AutoTCL: Automated Time Series Contrastive Learning with Adaptive Augmentations

Xu Zheng[1], **Tianchun Wang**[2], **Wei Cheng**[3],
Aitian Ma[1], **Haifeng Chen**[3], **Mo Sha**[1], **Dongsheng Luo**[1]
[1] Knight Foundation School of Computing and Information Sciences, Florida International University
[2] College of Information Science and Technology, Pennsylvania State University,
[3] NEC Lab America
{xzhen019,aima,msha,dluo}@fiu.edu, tkw5356@psu.edu, {weicheng,haifeng}@nec-labs.com

Abstract

Modern techniques like contrastive learning have been effectively used in many areas, including computer vision, natural language processing, and graph-structured data. Creating positive examples that assist the model in learning robust and discriminative representations is a crucial stage in contrastive learning approaches. Usually, preset human intuition directs the selection of relevant data augmentations. Due to patterns that are easily recognized by humans, this rule of thumb works well in the vision and language domains. However, it is impractical to visually inspect the temporal structures in time series. The diversity of time series augmentations at both the dataset and instance levels makes it difficult to choose meaningful augmentations on the fly. Thus, although prevalent, contrastive learning with data augmentation has been less studied in the time series domain. In this study, we analyze data augmentation for time series based on information theory and summarize the most adopted transformations in a unified form. On top of that, we generalize it to a parameterized augmentation method to support adaptive usage in time series representation learning. Experiments on benchmark datasets demonstrate the highly competitive results of our method, AutoTCL, with an average 10.3% reduction in MSE and 7.0% in MAE over the leading baselines.

1 Introduction

Time series data is complex, unstructured, and high-dimensional, making it more difficult to gather the label than images or languages. This property hinders the deployment of powerful deep learning methods, which typically require a large amount of labeled data for training[Eldele *et al.*, 2021a]. Self-supervised learning is a promising solution due to its capacity in learning from unlabelled data. Similar to unsupervised learning, self-supervised learning methods learn a fixed-dimension embedding of the time series data that preserves its inherent features with better transferability and generalization capacity. A representative framework, contrastive learning, has been successful in representation learning for various types of data including vision, language, and graphs[Chen *et al.*, 2020; Xie *et al.*, 2019; You *et al.*, 2020]. These methods train an encoder to map instances to an embedding space where similar instances (positive pairs) are easily distinguished from dissimilar ones (negative pairs). As a result, model predictions are unaffected by minor noise introduced into the inputs or hidden states. As a key component, data augmentation such as jittering, scaling, permutation, and subsequence extraction [Fan *et al.*, 2020; Wen *et al.*, 2021], is usually adopted to produce positive pairs.

Compared to other forms of data, the time series domain has seen less research on contrastive learning [Eldele *et al.*, 2021a; Franceschi *et al.*, 2019; Fan *et al.*, 2020; Tonekaboni *et al.*, 2021]. Due to the diversity and variability of real-time series data, it is challenging to apply a general augmentation technique to all datasets. As a result, current approaches to contrastive learning for time series data frequently need particular data augmentation techniques that are guided by domain knowledge and necessitate trial and error to identify the most appropriate augmentations. Attempts have recently been made to study the theory behind adaptive augmentation selection for contrastive learning [Tian *et al.*, 2020; Suresh *et al.*, 2021; Xu *et al.*, 2021]. Good augmentations, according to InfoMin [Tian *et al.*, 2020], produce label-preserving views with less shared information. They discover the best perspectives through adversarial training in unsupervised or self-supervised environments by adding an invertible flow-based generative model. The InfoMin principle performs well in the vision area and has been successfully applied to graph-structured data [Xu *et al.*, 2021; Suresh *et al.*, 2021; Yin *et al.*, 2022; You *et al.*, 2022]. However, in a self-regulated environment, the majority of existing studies soften the label-preserving property and place a more significant emphasis on enhancing diversity by reducing the exchange of information between different views. They frequently use stronger transformers as augmentations and undermine the semantics, which is inapplicable to time series data.

In order to accommodate various augmentation tactics for time series contrastive learning, we investigate the data augmentation for time series from the information theory perspective and provide a theoretical sound definition of good augmentations based on input factorization. We further

present a parameterized framework, AutoTCL, to adaptively transform data for contrastive time series learning based on the proposed factorization technique, which can prevent ad-hoc decisions or laborious trial-and-error tuning. Specifically, we utilize a parametric neural network to learn to factorize an instance into two parts: the task-irrelevant part and the informative part. The informative component is then applied to a lossless transform function to keep the instance's semantics. The adaptive transformation produces a prior mask for the input instance to generate workable positive samples. We demonstrate how the most effective time series data augmentation methods can be viewed as specialized forms of the suggested mask-based transformation. By adding another random variable with adequate variance, the diversity of the augmented view is further increased. In order to learn representations through contrast, augmented pairs are then fed into a time series encoder along with randomly chosen negative pairings. Parameters in the factorization network and transform function, are optimized in tandem with contrastive learning. We also introduce a reparameterization strategy to enable efficient backpropagation. Our main contributions are summarized as follows.

- We introduce a novel factorization-based framework to guide data augmentations for contrastive self-supervised learning without prefabricated knowledge.
- To automatically learn workable transformations for time series data, we provide a straightforward yet effective instantiation that can handle a variety of frequently applied augmentations.
- With comprehensive experimental studies, we empirically verify the advantage of the proposed method on benchmark time series forecasting datasets. We achieve highly competitive performances with a 6.5% reduction in MSE on 4.7% in MAE on univariate forecasting and a 2.9% reduction in MSE on 1.2% in MAE on multivariate forecasting. In classification tasks, our method achieves a 1.2% increase in average accuracy.

2 Related work

2.1 Contrastive learning for time series.

Contrastive learning has been widely used in representation learning, achieving superior results across various domains [Chen *et al.*, 2020; Xie *et al.*, 2019; You *et al.*, 2020]. Recently, there have been efforts to apply contrastive learning to the time series domain [Oord *et al.*, 2018; Franceschi *et al.*, 2019; Fan *et al.*, 2020; Eldele *et al.*, 2021a; Tonekaboni *et al.*, 2021; Yue *et al.*, 2021]. In [Franceschi *et al.*, 2019], Franceschi et.al. utilize subsequences to generate positive and negative pairs. TNC uses a debiased contrastive objective to make sure that in the representation space, signals from the local neighborhood are distinct from those that are not neighbors [Tonekaboni *et al.*, 2021]. SelfTime uses several custom-designed augmentations for unsupervised contrastive learning of time series by examining both the relations between different samples and those within samples [Fan *et al.*, 2020]. TS2Vec uses hierarchical contrastive learning to acquire a representation for each time stamp [Yue *et al.*, 2021]. TF-C utilizes the distance between time and frequency components as the self-supervised signal for representation learning. Each component is independently optimized by contrastive estimation [Zhang *et al.*, 2022]. BTSF further includes spectral information and utilizes a simple dropout to capture the long-term relationship within each instance [Yang and Hong, 2022]. In [Nonnenmacher *et al.*, 2022], the authors introduce an approach that incorporates expert knowledge into time-series representation learning using expert features, surpassing existing methods in unsupervised and semi-supervised learning on real-world datasets. CLUDA [Ozyurt *et al.*, 2022], a novel framework for unsupervised domain adaptation of time series data, utilizes contrastive learning to learn contextual representations that preserve label information, achieving state-of-the-art performance in time series unsupervised domain adaptation. However, data augmentations in these methods are not widely applied in complex real-life datasets due to the limitations of being either universal or chosen through trial and error.

2.2 Adaptive data augmentation

Data augmentation is a crucial aspect of contrastive learning. Previous studies have shown that the choice of optimal augmentation methods depends on the specific task and dataset being used [Chen *et al.*, 2020; Fan *et al.*, 2020]. Some studies have explored the adaptive selection of augmentation methods for contrastive learning in the visual domain, such as AutoAugment [Cubuk *et al.*, 2019], which uses a reinforcement learning method to search for the best combination of translation policies. Faster-AA [Hataya *et al.*, 2020], which improves the searching pipeline for data augmentation using a differentiable policy network. DADA introduces an unbiased gradient estimator that enables a more efficient one-pass optimization strategy [Li *et al.*, 2020]. In the contrastive learning frameworks, the InfoMin theory is applied to guide the selection of good views for contrastive learning in the vision domain [Tian *et al.*, 2020], it further proposes a flow-based generative model to transfer images from natural color spaces into novel color spaces for data augmentation. [Rommel *et al.*, 2022] investigates automatic data augmentation algorithms for deep learning, focusing on class-wise policies and complex data types like neuroscience signals. The proposed differentiable relaxation approach outperforms other methods and introduces novel augmentation operations for EGG data classification. In [Aboussalah *et al.*,], RIM, a recursive framework for time series augmentation, was proposed. However, given the complexity of time series data, directly applying the InfoMin framework may not be suitable. Different from previous works, our focus is on the time series domain and we propose an end-to-end differentiable method to automatically learn the optimal augmentations for each time series instance.

3 Methodology

In this section, we first describe the notations used in this paper. Then, we try to answer the following research questions. (1) What are the good views for contrastive learning in the

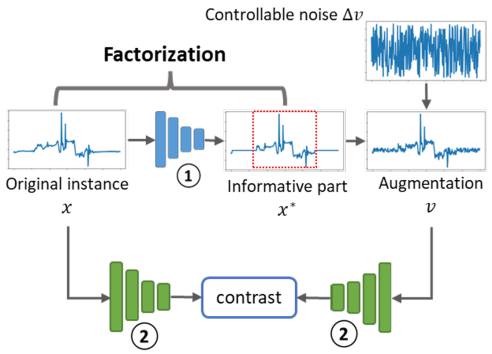


Figure 1: The framework of the proposed AutoTCL. ① is the factorization function to extract the informative part from the original instance guided by the Principle of Relevant Information. ② is the encoder network, which is optimized with the contrastive objective.

self-supervised setting? (2) How to obtain good views for each time series instance for contrastive learning?

3.1 Notations

We use a $T \times F$ matrix to represent a time series instance x, where T is the length of its sequence and F is the dimension of features. With $F > 1$, x is a multivariate time series instance. Otherwise, with $F = 1$, x is a single variate instance. Self-supervised contrastive learning aims to learn an encoder $f_{\boldsymbol{\theta}}$ that maps x from $\mathbb{R}^{T\times F}$ to a vector space $\mathbb{R}^D$, where $\boldsymbol{\theta}$ represents the parameters in the encoder network and D is the dimension of embedding vectors. In the paper, to distinguish random variables and instances, we use the Sans-serif style lowercase letters, such as x, to represent random time series variables, and italic lowercase letters, such as x, for real instances.

3.2 What makes good views for contrastive self-supervised learning?

In the literature, a well-accepted intuitive principle for view generation is that good views preserve the semantics and provide sufficient variances [Tian *et al.*, 2020; Yin *et al.*, 2022; Suresh *et al.*, 2021]. In the supervised setting, where training labels are available, the semantics of an instance is usually approximated with the label. On the other hand, semantics-preserving is much under-explored in the more popular self-supervised learning. Moreover, unlike images and natural language sentences, whose semantics can be manually verified, the underlying semantics of time series data are not recognizable to humans, making it more challenging, if not impossible, to include strong yet faithful data augmentations for such complicated data. To avoid the degenerate solutions caused by dismissing the semantics-preserving, InfoMin utilizes an invertible flow-based function, denoted by g, to generate a view v for an input x [Tian *et al.*, 2020]. Such that x can be restored by $x = g^{-1}(v)$. However, from the information theory perspective, invertible functions fail to include extra variance to the original variable. Formally, we have the following property.

Property 1. *If view* v *is generated from* x *with an invertible function* $\mathsf{v} = g(\mathsf{x})$*. Then* $H(\mathsf{v}) = H(\mathsf{x}) = MI(\mathsf{x};\mathsf{v})$*, where* $H(\mathsf{x}), H(\mathsf{v})$ *are entropy of variables* x *and* v*, respectively;* $MI(\mathsf{v};\mathsf{x})$ *is mutual information between* v *and* x*.*

The detailed proof can be found in Appendix. This property shows that the entropy of the augmented view, $H(\mathsf{v})$, is no larger than that of original data, $H(\mathsf{x})$, indicating that the existing data augmentation methods don't bring new information for input instances, which limits their expressive power for time series contrastive learning. To address the challenge and facilitate powerful self-supervised learning in the time series domain, we propose a novel factorized augmentation technique. Specifically, given an instance x, we assume that x can be factorized into two parts, informative x^* and noise/task-irreverent part Δx. Formally,

$$x = x^* + \Delta x. \tag{1}$$

As the informative part, x^* encodes the semantics of the original x. Motivated by the intuitive principle, we formally define good views for contrastive learning as follows.

