

Figure 3: **Tuning the discrete hyperparameter (anomaly type) with TSAP. Top:** For true Trend anomaly (black), val. loss favors Trend (green) and Mean shift (blue), both achieving high AUROC (middle) and strong resemblance (right), while rejecting Platform (orange) with poor performance. **Bottom:** For Jump anomalies in MoCap A with unknown type (black), val. loss favors Platform (purple), leading to high AUROC (middle) and similarity to the true anomaly (right), while rejecting Frequency (red) with poor performance.

Quantitative Results Table 1 shows the detection results for these six TSAD tasks. TSAP **ranks the best overall** in both F_1 and AUROC, showing that detector $f_{\text{det}}(\cdot;\boldsymbol{\theta}^*)$, trained through the alternating mechanism of TSAP, is able to generalize to unseen, unlabeled anomalies in $\mathcal{D}_{\text{test}}$. **While competing methods perform well on certain tasks, they lack consistency across all TSAD tasks.** LOF, for instance, thrives on PhysioNet A and B due to its abrupt, manually injected anomalies, but fails on MoCap, where anomalies emerge more gradually as activities transition naturally. SR-CNN and ARIMA show similar behavior. See Appx. B.2–B.3 for model/baseline configuration details.

Reconstruction-based methods like EncDec-LSTM and USAD struggle on PhysioNet due to high variability among inliers, yet excel on MoCap, which exhibits consistent, near-periodic patterns. NeuTraL-AD, despite its augmentation-based approach, struggles with PhysioNet's real-world ECG signal variability, suggesting its augmentation functions lack robustness in noisy conditions. We remark that only TSAP provides robust and consistent performance across all TSAD tasks.

Qualitative Results Key to TSAP's consistent performance is $\mathcal{L}_{\text{val}}$ through which TSAP automatically learns the augmentation hyperparameters $\mathbf{a}$. Once $\mathbf{a}$ is determined, the task reduces to a supervised learning problem. Next, we show that TSAP not only **effectively tunes the continuous augmentation hyperparameters** $\mathbf{a}$, but also that $\mathcal{L}_{\text{val}}$ guides the **accurate selection of the discrete hyperparameter (i.e. anomaly type)**. Consider PhysioNet A, where we aim to tune the continuous hyperparameter a, i.e. `level`, of the Platform anomalies present in $\mathcal{D}_{\text{test}}$. That is, the `level` in $\mathcal{D}_{\text{test}}$ is fixed and tuning aims to estimate its value using TSAP while the other hyperparameters (`location, length`) are randomized. Fig. 2 (top) shows TSAP's estimation process for different initializations of a. We observe that the initialization for $a \in \{-0.4, 0.6\}$ leads to the true a=0.2 (left). Simultaneously, $\mathcal{L}_{\text{val}}$ drops substantially once TSAP has arrived at the true a (center). This is also reflected in the performance of f_{det} on $\mathcal{D}_{\text{val}}$ which soars upon estimation of the true a (right). Conversely, the initialization for $a \in \{0.8, -0.8\}$ leads to a high $\mathcal{L}_{\text{val}}$, indicating poor alignment between $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{val}}$. Indeed, the performance of f_{det} on $\mathcal{D}_{\text{val}}$ now suffers from poor alignment. For PhysioNet B, we estimate both `level` and `length` while `location` is randomized. Fig. 2 (bottom) demonstrates TSAP's ability to accurately estimate the `level` and `length`. Further, Fig. 3 (top) showcases TSAP's ability to perform discrete hyperparameter selection. Here, TSAP has been initialized and trained with three anomaly types (Mean shift, Platform, Trend) on PhysioNet C (true anomaly type: Trend). $\mathcal{L}_{\text{val}}$ indicates a misalignment between the Platform and Trend types (left), also reflected in the AUROC of f_{det} on $\mathcal{D}_{\text{val}}$ (center). Note how the Mean shift anomaly type has a low $\mathcal{L}_{\text{val}}$ at the end of the training epochs, reflected in the high AUROC on $\mathcal{D}_{\text{val}}$. This shows that the true underlying anomaly type is not necessarily the only type that yields high alignment, and in turn a high-performing detector. In MoCap datasets, where anomaly types are a priori unknown, TSAP is initialized with different augmentation types (Frequency and Platform) to perform discrete

hyperparameter selection. Fig. 3 (bottom) highlights its effectiveness as $\mathcal{L}_{\text{val}}$ clearly prefers one type over the other. Indeed, the natural anomalies defined by jumping signals in MoCap A have close resemblance to platform anomalies. See Appx. D for additional results and ablation studies.

4 Conclusion

We introduced TSAP for self-supervised time series anomaly detection, which is the first attempt that automatically (self-)tunes the augmentation hyperparameters on time series data in an unsupervised manner through a differentiable augmentation model and an unsupervised validation loss to help align augmented and test data. While being the first self-tuning SSL solution to TSAD, our work opens avenues for an expanded catalog of anomaly types and extensions to multivariate time series data.

