Cross-Domain Pre-training with Language Models for Transferable Time Series Representations

Mingyue Cheng
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, Anhui Province, China
mycheng@ustc.edu.cn

Xiaoyu Tao
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, Anhui Province, China
txytiny@mail.ustc.edu.cn

Qi Liu*
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, Anhui Province, China
qiliuql@ustc.edu.cn

Hao Zhang
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, Anhui Province, China
zh2001@mail.ustc.edu.cn

Yiheng Chen
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, Anhui Province, China
cyh_20@mail.ustc.edu.cn

Defu Lian
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, Anhui Province, China
liandefu@ustc.edu.cn

ABSTRACT

Pre-training universal models across multiple domains to enhance downstream tasks is a prevalent learning paradigm. However, there has been minimal progress in pre-training transferable models across domains for time series representation. This dilemma is incurred by two key factors: the limited availability of training set within each domain and the substantial differences in data characteristics between domains. To address these challenges, we present a novel framework, namely CrossTimeNet, designed to perform cross-domain self-supervised pre-training to benefit target tasks. Specifically, to address the issue of data scarcity, we utilize a pre-trained language model as the backbone network to effectively capture the sequence dependencies of the input sequence. Meanwhile, we adopt the recovery of corrupted inputs as a self-supervised optimization objective, taking into account the locality of time series. To address discrepancies in data characteristics, we introduce a novel tokenization module that converts continuous time series inputs into discrete token sequences using vector quantization techniques. This approach facilitates the learning of transferable time series models across different domains. Extensive experimental results on diverse time series tasks, including classification and forecasting, demonstrate the effectiveness of our approach. Our codes are publicly available[1].

KEYWORDS

Cross Domain Transferring, Self-Supervised Learning, Time Series Analysis

1 INTRODUCTION

Time series analysis [1, 2] is a critical challenge in data science, with a wide range of applications that include healthcare diagnostics (e.g., physiological signals) and industrial monitoring (e.g., sensors and the Internet of Things). Recent works [3–5] show that deep learning methods have significantly resolved this area, due to their offering unparalleled scalability and the ability to model complex, nonlinear relationships within domain-specific data compared to classical models.

However, directly applying deep learning-based models does not always yield satisfactory results in real-world applications [6]. The primary issue is that most advanced network architectures, such as Transformer-based models [7], are data-intensive, necessitating the collection of extensive labeled training data from the specific application scenario. To overcome this limitation, a significant amount of recent research proposes utilizing self-supervised learning (SSL) on vast amounts of unlabeled time series data. The core idea is to leverage the knowledge gained during the self-supervised pre-training phase, thereby minimizing the requirement for extensive training resources. The main reason behind this phenomenon is that pre-trained models can enhance performance by discerning valuable patterns within the data from the extensive pre-training datasets. Therefore, various strategies have been explored to extract general-purpose features from different angles, including methods like contrastive learning algorithms [8] and denoising autoencoder models [9, 10]. Specifically, the contrastive learning approach is commonly used in discriminative pretraining methods, where models are trained to learn representations by working with constructed positive and negative pairs [11].

Although these current methods are effective, there are still two limitations. The first limitation is that the time series field often faces the challenge of insufficient training data compared to other fields, making it difficult for models to achieve ideal results with a limited training set. The second limitation is that these existing methods often assume that self-supervised training is confined within the same domain, overlooking the importance of learning transferable knowledge across domains. Meanwhile, learning universal representations has almost become the default setting. From our perspective, cross-domain self-supervised pre-training naturally brings several benefits. To begin with, cross-domain learning helps discover underlying correlations and patterns, thereby enhancing the model's feature understanding and improving its predictive capabilities. In addition, by leveraging a large amount of unlabeled data in different fields, cross-domain learning helps to obtain transferable generalized features and effectively solve the

[1] https://github.com/Mingyue-Cheng/CrossTimeNet

data scarcity problem in specific fields. Furthermore, cross-domain pre-training significantly improves the overall generalization ability of the model, enabling it to perform well on a variety of tasks with minimal domain-specific fine-tuning.

Inspired by the above analysis, we decide to explore a promising but under-studied self-supervised pre-training paradigm to enhance time series representation capabilities. This research is full of challenges. For one thing, to address the data sparsity problem, we need to design encoders that are easy to train to efficiently capture key features and patterns with limited data, thereby improving prediction performance and model transferability. For another, to cope with the data differences between different domains, we need to develop a method to unify instances from various domains, which is crucial for building a cohesive pre-trained model.

In this work, we propose CrossTimeNet, a novel framework specifically designed for self-supervised pre-training of time series data across different domains. First, current research tends to use convolutional [8] or Transformer-based networks [10], which are usually randomly initialized and thus require a large amount of training data to learn effective features. To address this issue, we adopt a pre-trained language model as the backbone network to effectively capture the sequential dependencies of the input data using existing knowledge and features. Considering the locality of time series data, we use the restoration of the corrupted input as the self-supervised optimization objective. This approach can capture bidirectional representations and thus better utilize the contextual information of sequence data. Second, to unify instances from multiple domains, we provide a carefully designed tokenizer that can convert continuous time series into discrete tokens through a reconstruction optimization process. This tokenizer enables each segment of the raw time series to be assigned its own identity code, effectively bridging the gap caused by data discrepancies across domains. Using these strategies, we can develop a general pre-trained time series model that extracts various types of knowledge and patterns from different domains. To evaluate the effectiveness of CrossTimeNet, we conduct comprehensive experiments on time series classification and forecasting tasks using multiple real-world datasets. The experimental results clearly confirm the superiority of our CrossTimeNet from multiple perspectives. We hope that CrossTimeNet can inspire further research to develop general time series representation models.

2 RELATED WORK

The related research primarily falls into two categories: (1) self-supervised time series representation and (2) time series analysis.

2.1 Self-supervised Time Series Representation

A considerable amount of recent research [12] has been centered around self-supervised learning for time series representation. Upon reviewing and synthesizing the current body of work in this area, self-supervised pre-training efforts for time series data can be broadly categorized into the following approaches: encoder-decoder models, contrastive learning-based techniques, and denoising auto-encoder based. **Encoder-decoder models**: The primary philosophy behind this category is to leverage an encoder to transform input time series data into a latent representation, which is then reconstructed back to the original input (or some variant of it) by a decoder. This approach encourages the model to capture essential temporal dynamics and dependencies in the data. **Contrastive learning-based** techniques: This paradigm focuses on learning representations by distinguishing between similar (positive) and dissimilar (negative) pairs of time series segments [13]. Techniques such as TNC [14], TS-TCC [15], and TS2Vec [8] fall under this umbrella, each employing unique mechanisms to define and utilize positive and negative samples for training robust time series representations. **Denoising auto-encoder based** approaches: Methods like TST [9] and SimMTM [10] adopt a reconstructive strategy, where the model is trained to predict missing parts of the input time series or reconstruct the series from distorted versions. In addition, [16–20] have begun to explore the potential of training base models for time series tasks, but this area is still in the exploratory stage and has not yet been fully developed.

2.2 Time Series Analysis

Time series analysis has gained significant attention in recent years [2, 21], In classification tasks, its methods can be roughly divided into distance-based methods, interval-based techniques, shapelet-based [22], and dictionary-based techniques. **Distance-based** methods, such as dynamic time warping (DTW) combined with the nearest neighbor classifier (NN-DTW) [23], form the foundation of traditional TSC. **Interval-based** like time series forest (TSF) [24] extract features from specific time intervals, while **shapelet-based** methods focus on identifying predictive sub-sequences. **Dictionary-based** techniques, exemplified by Symbolic Aggregate approXimation (SAX) [25], transform time series into symbolic representations for analysis. Recently, **deep Learning-based** methods, particularly convolutional neural networks (CNNs) [3, 26, 27] and Transformer architectures [4, 28, 29], have been at the forefront of this wave. Meanwhile, time series forecasting aims to predict future values based on historical data, with applications in finance and supply chain management. Forecasting models range from classic statistical approaches like ARIMA [30], which are limited to stationary sequences, to recurrent neural networks (RNNs) such as LSTM [31] and GRU [32], which face challenges in training efficiency and long-term dependency modeling. Recently, Transformer models have shown promise in this domain, with notable examples including Informer [33] and Autoformer [1], which enhance long-term dependency management and computational efficiency.Although deep neural networks have powerful nonlinear modeling capabilities in the field of time series analysis and have the advantage of not requiring manual feature engineering - thus helping to learn more complex temporal features, their main disadvantage is their huge demand for data. These models require a large number of labeled training sets, without which they are prone to overfitting.

3 PRELIMINARIES

First, we introduce the studied problem, using notation and concepts. Then, we briefly present the relative models in our work.

