Self-Supervised Contrastive Pre-Training for Time Series via Time-Frequency Consistency

Xiang Zhang[†] [*]
Harvard University
xiang_zhang@hms.harvard.edu

Ziyuan Zhao[*]
Harvard University
ziyuanzhao@college.harvard.edu

Theodoros Tsiligkaridis
MIT Lincoln Laboratory
ttsili@ll.mit.edu

Marinka Zitnik
Harvard University
marinka@hms.harvard.edu

Abstract

Pre-training on time series poses a unique challenge due to the potential mismatch between pre-training and target domains, such as shifts in temporal dynamics, fast-evolving trends, and long-range and short-cyclic effects, which can lead to poor downstream performance. While domain adaptation methods can mitigate these shifts, most methods need examples directly from the target domain, making them suboptimal for pre-training. To address this challenge, methods need to accommodate target domains with different temporal dynamics and be capable of doing so without seeing any target examples during pre-training. Relative to other modalities, in time series, we expect that time-based and frequency-based representations of the same example are located close together in the time-frequency space. To this end, we posit that time-frequency consistency (TF-C) — embedding a time-based neighborhood of an example close to its frequency-based neighborhood — is desirable for pre-training. Motivated by TF-C, we define a decomposable pre-training model, where the self-supervised signal is provided by the distance between time and frequency components, each individually trained by contrastive estimation. We evaluate the new method on eight datasets, including electrodiagnostic testing, human activity recognition, mechanical fault detection, and physical status monitoring. Experiments against eight state-of-the-art methods show that TF-C outperforms baselines by 15.4% (F1 score) on average in one-to-one settings (*e.g.*, fine-tuning an EEG-pretrained model on EMG data) and by 8.4% (precision) in challenging one-to-many settings (*e.g.*, fine-tuning an EEG-pretrained model for either hand-gesture recognition or mechanical fault prediction), reflecting the breadth of scenarios that arise in real-world applications. The source code and datasets are available at https://github.com/mims-harvard/TFC-pretraining.

1 Introduction

Time series plays important roles in many areas, including clinical diagnosis, traffic analysis, and climate science [1, 2, 3, 4, 5, 6]. While representation learning has considerably advanced analysis of time series [7, 8, 9] more broadly [10], learning generalizable representations for temporal data remains a fundamentally challenging problem [8, 11]. There are numerous immediate benefits from generating such representations, of which pre-training capability is particularly desirable and of

[*]These authors contributed equally to this work.
[†]Present address: University of North Carolina at Charlotte, xiang.zhang@uncc.edu

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

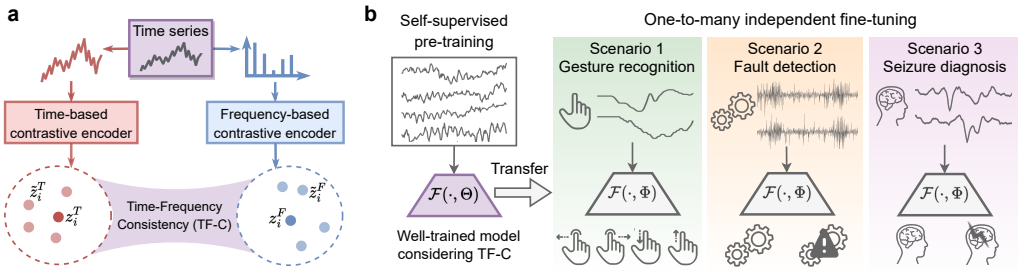

Figure 1: a. Illustration of Time-Frequency Consistency (TF-C). Time-based embedding $\boldsymbol{z}_i^{\mathrm{T}}$ and frequency-based embedding $\boldsymbol{z}_i^{\mathrm{F}}$ of time series sample $\boldsymbol{x}_i^{\mathrm{T}}$, along with $\tilde{\boldsymbol{z}}_i^{\mathrm{T}}$ and $\tilde{\boldsymbol{z}}_i^{\mathrm{F}}$ learned from augmentations of $\boldsymbol{x}_i^{\mathrm{T}}$, should be close to each other in the latent time-frequency space. **b.** Leveraging TF-C property in time series to optimize a pre-training model $\mathcal{F}$ with parameters Θ that get fine-tuned to Φ on a small scenario-specific dataset.

great practical importance [12, 13]. Central to pre-training is a question of how to process time series in a diverse dataset to greatly improve generalization on new time series coming from different datasets [14, 15, 10]. By training a neural network model on a dataset and transferring it to a new target dataset for fine-tuning, *i.e.*, without explicit retraining on that target data, we expect the resulting performance to be at least as good as that of state-of-the-art models tailored to the target dataset.

However, unfortunately, the expected performance gains are often not realized for a variety of reasons (*e.g.*, distribution shifts, properties of the target dataset unknown during pre-training) [16, 17] that get compounded by the complexity of time series: large variations of temporal dynamics across datasets, varying semantic meaning, irregular sampling, system factors (*e.g.*, different devices or subjects), etc. [18, 17]. This complexity of time series limits the utility of knowledge transfer for pre-training [19, 20]. For example, pre-training a model on a diverse time series dataset with mostly low-frequency components (smooth trends) may not lead to positive transfer on downstream tasks with high-frequency components (transient events) [17]. Examining these challenges can provide clues to what kind of inductive biases could facilitate generalizable representations of time series – this paper offers a strategy for that through a novel time-frequency consistency principle.

In addition, target datasets are not available during pre-training (different from domain adaption [21]; Appendix A), requiring that the pre-training model captures a latent property that holds true for previously unseen target datasets. At the center of this desideratum is the idea of a property that would be shared between pre-training and target datasets and would enable knowledge transfer from pre-training to fine-tuning. In computer vision (CV), pre-training is driven by findings that initial neural layers capture universal visual elements, such as edges and shapes, that are relevant regardless of image style and tasks [22]. In natural language processing (NLP), the foundation for pre-training is given by linguistic principles of semantics and grammar shared across different languages [23]. However, due to the aforementioned temporal complexity, such a principle for pre-training on time series has not yet been established. Moreover, supervised pre-training requires access to large annotated datasets, which limits its use in domains where richly labeled datasets are scarce [24, 25]. For example, in medical applications, labeling data at scale is often infeasible or can be expensive and noisy (experts can disagree on ground-truth labeling [26, 27], *e.g.*, whether an ECG signal indicates a normal vs. abnormal rhythm) [28, 29]. To mitigate these issues, self-supervised learning emerged as a promising strategy to sidestep the lack of labeled datasets [30].

Present work. We introduce a strategy for self-supervised pre-training in time series by modeling Time-Frequency Consistency (TF-C). TF-C specifies that time-based and frequency-based representations, learned from the same time series sample, should be closer to each other in the time-frequency space than representations of different time series samples. Specifically, we adopt contrastive learning in time-space to generate a time-based representation. In parallel, we propose a set of novel augmentations based on the characteristic of the frequency spectrum and produce a frequency-based embedding through contrastive instance discrimination. This is the first work to develop frequency-based contrastive augmentation to leverage rich spectral information and explore time-frequency consistency in time series. The pre-training objective is to minimize the distance between time-based and frequency-based embeddings using a novel consistency loss (Figure 1 (a)). The self-supervised loss is used to optimize the pre-training model and enforce consistency between time and frequency domains in the latent space. The learned relationship encoded in model parameters are transferred to initialize the fine-tuning model and improve performance in datasets of interest (Figure 1 (b)).

We evaluate the TF-C model on eight time series datasets under two evaluation settings (*i.e.*, one-to-one and one-to-many). The eight datasets cover a large set of variations: different numbers of channels (from univariate to 9-channel multivariate), varying time series lengths (from 128 to 5,120), different sampling rates (from 16 Hz to 4,000 Hz), different scenarios (neurological healthcare, human activity recognition, mechanical fault detection, physical status monitoring, etc.) and diverse types of signals (EEG, EMG, ECG, acceleration, and vibration). We compare TF-C approach to eight state-of-the-art baselines. Results show that TF-C achieves positive transfer, outperforming all baselines by a large margin of 15.4% (F1 score) on average. Further, the approach outperforms the strongest baselines with an improvement of up to 7.2% in the F1 score. Finally, the TF-C approach improves prior work by 8.4% in precision (when pre-training the model on sleep EEG signals and fine-tuning it on hand-gesture recognition) in challenging one-to-many setups that apply the same pre-trained model to multiple independent fine-tuning datasets.

2 Related Work

Pre-training for time series. Although there are studies on self-supervised representation learning for time series [7, 8, 31, 32] and self-supervised pre-training for images [33, 34, 35, 24], the intersection of these two areas, *i.e.*, self-supervised pre-training for time series, remains underexplored. In time series, it's not obvious what reasonable assumptions can bridge pre-training and target datasets. Hence, pre-training models in CV [36, 37, 14] and NLP [10, 15, 38] are not directly applicable due to data modality mismatch, and the existing results leave room for improvement [31, 39, 40]. Shi *et al.* [12] developed the only model to date that is explicitly designed for self-supervised time series pre-training. The model captures the local and global temporal pattern, but it is not convincing why the designed pretext task can capture generalizable representations. Although several studies applied transfer learning in the context of time series [7, 8, 18, 41], there is no foundation yet of which conceptual properties are most suitable for pre-training on time series and why. Addressing this gap, we show that TF-C, designed to be invariant to different time-series datasets, can produce generalizable pre-training models.

Unlike domain adaptation [21, 42] that requires access to target datasets during training, pre-training models do not have access to fine-tuning datasets. As a result, one needs to identify a generalizable time-series property to benefit from pre-training. Further, self-supervised domain adaptation does not need labels in the target dataset but still requires labels for model training [43, 44]. In contrast, TF-C does not need any labels during pre-training.

Contrastive learning with time series. Contrastive learning, a popular type of self-supervised learning, aims to learn an encoder that maps inputs into an embedding space such that positive sample pairs (original augmentation and another alternative augmentation/view of the same input sample) are pulled closer and negative sample pairs (original augmentation and an alternative input sample augmentation) are pushed apart [30, 45]. Contrastive learning in time series is less investigated in comparison, partly due to the challenge of identifying augmentations that capture key invariance properties in time series data. For example, CLOCS defines adjacent time segments as positive pairs [41], and TNC assumes overlapping temporal neighborhoods have similar representations [46]. These methods leverage temporal invariance to define positive pairs which are used to calculate contrastive loss, but other invariances, such as transformation invariance (*e.g.*, SimCLR [40]), contextual invariance (*e.g.*, TS2vec [47] and TS-TCC [48]) and augmentations are possible. In this work, we propose an augmentation bank that exploits multiple invariances to generate diverse augmentations (Sec. 4.1), which adds richness to the pre-training model [48]. Importantly, we propose frequency-based augmentations by perturbing the frequency spectrum of time series (*e.g.*, adding or removing the frequency components and manipulating their amplitude; more details in Sec. 4.2) to learn better representations by exposing the model to a local range of frequency variations. In previous work, CoST processes sequential signals through the frequency domain, but the augmentations are still implemented in time space [49]. Similarly, although BTSF [50] involves frequency domain, its data transformation is solely implemented in the time domain using instance-level dropout. Additional commentary on differences between CoST and BTSF is in Appendix B. To the best of our knowledge, this is the first work that directly perturbs the frequency spectrum to leverage frequency-invariance for contrastive learning. Further, we develop a pre-training model that subjects to TF-C upon two individual contrastive encoders.

3 Problem Formulation

We are given a pre-training dataset $\mathcal{D}^{\text{pret}} = \{\boldsymbol{x}_i^{\text{pret}} \mid i = 1, \ldots, N\}$ of unlabeled time series samples where sample $\boldsymbol{x}_i^{\text{pret}}$ has K^{pret} channels and L^{pret} timestamps. Let $\mathcal{D}^{\text{tune}} = \{(\boldsymbol{x}_i^{\text{tune}}, y_i) \mid i = 1, \ldots, M\}$ be a fine-tuning (*i.e.*, target; target and fine-tuning are used interchangeably) dataset of labeled time series samples, each having K^{tune} channels and L^{tune} timestamps. Furthermore, every sample $\boldsymbol{x}_i^{\text{tune}}$ is associated with a label $y_i \in \{1, \ldots, C\}$, where C is the number of classes. Without loss of generality, in the following descriptions, we focus on univariate (single-channel) time series, while noting that our approach can accommodate multivariate time series of varying lengths across datasets (shown in experiments in Sec. 5.2). We use superscript symbol $\tilde{}$ to denote contrastive augmentations. We note that $\boldsymbol{x}_i^{\text{T}} \equiv \boldsymbol{x}_i$ denotes an input time series sample, and $\boldsymbol{x}_i^{\text{F}}$ denotes discrete frequency spectrum of $\boldsymbol{x}_i$.

Problem (Self-Supervised Contrastive Pre-Training For Time Series). *Given are an unlabeled pre-training dataset $\mathcal{D}^{pret}$ with N samples and a target dataset $\mathcal{D}^{tune}$ with M samples ($M \ll N$). The goal is to use $\mathcal{D}^{pret}$ to pre-train a model $\mathcal{F}$ so that by fine-tuning model parameters on $\mathcal{D}^{tune}$, the fine-tuned model produces generalizable representations $\boldsymbol{z}_i^{tune} = \mathcal{F}(\boldsymbol{x}_i^{tune})$ for every $\boldsymbol{x}_i^{tune}$.*

We follow an established setup, *e.g.*, [41]: for pre-training, only the unlabeled dataset $\mathcal{D}^{\text{pret}}$ is available while, for fine-tuning, a small labeled dataset $\mathcal{D}^{\text{tune}}$ can be used. In short, a model $\mathcal{F}$ is pre-trained on the unlabeled time series dataset $\mathcal{D}^{\text{pret}}$ and its optimized model parameters Θ are fine-tuned to go from $\mathcal{F}(\cdot, \Theta)$ to $\mathcal{F}(\cdot, \Phi)$ using the dataset $\mathcal{D}^{\text{tune}}$. The Φ denotes fine-tuned model parameters. Note that this problem (*i.e.*, $\mathcal{D}^{\text{pret}}$ is independent of the target dataset) is distinct from domain adaptation as fine-tuning dataset $\mathcal{D}^{\text{tune}}$ is not accessed during pre-training. As a result, the pre-trained model can be used with many different fine-tuning datasets without re-training.

Rationale for Time-Frequency Consistency (TF-C). The central idea is to identify a general property that is preserved across time series datasets and use it to induce transfer learning for effective pre-training. The time domain shows how sensor readouts change with time, whereas the frequency domain shows how much of the signal lies within each frequency component over the entire spectrum [51]. Explicitly considering the frequency domain can provide an understanding of time series behavior that cannot be directly captured solely in the time domain [52]. However, existing contrastive methods (*e.g.*, [47, 48]) focus exclusively on modeling the time domain and ignore the frequency domain altogether. One can argue that approach is sufficient in the case of high-capacity methods as time and frequency domains are different views of the same data [53], which can be cross-translated using transformation, such as Fourier and inverse Fourier [54, 52]. The relationship between the two domains, grounded in signal processing theory, provides an invariance that is valid regardless of the time series distribution [55, 56] and thus can serve as an inductive bias for pre-training. Appendix C provides a commentary with analogies for images. Approaching this invariance through the lens of representation learning, we next formulate Time-Frequency Consistency (TF-C). The TF-C property postulates there exists a latent time-frequency space such that for every sample $\boldsymbol{x}_i$, time-based representation $\boldsymbol{z}_i^{\text{T}}$ and frequency-based representation $\boldsymbol{z}_i^{\text{F}}$ of the same sample, together with their local augmentations (defined later), are close to each other in the latent space.

Representational Time-Frequency Consistency (TF-C). *Let $\boldsymbol{x}_i$ be a time series and $\mathcal{F}$ be a model satisfying TF-C. Then, time-based representation $\boldsymbol{z}_i^{\text{T}}$ and frequency-based representation $\boldsymbol{z}_i^{\text{F}}$ as well as representations of $\boldsymbol{x}_i$'s local augmentations are proximal in the latent time-frequency space.*

Our strategy is to use dataset $\mathcal{D}^{\text{pret}}$ to induce TF-C in $\mathcal{F}$'s model parameters Θ, which, in turn, are used to initialize the target model on $\mathcal{D}^{\text{tune}}$ and produce generalizable representations for downstream prediction. The invariant nature of TF-C means that the approach can bridge $\mathcal{D}^{\text{pret}}$ and $\mathcal{D}^{\text{tune}}$ even when large discrepancies exist between them (in terms of temporal dynamics, semantic meaning, etc.), providing a vehicle for a general pre-training on time series.

To realize TF-C, our model $\mathcal{F}$ has four components (Figure 2): a time encoder G_{T}, a frequency encoder G_{F}, and two cross-space projectors R_{T} and R_{F} that map time-based and frequency-based representations, respectively, to the same time-frequency space. Together, the four components provide a way to embed $\boldsymbol{x}_i$ to the latent time-frequency space such that the time-based embedding $\boldsymbol{z}_i^{\text{T}} = R_{\text{T}}(G_{\text{T}}(\boldsymbol{x}_i^{\text{T}}))$ and the frequency-based embedding $\boldsymbol{z}_i^{\text{F}} = R_{\text{F}}(G_{\text{F}}(\boldsymbol{x}_i^{\text{F}}))$ are close together.