Acknowledgments

Funding: This work was supported by the Research Foundation - Flanders (FWO) [grant number G0C6721N] and [grant number V436123N].

References

[1] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, 54(3):1–33, 2021.

[2] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE TKDE*, 26(9):2250–2267, 2013.

[3] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. Cutpaste: Self-supervised learning for anomaly detection and localization. In *CVPR*, 2021.

[4] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. In *NeurIPS*, 2018.

[5] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. CSI: novelty detection via contrastive learning on distributionally shifted instances. In *NeurIPS*, 2020.

[6] Jaemin Yoo, Tiancheng Zhao, and Leman Akoglu. Data augmentation is a hyperparameter: Cherry-picked self-supervision for unsupervised anomaly detection is creating the illusion of success. *TMLR*, 2023.

[7] Chen Qiu, Timo Pfrommer, Marius Kloft, Stephan Mandt, and Maja Rudolph. Neural transformation learning for deep anomaly detection beyond images. In *ICML*, pages 8703–8714, 2021.

[8] Jaemin Yoo, Lingxiao Zhao, and Leman Akoglu. End-to-end augmentation hyperparameter tuning for self-supervised anomaly detection. *arXiv preprint*, 2023.

[9] Gari Clifford, Chengyu Liu, Benjamin Moody, Li-Wei Lehman, Ikaro Silva, Qiao Li, Alistair Johnson, and Roger Mark. AF classification from a short single lead ECG recording. In *CinC 2017*. Computing in Cardiology, September 2017.

[10] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

[11] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *NeurIPS*, volume 26. Curran Associates, Inc., 2013.

[12] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[13] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endow.*, 15(9):1779–1797, may 2022.

[14] Bernhard Schölkopf, Robert C. Williamson, Alexander J. Smola, John Shawe-Taylor, and John C. Platt. Support vector method for novelty detection. In *NIPS*, 1999.

[15] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Identifying density-based local outliers. In *SIGMOD*, SIGMOD, page 93–104, 2000.

[16] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[17] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[18] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *IEEE ICDM*, pages 1317–1322, 2016.

[19] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[20] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proc. of the 25th ACM SIGKDD*, KDD '19, 2019.

[21] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. USAD: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD*, page 3395–3404. ACM, 2020.

[22] Azul Garza and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

[23] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, National Taiwan University, 2003.

Supplemental Materials

A Types of Time Series Anomalies

The anomaly generation scheme g accommodates six types of time series anomalies as discussed in Sec. 2.1. Based on g, the goal for f_{aug} is then to learn to inject anomalies into inliers, conditional on the discrete hyperparameter, anomaly type, along with their corresponding hyperparameters as described below. A visual overview of the anomaly types is provided in Fig. 4.

- **Platform:** Starting at timestamp `location`, the values of a duration `length` in the time series are **equal to** a constant value `level`.
- **Mean shift:** Starting at timestamp `location`, a constant value `level` is **added to** the values of a duration `length` in the time series.
- **Amplitude:** Starting at timestamp `location`, a constant value `level` is **multiplied with** the values of a duration `length` in the time series.
- **Trend:** Starting at timestamp `location`, a **series of values** at **is added to** the duration `length`, where a is the `level` and t is the timestamp in that duration.
- **Extremum (a.k.a. Spike):** A large (either positive or negative) value `level` is **assigned to** a single timestamp `location` in the time series.
- **Frequency shift:** Starting at phase `location`, the frequency of the duration with `length` phases is **increased by** a constant value `level`.