3.1 Cross-domain Self-supervised

In this work, our primary goal is to create a unified self-supervised pre-training framework that can efficiently handle time series data

from various scenarios, each referred to as a "domain". These domains encompass distinct characteristics and patterns within the time series data. To tackle the challenge of self-supervised pre-training across multiple domains, we introduce $\mathcal{D} = \mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N$, a collection of datasets representing N distinct domains. Each domain dataset $\mathcal{D}_n$ comprises unlabeled time series data $\mathbf{x}_i^n$, with $\mathbf{x}_i^n = [x_{i1}^n, x_{i2}^n, \ldots, x_{iT}^n]$ representing the i-th time series instance in the n-th domain, consisting of T time points. The key challenge lies in leveraging the vast, unlabeled data across these varied domains to train a versatile pre-trained model $\mathcal{P}$. This model is designed to capture universal features and patterns inherent in time series data, going beyond the specific characteristics of individual domains. The self-supervised learning approach enables the model to uncover and utilize the intrinsic structure of the data without depending on explicit class labels, thereby tapping into the unexploited wealth of unlabeled data across different domains.

Once the pre-trained model $\mathcal{P}$ is established, it can be fine-tuned for corresponding downstream tasks, denoted as $\mathcal{T} = \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N$. These tasks may either fall within the same domains as those used in the pre-training phase or extend across different domains, showcasing the adaptability and transferability of the representations learned by the model. The ultimate aim is to employ the pre-trained model $\mathcal{P}$ to boost the performance and generalization capabilities of models $\mathcal{F}_n$ designated for time series analysis tasks within specific domains. Each model $\mathcal{F}_n : \mathbf{x}_i^n \mapsto y_i^n$ is responsible for mapping a time series to its accurate class label within the respective task domain $\mathcal{T}_n$, thereby enhancing the efficacy of time series classification and forecasting across a broad spectrum of domains.

3.2 Pre-trained Language Model

Recently, pre-trained language models (PLM) [34] have revolutionized the field of natural language processing (NLP) by providing a powerful framework for learning rich linguistic representations from vast amounts of textual data. At their core, PLMs are models trained on a large corpus of text in a self-supervised manner, meaning they learn to predict parts of the text based on other parts, without the need for explicit annotations or labels. This training approach allows them to capture various language phenomena and contextual nuances, making them highly versatile and effective for various NLP tasks. The typical text processing pipeline with a PLM begins with tokenization, where the input text is broken down into manageable pieces, often words or subwords, known as tokens. These tokens are then fed into a designed neural network, (e.g., Transformers [7]), which processes them to extract dynamic contextual features. As a result, PLMs have paved the way for breakthroughs in NLP by enabling models to generalize across tasks with impressive accuracy and minimal task-specific tuning. Due to the powerful capacity shown in PLM, it has nearly become a default paradigm in the NLP domain.

4 THE PROPOSED CROSSTIMENET

4.1 Overview of Model Architecture

As depicted in Figure 1, the CrossTimeNet encompasses three core components: (a) time series tokenization, (b) self-supervised pre-training across domains, and (c) domain-specific task fine-tuning. During time series tokenization, a tailor-made tokenizer converts continuous time series data into discrete tokens, establishing a uniform representation suitable for cross-domain applications. Following this, the self-supervised pre-training phase employs a bidirectional token prediction task—where random tokens are masked—to compel the model to deduce missing information, thereby learning potent representations of the time series data. Finally, in the downstream task fine-tuning stage, the model undergoes specific adjustments to excel in domain-related tasks, such as classification, harnessing the extensive knowledge gained from the pre-training process. This fine-tuning is meticulously carried out to ensure the model's proficiency in specialized tasks while retaining the extensive insights from its earlier cross-domain exposure.

4.2 Time Series Tokenization

Time series data poses distinct challenges for cross-domain analysis due to its inherent variability in structure, unlike more consistent modalities like text and vision. Variations in channel numbers, physical phenomena representation, and temporal resolutions across domains complicate the development of universally applicable pre-trained models. For example, the simplicity of financial time series contrasts sharply with the complexity of multichannel EEG data. Addressing this, one may wonder that channel-independent modeling [35] emerges as a promising approach, treating each channel as separate to bridge differences between datasets. However, this method may neglect crucial inter-channel dependencies, which are especially vital in fields like healthcare, where understanding the relationships between different physiological signals is key. This oversight could result in a less comprehensive interpretation of the data, highlighting the need for models that can balance generalization with the retention of critical inter-channel information.

In this work, we decide to adopt a direct yet under-explored path - time series discretation during pre-training time series representations. The main idea is to transform the time series into a format that maintains the essential inter-channel relationships while still accommodating the diverse characteristics of data across domains. This discretization not only standardizes the input data but also preserves the rich, multi-faceted information that is crucial for the subsequent stages of self-supervised pre-training and downstream task fine-tuning. Through this approach, we endeavor to construct a robust and versatile pre-trained model, one that harnesses the full spectrum of information within multi-domain time series data while overcoming the inherent challenges posed by their variability. Although some previous works have explored the discretization of time series, including SAX [25] and SFA [36, 37], these methods suffer from a significant drawback: they are computationally expensive. In addition, these methods exhibit significant limitations when applied to self-supervised pre-training of time series data. They struggle to compress complex temporal sequences effectively, often resulting in substantial information loss. Moreover, these techniques typically require manual tuning, which can introduce subjectivity and limit their scalability.

In light of this, inspired by the success of image compression [38], we employ a auto-encoder framework to achieve efficient time series compression, referred to as time series tokenizer. The tokenizer involves several steps. First, we transform the time series along the time dimension into sequence patches. Formally, for a given

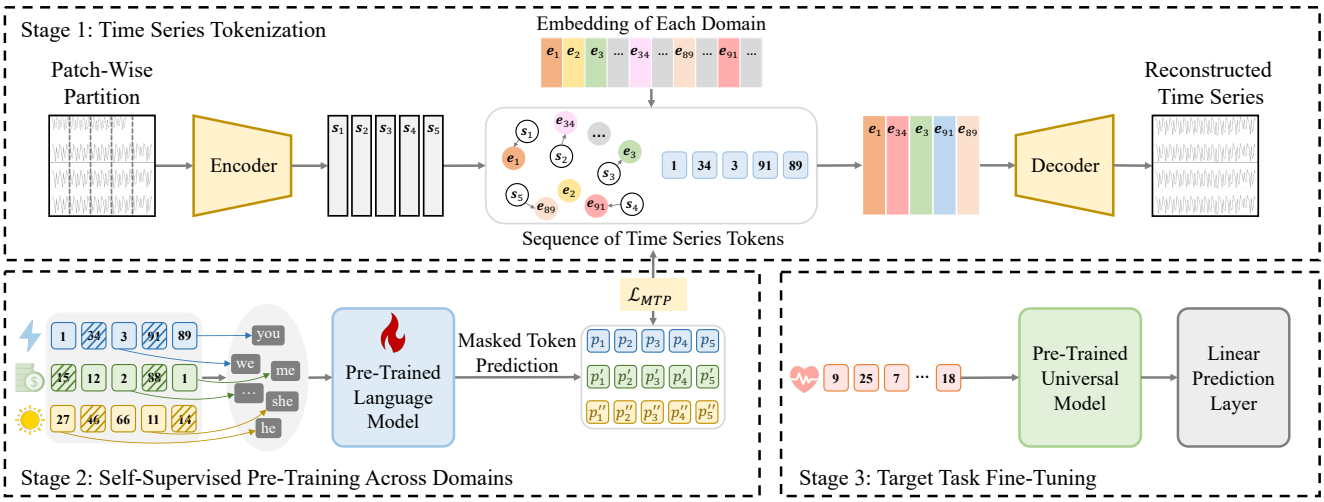


Figure 1: Overview of CrossTimeNet: a unified framework for self-supervised pre-training of cross-domain time series data.

time series instance $\mathbf{x^n} = [x_1^n, x_2^n, \ldots, x_T^n]$ in the n-th domain, it is first segmented into L non-overlap patches $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_L$, where each patch $\mathbf{s}_l$ contains a subset of consecutive time points from the original series. These patches are then fed into the auto-encoder architecture. In the following sections, we omit the mathematical notation for domains for brevity.

In this work, we propose to map each patch $\mathbf{s}_l$ to a latent representation $\mathbf{z}_n$. The TCN architecture [39] is chosen for its ability to capture long-range dependencies and for its efficient computation:

$$\mathbf{z}_l = \text{EncoderTCN}(\mathbf{s}_l). \tag{1}$$

Following the encoding, the vector quantization step occurs. Here, each latent representation $\mathbf{z}_l$ is mapped to the nearest vector in a learned codebook $\mathbf{e} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_K]$, where K is the size of the codebook. This mapping is given by:

$$\mathbf{z}_l^q = \text{argmin } \mathbf{e}_k \in \mathbf{e} |\mathbf{z}_l - \mathbf{e}_k|^2. \tag{2}$$

where $\mathbf{z}_l^q$ denote the assigned code selected from codebook. The decoder then reconstructs the time series from the quantized vectors, aiming to minimize the reconstruction error and thus preserve as much of the original sequence information as possible:

$$\hat{\mathbf{x}} = \text{Decoder}_{\text{TCN}}(\mathbf{z}^q). \tag{3}$$

By using this auto-encoder framework with TCNs, we can effectively compress the time series into a discrete representation that is amenable to self-supervised learning tasks while preserving the critical temporal dynamics that are essential for downstream applications. Our optimization process aligns with the approach used in the prior work of VQ-VAE [39], where it is particularly noteworthy that the nearest-neighbor selection process is non-differentiable, leading to challenges in gradient derivation. Due to space constraints in this paper, we omit a detailed discussion, and interested readers may refer to the relevant literature for further information. The novel contribution of our work is that a novel time series discretization is implemented, which is greatly different from previous works.