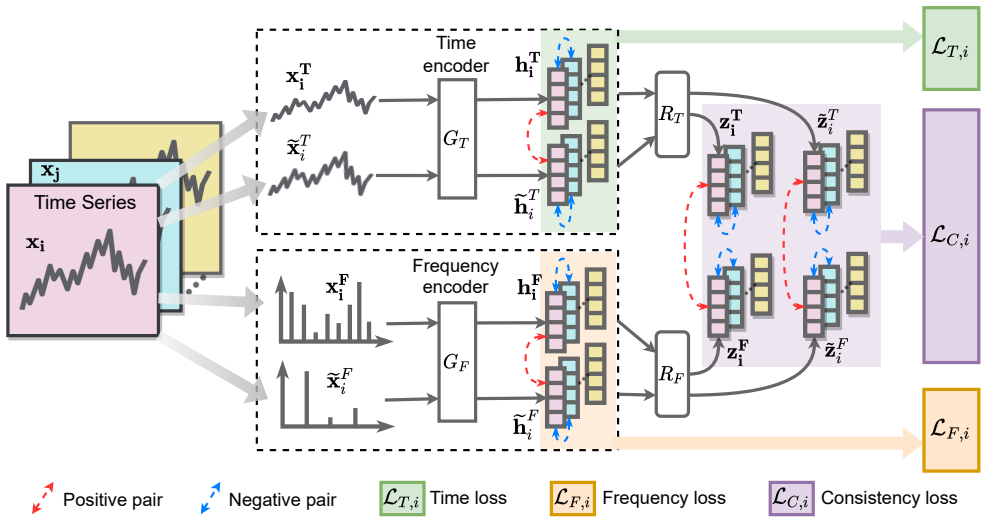

Figure 2: Overview of TF-C approach. Our TF-C pre-training model $\mathcal{F}$ has four components: a time encoder G_{T}, a frequency encoder G_{F}, and two cross-space projectors R_{T} and R_{F}. For an input time series $\boldsymbol{x}_i$, the model produces time-based representations (*i.e.*, $\boldsymbol{z}_i^{\text{T}}$ and $\widetilde{\boldsymbol{z}}_i^{\text{T}}$ of input $\boldsymbol{x}_i$ and its augmented version, respectively) and frequency-based representations (*i.e.*, $\boldsymbol{z}_i^{\text{F}}$ and $\widetilde{\boldsymbol{z}}_i^{\text{F}}$ of input $\boldsymbol{x}_i$ and its augmented version, respectively). The TF-C property is realized by promoting the alignment of time- and frequency-based representations in the latent time-frequency space, providing a vehicle for transferring $\mathcal{F}$ to a target dataset not seen before.

4 Our Approach

Next, we present the architecture of the developed self-supervised contrastive pre-training model $\mathcal{F}$. Unless specified otherwise, the data mentioned in this section are from pre-training dataset and the superscript ^{pret} is omitted for simplification. Here we describe the model using univariate time series as an example, but our model can be straightforwardly applied to multivariate time series (Sec 5).

4.1 Time-based Contrastive Encoder

For a given input time series sample $\boldsymbol{x}_i$, we generate an augmentation set $\mathcal{X}_i^{\text{T}}$ through a time-based augmentation bank $\mathcal{B}^{\text{T}} : \boldsymbol{x}_i^{\text{T}} \rightarrow \mathcal{X}_i^{\text{T}}$. Each element $\widetilde{\boldsymbol{x}}_i^{\text{T}} \in \mathcal{X}_i^{\text{T}}$ is augmented from $\boldsymbol{x}_i$ based on the temporal characteristics. Here, the time-based augmentation bank includes jittering, scaling, time-shifts, and neighborhood segments, all well-established in contrastive learning [40, 48, 41]. We develop an augmentation bank to produce diverse augmentations (rather than a single type of augmentation) and expose the model to complex temporal dynamics, which produces more robust time-based embeddings [48].

For the input $\boldsymbol{x}_i$, we randomly select an augmented sample $\widetilde{\boldsymbol{x}}_i^{\text{T}} \in \mathcal{X}_i^{\text{T}}$ and feed into a contrastive time encoder G_{T} that maps samples to embeddings. We have $\boldsymbol{h}_i^{\text{T}} = G_{\text{T}}(\boldsymbol{x}_i^{\text{T}})$ and $\widetilde{\boldsymbol{h}}_i^{\text{T}} = G_{\text{T}}(\widetilde{\boldsymbol{x}}_i^{\text{T}})$. As $\widetilde{\boldsymbol{x}}_i^{\text{T}}$ is generated based on $\boldsymbol{x}_i^{\text{T}}$, after passing through G_{T}, we assume the embedding of $\boldsymbol{x}_i^{\text{T}}$ is close to the embedding of $\widetilde{\boldsymbol{x}}_i^{\text{T}}$ but far away from the embedding of $\boldsymbol{x}_j^{\text{T}}$ and $\widetilde{\boldsymbol{x}}_j^{\text{T}}$ that are derived from another sample $\boldsymbol{x}_j^{\text{T}} \in \mathcal{D}^{\text{pret}}$ [34, 47, 41]. In specific, we select the positive pair as $(\boldsymbol{x}_i^{\text{T}}, \widetilde{\boldsymbol{x}}_i^{\text{T}})$ and negative pairs as $(\boldsymbol{x}_i^{\text{T}}, \boldsymbol{x}_j^{\text{T}})$ and $(\boldsymbol{x}_i^{\text{T}}, \widetilde{\boldsymbol{x}}_j^{\text{T}})$ [34].

Contrastive time loss. To maximize the similarity within a positive pair and minimize the similarity within a negative pair, we adopt the NT-Xent (the normalized temperature-scaled cross entropy loss) as distance function d which is widely used in contrastive learning [34, 40]. In specific, we define the loss function of the time-based contrastive encoder in terms of sample $\boldsymbol{x}_i^{\text{T}}$ as:

$$\mathcal{L}_{\text{T},i} = d(\boldsymbol{h}_i^{\text{T}}, \widetilde{\boldsymbol{h}}_i^{\text{T}}, \mathcal{D}^{\text{pret}}) = -\log \frac{\exp(\text{sim}(\boldsymbol{h}_i^{\text{T}}, \widetilde{\boldsymbol{h}}_i^{\text{T}})/\tau)}{\sum_{\boldsymbol{x}_j \in \mathcal{D}^{\text{pret}}} \mathbb{1}_{i \neq j} \exp(\text{sim}(\boldsymbol{h}_i^{\text{T}}, G_{\text{T}}(\boldsymbol{x}_j))/\tau)}, \tag{1}$$

where $\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^T \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\|$ denotes the cosine similarity, the $\mathbb{1}_{i \neq j}$ is an indicator function that equals to 0 when $i = j$ and 1 otherwise, and τ is a temporal parameter to adjust scale. The $\boldsymbol{x}_j \in \mathcal{D}^{\text{pret}}$ refers to a different time series sample or its augmented sample. This loss function

urges the time encoder G_{T} to generate closer time-based embeddings for positive pairs and push the embeddings for negative pairs apart from each other.

4.2 Frequency-based Contrastive Encoder

We generate the frequency spectrum $\boldsymbol{x}_i^{\mathrm{F}}$ from a time series sample $\boldsymbol{x}_i^{\mathrm{T}}$ through a transform operator (*e.g.*, Fourier Transformation [54]). The frequency information in time series is universal and plays a key role in classic signal processing [57, 53, 55], but it is rarely investigated in self-supervised contrastive representation learning for time series [58]. In this section, we develop augmentation method to perturb $\boldsymbol{x}_i^{\mathrm{F}}$ based on characteristics of frequency spectra and show how to generate frequency-based representations.

As every frequency component in the frequency spectrum denotes a basis function (*e.g.*, sinusoidal function for Fourier transformation) with the corresponding frequency and amplitude, we perturb the frequency spectrum by adding or removing frequency components. A small perturbation in the frequency domain may cause large changes to the temporal patterns in the time domain [55]. To make sure the perturbed time series is still similar to the original sample (not only in frequency domain but also in time domain; Figure 6), we use a small budget E in the perturbations where E denotes the number of frequency components we manipulate. While removing frequency components, we randomly select E frequency components and set their amplitudes to 0. While adding frequency components, we randomly choose E frequency components from the ones have smaller amplitude than $\alpha \cdot A_m$, and increase their amplitude to $\alpha \cdot A_m$. The A_m is the maximum amplitude in the frequency spectrum and α is a pre-defined coefficient to adjust the scale of the perturbed frequency component ($\alpha = 0.5$ in this work). We produce an augmentation set $\mathcal{X}_i^{\mathrm{F}}$ for $\boldsymbol{x}_i^{\mathrm{F}}$ through frequency-augmentation bank $\mathcal{B}^{\mathrm{F}} : \boldsymbol{x}_i^{\mathrm{F}} \rightarrow \mathcal{X}_i^{\mathrm{F}}$. As described above, we have two augmentation methods (*i.e.*, removing or adding frequency components) in $\mathcal{B}^{\mathrm{F}}$, $|\mathcal{X}_i^{\mathrm{F}}| = 2$. Details on the exploration of frequency augmentation strategies are covered in Appendix J.

We utilize a frequency encoder G_{F} to map the frequency spectrum (*e.g.*, $\boldsymbol{x}_i^{\mathrm{F}}$) to a frequency-based embedding (*e.g.*, $\boldsymbol{h}_i^{\mathrm{F}} = G_{\mathrm{F}}(\boldsymbol{x}_i^{\mathrm{F}})$). We assume the frequency encoder G_{F} can learn similar embedding for the original frequency spectrum $\boldsymbol{x}_i^{\mathrm{F}}$ and a slightly perturbed frequency spectrum $\widetilde{\boldsymbol{x}}_i^{\mathrm{F}} \in \mathcal{X}_i^{\mathrm{F}}$. Thus, we set the positive pair as $(\boldsymbol{x}_i^{\mathrm{F}}, \widetilde{\boldsymbol{x}}_i^{\mathrm{F}})$ and the negative pairs as $(\boldsymbol{x}_i^{\mathrm{F}}, \boldsymbol{x}_j^{\mathrm{F}})$ and $(\boldsymbol{x}_i^{\mathrm{F}}, \widetilde{\boldsymbol{x}}_j^{\mathrm{F}})$.

Contrastive frequency loss. We calculate frequency-based contrastive loss for sample $\boldsymbol{x}_i$ as:

$$\mathcal{L}_{\mathrm{F},i} = d(\boldsymbol{h}_i^{\mathrm{F}}, \widetilde{\boldsymbol{h}}_i^{\mathrm{F}}, \mathcal{D}^{\text{pret}}) = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{h}_i^{\mathrm{F}}, \widetilde{\boldsymbol{h}}_i^{\mathrm{F}})/\tau)}{\sum_{\boldsymbol{x}_j \in \mathcal{D}^{\text{pret}}} \mathbb{1}_{i \neq j} \exp(\mathrm{sim}(\boldsymbol{h}_i^{\mathrm{F}}, G_{\mathrm{F}}(\boldsymbol{x}_j))/\tau)}. \tag{2}$$

In preliminary experiments, we find that the value of τ has little effect on performance and use the same τ throughout all experiments. The $\mathcal{L}_{\mathrm{F},i}$ yield a frequency encoder G_{F} producing embeddings invariant to frequency spectrum perturbations.

4.3 Time-Frequency Consistency

We develop a consistency loss item $\mathcal{L}_{\mathrm{C},i}$ to urge the learned embeddings to satisfy TF-C: for a given sample, its time-based and frequency-based embeddings (and their local neighborhoods) are supposed to be close to each other (see Sec. 3 for justification). To make sure the distance between embeddings is measurable, we map $\boldsymbol{h}_i^{\mathrm{T}}$ from time space and $\boldsymbol{h}_i^{\mathrm{F}}$ from frequency space to a joint time-frequency space through projectors R_{T} and R_{F}, respectively. In specific, for every input sample $\boldsymbol{x}_i$, we have four embeddings, which are $\boldsymbol{z}_i^{\mathrm{T}} = R_{\mathrm{T}}(\boldsymbol{h}_i^{\mathrm{T}})$, $\widetilde{\boldsymbol{z}}_i^{\mathrm{T}} = R_{\mathrm{T}}(\widetilde{\boldsymbol{h}}_i^{\mathrm{T}})$, $\boldsymbol{z}_i^{\mathrm{F}} = R_{\mathrm{F}}(\boldsymbol{h}_i^{\mathrm{F}})$, and $\widetilde{\boldsymbol{z}}_i^{\mathrm{F}} = R_{\mathrm{F}}(\widetilde{\boldsymbol{h}}_i^{\mathrm{F}})$. The first two embeddings are generated based on temporal characteristics and the latter two embeddings are produced based on the properties of frequency spectrum.

To enforce the embeddings in the time-frequency space subject to TF-C, we design a consistency loss $\mathcal{L}_{\mathrm{C},i}$ that measures the distance between a time-based embedding and a frequency-based embedding. We use $S_i^{\mathrm{TF}} = d(\boldsymbol{z}_i^{\mathrm{T}}, \boldsymbol{z}_i^{\mathrm{F}}, \mathcal{D}^{\text{pret}})$ to denote the distance between $\boldsymbol{z}_i^{\mathrm{T}}$ and $\boldsymbol{z}_i^{\mathrm{F}}$). Similarly, we define $S_i^{\widetilde{\mathrm{TF}}}$, $S_i^{\widetilde{\mathrm{T}}\mathrm{F}}$, and $S_i^{\widetilde{\mathrm{TF}}}$. Note, in this time-frequency space, we don't consider the distance between $\boldsymbol{z}_i^{\mathrm{T}}$ and $\widetilde{\boldsymbol{z}}_i^{\mathrm{T}}$ where the two embeddings are from the same domain (*i.e.*, time domain). The same applies to pair the distance between $\boldsymbol{z}_i^{\mathrm{F}}$ and $\widetilde{\boldsymbol{z}}_i^{\mathrm{F}}$. We have already considered information of above two pairs in the calculation of $\mathcal{L}_{\mathrm{T},i}$ and $\mathcal{L}_{\mathrm{F},i}$.

Next, let's closely observe S_i^{TF} and $S_i^{\mathrm{T}\widetilde{\mathrm{F}}}$ that involve three embeddings: $\boldsymbol{z}_i^{\mathrm{T}}$, $\boldsymbol{z}_i^{\mathrm{F}}$, and $\widetilde{\boldsymbol{z}}_i^{\mathrm{F}}$. Here, $\boldsymbol{z}_i^{\mathrm{T}}$ and $\boldsymbol{z}_i^{\mathrm{F}}$ are learned from the original sample ($\boldsymbol{x}_i^{\mathrm{T}}$ and $\boldsymbol{x}_i^{\mathrm{F}}$) while $\widetilde{\boldsymbol{z}}_i^{\mathrm{F}}$ is learned from the augmented $\widetilde{\boldsymbol{x}}_i^{\mathrm{F}}$. Thus, intuitively, $\boldsymbol{z}_i^{\mathrm{T}}$ should be closer to $\boldsymbol{z}_i^{\mathrm{F}}$ in comparison to $\widetilde{\boldsymbol{z}}_i^{\mathrm{F}}$. Motivated by the relative relationship, we encourage the proposed model to learn a S_i^{TF} that is smaller than $S_i^{\mathrm{T}\widetilde{\mathrm{F}}}$. Inspired by the triplet loss [59], we design $(S_i^{\mathrm{TF}} - S_i^{\mathrm{T}\widetilde{\mathrm{F}}} + \delta)$ as a term of consistency loss $\mathcal{L}_{\mathrm{C},i}$ where δ is a given constant margin to keep negative samples far apart [60]. This term optimizes the model towards a smaller S_i^{TF} and relatively larger $S_i^{\mathrm{T}\widetilde{\mathrm{F}}}$. Similarly, S_i^{TF} is supposed to be smaller than $S_i^{\widetilde{\mathrm{T}}\mathrm{F}}$ and $S_i^{\widetilde{\mathrm{T}}\widetilde{\mathrm{F}}}$. In summary, we calculate the consistency loss $\mathcal{L}_{\mathrm{C},i}$ for sample $\boldsymbol{x}_i$ by:

$$\mathcal{L}_{\mathrm{C},i} = \sum_{S^{\mathrm{pair}}} (S_i^{\mathrm{TF}} - S_i^{\mathrm{pair}} + \delta), \quad S^{\mathrm{pair}} \in \{S_i^{\mathrm{T}\widetilde{\mathrm{F}}}, S_i^{\widetilde{\mathrm{T}}\mathrm{F}}, S_i^{\widetilde{\mathrm{T}}\widetilde{\mathrm{F}}}\}, \tag{3}$$

where S_i^{pair} denotes the distance between a time-based embedding (*e.g.*, $\boldsymbol{z}_i^{\mathrm{T}}$ or $\widetilde{\boldsymbol{z}}_i^{\mathrm{T}}$) and a frequency-based embedding (*e.g.*, $\boldsymbol{z}_i^{\mathrm{F}}$ or $\widetilde{\boldsymbol{z}}_i^{\mathrm{F}}$). In each pair, there is at least one embedding that is derived from augmented sample instead of the original sample. The δ is a pre-defined constant. By combining all the triplet loss items, $\mathcal{L}_{\mathrm{C}}$ encourages the pre-training model to capture the consistency between time-based and frequency-based embeddings in model optimization. Note, although the Eq. 3 does not explicitly measure the loss across different time series samples (*e.g.*, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$), the cross-sample relationships are implicitly covered in the calculation of S_i^{TF} and S_i^{pair}.