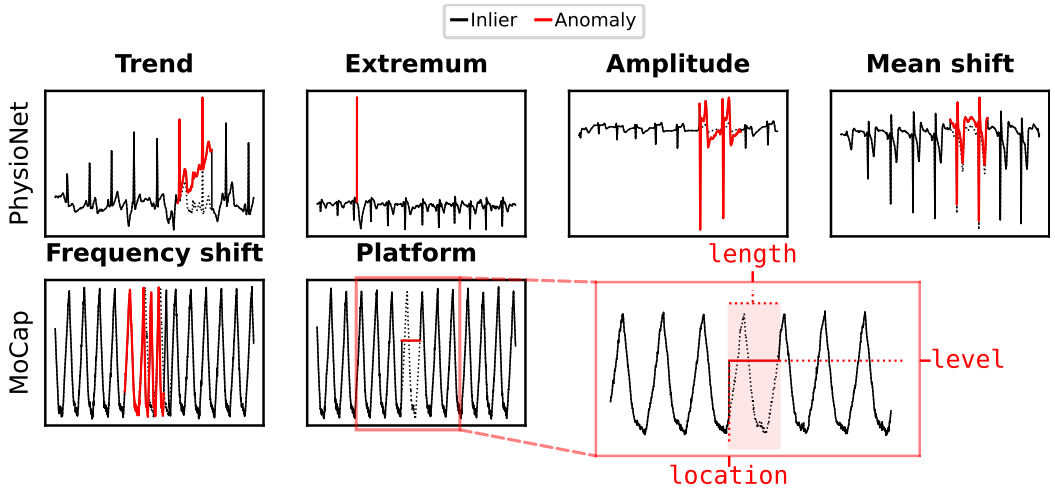

Figure 4: Examples of six different types of time series anomalies; (black) original real-world time series, (red) pseudo anomalies generated by g. The goal for f_{aug} is to learn to inject anomalies into inliers, conditional on the discrete (anomaly type) as well as continuous hyperparameters such as level (magnitude), location (start point) and length (see bottom right).

B Model Configurations

B.1 Validation Loss

We measure the degree of alignment between $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{val}}$ in the embedding space using the Wasserstein distance [10] as described in Sec. 2.2. The Wasserstein distance is a distance measure

between probability distributions, of order p between any two marginal distributions μ and ν, given by:

$$W_p(\mu,\nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \mathbb{E}_{(x,y)\sim\gamma}[d(x,y)^p] \right)^{1/p}, \tag{2}$$

where $\Gamma(\mu,\nu)$ is the set of all joint distributions (or couplings) with marginals μ and ν, respectively. That is, γ satisfies two conditions: $\int \gamma(x,y)dy = \mu(x)$ and $\int \gamma(x,y)dx = \nu(y)$. However, computing W_p directly is often computationally challenging. Thus, we employ the Sinkhorn algorithm to feasibly apply the Wasserstein distance in the machine learning context, which provides an efficient approach to approximate it via entropy regularization [11].

B.2 TSAP configuration

The Encoder$_\phi$ in f_{aug} and Encoder$_\theta$ in f_{det} are constructed using 1D CNN blocks [12] (transposed 1D CNN for Decoder$_\phi$) for efficient temporal feature extraction. We carefully choose the number of epochs T to allow sufficient time for the convergence of $\mathbf{a}$, with empirical evidence suggesting that $T = 100$ typically suffices. For the number of inner-loops L, we opt for $L = 5$, aligned with [8], such that f_{det} has adequate time to learn effective discriminative embeddings for $\mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{trn}}$. Table 2 provides a comprehensive overview of the configuration details for the different components of TSAP.

Table 2: Configuration details for f_{aug} and f_{det} of TSAP

f_{aug} configuration	
Encoder$_\phi$	
Conv Layer 1	{In: 1, Out: 64, Kernel: 100, Stride: 4, ReLU, BatchNorm}
Conv Layer 2	{In: 64, Out: 64, Kernel: 100, Stride: 4, ReLU}
Decoder$_\phi$	
TransConv Layer 1	{In: 64, Out: 64, Kernel: 100, Stride: 4, ReLU, BatchNorm}
TransConv Layer 2	{In: 64, Out: 1, Kernel: 100, Stride: 4}
MLP$_\phi$	
MLP Layer 1	{In: 3, Out: 16, ReLU}
MLP Layer 2	{In: 16, Out: dim($\mathcal{Z}_{\text{trn}}$), ReLU}
General Parameters	
Batch Size	64
# Epochs	500
Optimizer	Adam (LR: 0.002)
f_{det} configuration	
Encoder$_\theta$	
Conv Layer 1	{In: 1, Out: 32, Kernel: 10, Stride: 2, ReLU, BatchNorm}
Conv Layer 2	{In: 32, Out: 16, Kernel: 10, Dilation: 2, Stride: 2}
Conv Layer 3	{In: 16, Out: 8, Kernel: 10, Dilation: 4, Stride: 4}
Avg Pooling + Flatten	{Kernel: 10, Stride: 3}
Linear Layer	{In: 400, Out: 10}
MLP$_\theta$	
MLP Layer 1	{In: 10, Out: 1}
General Parameters	
Dropout	0.2
Batch Size	64
Warm Start # Epochs f_{det}	3
# Epochs	100
Optimizer for $\mathbf{a}$	Adam (LR: 0.001)
Optimizer for f_{det}	Adam (LR: 0.002)
Mixing Rate (cf. Phase ii)	0.15

B.3 Baseline configurations

We compare TSAP with a selection of established baselines, including traditional and deep learning-based methods with demonstrated efficacy in TSAD [13]. The traditional methods consist of different modeling approaches; namely, One-Class Support Vector Machines (**OC-SVM**) [14]; Local Outlier Factor (**LOF**) [15]; (**ARIMA**) [16]; Isolation Forest (**IF**) [17]; and the Matrix Profile (**MP**) [18]. On the deep learning side, we benchmark against the Encoder-Decoder LSTM (**EncDec-LSTM**) [19]; the Spectral Residual Convolutional Neural Network (**SR-CNN**) [20]; the Unsupervised Anomaly Detection (**USAD**) for TSAD [21]; and a recent time series foundation model (**TimeGPT**) [22]. Lastly, we include a state-of-the-art competing method which learns augmentations in the embedding space, called Neural Transformation Learning for (TS)AD (**NeuTraL-AD**) [7]. This diverse set of baselines allows for a comprehensive analysis across different approaches within the TSAD domain. The details for the baseline configurations are provided below. All models were trained on a single NVIDIA Tesla P100 GPU.