4.3 Self-supervised Pre-training

4.3.1 Pre-trained Language Model as Encoders. A comprehensive review of the extant literature reveals that the backbone networks in the domain of time series self-supervised learning predominantly harness either convolutional neural networks (e.g., TCN in [39] or Transformer networks in [7]). In view of the sparsity of data in the time series field, the selected backbone architecture must be easy to train in order to efficiently utilize limited training data. Drawing inspirations from cognitive science literature [40], which posits that human learning is not initiated from a tabula rasa but rather resembles a form of pre-trained network, we ventured to adopt a similar paradigm for our backbone network. This approach aligns with the concept that infants' brains, though not fully developed, are equipped with a rudimentary but potent learning framework right from birth. In this vein, our experimental forays led us to an intriguing discovery: employing a pre-trained language model as the backbone network significantly amplifies performance, marking a substantial leap in achieving superior outcomes in time series analysis. This finding is particularly noteworthy as it challenges the conventional neglect of such a potentially efficacious setting in prior research.

Specifically, we draw upon the network architecture of BERT [34], renowned for its masked language modeling (MLM) and next sentence prediction (NSP) tasks, to serve as the foundational base model for our network initialization. This choice is predicated on BERT's proven versatility and robustness in capturing contextual dependencies, making it an ideal candidate for our cross-domain self-supervised learning framework. The adoption of a pre-trained language model as the backbone, therefore, represents a novel and promising avenue in the realm of time series analysis, setting the stage for further exploration and validation in future work.

Upon integrating a language model, we encountered the challenge of encoding time series tokens within the BERT network. Although the codebook embedding from tokenizer could be directly utilized, discrepancies in the embedding size and potential

gaps in the representational space compared to BERT's word embeddings were apparent. To address this, we employed a word mapping mechanism [41], randomly assigning each token a corresponding word selected from BERT's vocabulary. This approach effectively resolved the encoding issues of time series tokens, ensuring seamless integration with the language model framework. It should be noted that we provide a detailed comparison of the differences in word mapping to illustrate that the specific impact as shown in Table 13 and Table 14 presented in Appendix C.3.

4.3.2 Masked Token Prediction. In this subsection, we introduce the cross-domain self-supervised optimization objectives. In the realm of time series analysis, in our view, an ideal self-supervised optimization task should fulfill two pivotal objectives. Firstly, it must be capable of learning rich contextual information [42], ensuring that the locality of the sequence can be represented (*Objective 1*). This requirement stems from the inherent sequential nature of time series data, where the understanding of a given point is significantly enhanced by its preceding and succeeding elements. Secondly, maintaining a certain level of abstraction of the predicted targets in the self-supervised optimization loss can be beneficial to improving the transferability of pre-trained models (*Objective 2*). This idea is also consistent in [43], which thinks that directly formulating self-supervised signals in raw space would largely restrict the capacity of the model due to the noisy and unbound properties. Thirdly, the optimization challenge posed by the self-supervised pre-training should be sufficiently challenge [44] (*Objective 3*). This difficulty is crucial as it compels the model to learn a more profound and comprehensive set of knowledge, which can be leveraged to augment the performance on downstream tasks.

Given these considerations, our approach in this paper is to design a self-supervised optimization task characterized by a relatively high ratio (more than 30%) of *masked token prediction*. This design choice is predicated on the hypothesis that by obscuring a substantial portion of the input data, the model is compelled to infer the masked information based solely on the context provided by the visible data points. Precisely, let $\mathcal{M}$ denotes the set of masked positions within the time series data. Suppose the predicted corresponding to the masked inputs are p_l, with $l \in \mathcal{M}$. Formally, the reconstruction self-supervised optimization goal $\mathcal{L}_{MTP}$ can be described as follows:

$$\mathcal{L}_{MTP} = -\sum_{l \in \mathcal{M}} \log p(p_l | \mathbf{x}_{\setminus \mathcal{M}}; \Theta), \tag{4}$$

where $\mathbf{x}_{\setminus \mathcal{M}}$ signifies the sequence with the masked tokens removed, and Θ encapsulates the model parameters. This task not only encourages the model to learn robust representations by leveraging bidirectional context but also ensures that the optimization challenge is sufficiently demanding to facilitate the acquisition of valuable knowledge for downstream applications.

4.4 Downstream Task Adaptation

Upon the completion of cross-domain self-supervised pre-training, we have successfully developed a versatile foundation model. To assess the efficacy of the pre-trained model, it is crucial to employ rigorous evaluation techniques. While acknowledging the plethora of transfer learning strategies available, such as Adapter-based methods among others [45, 46], it is pertinent to note that these are beyond the scope of our current investigation and are earmarked for future research endeavors. In this work, we adopt two classical evaluation paradigms within the self-supervised learning framework: *linear evaluation* and *full fine-tuning*. In linear evaluation, we freeze the weights of the pre-trained model, preserving the learned representations. A linear layer for prediction or classification is then added on top of the model. This layer is the only component that is trained on the downstream task dataset. This approach allows us to evaluate the quality and transferability of the features learned during the self-supervised pre-training phase without modifying the underlying representations. Moreover, linear evaluation is particularly useful for assessing the generalizability of the pre-trained model across different domains with minimal computational cost. Contrary to linear evaluation, full fine-tuning involves adjusting the entire model, including both the pre-trained layers and the newly added task-specific layers. This approach allows the model to fine-tune the learned representations in conjunction with learning the downstream task, potentially leading to higher performance on the target task. Full fine-tuning is more computationally intensive but can result in a model that is more closely tailored to the specifics of the target task, leveraging both the generic representations learned during pre-training and the specific nuances of the new task.

5 EXPERIMENTS

5.1 Experimental Setup

To evaluate the effectiveness of our CrossTimeNet, we employ several prevalent real-world datasets representing distinct domains of time series analysis. For classification tasks, we utilize HAR, ECG, and EEG datasets, while for forecasting tasks, we use ETT-small, Weather, and Exchange datasets. We conduct comparative analyses against a spectrum of prevailing self-supervised baselines, including contrastive learning approaches (TNC [14], TS-TCC [15], TS2Vec [8]), denoising auto-encoder based methods (SimMTM [10], TST [9]). In addition, TST-Zero refers to a Transformer model initialized randomly, while TST-Plus denotes a randomly initialized model that incorporates a pre-trained language model (PLM) architecture. For classification, we apply Accuracy and F1 Score as measurement metrics, while for forecasting, we use Mean Absolute Error (MAE) and Mean Squared Error (MSE). Detailed dataset settings, baseline descriptions, and experimental configurations can be found in Appendix A and Appendix B.

5.2 Results and Analysis

5.2.1 Downstream Results. Based on the experimental results presented in Table 1 and Table 2, we can observe CrossTimeNet's performance across various time series tasks, including forecasting and classification. The findings from these tables demonstrate the model's capabilities in learning transferable time series representations and its effectiveness across diverse domains. Due to space limitations, the complete results including linear evaluation are presented in the Appendix C.6.

Time Series Classification. Table 1 shows that CrossTimeNet achieved the highest scores on all data sets. A key point worth paying attention to is the significant performance differences between different models, especially on ECG datasets. While CrossTimeNet

Table 1: Experimental results of time series classification task evaluated by Accuracy and F1 Score.

Models	TNC		TS-TCC		TS2Vec		SimMTM		TST		TST-Zero		TST-Plus		CrossTimeNet	
Metric	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
HAR	0.8961	0.8951	0.8832	0.8815	0.8968	0.8957	0.9200	0.9220	0.9203	0.9203	0.9121	0.9120	0.8550	0.8520	**0.9335**	**0.9347**
EEG	0.7603	0.4457	0.7291	0.4347	0.7565	0.4449	0.8165	0.6123	0.8086	0.5516	0.7938	0.5211	0.7929	0.5426	**0.8541**	**0.6402**
ECG	0.2081	0.3310	0.1178	0.3780	0.1302	0.2064	0.2565	0.3562	0.2206	0.3317	0.1810	0.3861	0.2134	0.3246	**0.4378**	**0.6278**

Table 2: Experimental results of time series forecasting task evaluated by MSE and MAE.