Definition 1 *(Good View). Given a random variable* x *with its semantics* x^**, a good view* v *for contrastive learning can be achieved by* $\mathsf{v} = g(\mathsf{x}^*) + \Delta\mathsf{v}$*, where* g *is an inverible function, and* $\Delta\mathsf{v}$ *is a task-irrelevant noise, satisfying* $H(\Delta\mathsf{v}) \geq H(\Delta\mathsf{x})$*.*

Intuitively, a good view, based on our definition, maintains the useful information in the original variable and at the same time, includes a larger variance to boost the robustness of encoder training. We theoretically show that the defined good view has the following properties.

Property 2 *(Task Agnostic Label Preserving). If a variable* v *is a good view of* x*, and the downstream task label* y *(although not visible to training) is independent to noise in* x*, the mutual information between* v *and* y *is equivalent to that between raw input* x *and* y*, i.e.,* $MI(\mathsf{v};\mathsf{y}) = MI(\mathsf{x};\mathsf{y})$*.*

Property 3 *(Containing More Information). A good view* v *contains more information comparing to the raw input* x*, i.e.,* $H(\mathsf{v}) \geq H(\mathsf{x})$*.*

Detailed proofs are given in Appendix. These properties show that in the self-supervised setting, adopting a good view for contrastive learning theoretically guarantees that we will not decrease the fidelity, regardless of the downstream tasks. Simultaneously, the good view is flexible to the choice of $\Delta\mathsf{v}$, meaning that strong augmentations may be utilized to incorporate enough diversity for training.

3.3 How to achieve good views?

The theoretical analysis suggests a factorized augmentation to preserve task-agnostic labels and improve the diversity of views. In this part, we introduce a practical instantiation to obtain good views based on parametric augmentations as demonstrated in Fig. 1. We first introduce a factorization function $h : \mathbb{R}^{T\times F} \to \{0,1\}^{T\times 1}$ to detect the informative components of the input. Formally,

$$\begin{aligned} &\text{Factorization mask} && \boldsymbol{m}^{(h)} = h(x) \\ &\text{Informative component} && x^* = \boldsymbol{m}^{(h)} \odot x \\ &\text{Noise component} && \Delta x = x - x^*, \end{aligned} \tag{2}$$

where $\odot$ is a generalized Hadamard product operation between a vector(matrix) and another vector(matrix). If both

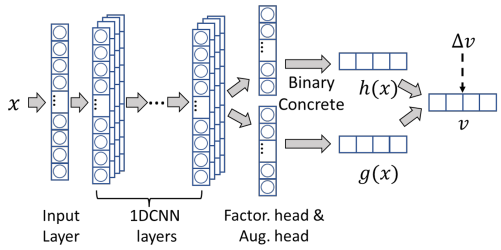


Figure 2: The architecture of our augmentation network.

two inputs are vectors, such as $\boldsymbol{v}$ and $\boldsymbol{m}$, then $\boldsymbol{v} \odot \boldsymbol{m}$ denotes the element-wise product; if the first operator a vector $\boldsymbol{v} \in \mathbb{R}^N$ and the second one is a matrix $\boldsymbol{M} \in \mathbb{R}^{N \times M}$, we first stack M copies of $\boldsymbol{v}$ to get a matrix $\boldsymbol{V} \in \mathbb{R}^{N \times M}$, then apply the normal Hadamard product on $\boldsymbol{V}$ and $\boldsymbol{M}$; if both inputs are matrices with the same shape, $\odot$ is the vanilla Hadamard product. For the invertible function applied on x^*, we present a mask-based instantiation. More sophisticated mechanisms, such as normalization flow [Kobyzev *et al.*, 2020], can also be used as a plug-and-play component. Specifically, we introduce a non-zero mask $\boldsymbol{m}^{(g)} \in \mathbb{R}_+^T$ to form the transformation. Specifically, we have $v^* = \boldsymbol{m}^{(g)} \odot x^*$. It is easy to show that such a non-zero mask transformation is lossless as the original x^* can be restored by

$$x^* = \frac{1}{\boldsymbol{m}^{(g)}} \odot v^*. \tag{3}$$

Considering the diverse nature of time series instances, which even differ inside a dataset, a universal mask is infeasible for all instances. For example, the cutout transform works well for a time series instance and the jitter transform is more suitable for another. To support the instance-level adaptive selection of suitable transforms, we propose a parametric mask generator, denoted by $g : \mathbb{R}^{T \times F} \rightarrow \mathbb{R}_+^T$, that learns to generate the non-zero mask for lossless transformation in a data-driven manner. Formally, $\boldsymbol{m}^{(g)} = g(x)$. By combing the factorization function, the mask generator, and a random noise for perturbation, Δv, Then, a good view v for contrastive learning can be achieved. Formally, we have

$$\begin{aligned} v &= v^* + \Delta v = \boldsymbol{m}^{(g)} \odot x^* + \Delta v, \\ &= g(x) \odot h(x) \odot x + \Delta v. \end{aligned} \tag{4}$$

Practical instantiation with augmentation neural network. According to the Universal Approximation Theorem (Chapter 6 in [Goodfellow *et al.*, 2016]), we implement $g(x)$ and $h(x)$ with neural networks, respectively. We first utilized the same input layer and a stacked dilated CNN module [Franceschi *et al.*, 2019; Yue *et al.*, 2021] for both $g(x)$ and $h(x)$, respectively. Then, we include two projectors heads, a factorization head for $h(x)$ and an augmentation head for $g(x)$. The architecture of the overall augmentation network is shown in Fig. 2. To ensure the binary output of the factorization function $h(x)$, we introduce a stochastic mechanism following the factorization head. Specifically, we assume that each element in the output $\boldsymbol{m}^{(h)}$, denoted by $m_i^{(h)}$, is drawn from a Bernoulli distribution parameterized by π_i, which is calculated by the factorization head.

To enable efficient optimization with gradient-based methods, we approximate the discrete Bernoulli processes with hard binary concrete distributions [Louizos *et al.*, 2017]. Specifically, we first draw $\tilde{m}_i^{(h)}$ from a binary concrete distribution with π_i indicating the location [Maddison *et al.*, 2016; Jang *et al.*, 2016]. Formally,

$$\tilde{m}_i^{(h)} = \sigma((\log \epsilon - \log(1-\epsilon) + \log \frac{\pi_i}{1-\pi_i})/\tau), \tag{5}$$

where $\epsilon \sim \text{Uniform}(0,1)$ is an independent variable, $\sigma(\cdot)$ is the sigmoid function, and τ is the temperature controlling the approximation. The rationality of such an approximation is that with temperature $\tau > 0$, the gradient $\frac{\partial \tilde{m}_i^{(h)}}{\partial \pi_i}$ is well-defined. The output of the binary concrete distribution is in the range of (0,1). To further facilitate the binary selection, we stretch $\tilde{m}_i^{(h)}$ the range to (γ, ζ), with $\gamma < 0$ and $\zeta > 1$. Then, the final masking element $m_i^{(h)}$ is obtained by clipping values to the range $[0, 1]$. Formally,

$$m_i^{(h)} = \min(1, \max(\tilde{m}_i^{(h)}(\zeta - \gamma) + \gamma, 0)). \tag{6}$$

Δv as random timestamp masking. To increase the variance of augmented views, inspired by Dropout [Srivastava *et al.*, 2014] and TS2Vec [Yue *et al.*, 2022], we implement Δv by masking the hidden representation at the first layer. Specifically, given a latent vector of a view v after the first hidden layer, we randomly mask it along the time dimension with a binary vector $\boldsymbol{m}_v \in \{0,1\}^T$. Each element in $\boldsymbol{m}_v$ is sampled independently from a Bernoulli distribution $\text{Bern}(0.5)$.

Regularization of temporal consistency. As shown in previous studies [Luo *et al.*, 2023], informative signals tend to be continuous. Thus, we include regularization of temporal consistency when generating the factorization mask. Specifically, given a batch $\mathbb{X}_B$, for each instance $x \in \mathbb{X}_B$, we randomly select a time point a as the anchor. Then we randomly selected a time point p from the left or right position of a to create a positive pair (a, p). Their mask values $m_{x,a}^{(h)}$ and $m_{x,p}^{(h)}$ should be similar, compared to another point n that is far away from a, whose mask value is denoted by $m_{x,n}^{(h)}$. Formally, we have the following triplet loss.

$$\mathcal{L}_{\text{r}} = \frac{1}{\mathbb{X}_B} \sum_{x \in \mathbb{X}_B} (|m_{x,a}^{(h)} - m_{x,p}^{(h)}| - |m_{x,a}^{(h)} - m_{x,n}^{(h)}|). \tag{7}$$

Relationship to existing augmentations for time series. In recent years, various types of data augmentation techniques have been applied to enhance the performance of deep learning on time series dasta [Wen *et al.*, 2021; Yue *et al.*, 2022; Fan *et al.*, 2020], including time domain, frequency domain, and their hybrids. Our instantiation can be considered a general augmentation framework in the time domain. Most

existing techniques in this category, such as cropping, flipping, scaling, and jittering can be unified in our framework. For example, cropping, which deletes a subsequence, can be achieved by letting $g(x) = x$ and the cropping time steps in $\boldsymbol{m}^{(h)}$ be 0; scaling, which multiplies the input time series by a scaling factor, either a constant or being sampled from a Gaussian distribution, can also be obtained by setting $\boldsymbol{m}^{(h)} = \mathbb{1}$ and $\boldsymbol{m}^{(g)}$ being the scaling factor.

3.4 Training algorithm

In the proposed framework, there are two parametric neural networks to be optimized, i.e., the encoder and the augmentation networks. In a nutshell, the augmentation network aims to generate perceptually realistic views with high diversities, and the encoder aims to distinguish the positive pairs from randomly sampled negative pairs. Our method is used as plug and play component and can be used in a wide range of contrastive learning frameworks, such as TS2Vec [Yue *et al.*, 2022] and CoST [Woo *et al.*,]. In this section, we first introduce a new objective to train the augmentation network followed by the alternating training of encoder and augmentation networks.

Training the augmentation network with the principle of relevant information. Existing optimization techniques for the learnable augmentation and the encoder generally follow the principle of Information Bottleneck (IB) [Tishby *et al.*, 2000], which aims to achieve sufficient and minimal representations [Tian *et al.*, 2020; Yin *et al.*, 2022; Suresh *et al.*, 2021]. However, IB relies on the class labels from the downstream task, making it unsuitable for self-supervised training where there are few or no labels. Instead of following previous works that adversarially train the encoder and augmentation networks, which may fail to preserve the semantics in time series data, we train the augmentation network based on the Principle of Relevant Information (PRI) [Principe, 2010]. Unlike supervised IB which relies on another variable as well as their joint distributions, PRI only exploits the self-organization of a random variable, making it fully unsupervised. Specifically, with PRI training the augmentation network, we aim to achieve a reduced statistical representation x^* by decomposing x as follows.

$$\mathcal{L}_{\text{PRI}} = \min \beta H(\mathsf{x}^*) + D(\mathbb{P}(\mathsf{x})||\mathbb{P}(\mathsf{x}^*)), \tag{8}$$

where β is the trade-off hyper-parameter, $H(\mathsf{x}^*)$ is the entropy of representation variable x^*, and $D(\mathbb{P}(\mathsf{x})||\mathbb{P}(\mathsf{x}^*))$ is the divergence between distributions of the original variable x and transformed variable x^*. The minimization of $H(\mathsf{x}^*)$ aims to reduce uncertainty and obtain statistical regularity in x^* and the second term is for preserving the descriptive power of x^* about x.

Given an instance x, the informative part x^* is obtained by applying a binary factorization mask $\boldsymbol{m}^{(h)} \in \{0,1\}^T$ on x, thus, minimizing the first term in Eq. (8), $H(\mathsf{x}^*)$, can be achieved by minimizing the number of non-zero elements in the factorization mask, i.e. $||\boldsymbol{m}^{(h)}||_0$. According to the calculation of $\boldsymbol{m}^{(h)}$ in Eq. (6), we have

$$||\boldsymbol{m}^{(h)}||_0 = \sum_{t=1}^{T} \left(1 - \sigma(\tau \log \frac{-\gamma(1-\pi_t)}{\zeta \pi_t})\right). \tag{9}$$

To preserve the descriptive power of x^* about x, we follow existing works and approximate the second term $D(\mathbb{P}(\mathsf{x})||\mathbb{P}(\mathsf{x}^*))$ with the maximum mean discrepancy(MMD) [Gretton *et al.*, 2012]. Formally, given a mini-batch $\mathbb{X}_B$, we have our practical objective as follows.

$$\mathcal{L}_{\text{PRI}} = \frac{1}{\mathbb{X}_B} \sum_{x \in \mathbb{X}_B} \beta||\boldsymbol{m}_x^{(h)}||_0 + \text{MMD}(\boldsymbol{z}_x, \boldsymbol{z}_{x^*}) \tag{10}$$

where $\boldsymbol{m}_x^{(h)}$ is the factorization mask of instance x,[1] and $\text{sim}(\boldsymbol{z}_x, \boldsymbol{z}_{x*})$ is the inner product between representation vectors $\boldsymbol{z}_x$ and $\boldsymbol{z}_{x^*}$. The final augmentation network loss could be formulated as:

$$\mathcal{L}_{\text{aug}} = \mathcal{L}_{\text{PRI}} + \lambda \mathcal{L}_{\text{r}} \tag{11}$$

Alternating training. To effectively train encoder and augmentation networks, we follow GAN [Goodfellow *et al.*, 2020] to use an alternating training schedule that trains the embedding network M times and then trains the augmentation network one time. M is a hyper-parameter determined by grid search.