- OC-SVM: We use author-recommended hyperparameters [23].
- LOF: We use author-recommended hyperparameters [15].
- IF: We use author-recommended hyperparameters [17].
- ARIMA: We use AutoARIMA to select hyperparameters [16].
- MP: We use author-recommendations to set the window size m [18]
- EncDec-LSTM: Similar to the original authors [19], we downsample our time series to obtain time series of length approx. equal to 200. For the remaining hyperparameters, we use the original authors' recommendations.
- SR-CNN: We use author-recommended hyperparameters [20].
- USAD: Similar to the original authors [21], we downsample our time series to obtain time series of length approx. equal to 200. For the remaining hyperparameters, we use the original authors' recommendations.
- NeuTraL-AD: We use author-recommended hyperparameters [7]. We tune augmentation type separately for each dataset using *labeled* validation data.
- TimeGPT: We tune the confidence interval [22] for each dataset using *labeled* validation data.

C Experimental Setup

C.1 Dataset Details

PhysioNet The 2017 PhysioNet Challenge dataset [9] comprises a diverse array of real-world 1-lead 300 Hz ECG recordings. We use ECG recordings, each 9 seconds in length with $K = 2700$ time-steps and standardized to have a zero mean and standard deviation of one. Injected anomalies represent 10% of the data. We include a total of seven controlled TSAD tasks based on the PhysioNet data as shown in Table 3, with PhysioNet A-D discussed in Sec. 3 and the remainder in Appx. D. For example in PhysioNet A and B, given the anomaly type (Platform), the task is to infer or tune, respectively, the hyperparameter(s) `level` only and both `level` and `length`, while anomaly location is random (hence not tuned). For PhysioNet C and D, the respective tuning tasks are the same but for a different anomaly type (Trend). Finally, Table 4 (top) shows the hyperparameter spaces used to train f_{aug} in PhysioNet. The hyperparameters `location` and `length` are normalized by K. Noting that the extremum anomaly always occurs on a single timestamp in the time series, thus `length` is always 1.

MoCap The CMU Motion Capture (MoCap) dataset[3] includes signal data from various sensors on subjects' bodies as they perform different activities (walking, jumping, or running). As we focus on a univariate setting, only the sensor signal on the left femur is used. We consider the walking signal as normal data, and the signals of jumping and running as anomalies. To generate normal signals, we stitch the walking signals by identifying the start and end points of each gait phase and add random noise; whereas to generate anomalous ones, we stitch walking or running signals at a random location in the normal signal. We further add random noises to augment the normal samples in the dataset. Each signal is normalized between -1 and 1 and truncated to length $K = 1500$. This yields two distinct TSAD tasks as shown in Table 3. For each, constructed anomalies represent 10% of the data. Different from PhysioNet A–G where we only tune the continuous hyperparameter(s)

[3] http://mocap.cs.cmu.edu/

Table 3: Anomaly profile of different TSAD tasks.

Dataset		Type	Level	Location	Length
PhysioNet	PhysioNet A	Platform	Fixed	Random	Random
	PhysioNet B	Platform	Fixed	Random	Fixed
	PhysioNet C	Trend	Fixed	Random	Random
	PhysioNet D	Trend	Fixed	Random	Fixed
	PhysioNet E	Mean shift	Fixed	Random	Random
	PhysioNet F	Mean shift	Fixed	Random	Fixed
	PhysioNet G	Extremum	Fixed	Random	N/A
MoCap	MoCap A	Jump	Fixed	Random	Random
	MoCap B	Run	Fixed	Random	Random

for the given (discrete) anomaly type, for MoCap A and B, we aim to tune *both* the unknown anomaly type that corresponds to Jump and Run behavior, respectively, as well as the (continuous) hyperparameter `level` while `location` and `length` take random values. Finally, Table 4 (bottom) shows the hyperparameter spaces used to train f_{aug} in MoCap. The hyperparameters `location` and `length` of platform anomaly are normalized by K. The hyperparameters `location` and `length` of frequency anomaly denote the starting gait phase and the length of gait phases, respectively.