Models	TNC		TS-TCC		TS2Vec		SimMTM		TST		TST-Zero		TST-Plus		CrossTimeNet	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.6401	0.5561	0.5962	0.5375	0.6775	0.5690	0.5991	0.5352	0.5718	0.5244	0.6463	0.5571	0.5778	0.5351	**0.5009**	**0.4944**
ETTh2	0.4087	0.4378	0.4122	0.4403	0.4099	0.4395	0.4537	0.4555	0.4171	0.4446	0.4208	0.4496	0.4169	0.4451	**0.3835**	**0.4233**
ETTm1	0.6618	0.6056	0.6323	0.5947	0.5897	0.5765	0.4935	0.4691	0.6030	0.5774	0.6125	0.5869	0.5507	0.5003	**0.4028**	**0.4171**
ETTm2	0.3123	0.3550	0.3225	0.3668	0.3175	0.3611	0.3827	0.3929	0.3113	0.3561	0.3031	0.3514	0.3146	0.3630	**0.2929**	**0.3474**
Weather	0.3523	0.4065	0.3023	0.3895	0.3502	0.4172	0.3134	0.3223	0.3067	0.3888	0.3184	0.4035	0.3028	0.3346	**0.2794**	**0.3089**
Exchange	0.5970	0.5606	0.6079	0.5644	0.6540	0.5904	**0.5684**	**0.5345**	0.6000	0.5605	0.6249	0.5734	0.6540	0.5908	0.5927	0.5499

achieved an accuracy of 0.4378 and an F1 score of 0.6278, other models such as TST-Zero and TNC performed significantly worse. This shows that CrossTimeNet performs better in handling the complexity and variability of ECG signals, which is a challenge for other models. From the perspective of time series locality, CrossTimeNet can effectively capture the correlation between adjacent data points, which is crucial for identifying short-term patterns and trends in time series. The extraction of local features enables the model to pay attention to key local changes when processing complex signals, thereby improving classification performance. Especially in ECG signals, subtle changes in heartbeats often contain important clinical information. A key observation from the table is the significant performance disparity between the models, particularly in the ECG dataset. While CrossTimeNet excels with an Accuracy of 0.4378 and an F1 Score of 0.6278, other models like TST-Zero and TNC show much lower performance. This suggests that CrossTimeNet is better equipped to manage the complexity and variability inherent in ECG signals, which are challenging for other models. Additionally, the table highlights the varying effectiveness of different models across datasets. For instance, while TST and SimMTM perform relatively well on the HAR dataset, their performance drops significantly on the EEG and ECG datasets. This variability underscores the importance of model selection based on the specific characteristics of the dataset and task at hand. CrossTimeNet's robust performance across all datasets suggests a strong generalization ability, making it a versatile choice for time-series classification tasks.

Time Series Forecasting. In our time series forecasting experiments, the lookback window length L is a crucial parameter that determines the number of past observations used to predict future values. We set L to 336 across all models and datasets, with horizon lengths H set to [96, 192, 336, 720]. This configuration allows us to evaluate the model's performance across different forecasting lengths and its ability to generalize across various time scales. The experimental results reported in Table 2, are averaged over all prediction horizons. The results show that CrossTimeNet generally achieves lower MSE and MAE in multiple datasets. However, on the Exchange dataset, CrossTimeNet performs worse than SimMTM. Specifically, SimMTM achieves an MSE of 0.5684 and an MAE of 0.5345 on the Exchange dataset, both better than CrossTimeNet's MSE of 0.5927 and MAE of 0.5499. This suggests that when dealing with complex financial time series such as foreign exchange data, SimMTM may be more effective in capturing patterns and changes in the data and providing more accurate predictions. This result also suggests the differences in the adaptability of different models when dealing with different types of data, emphasizing the importance of choosing the right model for the specific task and dataset. Overall, these results highlight the superior performance of CrossTimeNet in capturing complex time dependencies and providing accurate predictions, demonstrating its strong adaptability and generalization capabilities in different scenarios.

5.2.2 Study of Using PLM as Encoders. In this study, we evaluate the effectiveness of pre-trained models by comparing the performance of a Pre-trained BERT (PTB), a Randomly Initialized BERT (RIB), and a Randomly Initialized Transformer (RIT) across time series forecasting and classification. Figure 2 displays a performance comparison of these models, highlighting their accuracy in classification and MSE in forecasting tasks.

The exceptional performance of PTB can largely be attributed to its pre-training on a vast corpus of text data, which enables the model to learn complex patterns and dependencies. The rich contextual representations developed during pre-training allow PTB to more effectively parse and predict time series data. Furthermore, leveraging pre-trained knowledge, PTB can adapt more efficiently to specific tasks with limited data, thereby enhancing generalization capabilities and predictive accuracy.

In conclusion, Pre-trained BERT models show significant advantages in handling complex time series tasks. Future research should consider prioritizing pre-trained models to leverage these benefits, as they outperform models without such advanced training, like RIB and RIT, across both classification and forecasting metrics.

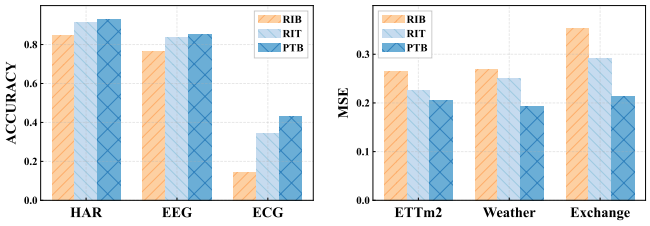
5.2.3 Influence of PLM Model Sizes and Structures. Figure 3 illustrates the performance of different PLM configurations, comparing BERT-Small to BERT-Large for classification tasks and BERT-Tiny to BERT-Large for forecasting tasks, including RoBERTa across three datasets in CrossTimeNet. Figure 3a shows that for classification tasks, the accuracy is consistently high across all BERT sizes, from BERT-Small to BERT-Large, indicating the minimal impact

of model size on performance. This suggests that these tasks have prominent features easily captured by the models.

Figure 3b presents the MSE for different BERT sizes across several forecasting tasks, which protection length is 96. Here, BERT-Tiny was included to examine the effect of an extremely small model. Surprisingly, larger models do not significantly reduce MSE and sometimes even increase it, likely due to overfitting, especially with insufficient or noisy data. The inclusion of BERT-Tiny shows that even very small models can perform competitively, indicating that simpler models can be effective and more robust to overfitting in certain cases.

From these findings, larger BERT models generally deliver superior performance in terms of accuracy and F1 score, with BERT-Large being the top performer. This highlights the advantage of extensive model architectures in capturing complex patterns. Inspired by this, scaling the model with mixture-of-expert techniques like Switch Transformer [47] could be a future direction. RoBERTa, despite its optimizations, does not consistently outperform BERT-Base, possibly due to the specific nature of time-series tasks. Smaller models like BERT-Small and BERT-Tiny still achieve commendable performance, balancing computational efficiency and accuracy. In summary, while model size does not significantly impact classification task performance, larger models may suffer from overfitting in forecasting tasks, leading to higher errors. Therefore, balancing model complexity with task and data characteristics is essential to avoid potential pitfalls of overly complex models.

5.2.4 Impact of Pre-training Across Domains. The results in Table 3 and Table 4 present the impact of cross-domain integration on both classification and forecasting tasks. Table 3 shows the classification performance on the HAR, EEG, and ECG datasets, measured by Accuracy and F1 Score. The integration of cross-domain information leads to slight improvements in performance. The gains are modest across all datasets, indicating that cross-domain integration provides some benefit but is not transformative for classification

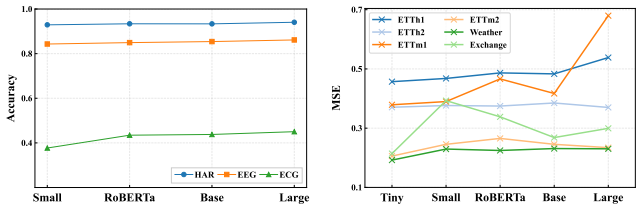


(a) PLM for classification (b) PLM for forecasting

Figure 2: Performance comparison between different PLM

Table 3: Cross-domain pre-training on classification tasks.

Compared Models	HAR		EEG		ECG	
	Acc	F1	Acc	F1	Acc	F1
w/o Cross Domain	0.9305	0.9305	0.8541	0.6327	0.4287	0.6161
w/ Cross Domain	**0.9335**	**0.9347**	**0.8541**	**0.6402**	**0.4378**	**0.6278**



(a) BERT configurations for classification (b) BERT configurations for forecasting

Figure 3: Performance of different BERT configurations

Table 4: Cross-domain pre-training on forecasting tasks.