4.4 Implementation and Technical Details

The overall loss function in pre-training has three terms. First, the time-based contrastive loss $\mathcal{L}_{\mathrm{T}}$ urges the model to learn embeddings invariant to temporal augmentations. Second, the frequency-based contrastive loss $\mathcal{L}_{\mathrm{F}}$ promotes learning of embeddings invariant to frequency spectrum-based augmentations. Third, the consistency loss $\mathcal{L}_{\mathrm{C}}$ guides the model to retain the consistency between time-based and frequency-based embeddings. In summary, the pre-training loss is defined as:

$$\mathcal{L}_{\text{TF-C},i} = \lambda(\mathcal{L}_{\mathrm{T},i} + \mathcal{L}_{\mathrm{F},i}) + (1-\lambda)\mathcal{L}_{\mathrm{C},i} \tag{4}$$

where λ controls the relative importance of the contrastive and consistency losses. We calculate the total loss by summing $\mathcal{L}_{\text{TF-C},i}$ across all pre-training samples. In implementation, the contrastive losses are calculated within the batch. From our problem definition, the model $\mathcal{F}$ we want to learn is the combination of neural networks G_{T}, R_{T}, G_{F}, and R_{F}. When pre-training is completed, we store parameters of entire model, and denote it as $\mathcal{F}(\cdot, \Theta)$ where Θ represents all trainable parameters. When a sample $\boldsymbol{x}_i^{\mathrm{tune}}$ is presented, fine-tuned model $\mathcal{F}$ generates an embedding $\boldsymbol{z}_i^{\mathrm{tune}}$ via concatenation as: $\boldsymbol{z}_i^{\mathrm{tune}} = \mathcal{F}(\boldsymbol{x}_i^{\mathrm{tune}}, \Phi) = [\boldsymbol{z}_i^{\mathrm{tune,T}}; \boldsymbol{z}_i^{\mathrm{tune,F}}]$ where Φ are fine-tuned model's parameters.

5 Experiments

We compare the developed TF-C model with 10 baselines on 8 diverse datasets. We investigate the time series classification tasks in the context of one-to-one and one-to-many transfer learning setups (the many-to-one setting is fundamentally different as discussed in Appendix K). We also assess TF-C in extensive downstream tasks including clustering and anomaly detection.

Datasets. (1) **SLEEPEEG** [61] has 371,055 univariate brainwaves (EEG; 100 Hz) collected from 197 individuals. Each sample is associated with one of five sleeping stages. (2) **EPILEPSY** [62] monitors the brain activities of 500 subjects with single-channel EEG sensor (174 Hz). A sample is labeled in binary based on whether the subject has epilepsy or not. (3) **FD-A** [63] gathers the vibration signals from rolling bearing from a mechanical system aiming at fault detection. Every sample has 5,120 timestamps and an indicator for one out of three mechanical device states. (4) **FD-B** [63] has the same setting as the **FD-A** but the rolling bearings are performed in different working conditions (*e.g.*, varying rotational speed). (5) **HAR** [64] has 10,299 9-dimension samples from 6 daily activities. (6) **GESTURE** [65] includes 440 samples that are collected from 8 hand gestures recorded by an accelerometer. (7) **ECG** [26] contains 8,528 single-sensor ECG recordings with sorted into four classes based on human physiology. (8) **EMG** [66] consists of 163 EMG samples with 3-class labels implying muscular diseases. Dataset labels are not used in pre-training. Further dataset statistics are in Appendix D and Table 3.

Baselines. We consider 10 baseline methods. This includes 8 state-of-the-art methods: TS-SD [12], TS2vec [47], CLOCS [41], Mixing-up [18], TS-TCC [48], SimCLR [40], TNC [46], and CPC [30].

Table 1: One-to-one pre-training evaluation (Scenario 3). Pre-training is performed on HAR, followed by fine-tuning on GESTURE. Results for other three scenarios are shown in Tables 4-6.

Models	Accuracy	Precision	Recall	F1 score	AUROC	AUPRC
Non-DL (KNN)	0.6766±0.0000	0.6500±0.0000	0.6821±0.0000	0.6442±0.0000	0.8190±0.0000	0.5231±0.0000
Random Init.	0.4219±0.0865	0.4751±0.0925	0.4963±0.1026	0.4886±0.0967	0.7129±0.1206	0.3358±0.1194
TS-SD	0.6937±0.0533	0.6806±0.0496	0.6883±0.0525	0.6785±0.0495	0.8708±0.0305	0.6261±0.0790
TS2vec	0.6453±0.0260	0.6287±0.0339	0.6451±0.0218	0.6261±0.0294	0.8890±0.0054	0.6670±0.0118
CLOCS	0.4731±0.0229	0.4639±0.0432	0.4766±0.0266	0.4392±0.0198	0.8161±0.0068	0.4916±0.0103
Mixing-up	0.7183±0.0123	0.7001±0.0166	0.7183±0.0123	0.6991±0.0145	**0.9127±0.0018**	0.7654±0.0071
TS-TCC	0.7593±0.0242	0.7668±0.0257	0.7566±0.0231	0.7457±0.0210	0.8866±0.0040	0.7217±0.0121
SimCLR	0.4383±0.0652	0.4255±0.1072	0.4383±0.0652	0.3713±0.0919	0.7721±0.0559	0.4116±0.0971
TF-C (Ours)	**0.7824±0.0237**	**0.7982±0.0496**	**0.8011±0.0322**	**0.7991±0.0296**	0.9052±0.0136	**0.7861±0.0149**

The TS2Vec, TS-TCC, SimCLR, TNC, and CPC are designed for representation learning on a single dataset rather than for transfer learning, so we apply them to fit our settings and make the results comparable. As the training of TNC and CPC are very time-consuming and relatively less competitive (Table 4), we only compare them in the one-to-one setting (scenario 1) while not in other experiments. To examine the utility of pre-training, we consider two additional approaches that are applied directly to fine-tuning datasets without any pre-training: Non-DL (a non-deep learning KNN model) and Random Init. (randomly initializes the fine-tuning model). The evaluation metrics are accuracy, precision (macro-averaged), recall, F1 score, AUROC, and AUPRC.

Implementation. We use two 3-layer 1-D ResNets [67] as backbones for encoders G_{T} and G_{F}. Our datasets contain long time series (samples in FD-A and FD-B have 5,120 observations), and preliminary experiments identified ResNet as a better option than a Transformer variant [68]. We use 2 fully-connected layers for R_{T} and R_{F}, with no sharing of parameters. We set $E = 1$ and $\alpha = 0.5$ in frequency augmentations and $\tau = 0.2$, $\delta = 1$, $\lambda = 0.5$ in loss functions. Reported are mean and standard deviation values across 5 independent runs (both pre-training and fine-tuning) on the same data split. Results for KNN (K=2) do not change so the standard deviation is zero. Method details and hyper-parameter selection are in Appendix E.

5.1 Results: One-to-One Pre-Training Evaluation

Setup. In one-to-one evaluation, we pre-train a model on *one* pre-training dataset and use it for fine-tuning on *one* target dataset only. Scenario 1 (SLEEPEEG → EPILEPSY): Pre-training is done on SLEEPEEG and fine-tuning on EPILEPSY. While both datasets describe a single-channel EEG, the signals are from different channels/positions on scalps, track different physiology (sleep vs. epilepsy), and are collected from different patients. Scenario 2 (FD-A → FD-B): Datasets describe mechanical devices that operate in different working conditions, including rotational speed, load torque, and radial force. Scenario 3 (HAR → GESTURE): Datasets record different activities (6 types of human daily activities vs. 8 hand gestures). While both datasets contain acceleration signals, HAR has 9 channels while GESTURE has 1 channel. Scenario 4 (ECG → EMG): While both are physiological datasets, the ECG records the electrical signal from the heart whereas EMG measures muscle response in response to a nerve's stimulation of the muscle. We note that the discrepancies between pre-training and fine-tuning datasets in the above four scenarios are substantial, and they cover a diverse range of variation in time series datasets: varying semantic meaning, sampling frequency, time series length, number of classes, and system factors (*e.g.*, number of devices or subjects). The setup is further challenged by the relatively small number of samples available for fine-tuning (EPILEPSY: 60; FD-B: 60; GESTURE: 480; EMG: 122). Further details are in Appendix F.

Results. The results for the four scenarios are shown in Table 1 and Tables 4-6. Overall, our TF-C model has won 16 out of 24 tests (6 metrics in 4 scenarios) and is the second-best performer in only 8 other tests. We report all metrics but discuss the F1 score in the following. On average, our TF-C model claims a large margin of 15.4% over all baselines. Although the strongest baseline is varying (such as TS-TCC in Scenario 2; Mixing-up in Scenario 3), our model outperforms the strongest baselines by 1.5% across all scenarios. Specifically, as shown in Table 1 (HAR → GESTURE; Scenario 3), TF-C achieves the highest performance of 79.91% in F1 score, which yields a margin of 7.2% over the best baseline TS-TCC (74.57%). One potential explanation is that Scenario 3 involves a complex dataset (HAR has 6 classes while GESTURE has 8 classes) that can be difficult to model. The complexity of Scenario 3 is further verified by poor performance of all models (±80%) relative to performance on other Scenarios (±90%): TF-C shows strong robustness by

Table 2: One-to-many pre-training evaluation. Pre-training is performed on SLEEPEEG, followed by an independent fine-tuning on EPILEPSY, FD-B, GESTURE, and EMG.

Scenarios	Models	Accuracy	Precision	Recall	F1 score	AUROC	AUPRC
SLEEPEEG → EPILEPSY	**Non-DL (KNN)**	0.8525±0.0000	0.8639±0.0000	0.6431±0.0000	0.6791±0.0000	0.6434±0.0000	0.6279±0.0000
	Random Init.	0.8983±0.0656	0.9213±0.1369	0.7447±0.1135	0.7959±0.1208	0.8578±0.2153	0.6489±0.1926
	TS-SD	0.8952±0.0522	0.8018±0.2244	0.7647±0.1485	0.7767±0.1855	0.7677±0.2452	0.7940±0.1825
	TS2vec	0.9395±0.0044	0.9059±0.0116	0.9039±0.0118	0.9045±0.0067	0.9587±0.0086	0.9430±0.0103
	CLOCS	**0.9507±0.0027**	0.9301±0.0067	**0.9127±0.0165**	**0.9206±0.0066**	0.9803±0.0023	0.9609±0.0116
	Mixing-up	0.8021±0.0000	0.4011±0.0000	0.5000±0.0000	0.4451±0.0000	0.9743±0.0081	0.9618±0.0104
	TS-TCC	0.9253±0.0098	0.9451±0.0049	0.8181±0.0257	0.8633±0.0215	0.9842±0.0034	0.9744±0.0043
	SimCLR	0.9071±0.0344	0.9221±0.0166	0.7864±0.1071	0.8178±0.0998	0.9045±0.0539	0.9128±0.0205
	TF-C (Ours)	0.9495±0.0249	**0.9456±0.0108**	0.8908±0.0216	0.9149±0.0534	**0.9811±0.0237**	**0.9703±0.0199**
SLEEPEEG → FD-B	**Non-DL (KNN)**	0.4473±0.0000	0.2847±0.0000	0.3275±0.0000	0.2284±0.0000	0.4946±0.0000	0.3308±0.0000
	Random Init.	0.4736±0.0623	0.4829±0.0529	0.5235±0.1023	0.4911±0.0590	0.7864±0.0349	0.7528±0.0254
	TS-SD	0.5566±0.0210	0.5710±0.0535	0.6054±0.0272	0.5703±0.0328	0.7196±0.0113	0.5693±0.0532
	TS2vec	0.4790±0.0113	0.4339±0.0092	0.4842±0.0197	0.4389±0.0107	0.6463±0.0130	0.4442±0.0162
	CLOCS	0.4927±0.0310	0.4824±0.0316	0.5873±0.0387	0.4746±0.0485	0.6992±0.0099	0.5501±0.0365
	Mixing-up	0.6789±0.0246	0.7146±0.0343	**0.7613±0.0198**	0.7273±0.0228	0.8209±0.0035	0.7707±0.0042
	TS-TCC	0.5499±0.0220	0.5279±0.0293	0.6396±0.0178	0.5418±0.0338	0.7329±0.0203	0.5824±0.0468
	SimCLR	0.4917±0.0437	0.5446±0.1024	0.4760±0.0885	0.4224±0.1138	0.6619±0.0219	0.5009±0.0477
	TF-C (Ours)	**0.6938±0.0231**	**0.7559±0.0349**	0.7202±0.0257	**0.7487±0.0268**	**0.8965±0.0135**	**0.7871±0.0267**
SLEEPEEG → GESTURE	**Non-DL (KNN)**	0.6833±0.0000	0.6501±0.0000	0.6833±0.0000	0.6443±0.0000	0.8190±0.0000	0.5232±0.0000
	Random Init.	0.4219±0.0629	0.4751±0.0175	0.4963±0.0679	0.4886±0.0459	0.7129±0.0166	0.3358±0.1439
	TS-SD	0.6922±0.0444	0.6698±0.0472	0.6867±0.0488	0.6656±0.0443	0.8725±0.0324	0.6185±0.0966
	TS2vec	0.6917±0.0333	0.6545±0.0358	0.6854±0.0349	0.6570±0.0392	0.8968±0.0123	0.6989±0.0346
	CLOCS	0.4433±0.0518	0.4237±0.0794	0.4433±0.0518	0.4014±0.0602	0.8073±0.0109	0.4460±0.0384
	Mixing-up	0.6933±0.0231	0.6719±0.0232	0.6933±0.0231	0.6497±0.0306	0.8915±0.0261	0.7279±0.0558
	TS-TCC	0.7188±0.0349	0.7135±0.0352	0.7167±0.0373	0.6984±0.0360	0.9099±0.0085	0.7675±0.0201
	SimCLR	0.4804±0.0594	0.5946±0.1623	0.5411±0.1946	0.4955±0.1870	0.8131±0.0521	0.5076±0.1588
	TF-C (Ours)	**0.7642±0.0196**	**0.7731±0.0355**	**0.7429±0.0268**	**0.7572±0.0311**	**0.9238±0.0159**	**0.7961±0.0109**
SLEEPEEG → EMG	**Non-DL (KNN)**	0.4390±0.0000	0.3772±0.0000	0.5143±0.0000	0.3979±0.0000	0.6025±0.0000	0.4084±0.0000
	Random Init.	0.7780±0.0729	0.5909±0.0625	0.6667±0.0135	0.6238±0.0267	0.9109±0.1239	0.7771±0.1427
	TS-SD	0.4606±0.0000	0.1545±0.0000	0.3333±0.0000	0.2111±0.0000	0.5005±0.0126	0.3775±0.0110
	TS2vec	0.7854±0.0318	**0.8040±0.0750**	0.6785±0.0396	0.6766±0.0501	**0.9331±0.0164**	**0.8436±0.0372**
	CLOCS	0.6985±0.0323	0.5306±0.0750	0.5354±0.0291	0.5139±0.0409	0.7923±0.0573	0.6484±0.0680
	Mixing-up	0.3024±0.0534	0.1099±0.0126	0.2583±0.0456	0.1541±0.0204	0.4506±0.1718	0.3660±0.1635
	TS-TCC	0.7889±0.0192	0.5851±0.0974	0.6310±0.0991	0.5904±0.0952	0.8851±0.0113	0.7939±0.0386
	SimCLR	0.6146±0.0582	0.5361±0.1724	0.4990±0.1214	0.4708±0.1486	0.7799±0.1344	0.6392±0.1596
	TF-C (Ours)	**0.8171±0.0287**	0.7265±0.0353	**0.8159±0.0289**	**0.7683±0.0311**	0.9152±0.0211	0.8329±0.0137

learning more generalizable representations. Additionally, we visualize the learned representations in time-frequency space (Appendix I), and the analyses provide further support for the TF-C property.

5.2 Results: One-to-Many Pre-Training Evaluation

Setup. In one-to-many evaluation, pre-training is done using *one* dataset followed by fine-tuning on *multiple* target datasets independently without starting pre-training from scratch. Out of eight datasets, SLEEPEEG has most complex temporal dynamics [69] and is the largest (371,055 samples). For that reason, we pre-train a model on SLEEPEEG and separately fine-tune a well-pre-trained model on EPILEPSY, FD-B, GESTURE, and EMG.

Results. Results are shown in Table 10. As there are fewer commonalities between EEG signals vs. vibration, and acceleration vs. EMG, we expect that transfer learning will be less effective for them than one-to-one evaluations. The pre-training and fine-tuning datasets are largely different in the bottom three blocks (SLEEPEEG → {FD-B, GESTURE, EMG}). The large gap reasonably leads to a deterioration in baseline performances, however, our model has a noticeably higher tolerance to knowledge transfer across datasets with large gaps. Notably, We find that the proposed model with TF-C earned the best performance in 14 out of 18 settings in the three challenging settings: indicating our TF-C assumption is universal in time series. For example, our approach outperforms the strongest baseline by 8.4% (in precision) when fine-tuning on GESTURE. Our model has great potential to serve as a universal model when there is no large pre-training dataset that is similar to the small fine-tuning dataset. Furthermore, the TF-C consistently outperforms KNN and Random Init. (which are not pre-trained) by a large margin of 42.8% and 25.1% (both in F1 score) on average.