4 Experiments

In this section, we compare AutoTCLwith representative and SOTA baselines on time series forecasting and classification tasks. We also conduct ablation studies to show insights into each component in AutoTCL. All experiments are conducted on a Linux machine with 8 NVIDIA A100 GPUs, each with 40GB of memory. The software environment is CUDA 11.6 and Driver Version 520.61.05. We used Python 3.9.13 and Pytorch 1.12.1 to construct our project. Due to the space limitation, detailed experimental setups, full experimental results, and extensive experiments are presented in Appendix.

4.1 Time series forecasting

Datasets and baselines. Six benchmark datasets for time series forecasting, ETTh1, ETTh2, ETTm1, [Zhou *et al.*, 2021], Electricity [Dua and Graff, 2017] dataset, Weather dataset[2], and Lora dataset are adopted for both univariate and multivariate settings. Lora dataset is a new proposed real-world dataset that captures the wireless signal data using the LoRa devices[3]. It contains 74 days of data with timestamps. The proposed AutoTCL model is compared to representative state-of-the-art methods such as TS2Vec [Yue *et al.*, 2022], Informer [Zhou *et al.*, 2021], StemGNN [Cao *et al.*, 2021], TCN [Bai *et al.*, 2018], LogTrans [Li *et al.*, 2019], LSTnet [Lai *et al.*, 2018], CoST [Woo *et al.*,], TNC[Tonekaboni *et al.*, 2021], TS-TCC[Eldele *et al.*, 2021b], InfoTS [Luo *et al.*, 2023] and N-BEATS [Oreshkin *et al.*, 2019], with N-BEATS being exclusive to univariate forecasting and StemGNN to multivariate.

Setup. We follow CoST [Woo *et al.*,] network architecture. A multi-layer dilated CNN module is used for the backbone

[1]We slightly reuse notations with different fonts. $\boldsymbol{m}_x^{(h)}$ is the mask vector of x and $m_i^{(h)}$ is the i-th element in a vector $\boldsymbol{m}^{(h)}$.

[2]https://www.ncei.noaa.gov/data/local-climatological-data/

[3]https://lora-alliance.org/

Table 1: Univariate time series forecasting results.

	AutoTCL		TS2Vec		Informer		LogTrans		N-BEATS		TCN		CoST		TNC		TS-TCC		InfoTS	
Dataset	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	**0.076**	**0.207**	0.110	0.252	0.186	0.347	0.196	0.365	0.218	0.375	0.263	0.431	0.091	0.228	0.150	0.303	0.168	0.316	0.091	0.227
$ETTh_2$	0.158	**0.299**	0.170	0.321	0.204	0.358	0.217	0.391	0.326	0.442	0.219	0.362	0.161	0.307	0.168	0.322	0.298	0.428	**0.149**	**0.299**
$ETTm_1$	**0.046**	**0.154**	0.069	0.186	0.241	0.382	0.270	0.416	0.162	0.326	0.200	0.349	0.054	0.164	0.069	0.191	0.158	0.299	0.050	0.157
Elec.	**0.366**	**0.345**	0.393	0.370	0.464	0.388	0.744	0.528	0.727	0.482	0.525	0.423	0.375	0.353	0.378	0.359	0.511	0.603	0.369	0.348
WTH	**0.160**	**0.287**	0.181	0.308	0.243	0.370	0.280	0.411	0.256	0.374	0.166	0.291	0.183	0.307	0.175	0.303	0.302	0.442	0.176	0.304
Lora	**0.177**	**0.273**	0.356	0.385	1.574	0.999	0.656	0.550	0.311	0.349	1.160	0.927	0.186	0.282	0.620	0.565	0.490	0.591	0.333	0.325
Avg.	**0.157**	**0.258**	0.207	0.301	0.486	0.477	0.382	0.441	0.320	0.388	0.419	0.465	0.168	0.271	0.256	0.340	0.315	0.441	0.188	0.274

Table 2: Multivariate time series forecasting results.

	AutoTCL		TS2Vec		Informer		LogTrans		StemGNN		TCN		CoST		TNC		TS-TCC		InfoTS	
Dataset	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	0.656	0.590	0.788	0.646	0.907	0.739	1.043	0.890	0.738	0.632	1.021	0.816	**0.650**	**0.585**	0.904	0.702	0.748	0.635	0.784	1.622
$ETTh_2$	**1.191**	**0.815**	1.566	0.937	2.371	1.199	2.898	1.356	1.940	1.077	2.574	1.265	1.283	0.851	1.869	1.053	2.120	1.109	1.474	0.914
$ETTm_1$	**0.409**	0.441	0.628	0.553	0.749	0.640	0.965	0.914	0.729	0.626	0.818	0.849	**0.409**	0.439	0.740	0.599	0.612	0.564	0.568	0.521
Elec.	0.175	0.272	0.319	0.397	0.495	0.488	0.351	0.412	0.501	0.489	0.332	0.404	**0.165**	**0.268**	0.387	0.446	0.511	0.602	0.289	0.376
WTH	0.423	**0.457**	0.451	0.474	0.574	0.552	0.645	0.617	**0.353**	0.593	0.440	0.461	0.430	0.464	0.441	0.466	0.483	0.535	0.455	0.472
Lora	0.346	0.372	0.356	0.384	0.743	0.586	0.766	0.520	**0.258**	0.492	1.013	0.814	0.350	0.378	0.590	0.518	0.490	0.591	0.345	**0.368**
Avg.	**0.545**	**0.499**	0.697	0.571	0.990	0.708	1.138	0.798	0.753	0.651	1.057	0.781	0.561	0.505	0.837	0.637	0.838	0.675	0.665	0.556

and we remove the seasonal feature disentangler module. The Augmentation network has the same feature extract architecture and two projectors as shown in Fig. **??**. In addition, the proposed AutoTCL is a general data augmentation framework that can also be combined with more recent methods, such as BTSF [Yang and Hong, 2022], TF-C [Zhang *et al.*, 2022], and LsST [Wang *et al.*, 2022] to further improve accuracy performances. We leave this as our future work. Due to the page limit, details of the experiments are provided in Appendix. Time series forecasting aims to predict future time stamps, using the last L_x observations. Following the method presented in [Yue *et al.*, 2021], a linear model regularized with L2 norm penalty, is trained to make predictions. In the univariate case, the model's output has a dimension of L_y, while in the multivariate case, it has a dimension of $L_y \times F$, where F is the number of features. The evaluation is based on standard regression metrics, Mean Squared Error (MSE), and Mean Absolute Error (MAE). To comprehensively evaluate the performances, we consider different prediction lengths, L_y.

Results. For each dataset, we calculate the average forecasting performances in both univariate and multivariate settings. The results are shown in Table 1 and Table 2, respectively. The detailed results of univariate and multivariate time series forecasting could be found in Table 5 and Table 6 in Appendix. From these tables, we have several observations. First, in general, contrastive learning methods, including AutoTCL, TSvec, CoST, and InfoTS, achieve better performances compared to traditional baselines, indicating the effectiveness of contrastive learning for learning time series representations. Second, the consistent improvement of our method over CoST indicates that universal data augmentations may not be the most informative for generating positive pairs in various datasets. Compared to CoST, AutoTCL decreases the average MSE by 6.5% and the average MAE by 4.8% in the univariate setting. This is because AutoTCL can adaptively learn the most suitable augmentations in a data-driven manner, preserving semantics and ensuring sufficient variance. Encoders trained with these informative augmentations lead to representations with higher quality. In the more complicated multivariate setting, AutoTCL decreases the average MSE by 2.9% and the average MAE by 1.2% on average. In particular, AutoTCL reduces the MSE and MAE by 7.2% and 4.2% on the dataset $ETTh_2$ compared to the second-best baseline CoST [Woo *et al.*,].

4.2 Time series classification

Datasets and baselines. For the classification task, we evaluate our method on the UEA dataset [Dau *et al.*, 2019], which contains 30 multivariate time series datasets. We compare our method with 8 state-of-the-art baselines, including TS2Vec [Yue *et al.*, 2022], T-Loss [Franceschi *et al.*, 2019], TNC [Tonekaboni *et al.*, 2021], TS-TCC [Eldele *et al.*, 2021b], TST [Zerveas *et al.*, 2021], DTW [Chen *et al.*, 2013], TF-C [Zhang *et al.*, 2022] and InfoTS [Luo *et al.*, 2023].

Setup. Similar to forecasting tasks, we use TS2Vec [Yue *et al.*, 2022] network architecture. In the training stage, we use the same strategy as the forecasting tasks which could be found in Appendix. We follow the previous setting [Yue *et al.*, 2022] that the evaluation is conducted in a standard supervised manner. A radial basis function kernel SVM classifier is trained on the training set and then makes predictions on test data. We report two metrics in the results, accuracy(ACC) and rank(RANK).

Results. The results on the 30 UEA datasets are summarized in Table 3. The detailed results can be found in Table 9 in Appendix. Overall, AutoTCL substantially outperforms baselines with an average rank value 2.3. As shown in Table 9, our method achieves the best results in 16 out of 30

Table 3: Classification result of the UEA dataset

Dataset	AutoTCL	TS2Vec	T-Loss	TNC	TS-TCC	TST	DTW	TF-C	InfoTS
Avg. ACC	**0.742**	0.704	0.658	0.670	0.668	0.617	0.629	0.298	0.730
Avg. RANK	**2.300**	3.700	4.667	5.433	5.133	6.133	5.400	8.200	2.367

Table 4: Ablation studies and model analysis

	AutoTCL		W/o $h(x)$		W/o $g(x)$		W/o Δv		W/o Aug		Cutout		Jitter		Adversarial	
Dataset	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	**0.076**	**0.207**	0.077	0.208	0.078	0.209	0.086	0.219	0.095	0.231	0.088	0.221	0.086	0.219	0.089	0.224
$ETTh_2$	**0.158**	**0.299**	0.168	0.305	0.178	0.312	0.176	0.311	0.170	0.309	0.160	0.306	0.173	0.317	0.187	0.319
$ETTm_1$	**0.046**	**0.154**	0.052	0.161	0.050	0.159	0.051	0.163	0.053	0.162	0.053	0.164	0.056	0.170	0.052	0.163
Elec.	**0.365**	0.348	0.371	0.349	**0.365**	0.348	0.366	0.347	0.368	0.354	0.367	**0.345**	0.366	0.344	**0.365**	**0.345**
WTH	**0.160**	**0.287**	0.172	0.301	0.166	0.295	0.164	0.293	0.183	0.309	0.167	0.294	0.174	0.304	0.166	0.294
Lora	**0.177**	**0.273**	0.237	0.309	0.489	0.385	0.304	0.361	0.711	0.412	0.783	0.442	0.285	0.346	0.445	0.373
Avg.	**0.157**	**0.258**	0.173	0.270	0.216	0.282	0.185	0.280	0.260	0.294	0.266	0.394	0.184	0.281	0.212	0.284

datasets. In addition, it improves the classification accuracy by 1.6%, on average, over the second-best baseline, InfoTS. The comprehensive comparison indicates the effectiveness of the proposed method.

4.3 Ablation study and model analysis.

In this set of experiments, we conduct ablatio studies to investigate the effectiveness of each component in the proposed method.

Effectiveness of automatic data augmentation and factorization

To present deep insights into the automatic data augmentation and factorization, we compare AutoTCL with multiple groups of variants. (1)**W/o** $h(x)$, **W/o** $g(x)$, and **W/o** Δv are ablation studies about the effectiveness of each part of AutoTCL. In our experiments, **W/o** $h(x)$ means the whole input instance would be regarded as the informative part. **W/o** $g(x)$ represents the augmentation head $g(x)$ would be replaced by all 1 vectors and No noise will be added in **W/o** Δv setting. (2) **Cutout** and **Jitter** are two commonly adopted data augmentation techniques for time series contrastive learning. We replace the augmentation network in AutoTCL with these two static transformations as variants. (3) **Adversarial** training is routinely adopted in the literature to learn views for contrastive learning. For this variant, we adversarially train the augmentation network by minimizing the mutual information between views and original instances, approximated by the InfoNCE [Tian *et al.*, 2020]. We report the averaged performances in Table 4 and The full results are shown in Table 7 in Appendix.

We have several observations in Table 4. First, by removing the factorization head, **W/o** $h(x)$ increase the MSE by 10.19% and MAE by 4.65% respectively, verifying the effectiveness of the proposed factorization-based augmentation. The comparison between AutoTCL, **W/o** $g(x)$, and **W/o** Δv indicates the importance of diversity in data augmentation. Specifically, **W/o** $g(x)$ increases the MSE by 37.6% and MAE by 9.3%; **W/o** Δv increases by 17.83% and 8.52%, respectively.