Compared Models		ETTh2		ETTm1		Exchange	
		MSE	MAE	MSE	MAE	MSE	MAE
w/o Cross Doamin	96	0.3706	0.4101	0.3790	0.4105	0.2142	0.3387
	192	0.3982	0.4302	0.4046	0.4269	0.3329	0.4264
	336	0.4065	0.4386	0.4523	0.4495	0.5140	0.5336
	720	0.4400	0.4612	0.5139	0.4799	1.2366	0.8358
w/ Cross Domain	96	**0.3589**	**0.4046**	**0.3430**	**0.3818**	0.2444	0.3674
	192	**0.3755**	**0.4167**	**0.3714**	**0.4046**	**0.3276**	**0.4171**
	336	**0.3731**	**0.4183**	**0.4200**	**0.4180**	0.6046	0.5830
	720	**0.4265**	**0.4537**	**0.4766**	**0.4638**	**1.1940**	**0.8321**

tasks. This could be because classification tasks often rely heavily on specific features inherent to the dataset, which may not be enhanced by cross-domain data. Table 4 illustrates the forecasting performance across different horizons (96, 192, 336, 720) for the ETTh2, ETTm1, and Exchange datasets, evaluated using MSE and MAE. Here, cross-domain integration results in more noticeable improvements, particularly for larger datasets like ETTh2 and ETTm1. For example, in the ETTh2 dataset, the MSE for a 96-point horizon decreases from 0.3706 to 0.3589 with cross-domain integration. However, for the Exchange dataset, the benefits are less pronounced or even detrimental, as seen in the 96-point horizon where MSE increases from 0.2142 to 0.2444. This indicates that cross-domain integration helps capture complex temporal dependencies that are more prevalent in extensive datasets. The mixed results on smaller datasets like Exchange suggest that the additional complexity introduced by cross-domain integration might not always be beneficial, potentially leading to overfitting or increased noise, which can degrade performance. This highlights the importance of tailoring model strategies to the dataset size and characteristics to maximize the benefits of cross-domain integration.

5.2.5 Effectiveness of Masked-style PLM. In this part, we aim to evaluate the effectiveness of initializing encoder networks with BERT [34] and GPT2 [48] parameters for different self-supervised strategies across three datasets for classification. Table 5 presents the performance of various model variants, including two-layer Transformers with Autoregressive (AR) and Masked Token Prediction (MTP) self-supervised pre-training strategies, as well as models utilizing GPT2 and BERT as encoders. The reported results reveal a clear trend: models initialized with BERT parameters outperform those with GPT2 across all metrics and tasks, particularly when

the entire model is fine-tuned. This suggests that BERT's bidirectional training framework may be more conducive to capturing the nuances of these diverse datasets. This superiority of BERT-based initialization could be attributed to the inherent design that allows it to better understand and integrate the context from both past and future data points in a time series, which is crucial for tasks like HAR, EEG, and ECG analysis where the significance of a data point often depends on its surrounding values. In contrast, GPT-2's forward-only context capture might limit its ability to fully utilize the available temporal information. In summary, the results highlight the importance of choosing an appropriate pre-trained model for initialization based on the nature of the task and the data. For time-series analysis, where contextual understanding from both directions can be crucial, BERT's bidirectional training framework offers a clear advantage over GPT-2's unidirectional approach.

5.2.6 Performance Comparison Across Varying Masking Ratios. In contrast to the common practice in BERT of using a 15% masking rate [49], our CrossTimeNet highlights a higher masking rate. Table 6 showcases the impact of varying masking ratios on the final performance. From the shown results, we find that a general trend is that the model's performance is sensitive to the masking ratio, with an optimal range appearing to be around 0.45, where the model achieves its peak performance across all datasets in terms of both accuracy and F1 score. A notable anomaly occurs at a masking ratio of 0.60, where a significant drop in performance is observed across all datasets and evaluation methods, indicating that excessive masking may hinder the model's ability to learn effective representations of the data. Additionally, these findings underscore the nuanced relationship between the masking ratio and model generalization, suggesting that higher masking rates are not always beneficial. These results suggest that while a certain level of input data masking encourages the model to learn more robust and generalizable features, there is a threshold beyond which further masking becomes detrimental, possibly due to the model receiving insufficient information for effective learning.

This highlights the importance of selecting an appropriate masking ratio tailored to the characteristics of the data and the specific learning task to optimize model performance.

Table 5: Performance of model variants using BERT and GPT2 as initialization parameters for the encoder network (Models A and B are two-layer transformers, with A using auto-regression (AR) and B using masking. Models C and D are pre-trained language models (PLMs), with C based on GPT and D on BERT).

Model	HAR		EEG		ECG	
	Acc	F1	Acc	F1	Acc	F1
A	0.8568	0.8534	0.8001	0.5472	0.2604	0.3978
B	0.7618	0.7527	0.7959	0.5234	0.165	0.2303
C	0.9258	0.9261	0.8353	0.5947	0.4101	0.5949
D	**0.9335**	**0.9347**	**0.8541**	**0.6402**	**0.4378**	**0.6278**

Table 6: Classification performance of CrossTimeNet across masking ratios.

Masking Ratio	HAR		EEG		ECG	
	ACC	F1	ACC	F1	ACC	F1
0.15	0.8914	0.8923	0.8346	0.6029	0.2081	0.3071
0.30	0.9187	0.9191	0.8332	0.5933	0.2222	0.3313
0.45	**0.8928**	**0.8939**	**0.8400**	**0.5997**	**0.2430**	**0.3610**
0.60	0.7944	0.7890	0.8395	0.6118	0.1940	0.2936

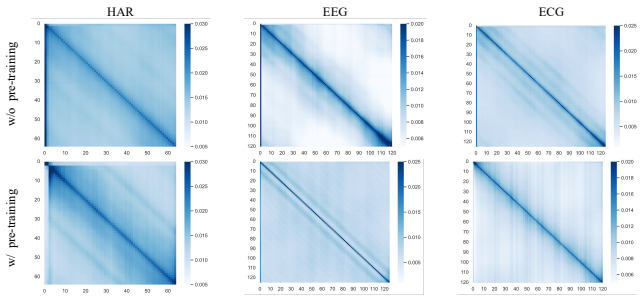
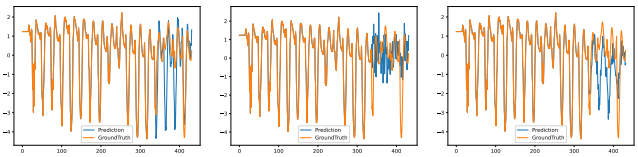


Figure 4: Visualization of attention weight regarding three different datasets.



(a) CrossTimeNet (b) TST (c) Fine-Zero

Figure 5: Visualization of forecasting case, generated by various models under the input-336-predict-96 setting, is presented.

5.2.7 Case study analysis. In the experimental analysis, we investigated the influence of pre-training on attention mechanisms across three distinct datasets: HAR, EEG, and ECG. As shown in Figure 4, we visualized attention weights using heatmaps to discern patterns indicative of the model's focus. For each dataset, we analyzed two conditions: with and without pre-training. The HAR data revealed a pronounced diagonal pattern without pre-training, suggesting a strong self-attention to temporal features, which became more dispersed upon pre-training, indicating a more complex relational understanding. The EEG data presented a less distinct diagonal pattern, with pre-training enhancing temporal structure comprehension. Conversely, the ECG data displayed a consistent focus on temporal autocorrelation, with minimal variations observed between pre-trained and non-pre-trained models. These visualizations underscore the varying impact of pre-training on attention-based models, reflecting the inherent complexities of each dataset.

Figure 5 visualizes forecasting cases generated by various models using 336 input steps and 96 prediction steps. The comparison clearly shows that our approach offers improved accuracy and robustness in capturing complex patterns, particularly in handling irregular fluctuations and intricate temporal dependencies within

time series data. This highlights the model's effectiveness in dealing with challenging time series scenarios, validating its superiority.

6 CONCLUSION

In this study, we proposed CrossTimeNet, a novel self-supervised pre-training method designed for time series representation pre-training. Our method's key feature is the time series data discretization, enabling cross-domain self-supervised pre-training. This approach empowers CrossTimeNet to harness temporal dynamics across diverse domains, leading to a versatile and transferable base model. Extensive experiments confirmed CrossTimeNet's effectiveness in learning meaningful and transferable representations, providing substantial benefits for downstream tasks. We hope the CrossTimeNet will inspire more work to be proposed.

7 LIMITATION ANALYSIS

Although the effectiveness of the CrossTimeNet, we also realize that there exist some inherent limitations within our work. This study proposed an innovative self-supervised pre-training method across domains but dose not explore the transferability across different tasks [50]. Despite showing promising results using PLM as the encoder, the research lacks a comprehensive theoretical explanation for its effectiveness. Additionally, our work has not investigated the potential of generative text models or even large language models for a more universal modeling approach, which could align with the emerging trend of a single model addressing multiple language modeling tasks [51, 52].

REFERENCES

[1] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems* 34 (2021), 22419–22430.

[2] Ahmed Shifaz, Charlotte Pelletier, François Petitjean, and Geoffrey I Webb. 2020. TS-CHIEF: a scalable and accurate forest algorithm for time series classification. *Data Mining and Knowledge Discovery* 34, 3 (2020), 742–775.

[3] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. 2014. Time series classification using multi-channels deep convolutional neural networks. In *International conference on web-age information management*. Springer, 298–310.

[4] Minghao Liu, Shengqi Ren, Siyuan Ma, Jiahui Jiao, Yizhou Chen, Zhiguang Wang, and Wei Song. 2021. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438* (2021).

[5] Zhiding Liu, Mingyue Cheng, Zhi Li, Zhenya Huang, Qi Liu, Yanhu Xie, and Enhong Chen. 2024. Adaptive normalization for non-stationary time series forecasting: A temporal slice perspective. *Advances in Neural Information Processing Systems* 36 (2024).