C.2 Evaluation Metrics

Our method calculates anomaly scores on an entire sequence level $\mathbf{x}$, similar to [7]. This is a different set-up compared to novelty detection in time series which typically operates on a point level. Detection on a sequence level can be especially important to spot long-term patterns (e.g. Trend anomalies). As such, we use the F_1 score and the Area Under the Receiver Operating Characteristic Curve (AUROC) as key performance metrics to quantify detection capability of anomalous sequences. All results are reported on the unseen $\mathcal{D}_{\text{test}}$. We determine the optimal F_1 score, by enumerating all possible thresholds, given by the anomaly scores for a given segment $\mathbf{x}$. We then compute the corresponding precision and recall for each threshold and select those that yield the highest F_1. As such, AUROC provides a balanced view whereas F_1 shows the optimal scenario. Both metrics range from 0 to 1, with higher values indicating superior performance.

D Additional Results

In this section, we present additional results for continuous (D.1) and discrete (D.2) augmentation hyperparameter tuning, as well as several ablation studies (D.3).

Table 4: Hyperparameter space for each type of anomaly.

PhysioNet			
Anomaly Types	`location`	`length`	`level`
Platform	$[100..2000]$	$[400..600]$	$\{0.2k-1 \mid k \in [0..10]\}$
Mean Shift	$[100..2000]$	$[400..600]$	$\{0.2k-1 \mid k \in [0..10]\}$
Amplitude	$[100..2000]$	$[400..600]$	$\{0.5k+1 \mid k \in [0..10]\}$
Trend	$[100..2000]$	$[400..600]$	$\{0.002k-0.01 \mid k \in [0..10]\}$
Extremum/Spike	$\{100k \mid k \in [1..26]\}$	$\{1\}$	$\{3k-15 \mid k \in [0..10]\}$
MoCap			
Anomaly Types	`location`	`length`	`level`
Platform	$[200..800]$	$[100..200]$	$\{0.2k-1 \mid k \in [0..10]\}$
Frequency Shift	$[1..3]$	$[1..6]$	$[1..3]$

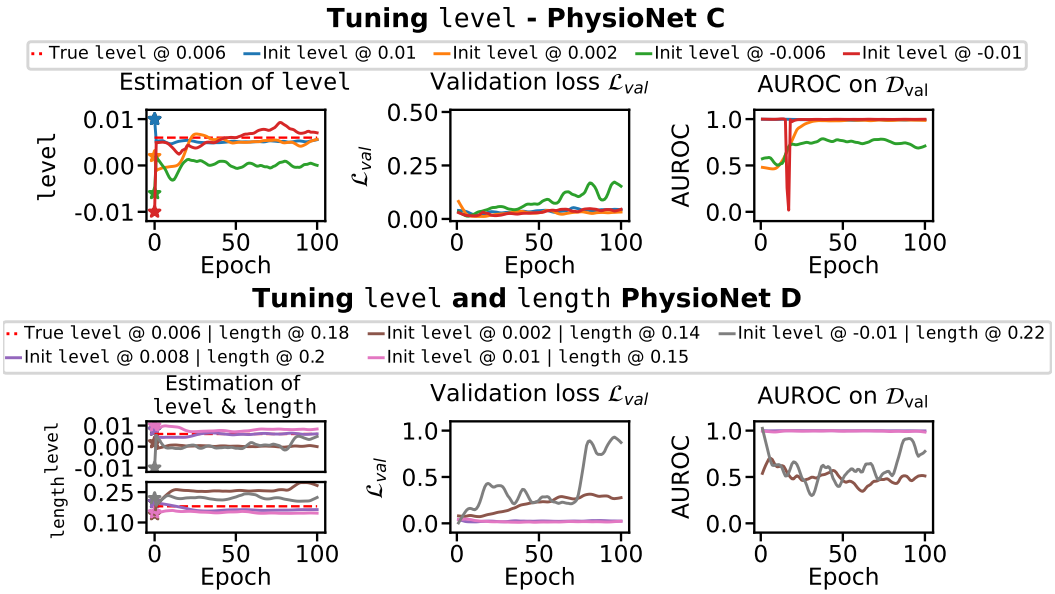

Figure 5: **Tuning the continuous augmentation hyperparameter(s) with TSAP for Trend anomalies**. **Top:** Given Trend anomalies at true `level` (red dashed line), various initializations converge near the true value (left), following the minimized values of val. loss (center), and leading to high detection AUROC performance (right). **Bottom:** Multiple continuous hyperparameters, here both `level` and `length` are accurately tuned to near true values (left), as guided by minimizing the val. loss (center), achieving high AUROC (right).

D.1 Continuous Augmentation Hyperparameter Tuning

In addition to the results on continuous hyperparameter tuning for PhysioNet A and B (see Fig. 2), we demonstrate TSAP's efficacy in tuning the continuous hyperparameters on five additional TSAD tasks. These include PhysioNet C and D, which feature Trend anomalies, as well as PhysioNet E and F, showcasing Mean shift anomalies, and PhysioNet G, containing Extremum anomalies (cf. Table 3).