Second, the comparison between **W/o Aug** and **Cutout** shows that universal and non-parametric augmentation techniques may harm the performances of time series contrastive learning. On average, **Cutout** performs even worse than **W/o Aug**. This observation is consistent with the conclusion drawn in TS2Vec [Yue *et al.*, 2022]. By adaptive learning suitable augmentations, our methods can consistently and significantly outperform these baselines.

Third, with the augmentation network trained in an adversarial manner, the variant, **Adversarial** improves the performances, indicating the necessity of adaptive augmentation for time series data. However, overlooking the semantic preservation may generate trivial augmented views, hindering the performance of downstream contrastive learning. On the other hand, our method achieves the best performances in most cases, especially for forecasting long periods, which verifies the advantage of our training algorithm.

5 Conclusion and future work

We present a novel factorization-based augmentation framework for time series representation learning in the self-supervised setting. Theoretical analysis from the information theory perspective shows that the proposed framework is more flexible to persevere semantics and includes sufficient variances to augment views. On top of that, we provide a simple and effective instantiation and an efficient training algorithm. With time series forecasting as the downstream task, we compare the proposed method, AutoTCL, with representative methods and verify its effectiveness.

As a general framework, our work is easy to combine with existing state-of-the-art methods and achieve a better result. In addition, AutoTCL exploits the informative part of time series data, which might help users better understand the time series data. We believe our work can bring a positive effect on such area and we do not see any negative side-effect of this work at present. In the future, we plan to include augmentations in the frequency domain into our factorization-based framework to further improve the performance of AutoTCL. Moreover, we will also extend the technique to other domains, such as graph-structured data.

References

[Aboussalah *et al.*,] Amine Mohamed Aboussalah, Minjae Kwon, Raj G Patel, Cheng Chi, and Chi-Guhn Lee. Recursive time series data augmentation. In *The Eleventh International Conference on Learning Representations*.

[Bai *et al.*, 2018] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[Cao *et al.*, 2021] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Conguri Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. Spectral temporal graph neural network for multivariate time-series forecasting. *arXiv preprint arXiv:2103.07719*, 2021.

[Chen *et al.*, 2013] Yanping Chen, Bing Hu, Eamonn Keogh, and Gustavo EAPA Batista. Dtw-d: time series semi-supervised learning from a single example. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 383–391, 2013.

[Chen *et al.*, 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607, 2020.

[Cubuk *et al.*, 2019] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, pages 113–123, 2019.

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

[Dua and Graff, 2017] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

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

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

[Fan *et al.*, 2020] Haoyi Fan, Fengbin Zhang, and Yue Gao. Self-supervised time series representation learning by inter-intra relational reasoning. *arXiv preprint arXiv:2011.13548*, 2020.

[Franceschi *et al.*, 2019] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *arXiv preprint arXiv:1901.10738*, 2019.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[Goodfellow *et al.*, 2020] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[Gretton *et al.*, 2012] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[Hataya *et al.*, 2020] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster autoaugment: Learning augmentation strategies using backpropagation. In *ECCV*, pages 1–16. Springer, 2020.

[Jang *et al.*, 2016] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kobyzev *et al.*, 2020] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[Lai *et al.*, 2018] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *SIGIR*, pages 95–104, 2018.

[Li *et al.*, 2019] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *NeurIPS*, pages 5243–5253, 2019.

[Li *et al.*, 2020] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M Robertson, and Yongxin Yang. Dada: Differentiable automatic data augmentation. *arXiv preprint arXiv:2003.03780*, 2020.

[Louizos *et al.*, 2017] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

[Luo *et al.*, 2021] Dongsheng Luo, Wei Cheng, Yingheng Wang, Dongkuan Xu, Jingchao Ni, Wenchao Yu, Xuchao Zhang, Yanchi Liu, Haifeng Chen, and Xiang Zhang. Information-aware time series meta-contrastive learning. 2021.

[Luo *et al.*, 2023] Dongsheng Luo, Wei Cheng, Yingheng Wang, Dongkuan Xu, Jingchao Ni, Wenchao Yu, Xuchao Zhang, Yanchi Liu, Yuncong Chen, Haifeng Chen, and Xiang Zhang. Time series contrastive learning with information-aware augmentations, 2023.

[Maddison *et al.*, 2016] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[Nonnenmacher *et al.*, 2022] Manuel T Nonnenmacher, Lukas Oldenburg, Ingo Steinwart, and David Reeb. Utilizing expert features for contrastive learning of time-series representations. In *International Conference on Machine Learning*, pages 16969–16989. PMLR, 2022.

[Oord *et al.*, 2018] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[Oreshkin *et al.*, 2019] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[Ozyurt *et al.*, 2022] Yilmazcan Ozyurt, Stefan Feuerriegel, and Ce Zhang. Contrastive learning for unsupervised domain adaptation of time series. *arXiv preprint arXiv:2206.06243*, 2022.

[Principe, 2010] Jose C Principe. *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science & Business Media, 2010.

[Rommel *et al.*, 2022] Cédric Rommel, Thomas Moreau, Joseph Paillard, and Alexandre Gramfort. Cadda: Class-wise automatic differentiable data augmentation for eeg signals. In *ICLR 2022-International Conference on Learning Representations*, 2022.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[Suresh *et al.*, 2021] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:15920–15933, 2021.

[Tian *et al.*, 2020] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *arXiv preprint arXiv:2005.10243*, 2020.

[Tishby *et al.*, 2000] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

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

[Wang *et al.*, 2022] Zhiyuan Wang, Xovee Xu, Goce Trajcevski, Weifeng Zhang, Ting Zhong, and Fan Zhou. Learning latent seasonal-trend representations for time series forecasting. In *Advances in Neural Information Processing Systems*, 2022.

[Wen *et al.*, 2021] Qingsong Wen, Liang Sun, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. In *AAAI*, 2021.

[Woo *et al.*,] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In *International Conference on Learning Representations*.

[Xie *et al.*, 2019] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

[Xu *et al.*, 2021] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-aware graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:30414–30425, 2021.

[Yang and Hong, 2022] Ling Yang and Shenda Hong. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *International Conference on Machine Learning*, pages 25038–25054. PMLR, 2022.

[Yin *et al.*, 2022] Yihang Yin, Qingzhong Wang, Siyu Huang, Haoyi Xiong, and Xiang Zhang. Autogcl: Automated graph contrastive learning via learnable view generators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8892–8900, 2022.

[You *et al.*, 2020] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *NeurIPS*, pages 5812–5823, 2020.

[You *et al.*, 2022] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. Bringing your own view: Graph contrastive learning without prefabricated data augmentations. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1300–1309, 2022.

[Yue *et al.*, 2021] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, and Bixiong Xu. Ts2vec: Towards universal representation of time series. *arXiv preprint arXiv:2106.10466*, 2021.

[Yue *et al.*, 2022] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8980–8987, 2022.

[Zerveas *et al.*, 2021] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *SIGKDD*, pages 2114–2124, 2021.

[Zhang *et al.*, 2022] Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. Self-supervised contrastive pre-training for time series via time-frequency consistency. *arXiv preprint arXiv:2206.08496*, 2022.

[Zhou *et al.*, 2021] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, 2021.

Appendix

A Detailed proofs

Property 1. *If view* v *is generated from* x *with an invertible function* $\mathsf{v} = g(\mathsf{x})$*. Then* $H(\mathsf{v}) = H(\mathsf{x}) = MI(\mathsf{x};\mathsf{v})$*, where* $H(\mathsf{x}), H(\mathsf{v})$ *are entropy of variables* x *and* v*, respectively;* $MI(\mathsf{v};\mathsf{x})$ *is the mutual information between* v *and* x*.*

Proof. Since g is an invertible function and $\mathsf{v} = g(\mathsf{x})$, we have an one-to-one mapping between variables v and x. Thus, $v = g(x)$ for each pair of x and v. We have $\mathbb{P}[\mathsf{x} = x] = \mathbb{P}[\mathsf{v} = v]$. From the definition of Shannon entropy, we have

$$\begin{aligned} H(\mathsf{x}) &= -\sum_x p(x)\log p(x) = -\sum_x \mathbb{P}[\mathsf{x}=x]\log\mathbb{P}[\mathsf{x}=x] \\ &= -\sum_v \mathbb{P}[\mathsf{v}=v]\log\mathbb{P}[\mathsf{v}=v] = -\sum_x p(v)\log p(v) \\ &= H(\mathsf{v}). \end{aligned}$$

From the definition of conditional entropy, we have

$$H(\mathsf{x}|\mathsf{v}) = \sum_{v,x} p(v,x)\log\frac{p(v,x)}{p(v)},$$

$$H(\mathsf{x}|\mathsf{v}) = \sum_{v,x} p(v)\log\frac{p(v)}{p(v)} = 0.$$

The above results in the mutual information between v and x, given by

$$\mathrm{MI}(\mathsf{v};\mathsf{x}) = H(\mathsf{x}) - H(\mathsf{x}|\mathsf{v}) = H(\mathsf{v}).$$

□

Property 2 *(Task agnostic label preserving). If a variable* v *is a good view of* x*, and the downstream task label* y *(although not visible to training) is independent to noise in* x*, the mutual information between* v *and* y *is equivalent to that between raw input* x *and* y*, i.e.,* $MI(\mathsf{v};\mathsf{y}) = MI(\mathsf{x};\mathsf{y})$*.*

Proof. From the definition of the good view, we have

$$\begin{aligned} \mathsf{x} &= \mathsf{x}^* + \Delta\mathsf{x} \\ \mathsf{v} &= g(\mathsf{x}^*) + \Delta\mathsf{v}. \end{aligned}$$

We first analyze the relationship between $\mathrm{MI}(\mathsf{x},\mathsf{y})$ and $\mathrm{MI}(\mathsf{x}^*,\mathsf{y})$.

$$\begin{aligned} \mathrm{MI}(\mathsf{x},\mathsf{y}) &= H(\mathsf{y}) - H(\mathsf{y}|\mathsf{x}) \\ &= H(\mathsf{y}) + \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)} \\ &= H(\mathsf{y}) + \sum_{x^*,\Delta x,y} p(x^*,\Delta x,y)\log\frac{p(x^*,\Delta x,y)}{p(x^*,\Delta x)} \\ &= H(\mathsf{y}) + \sum_{x^*,\Delta x,y} p(x^*,\Delta x,y)\log\frac{p(\Delta x,y|x^*)}{p(\Delta x|x^*)}. \end{aligned}$$

With the safe independence assumption, we have

$$p(\Delta x, y|x^*) = p(\Delta x|x^*)p(y|x^*).$$

Thus, we show that

$$\begin{aligned} \mathrm{MI}(\mathsf{x},\mathsf{y}) &= H(\mathsf{y}) + \sum_{x^*,\Delta x,y} p(x^*,\Delta x,y)\log\frac{p(x^*,y)}{p(x^*)} \\ &= H(\mathsf{y}) + \sum_{x^*,y} p(x^*,y)\log\frac{p(x^*,y)}{p(x^*)} \\ &= H(\mathsf{y}) - H(\mathsf{y}|\mathsf{x}^*) \\ &= \mathrm{MI}(\mathsf{x}^*,\mathsf{y}). \end{aligned}$$

Letting $\mathsf{v}^* = g(\mathsf{x}^*)$, from the Property 1, we have

$$\mathrm{MI}(\mathsf{x}^*,\mathsf{y}) = \mathrm{MI}(\mathsf{v}^*,\mathsf{y}).$$

Since $\Delta\mathsf{v}$ is a random noise and is independent to the label y, similarly, we have

$$\mathrm{MI}(\mathsf{v}^*,\mathsf{y}) = \mathrm{MI}(\mathsf{v},\mathsf{y}).$$

Combining them together results in

$$\mathrm{MI}(\mathsf{v},\mathsf{y}) = \mathrm{MI}(\mathsf{x},\mathsf{y}).$$

□

Property 3. *(Containing more information). A good view* v *contains more information comparing to the raw input* x*, i.e.,* $H(\mathsf{v}) \geq H(\mathsf{x})$*.*

Proof. Since $\Delta\mathsf{v}$ denotes the included random noise, we assume that its generation is independent of the augmented view v^*. Thus we have

$$\mathrm{MI}(\Delta\mathsf{v},\mathsf{v}^*) = 0. \tag{12}$$

Further, with our decomposing model, we can rewrite the entropy of x as the joint entropy of x^* and $\Delta\mathsf{x}$. Formally, we have

$$H(\mathsf{x}) = H(\mathsf{x}^*,\Delta\mathsf{x}) = H(\mathsf{x}^*) + H(\Delta\mathsf{x}) - \mathrm{MI}(\Delta\mathsf{x},\mathsf{x}^*).$$