[6] Sindhu Tipirneni and Chandan K Reddy. 2022. Self-supervised transformer for sparse and irregularly sampled multivariate clinical time-series. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16, 6 (2022), 1–17.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

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

[9] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2114–2124.

[10] Jiaxiang Dong, Haixu Wu, Haoran Zhang, Li Zhang, Jianmin Wang, and Mingsheng Long. 2024. Simmtm: A simple pre-training framework for masked time-series modeling. *Advances in Neural Information Processing Systems* 36 (2024).

[11] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. 2022. Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. *arXiv preprint arXiv:2202.01575* (2022).

[12] Kexin Zhang, Qingsong Wen, Chaoli Zhang, Rongyao Cai, Ming Jin, Yong Liu, James Zhang, Yuxuan Liang, Guansong Pang, Dongjin Song, et al. 2023. Self-Supervised Learning for Time Series Analysis: Taxonomy, Progress, and Prospects. *arXiv preprint arXiv:2306.10125* (2023).

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

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

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

[16] Azul Garza and Max Mergenthaler-Canseco. 2023. TimeGPT-1. *arXiv preprint arXiv:2310.03589* (2023).

[17] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. 2024. Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592* (2024).

[18] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. 2024. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885* (2024).

[19] Mingyue Cheng, Qi Liu, Zhiding Liu, Hao Zhang, Rujiao Zhang, and Enhong Chen. 2023. Timemae: Self-supervised representations of time series with decoupled masked autoencoders. *arXiv preprint arXiv:2303.00320* (2023).

[20] Zhiding Liu, Jiqian Yang, Mingyue Cheng, Yucong Luo, and Zhi Li. 2024. Generative Pretrained Hierarchical Transformer for Time Series Forecasting. *arXiv preprint arXiv:2402.16516* (2024).

[21] Matthew Middlehurst, Patrick Schäfer, and Anthony Bagnall. 2023. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *arXiv preprint arXiv:2304.13029* (2023).

[22] Lexiang Ye and Eamonn Keogh. 2009. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 947–956.

[23] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1542–1552.

[24] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. 2013. A time series forest for classification and feature extraction. *Information Sciences* 239 (2013), 142–153.

[25] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15 (2007), 107–144.

[26] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*. IEEE, 1578–1585.

[27] Mingyue Cheng, Jiqian Yang, Tingyue Pan, Qi Liu, and Zhi Li. 2024. ConvTimeNet: A Deep Hierarchical Fully Convolutional Model for Multivariate Time Series Analysis. *arXiv preprint arXiv:2403.01493* (2024).

[28] Mingyue Cheng, Qi Liu, Zhiding Liu, Zhi Li, Yucong Luo, and Enhong Chen. 2023. FormerTime: Hierarchical Multi-Scale Representations for Multivariate Time Series Classification. *arXiv preprint arXiv:2302.09818* (2023).

[29] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2022. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125* (2022).

[30] George EP Box and Gwilym M Jenkins. 1968. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 17, 2 (1968), 91–109.

[31] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[32] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[33] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.

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

[35] Lu Han, Han-Jia Ye, and De-Chuan Zhan. 2023. The Capacity and Robustness Trade-off: Revisiting the Channel Independent Strategy for Multivariate Time Series Forecasting. *arXiv preprint arXiv:2304.05206* (2023).

[36] Patrick Schäfer and Mikael Högqvist. 2012. SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th international conference on extending database technology*. 516–527.

[37] Patrick Schäfer. 2015. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery* 29 (2015), 1505–1530.

[38] Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems* 30 (2017).

[39] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[40] Ghislaine Dehaene-Lambertz. 2017. The human infant brain: A neural architecture able to learn language. *Psychonomic bulletin & review* 24 (2017), 48–55.

[41] Wei-Tsung Kao and Hung-yi Lee. 2021. Is BERT a Cross-Disciplinary Knowledge Learner? A Surprising Finding of Pre-trained Models' Transferability. *arXiv preprint arXiv:2103.07162* (2021).

[42] Xiangwen Kong and Xiangyu Zhang. 2023. Understanding masked image modeling via learning occlusion invariant feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6241–6251.

[43] Eric J Kostelich and Thomas Schreiber. 1993. Noise reduction in chaotic time-series data: A survey of common methods. *Physical Review E* 48, 3 (1993), 1752.

[44] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 16000–16009.

[45] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[46] Junchen Fu, Fajie Yuan, Yu Song, Zheng Yuan, Mingyue Cheng, Shenghui Cheng, Jiaqi Zhang, Jie Wang, and Yunzhu Pan. 2023. Exploring Adapter-based Transfer Learning for Recommender Systems: Empirical Studies and Practical Insights. *arXiv preprint arXiv:2305.15036* (2023).

[47] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research* 23, 1 (2022), 5232–5270.

[48] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[49] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.

[50] Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. 2023. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. *arXiv preprint arXiv:2310.04948* (2023).

[51] Jun Li, Che Liu, Sibo Cheng, Rossella Arcucci, and Shenda Hong. 2023. Frozen Language Model Helps ECG Zero-Shot Learning. *arXiv preprint arXiv:2303.12311*

(2023).
[52] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

APPENDIX

A DETAIL OF DATASETS

To evaluate the effectiveness of our proposed cross-domain self-supervised pre-training method, we selected three time series datasets for classification and six for forecasting, each drawn from distinct domains. This diverse selection represents a unique set of challenges and characteristics, allowing for a comprehensive assessment of our approach across varied scenarios. The chosen datasets emphasize the adaptability and robustness of our method.

A.1 Detail of Classification Datasets

For the classification task, we selected HAR, EEG, and ECG datasets. The following is a detailed introduction to the datasets.

- **HAR**: The Human Activity Recognition (HAR) dataset is a multi-class classification dataset that consists of sensor data collected from subjects performing various activities. The activities include walking, sitting, standing, and more complex activities like ascending or descending stairs. The data is captured using wearable sensors, providing a rich source of temporal patterns for recognizing human activities.
- **EEG**: The EEG datasets (a.k.a Sleep-EDF) comprise polysomnographic (PSG) recordings, which are used for multi-class sleep stage classification. The dataset includes Electroencephalogram (EEG) recordings among other physiological signals, collected from subjects under normal and pathological conditions. This dataset is pivotal for developing models that can automatically identify sleep stages, aiding in the diagnosis and study of sleep disorders.
- **ECG**: The China Physiological Signal Challenge (CPSC) dataset is a multi-label classification dataset containing Electrocardiogram (ECG) recordings. The dataset is designed for the detection of arrhythmias and other cardiac abnormalities. It provides a diverse set of ECG recordings, making it suitable for developing and evaluating models aimed at cardiac monitoring and diagnosis.

Table 7: Time series datasets for classification.

Dataset	#Train	#Test	Length	#Channel	#Class
HAR	8,823	2,947	128	9	6
EEG	12,787	1,421	3,000	2	8
ECG	10,854	1,206	5,000	12	27

Table 7 summarizes the key statistics and characteristics of these datasets, including the number of samples, channels, and classes. It is evident that there are significant differences among the datasets in terms of their dimensions and the diversity of classes they encompass. Note that both HAR and EEG data refer to multi-class classification task while ECG involve multi-label classification. It is also worth noting that for the construction of the pre-training dataset, these three datasets were mixed and shuffled together, preserving the train-test splits consistent with previous works in each domain. This approach ensures that our model is exposed to a diverse range of patterns and challenges, simulating real-world scenarios where domain shifts are common.

A.2 Detail of Forecasting Datasets

Due to resource constraints, datasets with too many channels, such as the traffic dataset, were not selected. Finally, the ETT, weather, and exchange rates datasets were selected. The following is a detailed introduction to these datasets.

- **ETT**: encompasses records of oil temperature and load metrics of electricity transformers. The data spans from July 2016 to July 2018, divided into four sub-datasets.
- **Weather**: consists of 21 weather indicators, including air temperature and humidity. The data was collected at 10-minute intervals throughout the year 2021.
- **Exchange**: contains daily exchange rates of eight different nations. The data spans from 1990 to 2016.

Table 8: Time series datasets for forecasting

Dataset	Variables	Frequency	Length	Scope
ETTh1/ETTh2	7	1 Hour	17420	Energy
ETTm1/ETTm2	7	15 Minutes	69680	Energy
Exchange	8	1 Day	7588	Finance
Weather	21	10 Minutes	52696	Weather

Table 8 presents comprehensive descriptions of the datasets utilized in this study, encompassing a variety of data scenarios and scales. Among these, the ETT dataset primarily focuses on electricity consumption, with four subsets based on varying frequencies. Exchange captures the daily exchange rates among eight countries, while Weather consists of 21 climate indicators, such as air temperature. In accordance with standard protocols, each dataset was split into training, validation, and testing sets based on chronological order. The split ratio for the ETT dataset is 6:2:2, while the Exchange and Weather datasets follow a 7:1:2 ratio.