Ablation study. We evaluate how relevant the model components are for effective pre-training. As shown in Table 9 (SLEEPEEG → GESTURE; Appendix H), removing $\mathcal{L}_{\text{C}}$, $\mathcal{L}_{\text{T}}$, and $\mathcal{L}_{\text{F}}$ result in performance degradation (precision) of 6.1%, 7.2%, and 6.7%, respectively. To validate that

the performance increment is not solely brought by a third loss term no matter what consistency it measures, we replaced consistency loss $\mathcal{L}_{\text{C}}$ with a loss term measuring the consistency within time space (named $\mathcal{L}_{\text{TT-C}}$) or within frequency space (named $\mathcal{L}_{\text{FF-C}}$). Results show our consistency loss outperforms $\mathcal{L}_{\text{TT-C}}$ and $\mathcal{L}_{\text{FF-C}}$ by 5.3% and 7.2% (accuracy), respectively.

5.3 Additional Downstream Tasks: Clustering and Anomaly Detection

Clustering Task. We evaluate the clustering performance of TF-C taking SLEEPEEG $\rightarrow$ EPILEPSY as an example. Specifically, we added a K-means (K=2), as Epilepsy has 2 classes, on top of z_i^{tune} in fine-tuning. We adopt commonly used evaluation metrics: Silhouette score, Adjusted Rand Index (ARI), and Normalized Mutual Information (NMI). Table 7 shows our TF-C obtains the best clustering surpassing the strongest baseline (TS-TCC) by a large margin (5.4% in Silhouette score). It conveys that TF-C can capture more distinctive representations with the knowledge transferred from pre-training, which is consistent with the superiority of TF-C in the above classification tasks.

Anomaly Detection Task. We assess how TF-C performs on a sample-level anomaly detection task. Note we work on the sample-level rather than the observation-level anomaly detection. Based on global patterns, the former aims to detect abnormal time series samples instead of outlier observations in a sample (as in BTSF [50] and USAD [70]) which emphasizes local context. Specifically, In the scenario of FD-A $\rightarrow$ FD-B, we built a small subset of FD-B with 1,000 samples, of which 900 are from undamaged bearings, and the remaining 100 are from bearings with inner or outer damage. Undamaged samples are considered "normal," and inner/outer damaged samples are "outliers." In fine-tuning, we used one-class SVM on top of learned representations z_i^{tune}. The experimental results (Table 8) show that our TF-C outperforms five competitive baselines with 4.5% in F-1 Score. Results show that the proposed TF-C is more sensitive to anomalous samples and can effectively detect the abnormal status in mechanical devices.

6 Conclusion

We develop a pre-training approach that introduces time-frequency consistency (TF-C) as a mechanism to support knowledge transfer between time-series datasets. The approach uses self-supervised contrastive estimation and injects TF-C into pre-training, bringing time-based and frequency-based representations and their local neighborhoods close together in the latent space.

Limitations and future directions. TF-C property can serve as a universal property for pre-training on diverse time series datasets. Additional generalizable properties, such as temporal autoregressive processes, could also be helpful for pre-training on time series. Further, while our method expects as input a regularly sampled time series, it can handle irregularly sampled time series by using an encoder (such as Raindrop [71] and SeFT [72]) that can embed irregular time series. For frequency encoder inputs $\boldsymbol{x}_i^{\text{F}}$, alternatives include resampling or interpolation to obtain regularly sampled signals and using regular or non-uniform FFT operations. Furthermore, TF-C's current embedding strategy and loss functions are favorable for classification, leveraging global information over tasks that use local context (*e.g.*, forecasting). Results show that the TF-C approach performs well across broad downstream tasks, including classification, clustering, and anomaly detection (Sec. 5.3).

Acknowledgments and Disclosure of Funding

We gratefully acknowledge support by US Air Force Contract No. FA8702-15-D-0001, Harvard Data Science Initiative, and awards from Amazon Research, Bayer Early Excellence in Science, AstraZeneca Research, and Roche Alliance with Distinguished Scientists. T.T. is supported by the Under Secretary of Defense for Research and Engineering under US Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funders.

References

[1] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. Scientific data, 6(1):1–18, 2019.

[2] Shahbaz Rezaei and Xin Liu. Deep learning for encrypted traffic classification: An overview. IEEE communications magazine, 57(5):76–81, 2019.

[3] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, et al. Skilful precipitation nowcasting using deep generative models of radar. Nature, 597(7878):672–677, 2021.

[4] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied soft computing, 90:106181, 2020.

[5] Bing Su and Ji-Rong Wen. Temporal alignment prediction for supervised representation learning and few-shot sequence classification. In ICLR, 2022.

[6] Yixiang Deng, Lu Lu, Laura Aponte, Angeliki M Angelidi, Vera Novak, George Em Karniadakis, and Christos S Mantzoros. Deep transfer learning and data augmentation improve glucose levels prediction in type 2 diabetes patients. NPJ Digital Medicine, 4(1):1–13, 2021.

[7] Quentin Rebjock, Baris Kurt, Tim Januschowski, and Laurent Callot. Online false discovery rate control for anomaly detection in time series. NeurIPS, 34:26487–26498, 2021.

[8] Fan-Keng Sun, Chris Lang, and Duane Boning. Adjusting for autocorrelated errors in neural networks for time series. NeurIPS, 34:29806–29819, 2021.

[9] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery, 34(5):1454–1495, 2020.

[10] Wenyong Huang, Zhenhe Zhang, Yu Ting Yeung, Xin Jiang, and Qun Liu. Spiral: Self-supervised perturbation-invariant representation learning for speech pre-training. ICLR, 2022.

[11] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. Data mining and knowledge discovery, 33(4):917–963, 2019.

[12] Pengxiang Shi, Wenwen Ye, and Zheng Qin. Self-supervised pre-training for time series classification. In IJCNN, pages 1–8, 2021.

[13] Weixia Dang, Biyu Zhou, Lingwei Wei, Weigang Zhang, Ziang Yang, and Songlin Hu. Ts-bert: Time series anomaly detection via pre-training model bert. In International Conference on Computational Science, pages 209–223. Springer, 2021.

[14] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In CVPR, pages 3558–3568, 2021.

[15] Kailai Sun, Zuchao Li, and Hai Zhao. Multilingual pre-training with universal dependency learning. NeurIPS, 34:8444–8456, 2021.

[16] Rui Ye and Qun Dai. Implementing transfer learning across different datasets for time series forecasting. Pattern Recognition, 109:107617, 2021.

[17] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Transfer learning for time series classification. In 2018 IEEE international conference on big data (Big Data), pages 1367–1376. IEEE, 2018.

[18] Kristoffer Wickstrøm, Michael Kampffmeyer, Karl Øyvind Mikalsen, and Robert Jenssen. Mixing up contrastive learning: Self-supervised representation learning for time series. PRL, 155:54–61, 2022.

[19] Priyanka Gupta, Pankaj Malhotra, Jyoti Narwariya, Lovekesh Vig, and Gautam Shroff. Transfer learning for clinical time series analysis using deep neural networks. Journal of Healthcare Informatics Research, 4(2):112–137, 2020.

[20] Amiel Meiseles and Lior Rokach. Source model selection for deep learning in the time series domain. IEEE Access, 8:6190–6200, 2020.

[21] Ankit Singh. Clda: Contrastive learning for semi-supervised domain adaptation. NeurIPS, 34:5089–5101, 2021.

[22] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In ICLR, 2019.

[23] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. OpenAI, 2018.

[24] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In NeurIPS, volume 33, pages 22243–22255, 2020.

[25] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In NeurIPS, volume 33, pages 12449–12460, 2020.

[26] Gari D Clifford, Chengyu Liu, Benjamin Moody, H Lehman Li-wei, Ikaro Silva, Qiao Li, AE Johnson, and Roger G Mark. Af classification from a short single lead ecg recording: The physionet/computing in cardiology challenge 2017. In 2017 Computing in Cardiology (CinC), pages 1–4. IEEE, 2017.

[27] Mitchell L Gordon, Kaitlyn Zhou, Kayur Patel, Tatsunori Hashimoto, and Michael S Bernstein. The disagreement deconvolution: Bringing machine learning performance metrics in line with reality. In CHI, pages 1–14, 2021.

[28] Simon Rogers, Derek Sleeman, and John Kinsella. Investigating the disagreement between clinicians' ratings of patients in icus. IEEE Journal of Biomedical and Health Informatics, 17(4):843–852, 2013.

[29] Leonard M Horowitz, Rita de Sales French, Kirk D Wallis, David L Post, and Ellen Y Siegelman. The prototype as a construct in abnormal psychology: Ii. clarifying disagreement in psychiatric judgments. Journal of Abnormal Psychology, 90(6):575, 1981.

[30] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. In arXiv:1807.03748, 2019.

[31] Pritam Sarkar and Ali Etemad. Self-supervised learning for ecg-based emotion recognition. In ICASSP, pages 3217–3221, 2020.

[32] Joseph Y Cheng, Hanlin Goh, Kaan Dogrusoz, Oncel Tuzel, and Erdrin Azemi. Subject-aware contrastive learning for biosignals. arXiv preprint arXiv:2007.04871, 2020.

[33] Sriram Ravula, Georgios Smyrnis, Matt Jordan, and Alexandros G Dimakis. Inverse problems leveraging pre-trained contrastive representations. NeurIPS, 34:8753–8765, 2021.

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

[35] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In CVPR, pages 1601–1610, 2021.

[36] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In Proceedings of the IEEE international conference on computer vision, pages 667–676, 2017.

[37] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In ICCV, pages 2959–2968, 2019.

[38] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[39] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case. In arXiv:2001.08317, 2020.

[40] Chi Ian Tang, Ignacio Perez-Pozuelo, Dimitris Spathis, and Cecilia Mascolo. Exploring contrastive learning in human activity recognition for healthcare. arXiv preprint arXiv:2011.11542, 2020.

[41] Dani Kiyasseh, Tingting Zhu, and David A Clifton. Clocs: Contrastive learning of cardiac signals across space, time, and patients. In ICML, pages 5606–5615, 2021.

[42] David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alex Kurakin. Adamatch: A unified approach to semi-supervised learning and domain adaptation. ICLR, 2022.

[43] Guoqiang Wei, Cuiling Lan, Wenjun Zeng, Zhizheng Zhang, and Zhibo Chen. Toalign: Task-oriented alignment for unsupervised domain adaptation. NeurIPS, 34:13834–13846, 2021.

[44] Tongkun Xu, Weihua Chen, Pichao Wang, Fan Wang, Hao Li, and Rong Jin. Cdtrans: Cross-domain transformer for unsupervised domain adaptation. ICLR, 2022.

[45] Bernd Illing, Jean Ventura, Guillaume Bellec, and Wulfram Gerstner. Local plasticity rules can learn deep representations using self-supervised contrastive predictions. NeurIPS, 34:30365–30379, 2021.

[46] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. In ICLR, 2021.

[47] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In AAAI, volume 36, pages 8980–8987, 2022.

[48] 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. In IJCAI, pages 2352–2359, 2021.

[49] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In ICLR, 2022.

[50] Ling Yang and Shenda Hong. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In ICML, pages 25038–25054. PMLR, 2022.

[51] Rob J Hyndman and George Athanasopoulos. Forecasting: principles and practice. OTexts, 2018.

[52] Ronald Newbold Bracewell and Ronald N Bracewell. The Fourier transform and its applications, volume 31999. McGraw-hill New York, 1986.

[53] Leon Cohen. Time-frequency analysis, volume 778. Prentice hall New Jersey, 1995.

[54] Henri J Nussbaumer. The fast fourier transform. In Fast Fourier Transform and Convolution Algorithms, pages 80–111. Springer, 1981.

[55] Patrick Flandrin. Time-frequency/time-scale analysis. Academic press, 1998.

[56] Antonia Papandreou-Suppappola. Applications in time-frequency signal processing. CRC press, 2018.

[57] Ryan Soklaski, Michael Yee, and Theodoros Tsiligkaridis. Fourier-based augmentations for improved robustness and uncertainty calibration. NeurIPS'W, 2021.

[58] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. Technologies, 9(1):2, 2020.

[59] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In International workshop on similarity-based pattern recognition, pages 84–92. Springer, 2015.

[60] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In Bmvc, volume 1, page 3, 2016.

[61] Bob Kemp, Aeilko H Zwinderman, Bert Tuk, Hilbert AC Kamphuisen, and Josefien JL Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg. IEEE Transactions on Biomedical Engineering, 47(9):1185–1194, 2000.

[62] Ralph G Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. Physical Review E, 64(6):061907, 2001.

[63] Christian Lessmeier, James Kuria Kimotho, Detmar Zimmer, and Walter Sextro. Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification. In PHM Society European Conference, volume 3, 2016.

[64] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra Perez, and Jorge Luis Reyes Ortiz. A public domain dataset for human activity recognition using smartphones. In ESANN, pages 437–442, 2013.

[65] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. Pervasive and Mobile Computing, 5(6):657–675, 2009.

[66] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. circulation, 101(23):e215–e220, 2000.

[67] Amrutha Ramanathan and James McDermott. Fall detection with accelerometer data using residual networks adapted to multi-variate time series classification. In IJCNN, pages 1–8, 2021.

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

[69] Xiang Zhang and Lina Yao. Deep Learning for EEG-Based Brain–Computer Interfaces: Representations, Algorithms and Applications. World Scientific, 2021.

[70] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 3395–3404, 2020.

[71] Xiang Zhang, Marko Zeman, Theodoros Tsiligkaridis, and Marinka Zitnik. Graph-guided network for irregularly sampled multivariate time series. In ICLR, 2022.

[72] Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set functions for time series. In ICML, pages 4353–4363, 2020.

Broader Impacts

Our approach for self-supervised pre-training improves classification performance on target datasets in different application scenarios. The recognition of time-frequency consistency as a universal property specific to time series data is a weak assumption that enables effective, task- and domain-agnostic transfer learning. We believe our work will inspire the research community to uncover other universal properties for transfer learning. We also hope our work will also attract more researchers to the more general problem of time series representation learning which is still underappreciated relative to problems from CV and NLP fields.

On the society level, our work, along the line of transfer learning, can facilitate more efficient use of time series data in various settings. For example, in medical settings, some diseases of clinical interest may have very small labelled dataset. In this case, unlabelled data from patients of different diseases but with similar underlying physiological conditions can be used to pre-train the model. However, practitioners need to be aware of the limitations of the model, including that it may make biased predictions. Specifically, bias may exist in the source dataset used for pre-training due to an imbalance of samples from subjects of different demographic attributes. Also, the standardized medical protocols for collecting these datasets might be unsuitable for subjects with certain physiological attributes, creating unforeseen bias that may be transferred to fine-tuning.

All datasets in this paper are publicly available and are not associated with any privacy or security concern. Furthermore, we have followed guidelines on responsible use specified by primary authors of the datasets used in the current work.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] In abstract and introduction, we claim that TF-C is a generalizable property of time series that can support pre-training, which is well-justified in Sec. 3 and experimentally demonstrated in Sec. 5 (our model consistently performs comparatively to or above baseline methods).
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Broader Impact on Page 10.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Yes, we include an anonymous link (see Abstract) that provides the source codes with all implementation details, implementation of baselines, and eight datasets. The link will be updated to an non-anonymous link after acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyper-parameters, how they were chosen)? [Yes] See implementation details in Sec. 5. See Appendix E for baseline architectures and hyper-parameter settings. More details can be found in the included URL.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We run experiments for 5 times and report the average value with standard deviation. See Table 1, Tables 4-6, and Table 10.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We used eight existing datasets and 6 state-of-the-art baselines in contrastive learning and pre-training for time series. We cited the creators for every exist asset we used. See Sec. 5.

(b) Did you mention the license of the assets? [Yes] All dataset licenses are mentioned in the Appendix D.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See the anonymous link in Abstract.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] All data we use is freely available for download, without any requirement to re-contact the data curator.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] Our datasets are public, well-established, and do not contain PII or offensive content

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix A Further information on the relationship between our pre-training approach and domain adaptation

Here we note our problem definition of pre-training is fundamentally different from domain adaptation [S1, S2, S3, S4, S5, S6][1] in order to prevent any confusion between this work and domain adaptation methods. DA applies a model trained on a pre-training dataset (*i.e.*, source dataset) to a different target dataset [21, 42]. In contrast, self-supervised pre-training has four key differences with domain adaptation. (1) First, our model only requires the pre-training dataset while domain adaptation techniques generally require access to the target dataset [S7, S8, S9, S10, S11, S12, S13, S14, S15, S16]. (2) Second, our model can be applied to multiple unseen target datasets (without re-training the pre-trained model for every target dataset) while domain adaptation approaches use the target dataset during model training, *e.g.*, [S17, S18, S19, S20, S12, S14, S21] (see also Sec. 5.2 for experimental results). (3) Third, our approach can be used in scenarios where the feature space in pre-training is different from that in the target dataset (see Scenarios 1, 3, and 4 in Sec. 5.1 for experimental results). In contrast, domain adaptation methods usually restrict pre-training and target datasets to have the same feature space (but possible different distributions), *e.g.*, [S22, S18, S19, S20, S13].