Then $H(\mathsf{x}^*) = H(\mathsf{v}^*)$ holds (Property 1). From the definition of the good view, we have $H(\Delta\mathsf{v}) \geq H(\Delta\mathsf{x})$. Thus, we have

$$\begin{aligned} H(\mathsf{x}) &= H(\mathsf{x}^*) + H(\Delta\mathsf{x}) - \mathrm{MI}(\Delta\mathsf{x},\mathsf{x}^*) \\ &\leq H(\mathsf{v}^*) + H(\Delta\mathsf{v}) \\ &= H(\mathsf{v}^*) + H(\Delta\mathsf{v}) - \mathrm{MI}(\Delta\mathsf{v},\mathsf{v}^*) \\ &= H(\Delta\mathsf{v},\mathsf{v}^*) = H(\mathsf{v}). \end{aligned}$$

□

Derivation of Eq. (9) As described in Section 3.3, the factorization mask $\boldsymbol{m}^{(h)}$ is generated with a hard concrete distribution. Thus, the number of non-zero entries in $\boldsymbol{m}^{(h)}$ can be reformulated with

$$||\boldsymbol{m}^{(h)}||_0 = \sum_t (1 - \mathbb{P}_{m_t^{(h)}}(0)),$$

where $\mathbb{P}_{m_i^{(h)}}(0)$ is the cumulative distribution function (CDF) of $m_i^{(h)}$ (before clipping). We let $\mathrm{S}(\cdot)$ be an affine function of the stretch process in Eq. (6), such that

$$m_i^{(h)} = \mathrm{S}(\tilde{m}_i^{(h)}) = \tilde{m}_i^{(h)}(\zeta - \gamma) + \gamma,$$

where $\gamma \in (-\infty, 0)$ and $\zeta \in (1, \infty)$. As derived in [Maddison *et al.*, 2016], the density of $m_i^{(h)}$ is

$$p_{m_t^{(h)}}(x) = \frac{\tau \alpha_t^{(h)} x^{-\tau-1}(1-x)^{-\tau-1}}{(\alpha_t^{(h)} x^{-\tau} + (1-x)^{-\tau})^2},$$

where $\alpha_t^{(h)} = \log \frac{\pi_t}{1-\pi_t}$. The CDF of variable $m_t^{(h)}$ reads

$$\mathbb{P}_{m_t^{(h)}}(x) = \sigma((\log x - \log(1-x))\tau - \alpha_t^{(h)}).$$

Thus, the probability density function of $m_t^{(h)}$ is

$$\begin{aligned} p_{m_t^{(h)}}(x) &= p_{\tilde{m}_i^{(h)}}(\mathrm{S}^{-1}(x)) \left| \frac{\partial}{\partial x} \mathrm{S}^{-1}(x) \right| \\ &= \frac{(\zeta-\gamma)\tau \alpha_t^{(h)} (x-\gamma)^{-\tau-1}(\zeta-x)^{-\tau-1}}{(\alpha_t^{(h)}(x-\gamma)^{-\tau} + (\zeta-x)^{-\tau})^2}. \end{aligned}$$

The CDF of $m_t^{(h)}$ is given by

$$\begin{aligned} &\mathbb{P}_{m_t^{(h)}}(x) = \mathbb{P}_{\tilde{m}_i^{(h)}}(\mathrm{S}^{-1}(x)) \\ &= \sigma((\log(x-\gamma) - \log(\zeta-x))\tau - \alpha_t^{(h)}). \end{aligned}$$

When setting $x = 0$, we have the

$$\mathbb{P}_{m_t^{(h)}}(0) = \sigma(\tau \log \frac{-\gamma}{\zeta} - \alpha_t^{(h)}).$$

B Implementation details

B.1 Training the encoder with local and global contrasts.

Similar to the augmentation network, our method can work with different architectures. We formulate the feature extraction encoder as $f_{\boldsymbol{\theta}}(x) : \mathbb{R}^{T \times F} \to \mathbb{R}^D$, where $\boldsymbol{\theta}$ represents the learnable parameters and D is the dimensionality of output embeddings. Following existing work [Luo *et al.*, 2021], we use both global and local contrastive losses.

Global contrast aims to improve the inter-instance robustness for representation learning. Given a batch of time series instances $\mathbb{X}_B \subseteq \mathbb{X}$, for each instance $x \in \mathbb{X}_B$, we generate an augmented view v. Such a pair of the original instance x and the corresponding view v is then used as a positive pair. Other pairs of instances and views are treated as negative pairs. Formally, (x, v') is a negative pair, where v' is an augmented view of x' and $x' \neq x$. Following [Chen *et al.*, 2020], we use the InfoNCE as the global-wise contrastive loss to train the encoder network. Formally, we have

$$\mathcal{L}_g = -\frac{1}{|\mathbb{X}_B|} \sum_{x \in \mathbb{X}_B} \log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_x, \boldsymbol{z}_v))}{\sum_{x' \in \mathbb{X}_B} \exp(\mathrm{sim}(\boldsymbol{z}_x, \boldsymbol{z}_{v'}))}. \quad (13)$$

Local contrast is designed to enhance the encoder network to capture the intra-instance relationship. Given an augmented view v, we first segment it into a set of subsequences $\mathbb{S}$, where each subsequence $s \in \mathbb{S}$ has length L. Following [Tonekaboni *et al.*, 2021], two close subsequences (s, p) are considered as a positive pair, and the ones with a large distance lead to a negative pair. Formally, the loss of local contrast is:

$$\mathcal{L}_{l_x} = -\frac{1}{|\mathbb{S}|} \sum_{s \in \mathbb{S}} \log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_s, \boldsymbol{z}_p))}{\exp(\mathrm{sim}(\boldsymbol{z}_s, \boldsymbol{z}_p)) + \sum_{j \in \bar{\mathcal{N}}_s} \exp(\mathrm{sim}(\boldsymbol{z}_s, \boldsymbol{z}_j))}, \quad (14)$$

where $\bar{\mathcal{N}}_s$ is the set of negative pairs for a subsequence s. Considering all instances in a batch, we have

$$\mathcal{L}_l = \frac{1}{|\mathbb{X}_B|} \sum_{x \in \mathbb{X}_B} \mathcal{L}_{l_x}. \quad (15)$$

With both local and global contrasts, we have our contrastive loss as follows.

$$\mathcal{L}_{\text{con}} = \mathcal{L}_g + \alpha \mathcal{L}_l, \quad (16)$$

where α is the hyper-parameter to achieve the trade-off between global and local losses.

B.2 Training algorithm

In the training stage, AutoTCL optimizes the augmentation network and encoder network simultaneously. Similar to GAN [Goodfellow *et al.*, 2016], these networks were randomly initialized. Different from GAN, AutoTCL is less affected by the problem of gradient explosion and mode collapse, because our encoder network aims to embed the information part from different views rather than distinguish them. Although our argumentation network tries to reduce the distribution between original instances and arguments, AutoTCL augmentations preserve the information part by using a reversible mapping function, which alleviates the mode collapse problem. Our training algorithm could be described as follows.

Algorithm 1 AutoTCL training algorithm

Require: augmentation network f_{aug}, encoder network f_{enc}, epochs E, a hyperparameter M,
epoch $\leftarrow 0$
while epoch $< E$ **do**
  count $= 0$
  for x in dataset **do**
    $x_a \leftarrow f_{\text{aug}}(x)$
    $\boldsymbol{z}_x \leftarrow f_{\text{enc}}(x)$
    $\boldsymbol{z}_a \leftarrow f_{\text{enc}}(x_a)$
    if count$\%M == 0$ **then**
      Compute loss with using Eq. (11)
      Update parameters in f_{aug} with backpropagation
    end if
    Compute loss with using Eq. (16)
    Update parameters in f_{enc} with backpropagation
  end for
  epoch $\leftarrow$ epoch $+ 1$
end while

C Experimental settings

C.1 Baseline settings

In forecasting tasks, we conducted baseline methods on six benchmark datasets by following the experiment setting of TS2Vec[Yue *et al.*, 2022] for most baseline methods, such as Informer [Zhou *et al.*, 2021], [Tonekaboni *et al.*, 2021], StemGNN [Cao *et al.*, 2021], TCN [Bai *et al.*, 2018], N-BEATS [Oreshkin *et al.*, 2019], etc. For TS2Vec[Yue *et al.*,

2022], CoST [Woo *et al.*,], we followed its code default setting for Lora and Weather datasets. The representation dimension was 320 and the learning rate and batch size were 0.001 and 8. For InfoTS [Luo *et al.*, 2023], We used the default setting to conduct experiments. As for TS-TCC [Eldele *et al.*, 2021b] in forecasting tasks, we used the Epilepsy config as the default config and modified the network model to make the input and output channels remain the same. Due to its pooling layers, the network would require 3 times the lengths of inputs of other baselines which is unfair for forecasting tasks. In the experiments, we used another interpolate layer to make the length of input data and prediction data the same. In classification tasks, similar to the forecasting task, we followed the experiment setting of TS2Vec[Yue *et al.*, 2022]. In TF-C [Zhang *et al.*, 2022] classification experiments, we use its HAR config as the default setting. Similar to TS-TCC, we modified the network so that the transformer encoder could fit the input length and the pre-train dataset is the same as finetune dataset.

C.2 Hyperparameters

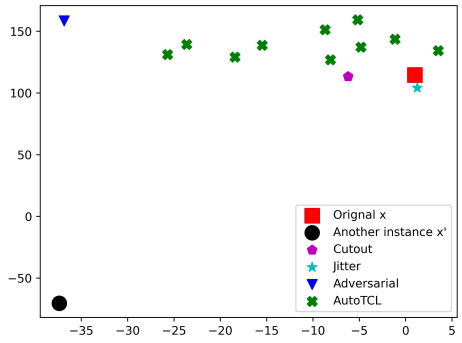
In our experiments, we used grid search to obtain the best performance. We used the same strategy in forecasting and classification tasks that each dataset had its own group of hyperparameters. We provided all of the hyperparameters as well as their configurations in the following:

- Optimizer: Two Adam optimizers [Kingma and Ba, 2014] were used for the augmentation network and feature extraction network with learning rate and other hyperparameters were setting with default decay rates setting to 0.001 and (0.9,0.999) respectively.
- Encoder architecture: The depth of the multi-layer dilated CNN module and the hidden dimension were designed to be able to change, which were searched in $\{6, 7, 8, 9, 10\}$ and $\{256, 128, 64, 32, 16, 8\}$. In training, we used a designed dropout rate to avoid overfitting, which was tuned in $[0.01, 1]$.
- Augmentation architecture: Same as encoder, the depth of multi-layer dilated CNN module and hidden dimension are hyperparameters, searched in $\{1, 2, 3, 4, 5\}$ and $\{256, 128, 64, 32, 16, 8\}$ and as mention in equation Eq. 6, ζ is another hyperparameter, tuned in $[0.0005, 0.5]$.
- Trade-off hyperparameters: β in Eq. (10), and λ in Eq. (11) are tuned in $[0, 0.3]$.
- Alternating training hyperparameters: M in Sec. (3.4) is tuned in $1, 2, 4, 8, 16, 32$.

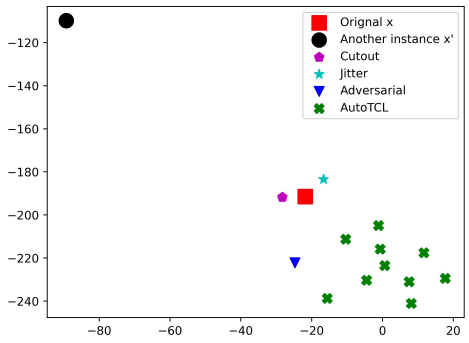
C.3 Extra experiments

Visualization of augmentation

In order to further explore the effectiveness of AutoTCL, we used T-SNE to visualize the embeddings of different augmented views in Figure 3. We chose an instance, denoted by x, from dataset ETTh_1 and compare different augmentation methods, including **Cutout**, **Jitter**, and **Adversarial**. To avoid the special case, we reported 10 augmented views for AutoTCL. We also include another x' instance as a reference. As shown in Figure 3, the instances augmented by AutoTCL include more diversity compared with **Jitter** and **Cutout**. Moreover, the augmentation generated by the **Adversarial** is closer to x' or x, indicating that it fails to preserve the underlying semantics.