PhysioNet C & D The tuning process of the continuous hyperparameters for the **Trend anomalies** in PhysioNet C and D is shown in Fig. 5. We observe several initializations for $\mathbf{a}$ that arrive closely to the true `level` for PhysioNet C, as well as to the true `level` and `length` for PhysioNet D. In turn, those initializations yield a detector f_{det} with high performance on $\mathcal{D}_{\text{val}}$. Note how, for PhysioNet C, $\mathcal{L}_{\text{val}}$ is low across the board. This is likely due to the fact that a Trend anomaly with a subtle slope, has similar characteristics to inliers (see e.g. Fig. 3 bottom left). Yet, TSAP effectively assigns a higher validation loss to the initializations that lead to misaligned cases. This shows the effectiveness of our method even in cases where anomalies are subtle.

PhysioNet E & F Similarly to Trend anomalies, **Mean shift anomalies** are inherently subtle, especially when the `level` is close to zero. We show in Fig. 6 how TSAP properly tunes the continuous hyperparameters for `level` and `length` in PhysioNet E and F for several initializations.

PhysioNet G Lastly, Fig. 7 showcases TSAP's ability to tune the `level` of the spike in the **Extremum anomalies** while the location is randomized. Note that Extremum anomalies have no `length` by definition. The ECG recordings in PhysioNet contain many natural spikes. As such, validation loss is low by default. Nonetheless, TSAP successfully tunes two out of four initializations and reflects – though subtly – this difference in the validation loss. This again leads to a well-tuned $f_{\text{det}}(\cdot; \boldsymbol{\theta}^*)$ that performs strongly on $\mathcal{D}_{\text{val}}$.

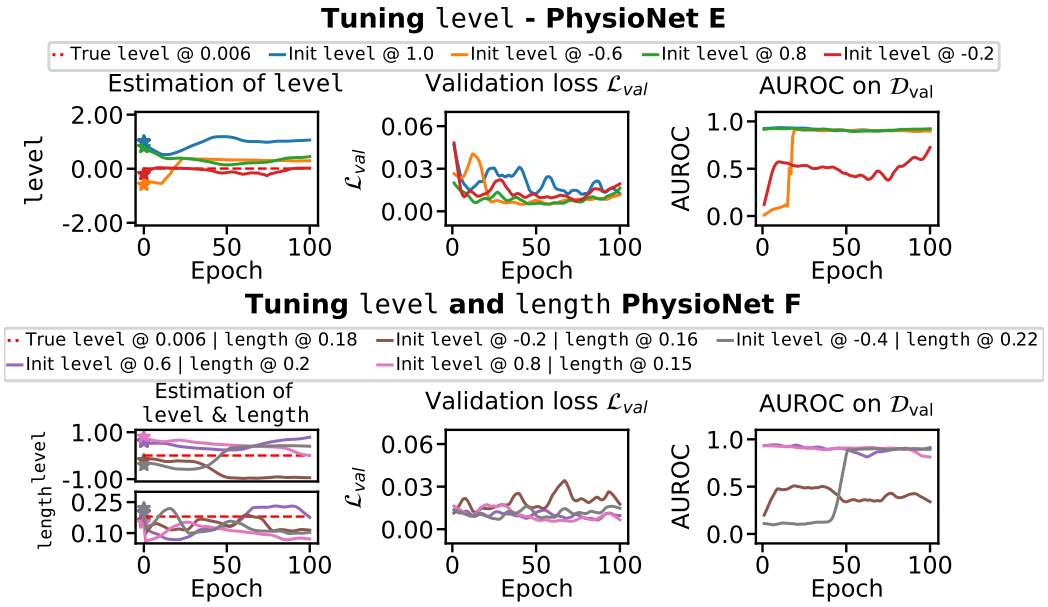

Figure 6: **Tuning the continuous augmentation hyperparameter(s) with TSAP for Mean shift anomalies**. **Top:** Given Mean shift anomalies at true `level` (red dashed line), various initializations converge near the true value (left), following the minimized values of val. loss (center), and leading to high detection AUROC performance (right). **Bottom:** Multiple continuous hyperparameters, both `level` and `length` are accurately tuned to near true values (left), as guided by minimizing the val. loss (center), achieving high AUROC (right). Mean shift anomalies have low val. loss by default due to the subtle nature of Mean shift anomalies.