In summary, to support transfer learning across different time series datasets, a pre-training approach needs a capability to capture a generalizable property of time series, one that is shared across different time series datasets regardless of the specific semantic meaning of a time series signal (*e.g.*, ECG, EMG, acceleration, vibration), conditions of data acquisition (*e.g.*, variation across subjects and devices), sampling frequencies, etc. This work develops a self-supervised contrastive pre-training strategy that fulfills these requirements by injecting an appropriate inductive bias (called Time-Frequency Consistency, TF-C, into the model (Sec. 3).

Further, we clarify that the term 'self-supervised' has different meanings in DA and in pre-training [S23, S24, S25, S26]. The 'self-supervised domain adaptation' [S27, S16, S21, S15] or 'unsupervised domain adaptation' [S1, S22, S28, S11, S14] means that there are no labels in the target dataset, however that still requires labels in the pre-training dataset. In contrast, 'self-supervised pre-training' [S29, S30, S31] (*i.e.*, the problem studied here, in line with a breadth of existing literature on pre-training) indicates the setting where no labels are available in pre-training.

Appendix B Detailed differences with CoST and BTSF

Up to the submission of this manuscript, there is no existing contrastive augmentations in time series' frequency domain. There are two models, CoST [49] and BTSF [50], that involved frequency domain in contrastive learning, however, the proposed TF-C is fundamentally different with them in the following aspects. We take BTSF as an example while the differences also apply to CoST.

- Problem definitions for both papers are different. Our method is designed to produce generalizable representations that can transfer to a different time series dataset (going from pre-training to a fine-tuning dataset) for the purpose of transfer learning. In contrast, BTSF attempts to learn embeddings within the same dataset for the purpose of representation learning. Our model captures the TF-C property invariant to different time series (in terms of various temporal dynamics, semantic meaning, etc.) and can thus serve as a vehicle for transfer learning. In contrast, BTSF learns embeddings invariant to perturbations (i.e., instance-level dropout) of the same time series.
- The modeling of the frequency domain is different in both papers. We developed augmentations in the frequency domain based on the spectral properties of time series. In contrast, although BTSF involves a frequency domain, its data transformation is solely implemented in the time domain (using instance-level dropout; Sec. 3.1 in BTSF). That is, the BTSF method applies the FFT after augmenting samples in the time domain which can lead to information loss.
- BTSF emphasizes fusing temporal and spectral features to generate discriminative embeddings. Unlike BTSF, our model leverages the consistency between time-based and frequency-based embeddings to produce generalizable time series representations. Our

[1] The supplementary document contains additional references, prefixed by 'S'. These additional references are listed at the end of the Appendix.

model maps every sample to a time-frequency embedding space and constrains the relative relationships between embeddings (through triplet loss) according to the TF-C property. The underlying consistency allows TF-C to realize transfer learning across time series datasets, which is TF-C's unique advantage.

In summary, TF-C introduces frequency domain augmentations in the sense that it directly perturbs the frequency spectrum. TF-C is the first method that uses frequency domain augmentations to enable transfer learning in time series.

Appendix C Time series invariance between time and frequency domains

Here, we provide an analogy, from images to time series, to aid understanding of the 'invariance' property of data representations.

In computer vision, it is well known that an image and its augmented views obtained by simple transformations (e.g., rotation, translation, scaling, etc.) can be used in different frameworks such as transformation prediction and contrastive instance discrimination to obtain invariant representations. These transformations are used as augmentations to guide the learning of self-supervised representations. For transformation prediction, the model learns representations that are equivariant to the selected transformation as the information embedded in the transformation needs to be embedded in the representation for the final layer to solve the pretext task. For contrastive instance discrimination, the representations are sensitive to the instances while learning invariances to transformations or views. The intuition behind such transformations is that the underlying object in the image is the same no matter how the image is rotated, translated, re-scaled, etc. In other words, the information carried by the original image and the transformed image is the same.

Similarly, the time domain and frequency domain representations carry the same information in time series. Thus it is reasonable to expect that by exploring local neighborhoods in the time domain and frequency domain and enforcing consistency of feature representations inter- and intra- domains, invariance properties are captured in our model via self-supervised learning. Thus, information carried in time and frequency domains of the same or similar time series sample should be the same. This invariance, formalized as 'Time-Frequency Consistency', is helpful for self-supervised pre-training. We will include the above discussion in the camera-ready version.

Appendix D Additional information on datasets and pre-training evaluation

D.1 Datasets

We use eight diverse time series datasets to evaluate our model. The datasets used in one-to-one and one-to-many pre-training evaluations are the same. The dataset statistics are shown in Table 3. Processed model-ready datasets are in our GitHub Repository (https://anonymous.4open.science/r/TFC-pre-training-6B07). Following is a detailed description of datasets.

SLEEPEEG [61]. The dataset contains 153 whole-night sleeping electroencephalography (EEG) recordings produced by a sleep cassette. Data are collected from 82 healthy subjects. The 1-lead EEG signal is sampled at 100 Hz. We segment the EEG signals into segments (window size is 200) without overlapping, and each segment forms a sample. Every sample is associated with one of the five sleeping patterns/stages: Wake (W), Non-rapid eye movement (N1, N2, N3), and Rapid Eye Movement (REM). After segmentation, we have 371,055 EEG samples. The raw dataset (https://www.physionet.org/content/sleep-edfx/1.0.0/) is distributed under the Open Data Commons Attribution License v1.0.

EPILEPSY [62]. The dataset contains single-channel EEG measurements from 500 subjects. For every subject, the brain activity was recorded for 23.6 seconds. The dataset was then divided and shuffled (to mitigate sample-subject association) into 11,500 samples of 1 second each, sampled at 178 Hz. The raw dataset features five classification labels corresponding to different states of subjects or measurement locations — eyes open, eyes closed, EEG measured in the healthy brain region, EEG measured in the tumor region, and whether the subject has a seizure episode. To emphasize the distinction between positive and negative samples in terms of epilepsy, We merge the first four classes into one, and each time series sample has a binary label describing if the

Table 3: Description of datasets use in the four different application scenarios. We have two datasets (a pre-training dataset and a fine-tuning dataset) in each scenario. For the number of samples in fine-tuning dataset, "A/B/C" denotes we use A samples for fine-tuning, B samples for validation, and C samples for test. To test our effectiveness on small datasets (which is practically meaningful), we limit the fine-tuning set to a very small set with less than 320 samples. We ensure the fine-tuning set is balanced in terms of classes.

Scenario #		**Dataset**	# Samples	# Channels	# Classes	Length	Freq (Hz)
1	Pre-training	**SLEEPEEG**	371,055	1	5	200	100
	Fine-tuning	**EPILEPSY**	60/20/11,420	1	2	178	174
2	Pre-training	**FD-A**	8,184	1	3	5,120	64K
	Fine-tuning	**FD-B**	60/21/13,559	1	3	5,120	64K
3	Pre-training	**HAR**	10,299	9	6	128	50
	Fine-tuning	**GESTURE**	320/120/120	3	8	315	100
4	Pre-training	**ECG**	43,673	1	4	1,500	300
	Fine-tuning	**EMG**	122/41/41	1	3	1,500	4,000

associated subject is experiencing a seizure or not. There are 11,500 EEG samples in total. To evaluate the performance of a pre-trained model on a small fine-tuning dataset, we choose a small set (60 samples; 30 samples for each class) for fine-tuning and assess the model with a validation set (20 samples; 10 samples for each class). Finally, the model with the best validation performance is used to make predictions on the test set (*i.e.*, the remaining 11,420 samples). Statistics of fine-tuning, validation, and test sets in the other three target datasets are in Appendix 3. The raw dataset (https://repositori.upf.edu/handle/10230/42894) is distributed under the Creative Commons License (CC-BY) 4.0.

FD-A and **FD-B** [63]. The dataset is generated by an electromechanical drive system that monitors the condition of rolling bearings and detects their failures. Four subsets of data are collected under various conditions, whose parameters include rotational speed, load torque, and radial force. Each rolling bearing can be undamaged, inner damaged, and outer damaged, which leads to three classes in total. We denote the subsets corresponding to condition A and condition B as Faulty Detection Condition A (**FD-A**) and Faulty Detection Condition B (**FD-B**), respectively. Each original recording has a single channel with a sampling frequency of 64k Hz and lasts 4 seconds. To deal with lengthy recordings, we follow the procedure described by Eldele *et al.* [48]. Specifically, we use a sliding window length of 5,120 observations and a shifting length of 1,024 or 4,096 to ensure that samples are relatively balanced between classes. The raw dataset (https://mb.uni-paderborn.de/en/kat/main-research/datacenter/bearing-datacenter/data-sets-and-download) is distributed under the Creative Commons Attribution-Non Commercial 4.0 International License.

HAR [64]. This dataset contains recordings of 30 health volunteers performing daily activities, including walking, walking upstairs, walking downstairs, sitting, standing, and lying. Prediction labels are the six activities. The wearable sensors on a smartphone measure triaxial linear acceleration and triaxial angular velocity at 50 Hz. After preprocessing and isolating gravitational acceleration from body acceleration, there are nine channels (*i.e.*, 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer) in total. The raw dataset (https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones) is distributed as-is. Any commercial use is not allowed.

GESTURE [65]. The dataset contains accelerometer measurements of eight simple gestures that differ based on the paths of hand movement. The eight gestures are: hand swiping left, right, up, and down, hand waving in a counterclockwise or clockwise circle, hand waving in a square, and waving a right arrow. The classification labels are these eight different kinds of gestures. The original paper reports the inclusion of 4,480 gesture samples, but through the UCR database, we can only recover 440 samples. The dataset is balanced, with 55 samples in each class, and is of an appropriate size for the fine-tuning. The original paper does not explicitly report sampling frequency but is presumably 100 Hz. The dataset uses three channels corresponding to three coordinate directions of acceleration. Note, when transferring knowledge from **HAR** (nine channels) to **GESTURE** (three channels): we use all the nine channels in pre-training if the model (*i.e.*, TF-C, CLOCS, SimCLR, and TS-TCC) has channel generalization ability (*i.e.*, allows different channel numbers in pre-training and fine-tuning datasets); otherwise, if the model (*i.e.*, KNN, TS-SD, Mixing-up, and TS2vec) don't have channel generalization ability, we only use three acceleration channels in **HAR** for pre-training. The raw dataset is accessible

Table 4: Performance on one-to-one setting (scenario 1): pre-training on SLEEPEEG and fine-tuning on EPILEPSY.

Models	Accuracy	Precision	Recall	F1 score	AUROC	AUPRC
Non-DL (KNN)	0.8525±0.0000	0.8639±0.0000	0.6431±0.0000	0.6791±0.0000	0.6434±0.0000	0.6279±0.0000
Random Init.	0.8983±0.0656	0.9213±0.1369	0.7447±0.1135	0.7959±0.1208	0.8578±0.2153	0.6489±0.1926
TS-SD	0.8952±0.0522	0.8018±0.2244	0.7647±0.1485	0.7767±0.1855	0.7677±0.2452	0.7940±0.1825
TS2vec	0.9395±0.0044	0.9059±0.0116	0.9039±0.0118	0.9045±0.0067	0.9587±0.0086	0.9430±0.0103
CLOCS	**0.9507±0.0027**	0.9301±0.0067	**0.9127±0.0165**	**0.9206±0.0066**	0.9803±0.0023	0.9609±0.0116
Mixing-up	0.8021±0.0000	0.4011±0.0000	0.5000±0.0000	0.4451±0.0000	0.9743±0.0081	0.9618±0.0104
TS-TCC	0.9253±0.0098	0.9451±0.0049	0.8181±0.0257	0.8633±0.0215	0.9842±0.0034	0.9744±0.0043
SimCLR	0.9071±0.0344	0.9221±0.0166	0.7864±0.1071	0.8178±0.0998	0.9045±0.0539	0.9128±0.0205
TNC	0.6701±0.0071	0.5336±0.2742	0.5011±0.0016	0.4024±0.0033	0.5853 ± 0.0487	0.6276±0.0438
CPC	0.662± 0.0110	0.4340±0.2215	0.5028±0.0064	0.406 ± 0.0134	0.5722±0.0748	0.6145±0.0575
TF-C (Ours)	0.9495±0.0249	**0.9456±0.0108**	0.8908±0.0216	0.9149±0.0534	**0.9811±0.0237**	**0.9703±0.0199**

through http://www.timeseriesclassification.com/description.php?Dataset=UWaveGestureLibrary. While the distribution license is not explicitly mentioned, the dataset is a public resource based on [65].

ECG [26]. This is the 2017 PhysioNet Challenge focusing on classifying ECG recordings. The single-lead ECG measures four different underlying conditions of cardiac arrhythmias. More specifically, these classes correspond to the recordings of normal sinus rhythm, atrial fibrillation (AF), alternative rhythm, or others (too noisy to be classified). The recordings are sampled at 300 Hz. Furthermore, the dataset is imbalanced, with much fewer samples from the atrial fibrillation and noisy classes out of all four. To preprocess the dataset, we use the code from the CLOCS paper, which applied a fixed-length window of 1,500 observations to divide long recordings into short samples of 5 seconds that are physiologically meaningful. Because of the imbalanced dataset, we report AUROC and AUPRC (insensitive to label distribution) in our results. The raw dataset (https://physionet.org/content/challenge-2017/1.0.0/) is distributed under the Open Data Commons Attribution License v1.0.

EMG [66]. Electromyograms (EMGs) measure muscle responses as electrical activity in response to neural stimulation, and they can be used to diagnose certain muscular dystrophies and neuropathies. The dataset consists of single-channel EMG recording from the tibialis anterior muscle of three healthy volunteers suffering from neuropathy and myopathy. The recordings are sampled at a frequency of 4K Hz. Each patient (*i.e.*, disorder) is a classification category. Then the recordings are split into time-series samples using a fixed-length window of 1,500 observations. The raw dataset (https://physionet.org/content/emgdb/1.0.0/) is distributed under the Open Data Commons Attribution License v1.0.

D.2 Varying lengths and dimensionality of time series in pre-training vs. target datasets

We use established strategies to address the challenging question of how to process time series with the variable number of dimensions (channels) and measurements (note that this study focuses on designing a pre-training model that can transfer knowledge across disparate time-series datasets). When the dimensionality or length of time series is different between pre-training and target datasets, we view the pre-training dataset as an anchor according to which we adjust (pre-process) the target dataset. For length adjustments, we use zero padding to increase the number of observations or use downsampling to reduce the number of observations. For dimensionality adjustments, our model applies to time series at the level of a single lead, enabling the ability to generalize to multivariate time series when we make different channels share the model (*i.e.*, time encoder, frequency encoder, and two projectors) parameters. In this way, we have a $\boldsymbol{z}_i^{\text{TUNE}}$ for each channel of the time series (Sec. 4) and concatenate the learned representations across all channels to form the final representation of the input time series sample.

Table 5: Performance on one-to-one setting (scenario 2): pre-training on **FD-A** and fine-tuning on **FD-B**.

Models	Accuracy	Precision	Recall	F1 score	AUROC	AUPRC
Non-DL (KNN)	0.4473±0.0000	0.2846±0.0000	0.3275±0.0000	0.2284±0.0000	0.4946±0.0000	0.3307±0.0000
Random Init.	0.4736±0.1085	0.4829±0.1235	0.5235±0.0956	0.4911±0.1123	0.7864±0.2659	0.7528±0.1233
TS-SD	0.5465±0.0417	0.5795±0.0608	0.6613±0.0324	0.5567±0.0478	0.7148±0.0014	0.6370±0.0231
TS2vec	0.6494±0.0379	0.6614±0.0599	0.7260±0.0377	0.6797±0.0553	0.8089±0.0254	0.7138±0.0593
CLOCS	0.7118±0.0361	0.7607±0.0830	0.7877±0.0294	0.7638±0.0696	0.8161±0.0146	0.8133±0.0295
Mixing-up	0.7821±0.0110	0.8887±0.0039	0.8404±0.0081	0.8307±0.0104	0.9339±0.0003	0.9348±0.0010
TS-TCC	0.8497±0.0114	0.8906±0.0087	**0.8899±0.0083**	0.8898±0.0083	0.9145±0.0138	0.8957±0.0198
SimCLR	0.5439±0.0562	0.6372±0.0568	0.6128±0.1344	0.5799±0.1286	0.7383±0.0622	0.6548±0.1152
TF-C (Ours)	**0.8934±0.0379**	**0.9209±0.0234**	0.8537±0.0486	**0.9162±0.0826**	**0.9435±0.0259**	**0.9527±0.0134**

Table 6: Performance on one-to-one setting (scenario 4): pre-training on **ECG** and fine-tuning on **EMG**.