(a) Samples a

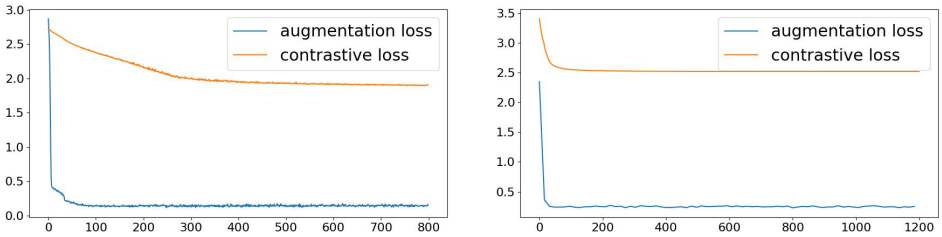


(b) Samples b

Figure 3: T-SNE visualization of different augmentation instances. In samples a and b, AutoTCL-generated samples are closer to the original instance x than other instances x' with large variety

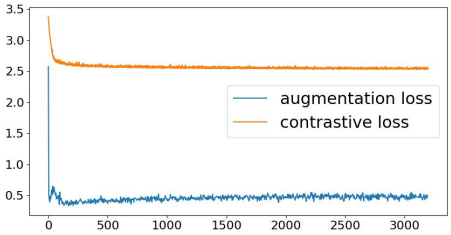
Visualization of convergence

To show the convergence of our method, we plotted the curves of Eq.(11) and Eq. (16) on different datasets. As shown in Figure 4, our method converged easily in both the argumentation network and the embedding network. In Figure 4(a) and 4(d), we observed that after the argumentation network converged to a certain level, the encoding network still benefited from that. In Figure 4(b), 4(c), and 4(e), they have the same patterns that the augmentation loss arrived the convergence level almost the same as the contrastive loss. While the situation was different in Figure 4(f), at the beginning the augmentation network benefited from encoding loss, then two losses converged gradually.

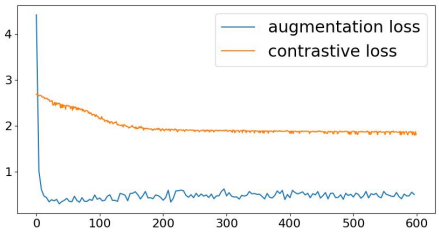


(a) ETTh_1

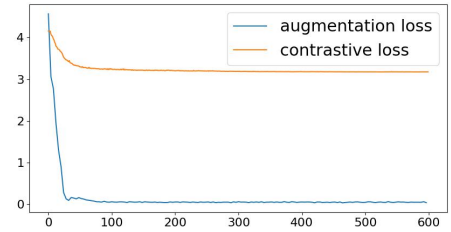
(b) ETTh_2



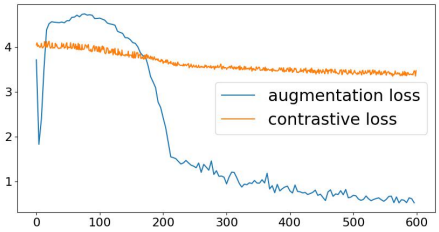
(c) ETTm_1



(d) Elec.



(e) WTH

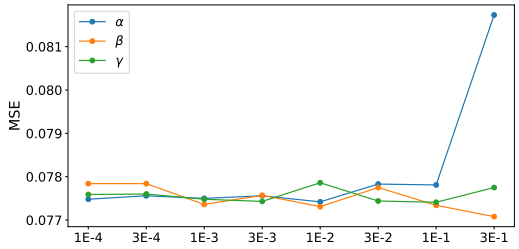


(f) Lora

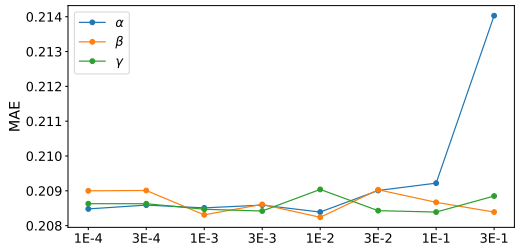
Figure 4: The augmentation loss, Eq. (11) and contrastive loss, Eq. (16), in the training process

Parameter sensitivity studies

In the proposed AutoTCL, we have three hyper-parameters, α in Eq. (16), β in Eq. (10), and γ in Eq. (11), to get the trade-off in training the augmentation network and the encoder network. In this part, we chose different values for these three variables in the range from 0.0001 to 0.3 and reported MSE and MAE scores in the ETTh1 dataset. The results of this part could be found in Figure 5 in Apendix C.3. The sensitivity studies result of three hyper-parameters are shown in Figure 5. From this figure, some results could be observed that our method is able to achieve comparative performances with a wide range of choices for these three hyperparameters, indicating the robustness of our method. The β in Eq. (10), and γ in Eq. (11) have the opposite effect as the weight goes up. Second, we observe that small values, such as 0.001, give good performances on ETTh1 datasets as well as others.



(a) MSE



(b) MAE

Figure 5: Parameter sensitivity studies on ETTh_1.

Case study

To further explore the augmentation of AutoTCL, we have done the case study in this section. We selected three instances to show the effectiveness of our method in Figure 6. As shown in Figure 6, we used the CrocketX dataset as input instances and got the informative part by using the augmentation network to get masks, the result of $h(x)$. From the results, our method could find the informative part in the whole input. With the regularization loss help in Eq. (7), our method could have a continuous mask that makes the informative part more consistent.

Performance with TS2vec as backbone

As a general framework for time series contrastive learning, AutoTCL can be used as a plug-and-play component to boost performance. To further verify the generalization capacity of AutoTCL, in this part, we adopt Ts2vec [Yue *et al.*, 2022] as the backbone. Comparison results are shown in Table 8. We can draw similar conclusions that by adaptively selecting the optimal augmentations with the principle of relevant information, AutoTCL can outperform the vanilla Ts2vec and other baselines.

C.4 Full experiments

Univariate forecasting. Full experiment results of univariate time series forecasting results can be found in Table 5. In these experiments, AutoTCL achieved minimum error in most cases. Compare to the state-of-the-art CoST method, AutoTCL reduced the average MSE error by 6.5% and the average MAE by 4.8%.

Multivariate forecasting. We provided our full experiment results of multivariate time series forecasting results in Table 6. In multivariate forecasting tasks, our method achieved fewer best results than univariate forecasting. AutoTCL reduced the average MSE error by 2.9% and the average MAE by 1.2% than CoST. In the column of stemGNN, because of the error out-of-memory, we can't report part results.

Effectiveness of automatic data augmentation and factorization. The full ablation studies with CoST/TS2Vec backbone are provided in Table 7 and Table 8. From the results, we found **Jitter** is the second-best effective augmentation on average while **Adversial** has almost the same result as **No** $g(x)$. These results show both $g(x)$ and ΔV are necessary for augmentation in AutoTCL.

Classification. In Table 9, the full results of 30 class datasets are provided. AutoTCL is the most powerful method than other baselines with the highest average accuracy rate and ranks. Due to the out-of-memory errors, some items could not be filled with accurate results and we left them blank.

Table 5: Univariate time series forecasting results.

		AutoTCL		TS2Vec		Informer		LogTrans		N-BEATS		TCN		CoST		TNC		TS-TCC		InfoTS	
Dataset	L_y	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	**0.037**	**0.148**	0.039	0.152	0.098	0.247	0.103	0.259	0.094	0.238	0.075	0.210	0.040	0.152	0.057	0.184	0.103	0.237	0.039	0.149
	48	**0.054**	**0.176**	0.062	0.191	0.158	0.319	0.167	0.328	0.210	0.367	0.227	0.402	0.060	0.186	0.094	0.239	0.139	0.279	0.056	0.179
	168	**0.078**	**0.210**	0.134	0.282	0.183	0.346	0.207	0.375	0.232	0.391	0.316	0.493	0.097	0.236	0.171	0.329	0.253	0.408	0.100	0.239
	336	**0.093**	**0.231**	0.154	0.310	0.222	0.387	0.230	0.398	0.232	0.388	0.306	0.495	0.112	0.258	0.192	0.357	0.155	0.318	0.117	0.264
	720	**0.120**	**0.272**	0.163	0.327	0.269	0.435	0.273	0.463	0.322	0.490	0.390	0.557	0.148	0.306	0.235	0.408	0.190	0.337	0.141	0.302
$ETTh_2$	24	**0.079**	**0.206**	0.090	0.229	0.093	0.240	0.102	0.255	0.198	0.345	0.103	0.249	**0.079**	0.207	0.097	0.238	0.239	0.391	0.081	0.215
	48	0.117	**0.255**	0.124	0.273	0.155	0.314	0.169	0.348	0.234	0.386	0.142	0.290	0.118	0.259	0.131	0.281	0.260	0.405	**0.115**	0.261
	168	0.176	**0.319**	0.208	0.360	0.232	0.389	0.246	0.422	0.331	0.453	0.227	0.376	0.189	0.339	0.197	0.354	0.291	0.420	**0.171**	0.327
	336	0.193	0.344	0.213	0.369	0.263	0.417	0.267	0.437	0.431	0.508	0.296	0.430	0.206	0.360	0.207	0.366	0.336	0.453	**0.183**	**0.341**
	720	0.223	0.373	0.214	0.374	0.277	0.431	0.303	0.493	0.437	0.517	0.325	0.463	0.214	0.371	0.207	0.370	0.362	0.472	**0.194**	**0.357**
$ETTm_1$	24	0.016	0.091	0.015	0.092	0.030	0.137	0.065	0.202	0.054	0.184	0.041	0.157	0.015	0.088	0.019	0.103	0.089	0.228	**0.014**	**0.087**
	48	0.026	0.120	0.027	0.126	0.069	0.203	0.078	0.220	0.190	0.361	0.101	0.257	**0.025**	**0.117**	0.036	0.142	0.134	0.280	**0.025**	**0.117**
	96	**0.036**	0.145	0.044	0.161	0.194	0.372	0.199	0.386	0.183	0.353	0.142	0.311	0.038	0.147	0.054	0.178	0.159	0.305	**0.036**	**0.142**
	288	**0.063**	**0.191**	0.103	0.246	0.401	0.554	0.411	0.572	0.186	0.362	0.318	0.472	0.077	0.209	0.098	0.244	0.204	0.327	0.071	0.200
	672	**0.090**	**0.225**	0.156	0.307	0.512	0.644	0.598	0.702	0.197	0.368	0.397	0.547	0.113	0.257	0.136	0.290	0.206	0.354	0.102	0.240
Elec.	24	**0.241**	**0.262**	0.260	0.288	0.251	0.275	0.528	0.447	0.427	0.330	0.263	0.279	0.243	0.264	0.252	0.278	0.379	0.561	0.245	0.269
	48	**0.287**	**0.292**	0.319	0.324	0.346	0.339	0.409	0.414	0.551	0.392	0.373	0.344	0.292	0.300	0.300	0.308	0.453	0.600	0.294	0.301
	168	**0.394**	**0.365**	0.427	0.394	0.544	0.424	0.959	0.612	0.893	0.538	0.609	0.462	0.405	0.375	0.412	0.384	0.575	0.616	0.402	0.367
	336	0.543	0.460	0.565	0.474	0.713	0.512	1.079	0.639	1.035	0.669	0.855	0.606	0.560	0.473	0.548	0.466	0.637	0.633	**0.533**	**0.453**
WTH	24	**0.093**	**0.211**	0.096	0.215	0.117	0.251	0.136	0.279	0.136	0.264	0.109	0.217	0.096	0.213	0.102	0.221	0.221	0.386	0.101	0.222
	48	**0.131**	**0.256**	0.140	0.264	0.178	0.318	0.206	0.356	0.198	0.319	0.143	0.269	0.138	0.262	0.139	0.264	0.255	0.406	0.141	0.266
	168	**0.182**	**0.311**	0.207	0.335	0.266	0.398	0.309	0.439	0.309	0.420	0.188	0.319	0.207	0.334	0.198	0.328	0.339	0.458	0.199	0.328
	336	0.195	0.325	0.231	0.360	0.297	0.416	0.359	0.484	0.369	0.460	**0.192**	**0.320**	0.230	0.356	0.215	0.347	0.372	0.491	0.220	0.351
	720	**0.198**	0.330	0.233	0.365	0.359	0.466	0.388	0.499	0.270	0.406	**0.198**	**0.329**	0.242	0.370	0.219	0.353	0.322	0.467	0.218	0.353
Lora	24	**0.052**	**0.141**	0.212	0.268	0.917	0.720	0.264	0.371	0.072	0.170	0.981	0.899	0.053	0.144	0.206	0.273	0.365	0.514	0.058	0.149
	48	**0.080**	**0.181**	0.267	0.316	1.067	0.786	0.364	0.424	0.115	0.223	0.981	0.898	0.082	0.184	0.286	0.349	0.426	0.562	0.090	0.192
	168	**0.155**	**0.263**	0.355	0.389	1.745	1.067	0.452	0.465	0.286	0.350	1.276	0.946	0.166	0.274	0.523	0.549	0.481	0.587	0.156	0.267
	336	**0.229**	**0.335**	0.425	0.441	1.661	1.050	0.950	0.683	0.405	0.429	1.273	0.943	0.252	0.355	0.772	0.724	0.588	0.645	0.313	0.386
	720	**0.370**	**0.445**	0.523	0.509	2.482	1.370	1.248	0.807	0.679	0.573	1.290	0.950	0.379	0.451	1.313	0.929	0.592	0.649	1.047	0.635
Avg.		**0.157**	**0.258**	0.207	0.301	0.486	0.477	0.382	0.441	0.320	0.388	0.419	0.465	0.168	0.271	0.256	0.340	0.315	0.441	0.188	0.274

Table 6: Multivariate time series forecasting results.