B ADDITIONAL IMPLEMENTATION DETAILS

B.1 Compared Baselines and Implement Details

To evaluate the efficacy of our cross-domain self-supervised pre-training approach, we compare it against several recent and popular self-supervised pre-training methods for time series data. These baseline methods encompass both reconstruction-based and contrastive learning approaches, ensuring a comprehensive comparison across different self-supervised learning paradigms.

- **TNC** : A contrastive learning approach that treats close temporal segments as positive pairs and distant segments as negative pairs, encouraging the model to learn discriminative features by distinguishing between them.
- **TS-TCC** : This method extends the contrastive learning framework to time series data by leveraging temporal coherence as a signal for similarity, aiming to learn representations that are invariant to specific transformations while maintaining temporal structure.
- **TS2Vec**: A hierarchical contrastive learning approach that captures multi-scale temporal patterns by contrasting representations at different time scales, facilitating a comprehensive understanding of the time series data.

- **SimMTM** : Similar to TST, SimMTM employs a masked autoencoder framework for time series analysis, where portions of the input data are masked, and the model learns to reconstruct the original data. This process enables the model to effectively capture intrinsic temporal dynamics.
- **TST**: A reconstruction-based self-supervised method uses a Transformer architecture to model time series data by predicting missing segments or forecasting future values. To assess the impact of pre-training, we also consider a TST variant without self-supervised pre-training as a baseline.
- **TST-Zero**: A method that refers to training vanilla transformer encoder networks from scratch without the use of self-supervised pre-training.
- **TST-Plus**: A method that integrates the architecture of a pre-trained language model (PLM) but is initialized randomly instead of using pre-trained weights.

To ensure a fair comparison, all baseline methods employ an encoder network based on the Transformer architecture, maintaining consistency in the model's capacity and structural complexity. During the fine-tuning phase for downstream tasks, the task-specific layer remains identical across all models, ensuring that any observed performance differences can be attributed to the effectiveness of the pre-training strategy rather than variations in the network architecture or task-specific adaptations. In terms of the hyper-parameter settings, the batch size is set to 32, the remaining ones either strictly follow the specific settings suggested by the original paper or tune the validation sets. We report the results of each baseline under its optimal hyper-parameter settings.

B.2 Configurations of Our CrossTimeNet

Next, we present the details of implementing the CrossTimeNet approach, with a special focus on the time series tokenization process using an autoencoder reconstruction architecture. For the tokenization component, the encoder network is carefully designed with a multi-layer TCN network to ensure efficient encoding of time series data into a compact representation. Specifically, the encoder network contains four layers, which helps to achieve a powerful feature extraction mechanism. The embedding size is set to 64. For the codebook number, we consistently set it to 512 in all classification datasets and 256 in the prediction dataset. As for the patch size, we set it to 2, 25, and 40 in the HAR, EEG, and ECG datasets, respectively, and uniformly set it to 7 on the prediction task. The network is initialized in a random manner. As for the optimization settings, the learning rate for the classification task is set to $5e^{-4}$, and the learning rate for the prediction task is set to $4e^{-4}$, accompanied by the Adam optimizer. In the pre-training stage, we carefully tuned several hyperparameters to optimize the learning process. These include the learning rate (set to $1e^{-4}$) and the batch size (chosen as 32). For the fine-tuning phase, we explored two different strategies: full fine-tuning and linear evaluation. All parameters of the pre-trained CrossTimeNet model are updated during the training of the downstream task while retaining the same hyperparameter settings as the pre-training phase.

Table 9: Performance comparison of our CrossTimeNet with different number of token in codebook for classification.

Datasets	Size	Accuracy	F1 Score	Coverage	MSE
HAR	128	0.9298	0.931	1	0.0091
	256	0.9335	0.9348	1	0.0088
	384	0.9352	0.9366	1	0.0069
	512	0.9335	0.9347	0.7422	0.0082
	768	0.9389	0.9401	0.6237	0.0077
EEG	128	0.8656	0.6549	0.9922	0.0102
	256	0.8629	0.6335	0.4648	0.0103
	384	0.8635	0.6596	0.4453	0.0096
	512	0.8541	0.6402	0.2793	0.0099
	768	0.8543	0.8384	0.1849	0.0098
ECG	128	0.4187	0.5995	1	0.0055
	256	0.4146	0.6054	1	0.0051
	384	0.4179	0.5953	1	0.0047
	512	0.4378	0.6278	0.5918	0.0047
	768	0.4328	0.6118	0.4714	0.0048

C EXTENDED ANALYSIS OF EXPERIMENTAL RESULTS

C.1 Results Across Varying Codebook Sizes

Next, we analyze the performance of CrossTimeNet with varying numbers of tokens in the codebook, as reflected in Table 10 and Table 9. The results illustrate how different codebook sizes influence results across different datasets. The findings indicate a general trend where increasing the number of tokens in the codebook initially enhances both MSE and MAE metrics, particularly evident in the ETTh2 dataset, which shows improved accuracy with a codebook size of 512. However, larger codebook sizes correlate with decreased coverage across all datasets, suggesting potential overfitting. For example, in the Weather dataset, coverage drops from 0.8200 at a codebook size of 128 to 0.7000 at 768, indicating a loss of generalization ability. In parallel, classification experiments reveal similar trends, where increasing codebook sizes lead to improved accuracy and F1 scores up to a certain threshold, particularly in the HAR dataset, peaking at a codebook size of 768. However, this improvement also comes with increased MSE and decreased coverage, reinforcing the notion of a trade-off between model expressiveness and generalization capability. In summary, both forecasting and classification results highlight the importance of finding an optimal codebook size that maximizes performance while maintaining efficient data representation, ensuring that CrossTimeNet effectively processes sequential data across various applications.

C.2 Results Across Varying Patch Sizes

The performance of CrossTimeNet with varying patch sizes is summarized in the table12, which includes metrics such as MSE, MAE, and Coverage across four patch sizes: 7, 14, 21, and 28. The analysis reveals that a patch size of 7 yields the best overall performance, achieving the lowest MSE (0.3706) and MAE (0.4101) while maintaining a high coverage of 0.8600. This suggests that smaller patch sizes are more effective in capturing the underlying patterns of

Table 10: Performance comparison of our CrossTimeNet with different number of tokens in the codebook for forecasting.

Datasets	Size	MSE	MAE	Coverage	MSE
ETTh2	128	0.4501	0.4702	0.8800	0.2036
	256	0.3706	0.4101	0.8600	0.2036
	384	0.3908	0.4205	0.8400	0.2100
	512	0.4100	0.4300	0.8000	0.2150
	768	0.4305	0.4400	0.7500	0.2200
Weather	128	0.2100	0.2800	0.8200	0.1900
	256	0.1922	0.2538	0.8000	0.1850
	384	0.2000	0.2600	0.7800	0.1800
	512	0.2150	0.2750	0.7500	0.1750
	768	0.2300	0.2900	0.7000	0.1700
Exchange	128	0.2200	0.3400	0.8400	0.2000
	256	0.2142	0.3387	0.8100	0.1980
	384	0.2300	0.3500	0.7900	0.1950
	512	0.2400	0.3600	0.7500	0.1920
	768	0.2500	0.3700	0.7000	0.1900

Table 11: Performance comparison of our CrossTimeNet with different patch sizes for classification.

Patch Size	Accuracy	F1 Score	Coverage	MSE
20	0.4063	0.6051	1.0000	**0.0033**
40	**0.4287**	**0.6161**	0.5918	0.0047
60	0.4163	0.5935	0.4609	0.0057
80	0.3856	0.5627	0.4629	0.0066
100	0.3822	0.5295	0.2891	0.0078

the data, leading to better model predictions. As the patch size increases to 14 and 21, there is a slight increase in MSE and MAE, with corresponding decreases in coverage, indicating a potential trade-off between model precision and generalization ability. In parallel, classification experiments in tabel11 demonstrate similar trends, where smaller patch sizes improve accuracy and F1 scores. However, larger patch sizes can lead to overfitting, as evidenced by increased errors and decreased coverage. This reinforces the notion that while larger patch sizes may seem beneficial for capturing broader patterns, they can hinder the model's ability to generalize effectively. Overall, both forecasting and classification results highlight the critical impact of patch size on the efficacy of the CrossTimeNet model. Selecting an optimal patch size is essential for maximizing performance, with a patch size of 7 showing the most favorable results in both analyses. This underscores the necessity to balance model granularity with computational efficiency for optimal performance across different tasks.

C.3 Studying the Token Selection Strategies

Table 13 and Table 14 presents the performance of CrossTimeNet under different random word mappings (denoted as A, B, and C) across three distinct datasets. Observing the results, it is evident that the variations in word mappings have a relatively minor influence on the overall performance metrics. This result underlines the

Table 12: Performance comparison of our CrossTimeNet with different patch sizes for forecasting.

Patch Size	MSE	MAE	Coverage	MSE
7	0.3706	0.4101	0.8600	0.2063
14	0.3727	0.4166	0.7780	0.2795
21	0.3755	0.4176	0.6595	0.2833
28	0.3873	0.4289	0.6619	0.2846

effectiveness of the word mapping mechanism in bridging the representational gap between time series tokens and the BERT model's vocabulary, thereby enabling the CrossTimeNet to leverage pre-trained language model representations for time series analysis tasks.