Models	Accuracy	Precision	Recall	F1 score	AUROC	AUPRC
Non-DL (KNN)	0.4390±0.0000	0.3771±0.0000	0.5143±0.0000	0.3979±0.0000	0.6025±0.0000	0.4083±0.0000
Random Init.	0.878±0.1259	0.5909±0.1135	0.6667±0.1534	0.6238±0.2315	0.9109±0.1264	0.7771±0.1359
TS-SD	0.4606±0.0000	0.1544±0.0000	0.3333±0.0000	0.2111±0.0000	0.5031±0.0219	0.3805±0.0165
TS2vec	0.9704±0.0109	0.9666±0.0186	0.9751±0.0082	**0.9746±0.0141**	0.9948±0.0070	**0.9780±0.0305**
CLOCS	0.8829±0.0499	0.8492±0.0620	0.8134±0.1262	0.8037±0.1080	0.9385±0.0369	0.8178±0.0945
Mixing-up	0.9121±0.0872	0.8007±0.2158	0.8518±0.1776	0.8215±0.2010	**0.9999±0.0000**	0.9999±0.0000
TS-TCC	0.9590±0.0135	**0.9684±0.0140**	0.8994±0.0304	0.9244±0.0247	0.9800±0.0192	0.9663±0.0348
SimCLR	0.8878±0.0218	0.8209±0.0307	0.8533±0.0433	0.8225±0.0339	0.9565±0.0153	0.8337±0.0163
TF-C (Ours)	**0.9756±0.0071**	0.9444±0.0029	**0.9803±0.0000**	0.9596±0.0003	0.9801±0.0012	0.8867±0.0122

Appendix E TF-C and baseline architectures and implementation details

We compare our TF-C model against eight state-of-the-art baselines and two additional methods (Sec. 5): (1) directly fit a KNN (K=2) classifier with the target (fine-tuning) dataset; (2) randomly initialize a model (the model structure and experimental settings are the same with our TF-C model), ignore pre-training, and directly train from scratch with the target dataset.

We implement the baselines follow the corresponding papers including TS-SD [12], TS2vec [47], CLOCS [41], Mixing-up [18], TS-TCC [48], SimCLR [40], TNC [46], and CPC [30]. We use default settings for hyper-parameters as reported in the original works unless noted below. All pre-training and fine-tuning with baselines are done with a single Tesla V100 GPU with 32 Gb of allocated memory provided by Harvard Medical School's O2 High Performance Computing platform.

TF-C (our model) Our time-based contrastive encoder F_{T} adopts 1-D ResNet backbone similar to SimCLR. Specifically, after hyper-parameter tuning, the encoder consists of three layers convolutional blocks: the kernel sizes in all layers are 8; the strides are 8, 1, and 1, respectively; the the depths are 32, 64, and 128, respectively. We use max pooling after each convolutional layer and all the pooling kernel sizes and strides are set as 2. The cross-space projector R_{T} contains two fully-connected layers with hidden dimensions 256 and 128, respectively. In the transformation from time space to frequency space, we use the full spectrum (symmetrical) thus $\boldsymbol{x}^{\mathrm{T}}$ and $\boldsymbol{x}^{\mathrm{F}}$ have the same dimension. For simplify, the frequency encoder F_{F} and projector R_{F} have same structure (but different parameters) with their counterparts in time space. Our preliminary experiments show that the change of model structure has relatively small change on performance. For example, replacing F_{T} by 2-layer LSTM or 3-layer Transformer only cause a slight drop of F1 score ($\pm$ 1.2%; one-to-one setting; Scenario 1). In pre-training, we use Adam optimizer with learning rate of 0.0003 and 2-norm penalty coefficient of 0.0005. We use batch size of 64 and training epoch of 40. We record the best performance in terms of F1 score and save the parameters in the corresponding epoch as Θ. In fine-tuning, we initialize the model parameters with Θ and optimize the model on fine-tuning set. The hyper-parameters are the same as in pre-training. For the classification task in fine-tuning stage, we adopt a 2-layer fully-connected layer as a classifier. The hidden dimensions are 64 and the number classes in target dataset, respectively. We measure the classification loss via the cross entropy function and the loss is jointly optimized with the fine-tuning loss (*i.e.*, $\mathcal{L}_{\text{TF-C}}$).

TS-SD [12] uses a modified attention mechanism to encode latent features and use denoising or DTW-similarity prediction as pretext-tasks.The encoder network is based on self-attention mechanism [68] but instead of the linear layer that produce the K, Q, V matrices from the input sequence, TS-SD uses a single convolutional layer to replace the linear layer. With multihead attention, TS-SD sets a different kernel size for each head to attempt capture short- and long-range temporal patterns. Although we found that 12 heads was relatively redundant and may be replaced by fewer heads, we still use 12 heads to keep the same with the original work in [12]. For self-supervised pre-training, we adopt the denoising pretext task that attempts to minimize the mean square distance between the original input time series and the output of the encoder, for which the input is an augmented time series sample with noise added to a subseries. To cope with small numerical values, we use a learning rate of 3e-7 in pre-training and 3e-4 during fine-tuning. While training on pre-training dataset for the four scenarios, the batch sizes varied from 4 to 128 and were manually chosen to achieve reasonable performance for each scenario. During fine-tuning, we use # epochs ranging from 20 to 80 and a batch size of 16. Although the source code of TS-SD is not released, we implement the TS-SD model to our best understanding and tune the model to achieve the best performance as a strong baseline. We public the implemented TS-SD in our online repository.

TS2vec [47] introduces the notion of contextual consistency and uses a hierarchical loss function to capture long-range structure in time series. TS2vec is a powerful representative learning method and have a specially-designed architecture. The encoder network consists of three components. First, the input time series is augmented by selecting overlapping subseries. They are projected into a higher dimensional latent space. Then, latent vectors for input time series are masked at randomly chosen positions. Finally, a dilated CNN with residual blocks produce the contextual representations. To compute the loss, the representations are gradually pooled along the time dimension and at each step a loss function based on contextual consistency is applied. For the baseline experiment, we found that the original 10 layers of ResNet blocks is redundant, and we reduce residual blocks in the encoder from 10-layer to 2-layer without compromising model performances. To make the model comparable with other baselines, we also restrict the hidden dimension to 1 and use a batch size of 64, except in the second scenario we reduce it to 16 given the length of samples in the **FD-A**/**FD-B** datasets.

CLOCS [41] assumes that samples from different subjects are negative pairs in contrastive learning and applies it to classify ECG signals. CLOCS makes the basic assumption that time series samples from different patients (the application scenario is in learning cardiac signals) should have dissimilar representations. Based on that, they devised three slightly different architectures - CMSC, CMLC, and CMSMLC, where ECG recordings from non-overlapping temporal segments, different ECG leads, and both of these are treated as positive pairs in case if they come from the same patients. For this model, the encoder consists of alternating convolutional and pooling layers. So given the different input lengths, the hyper-parameters for these layers have to be modified accordingly. For example, we use kernel sizes of $7, 7, 7$, stride of $3, 3, 3$, pooling layer size of 2 for the last scenario of **ECG** to **EMG** but increased stride to 4 and increased representation size for a time series sample from 64 to 128 in case of **FD-A** to **FD-B** due to much longer time series samples.

Mixing-up [18] proposes new mixing-up augmentation and pretext tasks that aim to correctly predict the mixing proportion of two time series samples. In Mixing-up, the augmentation is chosen as the convex combination of two randomly drawn time series from the dataset, where the mixing parameter is random drawn from a beta distribution. The contrastive loss is then computed between the two inputs and the augmented time series. The loss is a minor modification of NT-Xent loss and is designed to encourage the correct prediction of the amount of mixing. We use the same beta distribution as reported in the original Mixing-up model.

TS-TCC [48] leverages contextual information with a transformer-based autoregressive model and ensures transferability by using both strong and weak augmentations. TS-TCC proposed a challenging pretext task. An input time series sample is first augmented by adding noise, scaling, and permuting time series. The views are then passed through an encoder consisting of three convolutional layers before processed by the temporal contrasting module. During temporal contrasting, for each view, a transformer architecture is used to learn a contextual representation. The learned representation is then used to predict latent observation of the other augmented view at a future time. The contextual representations are then projected and maximized similarity using NT-Xent loss. For this baseline, we mostly adopted the hyper-parameters presented in the original paper. We use a learning rate of 3e-4 throughout, and a batch size of 128 during pre-training and 16 during fine-tuning.

SimCLR [40] is a state-of-the-art model in self-supervised representation learning of images. It utilizes deep learning architectures to generate augmentation-based embeddings and optimize the model parameters by minimizing NT-Xent loss in the embedding space. [40] applies the original SimCLR model for time series data. In this work, we compare with the modified SimCLR as in [40]. The SimCLR contrastive learning framework consists of four major components. Although initially proposed for image data, it is readily adapted to time series, as shown in [40]. An input time series sample is first stochastically augmented into two related views. Then a base encoder extracts representation vectors. ResNet is used as encoder backbone for simplicity. Then the projection head transforms representations into a latent space where the NT-Xent loss is applied. For time series, SimCLR investigated different augmentations, including adding noise, scaling, rotation, negation, flipping in time, permutation of subseries, time warping, and channel shuffling. All the unmentioned hyper-parameters are kept the same with the original model.

TNC [46] focuses on learning representations that encode the underlying state of a non-stationary time series. This is achieved by ensuring that the distribution of observations in the latent space is different from the distribution of temporally separated observations. For each time point, the TNC model calculates a neighborhood size using the Augmented Dickey–Fuller test. The encoded representations of two different windows will be passed to a discriminator to predict the probability that they belong to the same temporal neighborhood. We implement the TFC model following the public GitHub repository provided by the authors. We observed that the ADF tests are extremely slow, so TNC is only used as a baseline in scenario 1 in one-to-one setting (classification task).

CPC [30] aims to learn the representations that encode the global information shared across different segments of a time series signal by the strategy of predictive coding. Instead of modelling the conditional distribution at future time points for prediction, the authors choose to encode the context and observation into a vector representation in a way that maximally preserves their mutual information. For the model architecture, a nonlinear encoder first maps the sequence of observations to latent representations. Then an autoregressive model summarizes past latent representations to produce a context representation at every time point. The two components are trained to jointly optimize a loss function which is inspired by noise contrastive estimation. We used the implementation provided by the authors of the TNC paper [46] and kept their default choices of parameters whenever reasonable.

The TS-SD is the only method, to our knowledge, that explicitly aims at time series pre-training but the underlying assumption that can support knowledge transfer across different time series datasets is not clearly provided. The TS2vec, CLOCS, Mixing-up, TS-TCC, TNC, and CPC are designed for representation learning (instead of pre-training) but involved transfer learning setting in their experiments. We modify them to perform pre-training task and make them comparable to our model. The SimCLR is a widely use baseline in computer vision and simply adopted to deal with time series by [40]. We compare our model with SimCLR to further show our superiority.

Appendix F Additional results on one-to-one evaluation

In one-to-one pre-training evaluation, we consider four different scenarios with paired pre-training and fine-tuning datasets from different fields (Sec. 5.1). In each scenario, the pre-training and fine-tuning datasets share same semantic meanings (*e.g.*, both EEG signals; Scenario 1) or similar semantic meanings (*e.g.*, ECG and EMG are both physiological signals; Scenario 4). For example, in the first scenario, **SLEEPEEG** and **EPILEPSY** both involve EEG measurements but the recordings are taken under different physical conditions.

For each scenario, we first train each baseline model using the pre-training dataset and record the converged parameters. The parameters are then taken to initialize the model on the fine-tuning phase which uses the target dataset. In fine-tuning, we allowed all parameters in the model to be optimized (no freezing of earlier layers; full fine-tuning). Our preliminary results show that the performance of full fine-tuning is slightly better than partial fine-tuning by 5.4% in F1 score (one-to-one setting; Scenario 1). After fine-tuning, we evaluate the final model on the test set and report metrics including accuracy, precision (macro-averaged), recall, F1-score, AUROC (one-versus-rest), and AUPRC. The evaluation results of four scenarios are shown in Table 1 (main paper) and Tables 4-6.

Table 7: Performance on downstream clustering. Pre-training is performed on SLEEPEEG dataset, followed by an independent fine-tuning on EPILEPSY. We compare with five baselines including two non-transfer-based baselines (Random Init. and Non-DL), the best performing baseline in the context of classification task (i.e., TS-TCC), and two new models (TNC and CPC).

Method	Silhouette	ARI	NMI
Non-DL(KNN)	0.1208±0.0271	0.0549± 0.0059	0.0096± 0.0014
Random Init.	0.3497± 0.0509	0.3216±0.0136	0.2408± 0.0287
TNC	0.2353± 0.0018	0.0211± 0.0061	0.0082± 0.0018
CPC	0.2223±0.0011	0.0153± 0.0196	0.0063± 0.0055
TS-TCC	0.5154± 0.0458	0.6307± 0.0325	0.5178± 0.0283
TF-C (Ours)	0.5439± 0.0417	0.6583±0.0259	0.5567±0.0172

Table 8: Performance on downstream anomaly detection. We pre-train the model on FD-A and fine-tune on an anomaly detection subset of FD-B. The subset is highly imbalance (90% normal samples and 10% abnormal samples). We evaluate the performance with precision, recall, F1 score and AUROC.

Models	Precision	Recall	F1 score	AUROC
Non-DL(KNN)	0.4785±0.0356	0.6159±0.0585	0.5061±0.0278	0.7653±0.0153
Random Init.	0.8219±0.05319	0.7131±0.02773	0.7639±0.0094	0.8174±0.0218
TNC	0.8354±0.0484	0.7882±0.0157	0.7957±0.0165	0.8231±0.0379
CPC	0.5967±0.0776	0.4896±0.0866	0.5238±0.0792	0.7859±0.0356
TS-TCC	0.6219±0.0183	0.4431±0.0658	0.4759±0.0562	0.7966±0.0441
TF-C (Ours)	0.8526±0.0367	0.7823±0.0299	0.8312±0.0186	0.8598±0.0283

Appendix G Results on downstream clustering and anomaly detection tasks

Beyond the common downstream classification tasks, we conducted extensive experiments to examine the proposed TF-C in diverse downstream tasks. The comparison results with state-of-the-art baselines are shown in Table 7 for clustering and in Table 8 for anomaly detection. The experimental setups and results analysis are shown in Section 5.

Appendix H Further results on ablation study

We conduct ablation studies (Sec. 5) to evaluate the importance of every component in the developed TF-C model. For the example of one-to-one setting (**SLEEPEEG** $\rightarrow$ **GESTURE**) when pre-training the model using **SLEEPEEG** dataset and fine-tuning on **GESTURE** dataset, ablation study results are shown in Table 9. Through comparison, we observe that the full TF-C model achieves the highest performance in every evaluation metrics, indicating every component, especially the novel consistency loss $\mathcal{L}_{\text{C}}$), contributes to the model's performance.

Appendix I Visualization of embeddings in time-frequency space

To better demonstrate the effectiveness of the developed consistency loss $\mathcal{L}_{\text{C}}$, we visualize the learned embeddings in time-frequency space. In one-to-one setting, we pre-train the TF-C model on **SLEEPEEG** and fine-tune on **EPILEPSY** (Sec. 5.1). We visualize the testing samples of **EPILEPSY** in fine-tuning stage. The learned time-based embeddings $\boldsymbol{z}_i^{\text{T}}$ and frequency-based embedding $\boldsymbol{z}_i^{\text{F}}$ have 128 dimensions. We map the embeddings to a 2-dimensional space through UMAP (20 neighbors; minimum distance as 0.2; cosine distance) for visualization.

In this work, we develop a novel consistency loss $\mathcal{L}_{\text{C}}$ to induce the model learn closer time-based and frequency-based embeddings in a time-frequency space (Sec. 4.3). As our problem setting is that the fine-tuning dataset has small scale ($M \ll N$; Sec. 3), we randomly select 100 samples from the testing set (**EPILEPSY**) for visualization. The visualization of the learned embeddings in time-frequency space is shown in Figure 3 (without $\mathcal{L}_{\text{C}}$ v.s. with $\mathcal{L}_{\text{C}}$). We can observe that: (1) when without $\mathcal{L}_{\text{C}}$ (panel a), the time-based and frequency-based embeddings are clustered separately; the embeddings are largely clustered together when having $\mathcal{L}_{\text{C}}$ (panel b). The observation demonstrates that the proposed $\mathcal{L}_{\text{C}}$ has the ability to push time-based and frequency-based embeddings closer to each other. (2) for a specific sample $\boldsymbol{x}_i$ (marked as dashed line), we can find that its time-based

Table 9: Ablation study (**SLEEPEEG** → **EPILEPSY**). If we remove $\mathcal{L}_{\text{T}}$, we cannot calculate $\mathcal{L}_{\text{C}}$, thus we remove both $\mathcal{L}_{\text{T}}$ and $\mathcal{L}_{\text{C}}$ (instead of only remove $\mathcal{L}_{\text{T}}$). The same for $\mathcal{L}_{\text{F}}$. "W/o $\mathcal{L}_{\text{C}}$ and $\mathcal{L}_{\text{T}}$" means removing the time encoder (Sec. 4.1) and the consistency module (Sec. 4.3), using $\boldsymbol{z}^{\text{F}}$ as final embedding for downstream tasks in fine-tuning. "W/o $\mathcal{L}_{\text{C}}$ and $\mathcal{L}_{\text{F}}$" means removing the frequency encoder (Sec. 4.2) and the consistency module, using $\boldsymbol{z}^{\text{T}}$ as final embedding for downstream tasks in fine-tuning. W/o $\mathcal{L}_{\text{C}}$ means removing the consistency module, using $[\boldsymbol{z}^{\text{T}}; \boldsymbol{z}^{\text{F}}]$ as the final embedding for downstream tasks in fine-tuning. "Replace $\mathcal{L}_{\text{C}}$ with $\mathcal{L}_{\text{FF-C}}$" refers to replacing Eq. 3 by the distance $d(\boldsymbol{z}_i^{\text{F}}, \widetilde{\boldsymbol{z}}_i^{\text{F}}, \mathcal{D}^{\text{pret}})$ between two frequency-based embeddings. "Replace $\mathcal{L}_{\text{C}}$ with $\mathcal{L}_{\text{TT-C}}$" refers to replacing Eq. 3 by the distance $d(\boldsymbol{z}_i^{\text{T}}, \widetilde{\boldsymbol{z}}_i^{\text{T}}, \mathcal{D}^{\text{pret}})$ between two time-based embeddings.