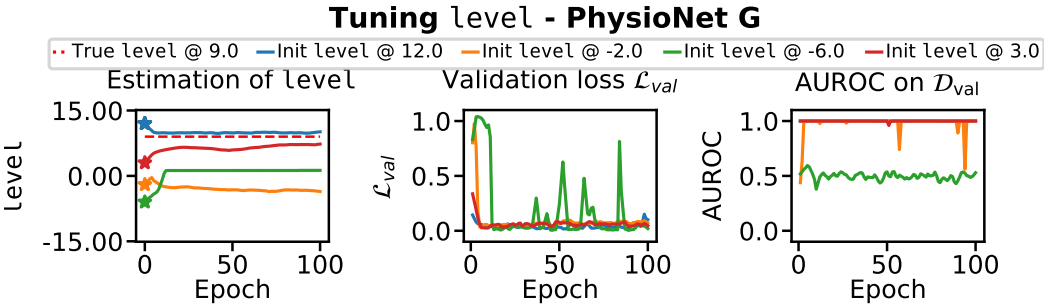

Figure 7: **Tuning the continuous augmentation hyperparameter(s) with TSAP for Extremum anomalies**. Given Extremum anomalies at true `level` (red dashed line), various initializations converge near the true value (left), following the minimized values of val. loss (center), and leading to high detection AUROC performance (right). Note how Extremum anomalies have low val. loss by default due to the natural presence of spikes in ECG data.

D.2 Discrete Augmentation Hyperparameter Tuning

We showcased TSAP's ability to tune the discrete hyperparameter, anomaly type, in Fig. 3 for controlled and natural TSAD tasks. Given the direct applicability and significance of discrete hyperparameter tuning in real-world contexts, we present extended results for discrete hyperparameter tuning.

MoCap B Fig. 8 shows the **discrete hyperparameter tuning for the unknown anomaly type** in MoCap B. TSAP was initialized twice: first with f_{aug} pre-trained for injecting Frequency shift anomalies, and second with f_{aug} pre-trained for injecting Platform anomalies. The validation loss (left) indicates a strong alignment between $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ and the unlabeled $\mathcal{D}_{\text{val}}$ when TSAP is

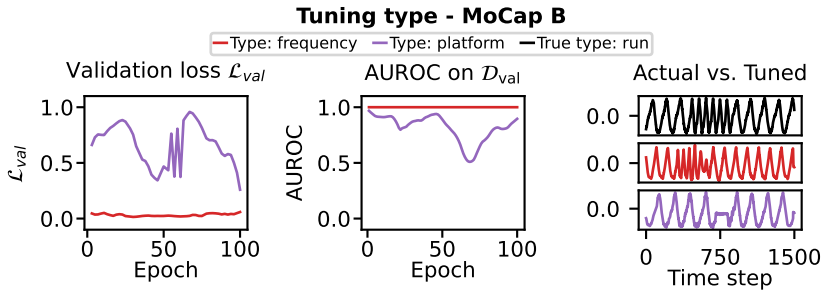

Figure 8: **Tuning discrete hyperparameter (anomaly type) with** TSAP. For Run anomalies in MoCap B with unknown type (black), val. loss favors Frequency shift (red) that leads to high AUROC (center) and mimics well true anomaly (right), and effectively rejects inferior type Platform (purple).

initialized with Frequency shift anomalies. This is also reflected in f_{det}'s performance on $\mathcal{D}_{\text{val}}$ (center). Visually, we can indeed confirm that Frequency shift anomalies (right – red) appear to be more similar to the running pattern (right – black) as opposed to Platform anomalies (right – purple).

D.3 Ablation Studies

We present four ablation studies on the controlled PhysioNet data to support various design strategies of TSAP: validation loss, self-tuning, second-order optimization, and embedding normalization.

Validation Loss In Fig. 9 (top), we illustrate the `level` estimation for PhysioNet C under the condition where our Wasserstein-based $\mathcal{L}_{\text{val}}$ is substituted with a point-wise metric, as used in [8]. This comparison shows that a point-wise validation loss tends to favor solutions where the `level` of the Trend anomaly approximates zero, essentially neutralizing the anomaly. Although this might produce high alignment, it leads to poor f_{det} performance in $\mathcal{D}_{\text{val}}$ (right). This shows that the **distributional characteristics captured by our $\mathcal{L}_{\text{val}}$ are a key contributing factor** to the success of TSAP.