		AutoTCL		TS2Vec		Informer		LogTrans		StemGNN		TCN		CoST		TNC		TS-TCC		InfoTS	
Dataset	L_y	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	0.389	0.439	0.599	0.534	0.577	0.549	0.686	0.604	0.614	0.571	0.767	0.612	**0.386**	**0.429**	0.708	0.592	0.516	0.508	0.564	0.520
	48	0.447	0.477	0.629	0.555	0.685	0.625	0.766	0.757	0.748	0.618	0.713	0.617	**0.437**	**0.464**	0.749	0.619	0.644	0.579	0.607	0.553
	168	**0.615**	**0.574**	0.755	0.636	0.931	0.752	1.002	0.846	0.663	0.608	0.995	0.738	0.643	0.582	0.884	0.699	0.678	0.619	0.746	0.638
	336	**0.802**	**0.671**	0.907	0.717	1.128	0.873	1.362	0.952	0.927	0.730	1.175	0.800	0.812	0.679	1.020	0.768	0.967	0.754	0.904	0.722
	720	1.028	0.789	1.048	0.790	1.215	0.896	1.397	1.291	–	–	1.453	1.311	0.970	0.771	1.157	0.830	**0.935**	**0.715**	1.098	0.811
$ETTh_2$	24	**0.337**	**0.433**	0.398	0.461	0.720	0.665	0.828	0.750	1.292	0.883	1.365	0.888	0.447	0.502	0.612	0.595	0.782	0.666	0.383	0.462
	48	0.572	**0.576**	0.578	0.573	1.457	1.001	1.806	1.034	1.099	0.847	1.395	0.960	0.699	0.637	0.840	0.716	1.357	0.881	**0.567**	0.582
	168	**1.470**	**0.947**	1.901	1.065	3.489	1.515	4.070	1.681	2.282	1.228	3.166	1.407	1.549	0.982	2.359	1.213	4.318	1.728	1.789	1.048
	336	**1.685**	**1.027**	2.304	1.215	2.723	1.340	3.875	1.763	3.086	1.351	3.256	1.481	1.749	1.042	2.782	1.349	2.097	1.145	2.120	1.161
	720	**1.890**	**1.092**	2.650	1.373	3.467	1.473	3.913	1.552	–	–	3.690	1.588	1.971	**1.092**	2.753	1.394	2.047	1.127	2.511	1.316
$ETTm_1$	24	0.256	0.339	0.443	0.436	0.323	0.369	0.419	0.412	0.620	0.570	0.324	0.374	**0.246**	**0.329**	0.522	0.472	0.403	0.455	0.391	0.408
	48	0.339	0.396	0.582	0.515	0.494	0.503	0.507	0.583	0.744	0.628	0.477	0.450	**0.331**	**0.386**	0.695	0.567	0.618	0.552	0.503	0.475
	96	**0.376**	0.422	0.622	0.549	0.678	0.614	0.768	0.792	0.709	0.624	0.636	0.602	0.378	**0.419**	0.731	0.595	0.607	0.572	0.537	0.503
	288	**0.464**	**0.484**	0.709	0.609	1.056	0.786	1.462	1.320	0.843	0.683	1.270	1.351	0.472	0.486	0.818	0.649	0.722	0.638	0.653	0.579
	672	**0.608**	**0.566**	0.786	0.655	1.192	0.926	1.669	1.461	–	–	1.381	1.467	0.620	0.574	0.932	0.712	0.708	0.601	0.757	0.642
Elec.	24	0.153	0.250	0.287	0.374	0.312	0.387	0.297	0.374	0.439	0.388	0.305	0.384	**0.136**	**0.242**	0.354	0.423	0.379	0.561	0.255	0.350
	48	0.167	0.264	0.307	0.388	0.392	0.431	0.316	0.389	0.413	0.455	0.317	0.392	**0.153**	**0.258**	0.376	0.438	0.453	0.600	0.279	0.368
	168	0.179	**0.275**	0.332	0.407	0.515	0.509	0.426	0.466	0.506	0.518	0.358	0.423	**0.175**	**0.275**	0.402	0.456	0.575	0.616	0.302	0.385
	336	0.199	0.297	0.349	0.420	0.759	0.625	0.365	0.417	0.647	0.596	0.349	0.416	**0.196**	**0.296**	0.417	0.466	0.637	0.633	0.320	0.399
WTH	24	0.302	0.364	0.307	0.363	0.335	0.381	0.435	0.477	**0.283**	0.507	0.321	0.367	0.298	**0.360**	0.320	0.373	0.356	0.463	0.316	0.369
	48	0.361	0.412	0.374	0.418	0.395	0.459	0.426	0.495	**0.337**	0.573	0.386	0.423	0.359	**0.411**	0.380	0.421	0.429	0.500	0.381	0.420
	168	0.455	**0.484**	0.491	0.506	0.608	0.567	0.727	0.671	**0.397**	0.652	0.491	0.501	0.464	0.491	0.479	0.495	0.511	0.550	0.490	0.501
	336	0.487	**0.505**	0.525	0.530	0.702	0.620	0.754	0.670	**0.394**	0.639	0.502	0.507	0.497	0.517	0.505	0.514	0.575	0.584	0.532	0.527
	720	0.508	0.519	0.556	0.552	0.831	0.731	0.885	0.773	–	–	**0.498**	**0.508**	0.533	0.542	0.519	0.525	0.545	0.577	0.554	0.543
Lora	24	0.198	0.252	0.212	0.267	0.376	0.345	0.456	0.394	**0.161**	0.373	0.854	0.775	0.202	0.259	0.264	0.302	0.365	0.514	0.198	**0.243**
	48	0.254	0.301	0.266	0.316	0.428	0.420	0.663	0.467	**0.204**	0.439	0.851	0.774	0.258	0.307	0.319	0.345	0.426	0.562	0.254	**0.297**
	168	0.346	0.377	0.354	0.389	0.734	0.597	0.682	0.510	**0.270**	0.536	1.118	0.839	0.350	0.383	0.474	0.477	0.481	0.587	0.345	**0.374**
	336	0.414	0.428	0.425	0.441	0.995	0.738	1.068	0.608	**0.395**	0.618	1.111	0.836	0.417	0.432	0.625	0.588	0.588	0.645	0.412	**0.427**
	720	0.517	0.502	0.522	0.509	1.181	0.831	0.959	0.622	–	–	1.131	0.844	0.524	0.507	1.266	0.876	0.592	0.649	**0.514**	**0.501**
Avg.		**0.545**	**0.499**	0.697	0.571	0.990	0.708	1.138	0.798	0.753	0.651	1.057	0.781	0.561	0.505	0.837	0.637	0.838	0.675	0.665	0.556

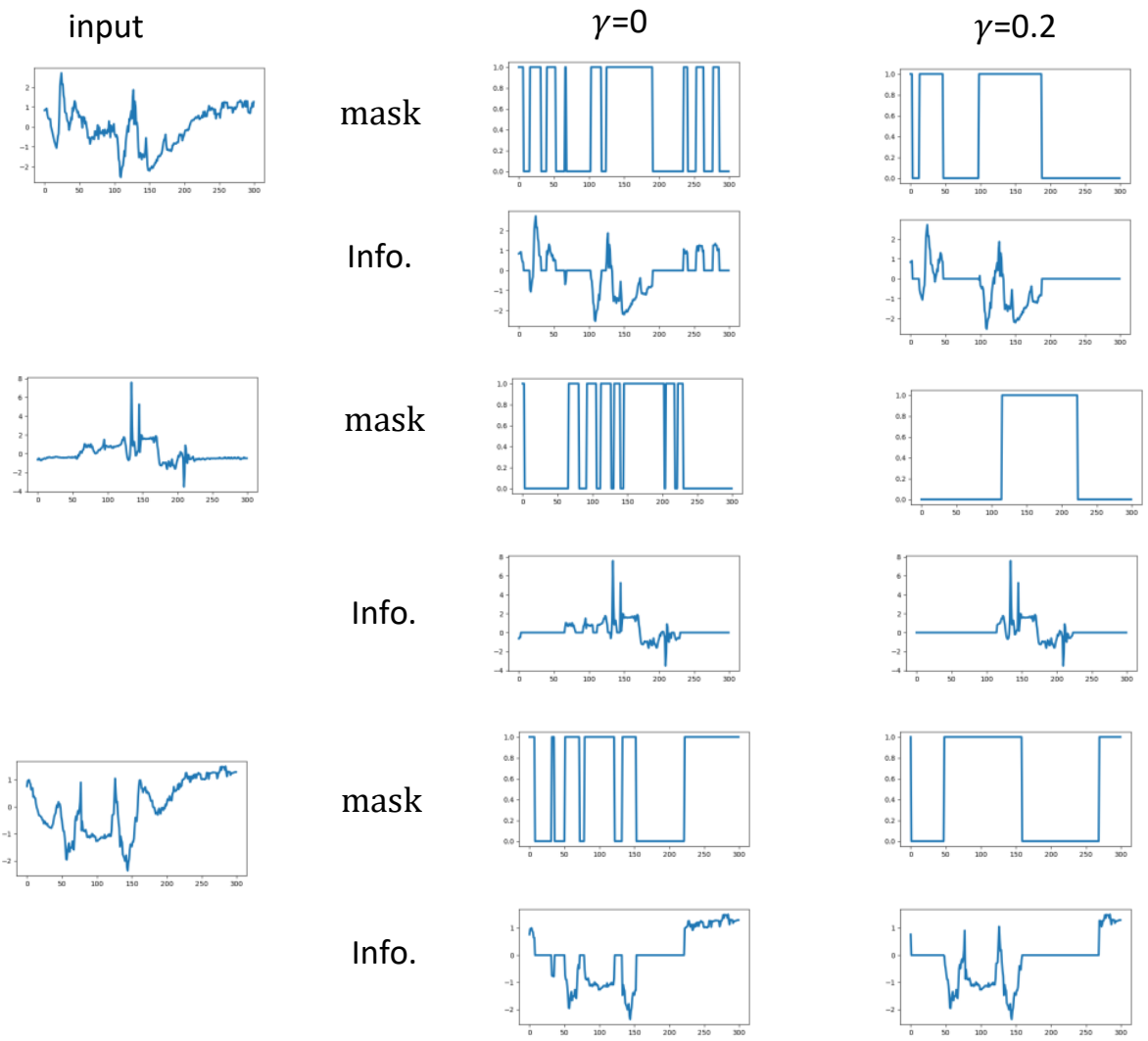


Figure 6: Case study of AutoTCL. The inputs are from the CricketX dataset, which is a univariate time series dataset. The first column is the original input instance and we demonstrate the augmented instances in two settings, w/ and w/o regularization loss in Eq.(11), which are the second and third columns. The odd rows are the masks, outputs of $g(x)$ and the even rows are informative parts of the original instances guided by the masks.