	Accuracy	**Precision**	**Recall**	**F1 score**	**AUROC**	**AUPRC**
W/o $\mathcal{L}_{\text{C}}$ and $\mathcal{L}_{\text{T}}$	0.7159±0.0128	0.7211±0.0428	0.7246±0.0428	0.7239±0.0429	0.8597±0.0236	0.7655±0.0386
W/o $\mathcal{L}_{\text{C}}$ and $\mathcal{L}_{\text{F}}$	0.7327±0.0328	0.7246±0.0311	0.7339±0.0307	0.7317±0.0356	0.8991±0.0279	0.7236±0.0278
W/o $\mathcal{L}_{\text{C}}$	0.7428±0.0297	0.7289±0.0278	0.7451±0.0263	0.7377±0.0308	0.9125±0.0167	0.7706±0.0135
Replace $\mathcal{L}_{\text{C}}$ with $\mathcal{L}_{\text{FF-C}}$	0.7259±0.0072	0.7319±0.0256	0.7338±0.0133	0.7341±0.0194	0.9015±0.0135	0.7529±0.0096
Replace $\mathcal{L}_{\text{C}}$ with $\mathcal{L}_{\text{TT-C}}$	0.7124±0.0091	0.7256±0.0169	0.7231±0.0197	0.7296±0.0209	0.8726±0.0098	0.7627±0.0107
Full Model (TF-C)	**0.7642±0.0196**	**0.7731±0.0355**	**0.7429±0.0268**	**0.7572±0.0311**	**0.9238±0.0159**	**0.7961±0.0109**

Table 10: One-to-many pre-training evaluation. Pre-training is performed on SLEEPEEG dataset, followed by an independent fine-tuning on EPILEPSY, FD-B, GESTURE, and EMG.

Scenarios	**Models**	**Accuracy**	**Precision**	**Recall**	**F1 score**	**AUROC**	**AUPRC**
SLEEPEEG ↓ EPILEPSY	**Non-DL (KNN)**	0.8525±0.0000	0.8639±0.0000	0.6431±0.0000	0.6791±0.0000	0.6434±0.0000	0.6279±0.0000
	Random Init.	0.8983±0.0656	0.9213±0.1369	0.7447±0.1135	0.7959±0.1208	0.8578±0.2153	0.6489±0.1926
	TS-SD	0.8952±0.0522	0.8018±0.2244	0.7647±0.1485	0.7767±0.1855	0.7677±0.2452	0.7940±0.1825
	TS2vec	0.9395±0.0044	0.9059±0.0116	0.9039±0.0118	0.9045±0.0067	0.9587±0.0086	0.9430±0.0103
	CLOCS	**0.9507±0.0027**	0.9301±0.0067	**0.9127±0.0165**	**0.9206±0.0066**	0.9803±0.0023	0.9609±0.0116
	Mixing-up	0.8021±0.0000	0.4011±0.0000	0.5000±0.0000	0.4451±0.0000	0.9743±0.0081	0.9618±0.0104
	TS-TCC	0.9253±0.0098	0.9451±0.0049	0.8181±0.0257	0.8633±0.0215	0.9842±0.0034	0.9744±0.0043
	SimCLR	0.9071±0.0344	0.9221±0.0166	0.7864±0.1071	0.8178±0.0998	0.9045±0.0539	0.9128±0.0205
	TF-C (Ours)	0.9495±0.0249	**0.9456±0.0108**	0.8908±0.0216	0.9149±0.0534	**0.9811±0.0237**	**0.9703±0.0199**
SLEEPEEG ↓ FD-B	**Non-DL (KNN)**	0.4473±0.0000	0.2847±0.0000	0.3275±0.0000	0.2284±0.0000	0.4946±0.0000	0.3308±0.0000
	Random Init.	0.4736±0.0623	0.4829±0.0529	0.5235±0.1023	0.4911±0.0590	0.7864±0.0349	0.7528±0.0254
	TS-SD	0.5566±0.0210	0.5710±0.0535	0.6054±0.0272	0.5703±0.0328	0.7196±0.0113	0.5693±0.0532
	TS2vec	0.4790±0.0113	0.4339±0.0092	0.4842±0.0197	0.4389±0.0107	0.6463±0.0130	0.4442±0.0162
	CLOCS	0.4927±0.0310	0.4824±0.0316	0.5873±0.0387	0.4746±0.0485	0.6992±0.0099	0.5501±0.0365
	Mixing-up	0.6789±0.0246	0.7146±0.0343	**0.7613±0.0198**	0.7273±0.0228	0.8209±0.0035	0.7707±0.0042
	TS-TCC	0.5499±0.0220	0.5279±0.0293	0.6396±0.0178	0.5418±0.0338	0.7329±0.0203	0.5824±0.0468
	SimCLR	0.4917±0.0437	0.5446±0.1024	0.4760±0.0885	0.4224±0.1138	0.6619±0.0219	0.5009±0.0477
	TF-C (Ours)	**0.6938±0.0231**	**0.7559±0.0349**	0.7202±0.0257	**0.7487±0.0268**	**0.8965±0.0135**	**0.7871±0.0267**
SLEEPEEG ↓ GESTURE	**Non-DL (KNN)**	0.6833±0.0000	0.6501±0.0000	0.6833±0.0000	0.6443±0.0000	0.8190±0.0000	0.5232±0.0000
	Random Init.	0.4219±0.0629	0.4751±0.0175	0.4963±0.0679	0.4886±0.0459	0.7129±0.0166	0.3358±0.1439
	TS-SD	0.6922±0.0444	0.6698±0.0472	0.6867±0.0488	0.6656±0.0443	0.8725±0.0324	0.6185±0.0966
	TS2vec	0.6917±0.0333	0.6545±0.0358	0.6854±0.0349	0.6570±0.0392	0.8968±0.0123	0.6989±0.0346
	CLOCS	0.4433±0.0518	0.4237±0.0794	0.4433±0.0518	0.4014±0.0602	0.8073±0.0109	0.4460±0.0384
	Mixing-up	0.6933±0.0231	0.6719±0.0232	0.6933±0.0231	0.6497±0.0306	0.8915±0.0261	0.7279±0.0558
	TS-TCC	0.7188±0.0349	0.7135±0.0352	0.7167±0.0373	0.6984±0.0360	0.9099±0.0085	0.7675±0.0201
	SimCLR	0.4804±0.0594	0.5946±0.1623	0.5411±0.1946	0.4955±0.1870	0.8131±0.0521	0.5076±0.1588
	TF-C (Ours)	**0.7642±0.0196**	**0.7731±0.0355**	**0.7429±0.0268**	**0.7572±0.0311**	**0.9238±0.0159**	**0.7961±0.0109**
SLEEPEEG ↓ EMG	**Non-DL (KNN)**	0.4390±0.0000	0.3772±0.0000	0.5143±0.0000	0.3979±0.0000	0.6025±0.0000	0.4084±0.0000
	Random Init.	0.7780±0.0729	0.5909±0.0625	0.6667±0.0135	0.6238±0.0267	0.9109±0.1239	0.7771±0.1427
	TS-SD	0.4606±0.0000	0.1545±0.0000	0.3333±0.0000	0.2111±0.0000	0.5005±0.0126	0.3775±0.0110
	TS2vec	0.7854±0.0318	**0.8040±0.0750**	0.6785±0.0396	0.6766±0.0501	**0.9331±0.0164**	**0.8436±0.0372**
	CLOCS	0.6985±0.0323	0.5306±0.0750	0.5354±0.0291	0.5139±0.0409	0.7923±0.0573	0.6484±0.0680
	Mixing-up	0.3024±0.0534	0.1099±0.0126	0.2583±0.0456	0.1541±0.0204	0.4506±0.1718	0.3660±0.1635
	TS-TCC	0.7889±0.0192	0.5851±0.0974	0.6310±0.0991	0.5904±0.0952	0.8851±0.0113	0.7939±0.0386
	SimCLR	0.6146±0.0582	0.5361±0.1724	0.4990±0.1214	0.4708±0.1486	0.7799±0.1344	0.6392±0.1596
	TF-C (Ours)	**0.8171±0.0287**	0.7265±0.0353	**0.8159±0.0289**	**0.7683±0.0311**	0.9152±0.0211	0.8329±0.0137

embedding (colored as light blue) and frequency-based (colored as dark blue) embedding are closer to each other in panel (b) than in panel (a). In particular, the cosine distance, between $\boldsymbol{z}_i^{\text{T}}$ and $\boldsymbol{z}_i^{\text{F}}$ of this sample, has reduced from 0.89 to 0.71 when using $\mathcal{L}_{\text{C}}$ in our model. (3) the frequency-based embeddings of 'epilepsy' and 'health' samples are largely overlapped when without $\mathcal{L}_{\text{C}}$ but clustered well when considering $\mathcal{L}_{\text{C}}$, indicating the learned embeddings (with $\mathcal{L}_{\text{C}}$) can potentially enhance downstream classification performance.

Furthermore, we attempt to quantify the impact of our consistency loss item on the quality of learned embeddings, using the one-to-many setting as shown in Figure 4. We pretrain the TF-C model on **SLEEPEEG** dataset and fine-tuning on **EPILEPSY**, **FD-B**, **GESTURE**, and **EMG**, respectively. Next, we explain how the cosine distance is calculated taking **EMG** dataset as an example. For

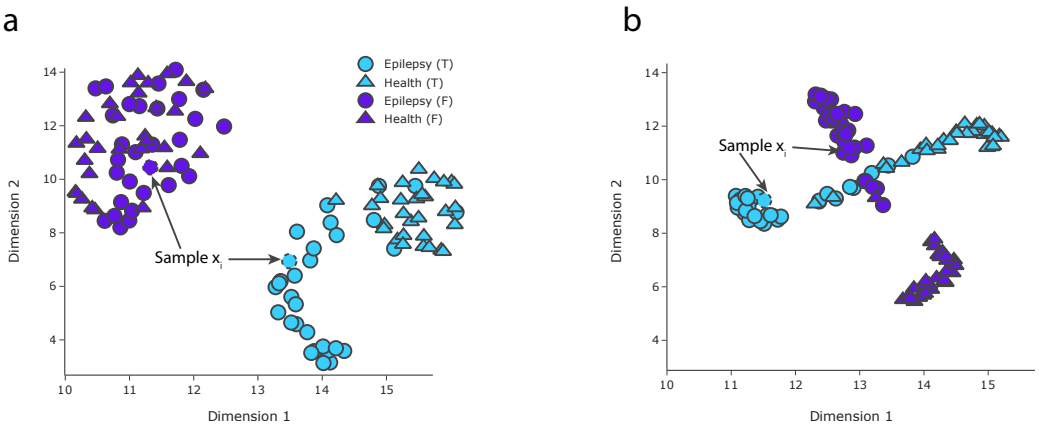

Figure 3: Visualizing time-based and frequency-based embeddings in **EPILEPSY** (one-to-one; pre-training on **SLEEPEEG**). (a) Without consistency loss item $\mathcal{L}_{\text{C}}$ in both pre-training and fine-tuning. (b) With consistency loss item in pre-training and fine-tuning. Dark blue and light blue denote time-based embedding $\boldsymbol{z}_i^{\text{T}}$ and frequency-based embedding $\boldsymbol{z}_i^{\text{F}}$, respectively. Each circle denotes an 'epilepsy' sample, i.e., the subject has a seizure episode, while each triangle denotes a sample associated with 'health', i.e., the subject is healthy. The annotated $\boldsymbol{x}_i$ (marked as dashed line) is the same sample in two panels.

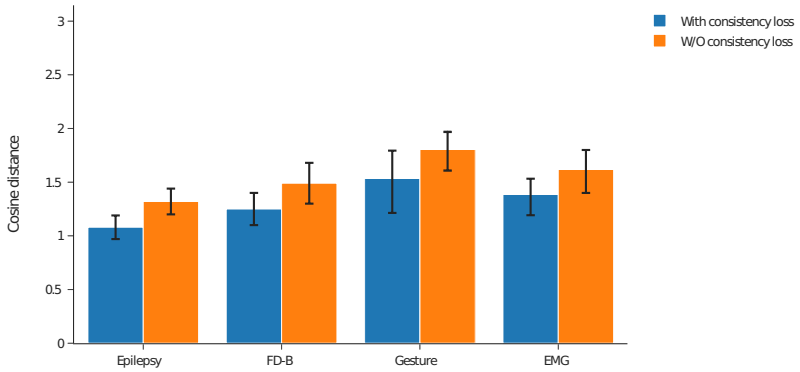

Figure 4: Cosine distance between *time-based and frequency-based embeddings*. Here, lower distance is better as it denotes higher consistency between time-based and frequency-based embeddings. The results are reported for the one-to-many setting, *i.e.*, pretraining the model on **SLEEPEEG** dataset and fine-tuning on **EPILEPSY**, **FD-B**, **GESTURE**, and **EMG**, respectively. In a specific dataset, *for each sample*, we measure the cosine distance between its time-based embedding $\boldsymbol{z}_i^{\text{T}}$ and frequency-based embedding $\boldsymbol{z}_i^{\text{F}}$ in a 128-dimension time-frequency space. In this figure, we report the cosine distance averaged across all testing samples. We observe that the cosine distance is larger when removing our proposed consistency loss item $\mathcal{L}_{\text{C}}$, indicating our model indeed pushes the time-based and frequency-based embeddings closer to each other.

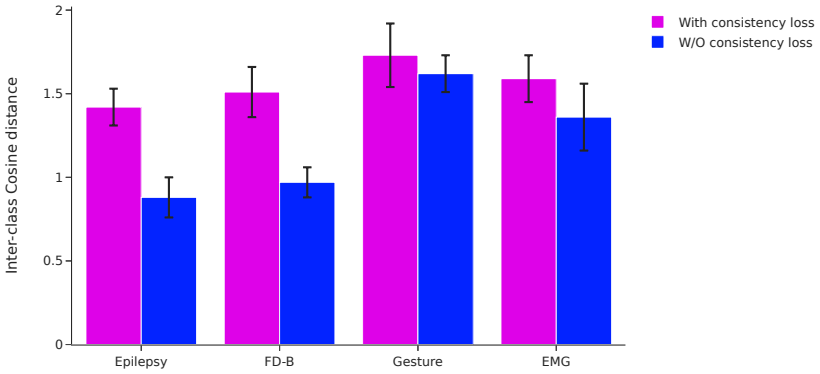

Figure 5: Cosine distance among *embeddings of samples belonging to different classes*. The higher distance indicates better discrimination and better downstream classification performance. As in Figure 4, the current experimental setup uses the one-to-many setting, but the calculation of cosine distance is slightly different. Take **EPILEPSY** as an example, we first measure the distance between a positive sample and all the negative samples, then take the averaged distance across all positive samples as inter-class distance. The distance is calculated based on the concatenation of time-based and frequency-based embeddings, *i.e.*, $\boldsymbol{z}_i^{\text{tune}}$. We observe that our TF-C model (with consistency loss $\mathcal{L}_{\text{C}}$ increase the inter-class distance from 0.88 to 1.42 with a remarkable margin of 61.4% (one-to-many setting; testing set of **EPILEPSY**), indicating our model indeed has the ability to increase the distinction of learned embeddings and bring better classification performance on the fine-tuning dataset of interest. The margins in **GESTURE** and **EMG** are relatively low, and one potential reason is that the testing set of them are very small (120 for **GESTURE**; 41 for **EMG**).

every sample $\boldsymbol{x}_i$ in **EMG**, we measure the cosine distance between its time-based embedding $\boldsymbol{z}_i^{\text{T}}$ and frequency-based embedding $\boldsymbol{z}_i^{\text{F}}$ in the 128-dimension time-frequency space. Then, we take the average of the distance across all testing samples and report in Figure 4. The number of testing samples of each dataset can be found in Table 3. We find that the cosine distance is larger when removing the proposed consistency loss item $\mathcal{L}_{\text{C}}$, which means our model, as designed and expected, indeed pushes the time-based and frequency-based embeddings closer to each other. For example, in **EPILEPSY**, the averaged cosine distance dropped 22.2% (from 1.32 to 1.08) when using consistency loss (Sec. 4.3). At the same time, the embeddings of samples drawn from different classes are more easily distinguished from each other, as shown in Figure 5: we observe that our TF-C model (with consistency loss $\mathcal{L}_{\text{T}}$ increase the inter-class cosine distance from 0.88 to 1.42 with a remarkable margin of 61.4% (one-to-many setting; testing set of **EPILEPSY**), indicating our model indeed has the ability to increase the distinctiveness of learned embeddings and bring forward better classification performance on the fine-tuning dataset of interest. The results of visualization increase our confidence that the developed TF-C pre-training model can capture generalizable properties and boost knowledge transfer across different time series datasets.