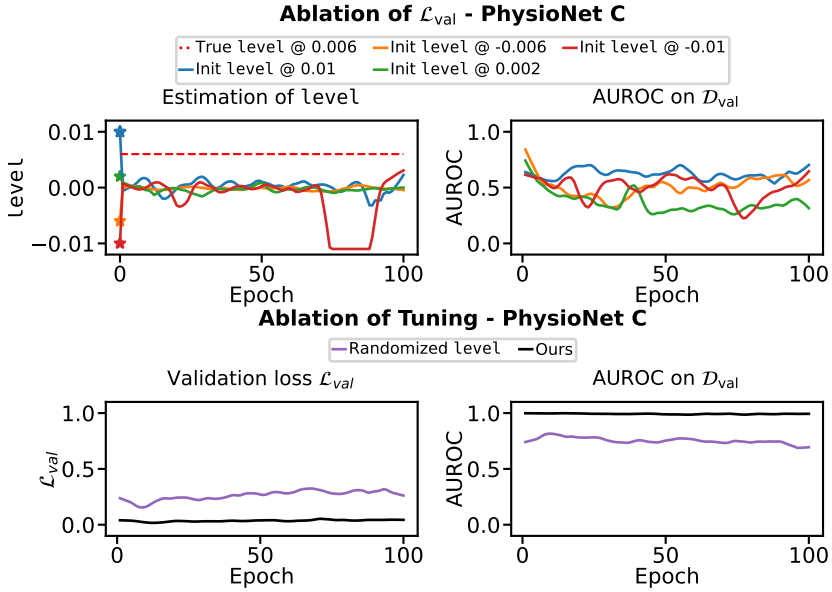

Figure 9: **Overview of ablation studies. Top:** TSAP's $\mathcal{L}_{\text{val}}$ is replaced by a point-wise val. loss leading to an erroneous estimation of a (left) and poor performance (right). **Bottom:** TSAP's self-tuning module is disabled, a is now randomized. Val. loss indicates poor alignment, reflected in poor performance.

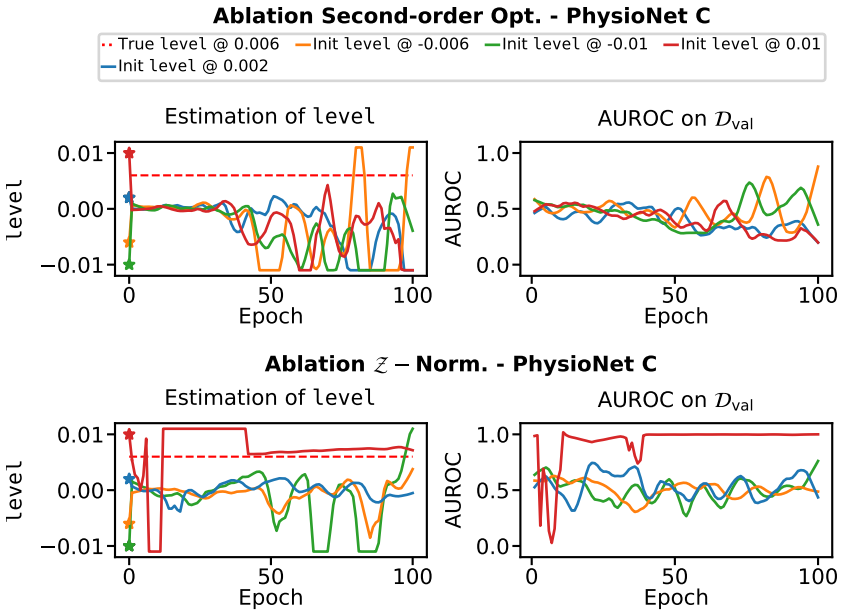

Figure 10: **Overview of additional ablation studies. Top:** We **disable second-order optimization** in TSAP, leading to a highly unstable estimation process of a (left) and poor performance of f_{det} (right). **Bottom:** We **disable normalization** of the embeddings in TSAP. Estimation of a (left) is volatile and does not converge well, in turn, performance of f_{det} on $\mathcal{D}_{\text{val}}$ is poor in most cases (right).

Randomization vs. Tuning In Fig. 9 (bottom), the self-tuning module is disabled for PhysioNet C, where `level` is instead randomized (along with `location`, and `length`). We observe substantially higher $\mathcal{L}_{\text{val}}$, indicating poor alignment. In turn, f_{det} struggles to detect the unlabeled anomalies in $\mathcal{D}_{\text{val}}$, showing the **utility of TSAP's systematic hyperparameter (self-)tuning over random choice**.

Second-order Opt. Ablation Fig. 10 (top) shows the `level`-estimation and performance of f_{det} on PhysioNet C when the second-order optimization is disabled. Note how the estimation process becomes highly unstable when second-order optimization is disabled. In turn, performance of f_{det} on $\mathcal{D}_{\text{val}}$ suffers severely.

Embedding Normalization Ablation We show the `level`-estimation and performance of f_{det} on PhysioNet C when the normalization of the embeddings $\{\mathcal{Z}_{\text{trn}}, \mathcal{Z}_{\text{aug}}, \mathcal{Z}_{\text{val}}\}$ obtained through $f_{\text{det}}^{\text{enc}}$ is disabled in Fig. 10 (bottom). While a initialized at 0.01 eventually leads to the correct `level`, the estimation process is highly volatile compared to when normalization is enabled as shown in Fig. 5 (top).