Table 7: Ablation studies using CoST backbone

		AutoTCL		W/o $h(x)$		W/o $g(x)$		W/o ΔV		W/o Aug		Cutout		Jitter		Adversarial	
Dataset	L_y	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	**0.037**	0.148	**0.037**	0.148	**0.037**	0.148	0.038	0.149	**0.037**	0.148	**0.037**	**0.147**	0.038	**0.147**	0.039	0.149
	48	0.054	0.176	**0.054**	0.176	0.054	0.177	0.055	0.180	0.055	0.178	**0.053**	**0.175**	0.054	0.176	0.056	0.180
	168	**0.078**	**0.210**	0.079	0.210	0.080	0.211	0.083	0.217	0.100	0.237	**0.078**	**0.210**	0.081	0.212	0.090	0.227
	336	0.093	0.231	0.093	0.231	0.094	0.232	0.096	0.234	0.108	0.251	**0.092**	**0.230**	0.095	0.233	0.106	0.250
	720	**0.120**	**0.272**	0.121	0.274	0.124	0.277	0.157	0.317	0.175	0.340	0.179	0.345	0.163	0.325	0.152	0.313
$ETTh_2$	24	0.079	0.206	0.077	**0.204**	**0.076**	0.205	0.079	0.209	0.078	0.208	**0.076**	**0.204**	0.092	0.217	0.078	0.207
	48	0.117	0.255	0.124	0.258	0.113	0.256	0.118	0.259	0.127	0.259	**0.110**	**0.253**	0.135	0.272	0.124	0.265
	168	**0.176**	**0.319**	0.191	0.329	0.212	0.346	0.240	0.358	0.220	0.347	0.191	0.340	0.207	0.356	0.227	0.361
	336	**0.193**	**0.344**	0.201	0.350	0.243	0.371	0.204	0.349	0.200	0.357	0.201	0.355	0.212	0.366	0.253	0.375
	720	0.223	**0.373**	0.246	0.384	0.246	0.380	0.238	0.379	0.227	0.374	0.220	0.376	**0.217**	0.374	0.251	0.385
$ETTm_1$	24	0.016	0.091	0.014	0.087	0.015	0.090	0.015	0.089	**0.013**	**0.085**	0.017	0.092	0.015	0.091	0.015	0.092
	48	0.026	0.120	0.025	0.119	0.025	0.117	0.027	0.122	**0.024**	**0.116**	0.028	0.123	0.027	0.124	0.028	0.125
	96	**0.036**	0.145	0.037	0.146	0.038	0.146	0.039	0.150	**0.036**	**0.144**	0.039	0.150	0.043	0.158	0.040	0.151
	288	**0.063**	**0.191**	0.074	0.205	0.072	0.204	0.072	0.205	0.080	0.216	0.078	0.211	0.082	0.218	0.075	0.205
	672	**0.090**	**0.225**	0.108	0.250	0.098	0.239	0.104	0.248	0.114	0.248	0.104	0.246	0.112	0.260	0.100	0.240
Elec.	24	**0.240**	0.266	0.244	0.266	0.241	0.267	0.242	**0.264**	0.243	0.272	0.241	**0.264**	**0.240**	**0.264**	0.242	0.265
	48	0.285	0.294	0.291	0.295	0.285	0.295	0.287	0.294	0.290	0.300	0.286	**0.291**	**0.284**	0.292	0.287	0.294
	168	**0.392**	0.371	0.400	0.371	**0.392**	0.366	0.394	0.367	0.398	0.372	0.394	0.365	0.395	**0.362**	0.393	0.364
	336	0.542	0.461	0.547	0.465	0.541	0.464	0.542	0.461	0.541	0.470	0.545	0.460	0.545	**0.457**	**0.539**	**0.457**
WTH	24	0.093	0.211	0.098	0.220	0.092	0.209	**0.091**	**0.207**	0.096	0.215	0.092	0.210	0.093	0.212	0.092	0.209
	48	0.131	0.256	0.139	0.266	0.130	0.255	**0.127**	**0.250**	0.141	0.266	0.129	0.252	0.134	0.260	0.131	0.257
	168	**0.182**	**0.311**	0.194	0.324	0.185	0.314	0.184	0.316	0.208	0.336	0.186	0.315	0.195	0.327	0.185	0.315
	336	**0.195**	**0.325**	0.210	0.342	0.208	0.342	0.206	0.341	0.231	0.357	0.209	0.339	0.215	0.349	0.203	0.335
	720	**0.198**	**0.330**	0.218	0.353	0.217	0.353	0.211	0.349	0.240	0.369	0.220	0.352	0.231	0.370	0.218	0.352
Lora	24	**0.052**	**0.141**	0.060	0.152	0.138	0.219	0.128	0.213	0.078	0.171	0.067	0.170	0.158	0.223	0.057	0.154
	48	**0.080**	**0.181**	0.092	0.196	0.117	0.225	0.181	0.264	0.127	0.223	0.112	0.218	0.185	0.257	0.084	0.189
	168	**0.155**	**0.263**	0.246	0.317	0.196	0.308	0.232	0.334	0.481	0.393	0.676	0.433	0.311	0.359	0.235	0.323
	336	**0.229**	**0.335**	0.302	0.372	0.395	0.444	0.363	0.433	0.941	0.532	1.403	0.619	0.378	0.429	0.535	0.475
	720	**0.370**	**0.445**	0.483	0.509	1.60	0.729	0.617	0.561	1.926	0.739	1.655	0.771	0.395	0.461	1.315	0.722
Avg.		**0.157**	**0.258**	0.173	0.270	0.216	0.282	0.185	0.280	0.260	0.294	0.266	0.394	0.184	0.281	0.212	0.284

Table 8: Ablation studies using TS2Vec backbone

		AutoTCL		W/o $h(x)$		W/o $g(x)$		W/o ΔV		W/o Aug		Cutout		Jitter		Adversarial	
Dataset	L_y	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	**0.039**	**0.146**	**0.039**	0.148	0.047	0.165	0.047	0.164	**0.039**	0.149	0.043	0.155	0.040	0.150	0.041	0.151
	48	**0.058**	**0.180**	0.060	0.185	0.075	0.212	0.073	0.205	0.063	0.190	0.063	0.187	0.063	0.190	0.062	0.186
	168	**0.106**	**0.245**	0.115	0.259	0.145	0.298	0.131	0.278	0.119	0.264	0.118	0.262	0.111	0.253	0.114	0.255
	336	**0.121**	**0.266**	0.139	0.289	0.159	0.317	0.148	0.303	0.141	0.291	0.133	0.285	0.130	0.277	0.132	0.280
	720	**0.154**	**0.314**	0.181	0.347	0.198	0.365	0.180	0.344	0.193	0.359	0.167	0.331	0.164	0.323	0.167	0.330
$ETTh_2$	24	0.106	0.252	0.108	0.250	**0.095**	**0.235**	0.104	0.248	0.108	0.251	0.105	0.250	0.105	0.249	0.107	0.252
	48	0.131	0.284	0.134	0.285	**0.129**	**0.279**	0.136	0.289	0.140	0.290	0.133	0.287	0.135	0.287	0.137	0.288
	168	**0.182**	**0.343**	0.185	0.344	0.212	0.365	0.211	0.366	0.203	0.360	0.198	0.355	0.194	0.353	0.195	0.353
	336	**0.190**	**0.351**	0.191	**0.351**	0.205	0.362	0.209	0.366	0.206	0.367	0.204	0.363	0.201	0.362	0.199	0.360
	720	0.204	0.370	0.204	0.368	0.203	0.366	0.200	**0.364**	0.205	0.369	0.205	0.367	0.200	**0.364**	**0.194**	0.359
$ETTm_1$	24	**0.014**	**0.085**	0.018	0.098	0.015	0.089	**0.014**	0.087	**0.014**	0.087	0.015	0.089	**0.014**	0.087	**0.014**	**0.085**
	48	**0.026**	**0.117**	0.027	0.121	0.028	0.123	**0.026**	0.120	0.027	0.121	0.027	0.121	0.027	0.121	**0.026**	**0.117**
	96	**0.038**	**0.147**	0.039	0.149	0.039	**0.147**	0.041	0.153	0.041	0.153	0.041	0.153	**0.038**	**0.147**	**0.038**	**0.147**
	288	**0.081**	0.216	0.083	0.219	0.082	0.216	0.084	0.222	0.084	0.222	0.084	0.222	**0.081**	**0.215**	**0.081**	0.216
	672	**0.119**	**0.263**	0.123	0.269	0.122	0.266	0.124	0.271	0.124	0.270	0.121	0.267	0.120	0.265	**0.119**	**0.263**
Elec.	24	**0.247**	**0.269**	0.249	0.271	0.248	**0.269**	**0.247**	0.270	0.250	0.271	0.248	0.270	0.249	0.273	0.250	0.270
	48	0.297	**0.301**	0.302	0.306	0.297	**0.301**	0.297	0.303	0.298	0.302	**0.296**	0.302	0.297	0.307	0.298	0.302
	168	**0.408**	0.380	0.413	0.381	**0.408**	0.380	0.410	0.380	**0.408**	**0.377**	**0.408**	0.383	0.410	0.384	**0.408**	**0.377**
	336	**0.541**	**0.468**	0.553	0.472	**0.541**	0.469	0.547	0.470	0.542	0.471	0.545	0.470	0.470	0.547	0.542	0.471
WTH	24	**0.093**	0.212	0.096	0.214	0.094	**0.210**	0.096	0.214	0.094	0.211	0.099	0.215	0.096	0.213	0.096	0.213
	48	0.133	0.258	0.134	0.259	**0.130**	**0.253**	0.134	0.258	0.134	0.257	0.132	0.256	0.135	0.259	0.133	0.254
	168	0.188	0.316	0.192	0.322	**0.184**	**0.313**	0.192	0.322	0.197	0.324	0.189	0.317	0.192	0.321	0.193	0.322
	336	**0.201**	**0.333**	0.208	0.341	0.202	0.335	0.208	0.341	0.216	0.347	0.208	0.338	0.211	0.342	0.212	0.344
	720	**0.204**	**0.339**	0.210	0.347	**0.204**	**0.339**	0.210	0.347	0.225	0.357	0.211	0.343	0.220	0.353	0.217	0.352
Lora	24	0.193	0.239	**0.192**	**0.238**	0.196	0.244	**0.192**	**0.238**	0.195	0.239	0.193	0.240	0.194	0.240	0.193	0.241
	48	0.250	**0.293**	0.251	0.295	0.253	0.299	**0.249**	**0.293**	0.251	**0.293**	0.252	0.297	0.251	0.294	0.253	0.297
	168	**0.339**	**0.370**	0.344	0.375	0.340	0.372	**0.339**	**0.370**	0.341	0.371	0.345	0.377	0.341	0.372	0.355	0.382
	336	**0.397**	**0.418**	0.409	0.427	0.398	**0.418**	0.399	**0.418**	0.400	0.420	0.410	0.428	0.402	0.421	0.431	0.438
	720	**0.478**	**0.479**	0.505	0.498	0.480	0.480	0.488	0.486	0.484	0.482	0.501	0.496	0.492	0.489	0.547	0.514
Avg.		**0.191**	**0.285**	0.197	0.291	0.198	0.293	0.198	0.293	0.198	0.292	0.196	0.290	0.195	0.289	0.198	0.290

Table 9: Classification result of the UEA dataset

Dataset	AutoTCL	TS2Vec	T-Loss	TNC	TS-TCC	TST	DTW	TF-C	InfoTS
ArticularyWordRecognition	0.983	0.987	0.943	0.973	0.953	0.977	0.987	0.467	**0.993**
AtrialFibrillation	**0.467**	0.200	0.133	0.133	0.267	0.067	0.200	0.040	0.267
BasicMotions	**1.000**	0.975	**1.000**	0.975	**1.000**	0.975	0.975	0.475	**1.000**
CharacterTrajectories	0.976	**0.995**	0.993	0.967	0.985	0.975	0.989	0.090	0.987
Cricket	**1.000**	0.972	0.972	0.958	0.917	**1.000**	**1.000**	0.125	**1.000**
DuckDuckGeese	**0.700**	0.680	0.650	0.460	0.380	0.620	0.600	0.340	0.600
EigenWorms	**0.901**	0.847	0.840	0.840	0.779	0.748	0.618	–	0.748
Epilepsy	0.978	0.964	0.971	0.957	0.957	0.949	0.964	0.217	**0.993**
ERing	0.944	0.874	0.133	0.852	0.904	0.874	0.133	0.167	**0.953**
EthanolConcentration	**0.354**	0.308	0.205	0.297	0.285	0.262	0.323	0.247	0.323
FaceDetection	**0.581**	0.501	0.513	0.536	0.544	0.534	0.529	0.502	0.525
FingerMovements	**0.640**	0.480	0.580	0.470	0.460	0.560	0.530	0.510	0.620
HandMovementDirection	0.432	0.338	0.351	0.324	0.243	0.243	0.231	0.405	**0.514**
Handwriting	0.384	0.515	0.451	0.249	0.498	0.225	0.286	0.051	**0.554**
Heartbeat	**0.785**	0.683	0.741	0.746	0.751	0.746	0.717	0.737	0.771
JapaneseVowels	0.984	0.984	**0.989**	0.978	0.930	0.978	0.949	0.135	0.986
Libras	0.833	0.867	0.883	0.817	0.822	0.656	0.870	0.067	**0.889**
LSST	0.554	0.537	0.509	**0.595**	0.474	0.408	0.551	0.314	0.593
MotorImagery	0.570	0.510	0.580	0.500	**0.610**	0.500	0.500	0.500	**0.610**
NATOPS	**0.944**	0.928	0.917	0.911	0.822	0.850	0.883	0.533	0.939
PEMS-SF	**0.838**	0.682	0.676	0.699	0.734	0.740	0.711	0.312	0.757
PenDigits	0.984	**0.989**	0.981	0.979	0.974	0.560	0.977	0.236	**0.989**
PhonemeSpectra	0.218	0.233	0.222	0.207	**0.252**	0.085	0.151	0.026	0.233
RacketSports	**0.914**	0.855	0.855	0.776	0.816	0.809	0.803	0.480	0.829
SelfRegulationSCP1	**0.891**	0.812	0.843	0.799	0.823	0.754	0.775	0.502	0.887
SelfRegulationSCP2	**0.578**	**0.578**	0.539	0.550	0.533	0.550	0.539	0.500	0.527
SpokenArabicDigits	0.925	0.932	0.905	0.934	0.970	0.923	0.963	0.100	**0.988**
StandWalkJump	**0.533**	0.467	0.333	0.400	0.333	0.267	0.200	0.333	0.467
UWaveGestureLibrary	0.893	0.884	0.875	0.759	0.753	0.575	0.903	0.125	**0.906**
InsectWingbeat	**0.488**	0.466	0.156	0.469	0.264	0.105	–	0.108	0.472
Avg. ACC	**0.742**	0.704	0.658	0.670	0.668	0.617	0.629	0.298	0.730
Avg. RANK	**2.300**	3.700	4.667	5.433	5.133	6.133	5.400	8.200	2.367