Appendix J Additional information on domain augmentations

To the best of our knowledge, this is the first work that direct perturbs the frequency domain of time series in contrastive learning. Here, we thoroughly explore various frequency augmentations and discuss the design of manipulation strategy in frequency domain. In particular, we analyze eight frequency augmentation policies and discuss their performance.

We consider three types of frequency augmentations:

- **Low- vs. High-band perturbations**: High-frequency components contribute to the fast varying parts of the time series (e.g., sharp transitions). In contrast, low-frequency components contribute to slow signal variations in the time domain. Low band perturbations correspond to the first half of the frequency spectrum.

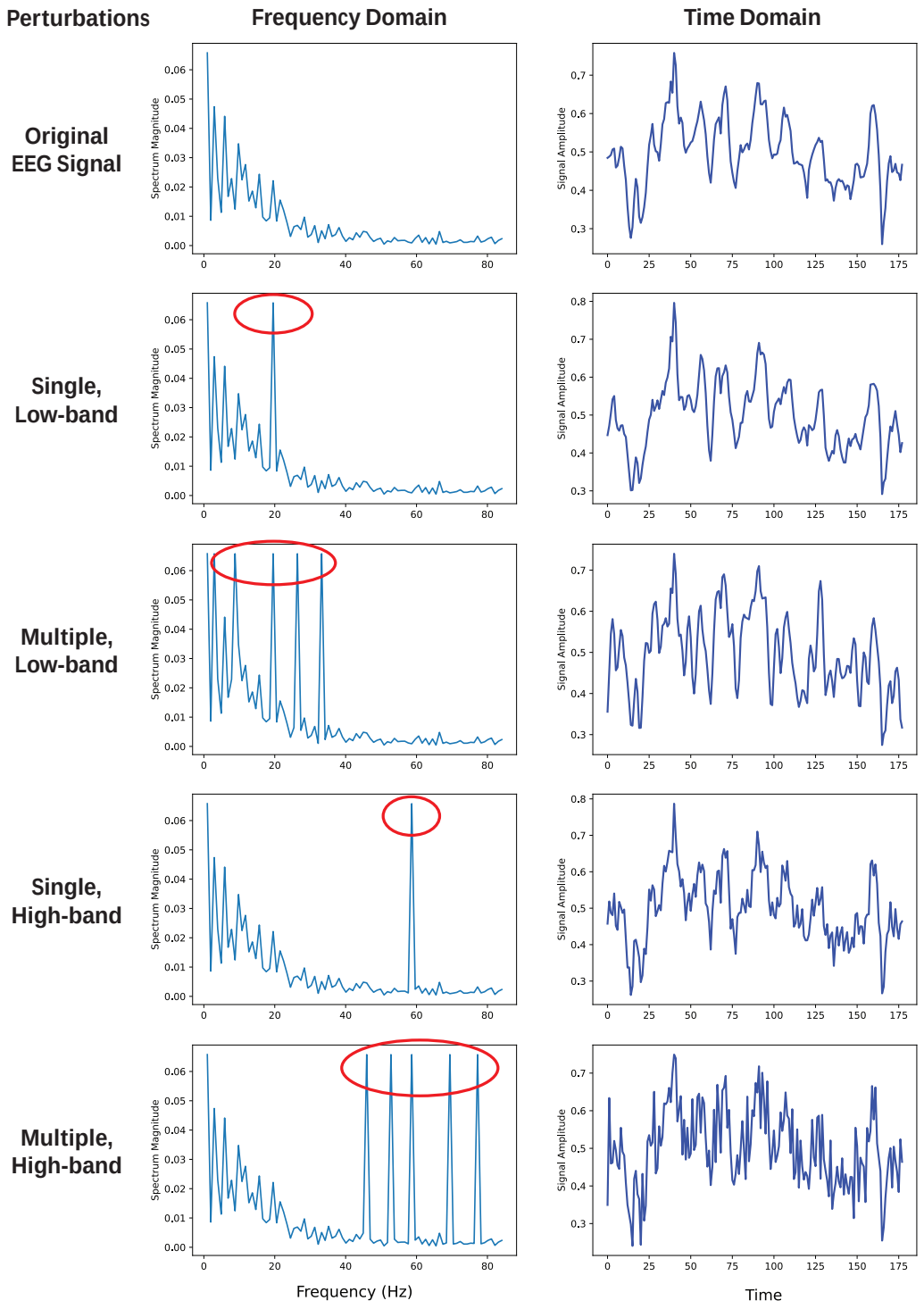

Figure 6: Illustration of augmentations in the frequency domain and the corresponding influence on the time domain. We considered the frequency augmentation from three aspects: low-band v.s. high-band, single-perturbation v.s. multiple perturbation, random selection v.s. distribution-based selection. The selection method has relative slight effect on time domain, thus, we mainly show how the perturbation budget (single or multiple) and frequency bands affect the temporal pattern in time domain. Here, for better visualization, we select $\alpha = 1$ which means the perturbed component has the maximum magnitude (we set $\alpha = 0.5$ in TF-C model) and set budget as 5 in multiple perturbation. The developed augmentations in this paper only perturb magnitude and keep phase information unchanged. We notice that perturbing phase is a crucial research direction in frequency domain augmentation in the future work.

Table 11: Results comparison of eight frequency augmentations. Considering the perturbing bands, the number of components, and the perturbation distribution, there are eight different combinations. Here we regard each factor as binary (e.g., Low v.s. High), we note that the more fine-grid factors (e.g., Delta band v.s. Alpha band v.s. Beta band v.s. Gamma band in the context of EEG signals) might be helpful for augmentation design.

No.	Low vs. High	Single vs. Multi	Random vs. Gaussian	F1 score
1	Low	Single	Random	0.9136
2	Low	Single	Gaussian	0.8957
3	Low	Multi	Random	0.8763
4	Low	Multi	Gaussian	0.8859
5	High	Single	Random	0.8972
6	High	Single	Gaussian	0.8651
7	High	Multi	Random	0.8818
8	High	Multi	Gaussian	0.8775

- **Single- vs. Multi-component perturbations** (aka varying budget size): Perturbing several frequency components (i.e., larger budget) leads to more substantial changes in the time domain. In this analysis, we set the budget to 5 for multi-component perturbations and 1 for single-component perturbations.
- **Random vs. Distributional perturbations**: The selection of components in the spectrum that get perturbed is important because it influences the resulting time-based representation of the augmented sample. Frequency components can be sampled uniformly at random or according to a user-specified distribution (i.e., Gaussian in the following analysis).

Based on the above considerations, we developed eight frequency augmentations. In Fig. 6 (EPILEPSY dataset)), we visualize a series of perturbations in the frequency domain and their effects in the time domain (through the inverse FFT).

In addition, we explore how different spectral perturbations affect the performance of TF-C. Table 11 shows results for the one-to-one setup (SLEEPEEG $\rightarrow$ EPILEPSY). We find the following:

- Perturbing low-band components performs better than augmenting high-band elements of the spectrum (0.8% change in F1 performance on average across the four setups). This can be explained by the nature of the underlying time series: in physiological signals, most information is carried in a low band (e.g., EEG signals are most informative in 0.5-70 Hz while > 70Hz is considered noise). This means that the choice of low- vs. high-band components depends on the signal context. For example, high-band components are informative for mechanical signals. We further verified this by external experiments in the FD-A $\rightarrow$ FD-B setting, where perturbing high-band components outperform low-band augmentations by 0.8% in F1. Thus, we consider perturbations across the entire spectral band in the paper to achieve broad generalization.
- In general, perturbing many frequency components degrades model performance. One reason for worse performance is that too big a budget incurs too significant changes in the time-based representation of augmented samples. In that case, the augmented samples would be easily distinguished by a contrastive model, leading to poor contrastive encoders. For this reason, we use single-component perturbations in the paper.
- We cannot draw a clear conclusion on whether the random selection of components is better than Gaussian distribution-based as both F1 scores are very similar. The optimal choice of sampling strategy likely depends on both the time-series signal and prediction task. We use random sampling in the paper.

Appendix K Discussion on many-to-one setting

We probed the one-to-one setting and one-to-many setting in knowledge transfer of time series (Sec. 5). Here, we discuss the many-to-one set up: pre-train on a mixture of multiple datasets and fine-tune on a single pure dataset.

We notice that the many-to-one setup fundamentally differs from the one-to-one and also from one-to-many configurations. Notably, in the one-to-one and one-to-many setups, the model is pre-trained on a single pre-training dataset presumed to originate from a data source exhibiting homogeneity. For

example, samples in the pre-training dataset could describe similar devices (e.g., different kinds of boring machines) with similar sampling rates, lengths, semantic meanings, and/or relevant temporal patterns that occur on comparable time scales, etc. When some level of homogeneity exists in a pre-training dataset, the method can learn patterns and apply them to different fine-tuning datasets. In contrast, in many-to-one setups, pre-training datasets can be incredibly heterogeneous. For example, a pre-training dataset might contain samples describing boring machines' behavior and recordings of EEG medical devices. For this reason, pre-training can be severely hampered by diversity and misalignments across samples in the pre-training dataset. When we merge many heterogeneous samples into a single pre-training dataset, the model might struggle to find patterns underlying the combined dataset. While models in computer vision have been successfully pre-trained on large, diverse datasets, such as ImageNet, the literature on pre-training for time series is much sparser. Further, heterogeneous pre-training datasets pose additional challenges due to the unique properties of time series. Learning universal representations from multi-sourced time series datasets (i.e., many-to-one setups) remains an open challenge. It is an interesting future direction, and we hope this study can facilitate future research.

We have explored this and provide preliminary experimental results in four many-to-one setups (named N-to-one for N = 1,2, 3, and 4). Take four-to-one as an example. We merge four pre-training datasets (SLEEPEEG,FD-A, HAR, and ECG) into one pre-training dataset. We had to preprocess the datasets to make a many-to-one setup feasible. To address the mismatch between the number of channels, for the multivariate dataset (HAR), we take a single channel (i.e., x-axis acceleration) to make sure all samples have the same number of channels. To align the length across datasets, we make all samples have 1,500 observations by zero-padding (for the ones shorter than 1,500) and cutting-off (for the ones longer than 1,500). Further, we take 25% samples (instead of 100%; for computational reasons only) from each of the four datasets to create a pre-training dataset with 108,300 samples that belong to 18 classes in total. Similarly, we built pre-training for three-to-one (merge SLEEPEEG,FD-A, and HAR) and two-to-one (merge SLEEPEEG and FD-A).

We take the above N-to-one pre-training datasets and repeat the following analysis. First, we pre-train our TF-C and then fine-tune it on EPILEPSY for classification. The AUROC for one-to-one, two-to-one, three-to-one, and four-to-one settings are 0.9819, 0.9525, 0.8607, and 0.7253, respectively. We observe AUROC decreases with the increasing heterogeneity of pre-training datasets, which is consistent with our analysis in the first part of this answer. Nevertheless, whether there exists a framework that overcomes the problems and enables many-to-one pre-training for time series remains to be seen, and we believe it's an exciting direction for future research.

Appendix References

[S1] Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. ACM Transactions on Intelligent Systems and Technology (TIST), 11(5):1–46, 2020.

[S2] Zhijun Zhou, Yingtian Zhang, Xiaojing Yu, Panlong Yang, Xiang-Yang Li, Jing Zhao, and Hao Zhou. Xhar: Deep domain adaptation for human activity recognition with smart devices. In 2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), pages 1–9. IEEE, 2020.

[S3] Felix Ott, David Rügamer, Lucas Heublein, Bernd Bischl, and Christopher Mutschler. Domain adaptation for time-series classification to mitigate covariate shift. arXiv:2204.03342, 2022.

[S4] Paulo Roberto de Oliveira da Costa, Alp Akçay, Yingqian Zhang, and Uzay Kaymak. Remaining useful lifetime prediction via deep domain adaptation. Reliability Engineering & System Safety, 195:106682, 2020.

[S5] Dezhi Hao and Xianwen Gao. Multi-weighted partial domain adaptation for sucker rod pump fault diagnosis using motor power data. Mathematics, 10(9):1519, 2022.

[S6] Penghao Zhang, Jiayue Li, Yining Wang, and Judong Pan. Domain adaptation for medical image segmentation: a meta-learning method. Journal of Imaging, 7(2):31, 2021.

[S7] Xiaoxia Wang, Haibo He, and Lusi Li. A hierarchical deep domain adaptation approach for fault diagnosis of power plant thermal system. IEEE Transactions on Industrial Informatics, 15(9):5139–5148, 2019.

[S8] Devis Tuia, Claudio Persello, and Lorenzo Bruzzone. Recent advances in domain adaptation for the classification of remote sensing data. arXiv:2104.07778, 2021.

[S9] Zhao-Hua Liu, Lin-Bo Jiang, Hua-Liang Wei, Lei Chen, and Xiao-Hua Li. Optimal transport-based deep domain adaptation approach for fault diagnosis of rotating machine. IEEE Transactions on Instrumentation and Measurement, 70:1–12, 2021.

[S10] Wei Lin, Anna Kukleva, Kunyang Sun, Horst Possegger, Hilde Kuehne, and Horst Bischof. Cycda: Unsupervised cycle domain adaptation from image to video. arXiv:2203.16244, 2022.

[S11] Dengyu Xiao, Chengjin Qin, Honggan Yu, Yixiang Huang, Chengliang Liu, and Jianwei Zhang. Unsupervised machine fault diagnosis for noisy domain adaptation using marginal denoising autoencoder based on acoustic signals. Measurement, 176:109186, 2021.

[S12] Youngeun Kim and Sungeun Hong. Adaptive graph adversarial networks for partial domain adaptation. IEEE Transactions on Circuits and Systems for Video Technology, 32(1):172–182, 2021.

[S13] Wei Zhang, Xiang Li, Hui Ma, Zhong Luo, and Xu Li. Universal domain adaptation in fault diagnostics with hybrid weighted deep adversarial learning. IEEE Transactions on Industrial Informatics, 17(12):7957–7967, 2021.

[S14] Ronghang Zhu, Xiaodong Jiang, Jiasen Lu, and Sheng Li. Cross-domain graph convolutions for adversarial unsupervised domain adaptation. IEEE Transactions on Neural Networks and Learning Systems, 2021.

[S15] Zhengyang Chen, Shuai Wang, and Yanmin Qian. Self-supervised learning based domain adaptation for robust speaker verification. In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5834–5838. IEEE, 2021.

[S16] Kazuma Fujii and Kazuhiko Kawamoto. Generative and self-supervised domain adaptation for one-stage object detection. Array, 11:100071, 2021.

[S17] Hao Guan and Mingxia Liu. Domain adaptation for medical image analysis: a survey. IEEE Transactions on Biomedical Engineering, 2021.

[S18] Wentao Mao, Ling Ding, Yamin Liu, Sajad Saraygord Afshari, and Xihui Liang. A new deep domain adaptation method with joint adversarial training for online detection of bearing early fault. ISA transactions, 122:444–458, 2022.

[S19] Haifeng Xia, Handong Zhao, and Zhengming Ding. Adaptive adversarial network for source-free domain adaptation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 9010–9019, 2021.

[S20] Muhammad Awais, Fengwei Zhou, Hang Xu, Lanqing Hong, Ping Luo, Sung-Ho Bae, and Zhenguo Li. Adversarial robustness for unsupervised domain adaptation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 8568–8577, 2021.

[S21] Yuxin Ma, Yang Hua, Hanming Deng, Tao Song, Hao Wang, Zhengui Xue, Heng Cao, Ruhui Ma, and Haibing Guan. Self-supervised vessel segmentation via adversarial learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 7536–7545, 2021.

[S22] Qiao Liu and Hui Xue. Adversarial spectral kernel matching for unsupervised time series domain adaptation. In Int Joint Conf Artif Intell, pages 2744–2750, 2021.

[S23] Yuan Yuan and Lei Lin. Self-supervised pretraining of transformers for satellite image time series classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14:474–487, 2020.

[S24] Colorado J Reed, Xiangyu Yue, Ani Nrusimha, Sayna Ebrahimi, Vivek Vijaykumar, Richard Mao, Bo Li, Shanghang Zhang, Devin Guillory, Sean Metzger, et al. Self-supervised pretraining improves self-supervised pretraining. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 2584–2594, 2022.

[S25] Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 699–708, 2020.

[S26] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. arXiv:2103.01988, 2021.

[S27] Peri Akiva, Matthew Purri, Kristin Dana, Beth Tellman, and Tyler Anderson. H2o-net: Self-supervised flood segmentation via adversarial domain adaptation and label refinement. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 111–122, 2021.

[S28] Xinyi Wu, Zhenyao Wu, Hao Guo, Lili Ju, and Song Wang. Dannet: A one-stage domain adaptation network for unsupervised nighttime semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15769–15778, 2021.

[S29] Alexei Baevski, Michael Auli, and Abdelrahman Mohamed. Effectiveness of self-supervised pre-training for speech recognition. arXiv:1911.03912, 2019.

[S30] Matthew McDermott, Brenda Yap, Peter Szolovits, and Marinka Zitnik. Structure inducing pre-training. arXiv:2103.10334, 2021.

[S31] W Hu, B Liu, J Gomes, M Zitnik, P Liang, V Pande, and J Leskovec. Strategies for pre-training graph neural networks. In International Conference on Learning Representations (ICLR), 2020.