C.4 Data Sparsity Analysis

Figure 6 illustrates the accuracy of fine-tuning a pre-trained model across varying degrees of training set sparsity in three datasets. Overall, the accuracy of the fine-tuned pre-trained model exhibits a general decline as dataset sparsity increases. This indicates that denser datasets tend to yield better fine-tuning outcomes, affirming the importance of data richness for model performance optimization. Despite the overall trend, the pre-trained BERT model demonstrates a notable degree of robustness to sparsity, maintaining higher accuracy relative to the model without pre-training and the RandomInit BERT. This robustness is especially evident in the ECG domain, where accuracy remains relatively stable until a sparsity level of 0.5, suggesting that pre-trained models can leverage their learned representations to effectively handle sparse data. In summary, while dataset sparsity negatively impacts model accuracy, pre-training can serve as a mitigating factor, enhancing the model's ability to maintain performance in sparse data conditions. This underscores the potential of pre-trained models in applications with data limitations.

C.5 Analyzing Semantic of Time Series Token

The case study employs t-SNE visualization to articulate the clustering of discrete tokens represented in a high-dimensional feature space, juxtaposed with their manifestation in the corresponding raw time series data. The t-SNE plot distinctly segregates the tokens into coherent clusters, demarcated by a trio of colors, each color signifying a unique token category. This delineation by the t-SNE algorithm evidences its capability to reduce dimensionality while preserving the topology of the dataset, facilitating an intuitive understanding of complex structures within the feature space.

Parallelly, the time series graph delineates these tokens across the temporal axis, with the color-coded segments reflecting the same categorical distinctions identified in the t-SNE visualization. The synchronization between the spatial clusters in the t-SNE plot and the temporal segmentation in the time series data provides a compelling narrative of the data's underlying dynamics. Each color-coded point, representing a discrete token, aligns with a specific behavior or state in the time series, thereby mapping a multidimensional data narrative onto a comprehensible two-dimensional framework.

Table 13: Classification performance with varied random word mappings across three repetitions, denoted by A, B, C.

Evaluation Manners	Models	HAR		EEG		ECG	
		Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
Full Fine-tuning	A	0.9325	0.9339	0.8529	0.6401	0.4395	0.6293
	B	0.9345	0.9357	0.8543	0.6403	0.4336	0.6218
	C	0.9335	0.9345	0.8550	0.6402	0.4403	0.6324
Linear Evaluation	A	0.9155	0.9152	0.8388	0.5923	0.2180	0.3259
	B	0.9182	0.9189	0.8276	0.5827	0.2231	0.3323
	C	0.9223	0.9232	0.8332	0.6050	0.2255	0.3358

Table 14: Forecasting performance with varied random word mappings across three repetitions, denoted by A, B, C.

Evaluation Manners	Models	ETTh2		Weather		Exchange	
		Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
Full Fine-tuning	A	0.3706	0.4101	0.1922	0.2538	0.2142	0.3387
	B	0.3710	0.4163	0.1930	0.2503	0.2198	0.3358
	C	0.3732	0.4168	0.1998	0.2535	0.2190	0.3337
Linear Evaluation	A	0.3906	0.4222	0.2313	0.2855	0.4467	0.4806
	B	0.3947	0.4257	0.2318	0.2877	0.4472	0.4814
	C	0.4175	0.4359	0.2359	0.2806	0.4571	0.4805

This analytical approach, combining t-SNE with time series visualization, serves as an effective method for interpreting the nuanced interactions of tokens over time, offering a profound lens through which data scientists can observe temporal patterns, detect anomalies, or even predict future states in sequential data models.

C.6 Full Results

The comprehensive results of CrossTimeNet on time series forecasting and classification tasks are presented in Tables 16 and Table 15. In Table 16, the upper portion displays the results for fine-tuning all layers of the model, while the lower portion presents the results for linear evaluation. The 'N/A' placeholder indicates instances where the model's accuracy is effectively zero due to the threshold of measurement precision.

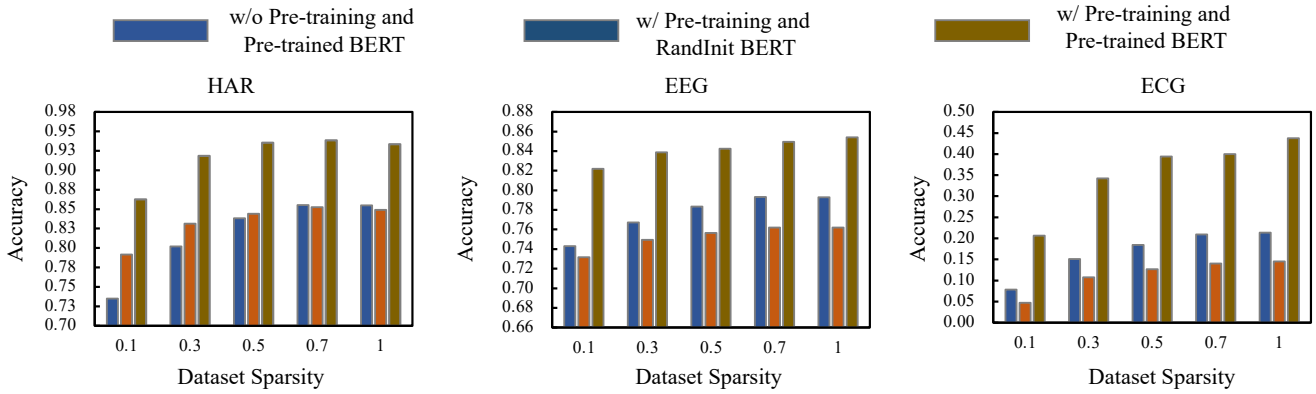


Figure 6: Investigating the impact of downstream training set sparsity on fine-tuning pre-trained model.

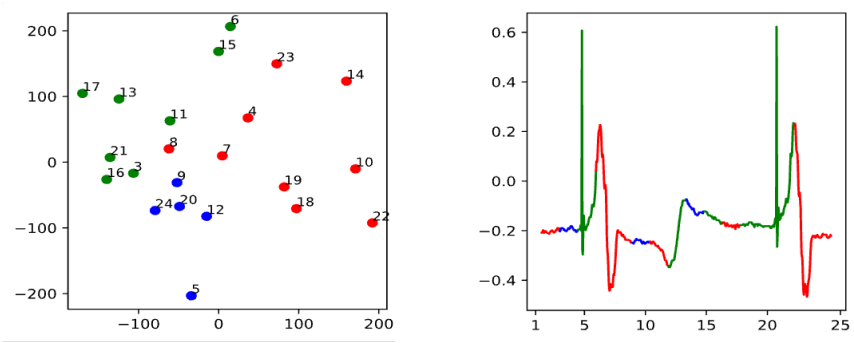


Figure 7: T-SNE visualization of sequence of patches in a instance and its corresponding raw time series.

Table 15: Full results of time series forecasting task evaluated by MSE and MAE.

			TNC		TS-TCC		TS2Vec		SimMTM		TST		TST-Zero		TST-Plus		CrossTimeNet	
		Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Full Fine-tuning	ETTh1	96	0.6271	0.5393	0.5817	0.5171	0.6638	0.5498	0.5463	0.4974	0.5587	0.5069	0.6362	0.5396	0.5059	0.4865	**0.4558**	**0.4650**
		192	0.6403	0.5489	0.5942	0.5272	0.6777	0.5601	0.5680	0.5112	0.5746	0.5179	0.6443	0.5478	0.5570	0.5191	**0.4946**	**0.4842**
		336	0.6405	0.5551	0.5961	0.5367	0.6724	0.5654	0.5620	0.5234	0.5736	0.5238	0.6416	0.5539	0.5914	0.5428	**0.5129**	**0.4960**
		720	0.6524	0.5812	0.6129	0.5690	0.6961	0.6008	0.7202	0.6089	0.5804	0.5488	0.6629	0.5869	0.6568	0.5921	**0.5404**	**0.5325**
	ETTh2	96	0.3844	0.4175	0.3914	0.4256	0.3892	0.4232	0.3947	0.4204	0.3934	0.4264	0.3874	0.4262	0.3917	0.4250	**0.3589**	**0.4046**
		192	0.4013	0.4318	0.4093	0.4367	0.4068	0.4365	0.4687	0.4544	0.4104	0.4400	0.4094	0.4432	0.4115	0.4421	**0.3755**	**0.4167**
		336	0.3978	0.4351	0.4004	0.4352	0.3996	0.4361	0.4638	0.4628	0.4072	0.4416	0.4121	0.4472	0.4042	0.4405	**0.3731**	**0.4183**
		720	0.4513	0.4668	0.4476	0.4638	0.4441	0.4623	0.4875	0.4843	0.4573	0.4705	0.4743	0.4818	0.4600	0.4728	**0.4265**	**0.4537**
	ETTm1	96	0.5710	0.5579	0.5174	0.5332	0.4522	0.5029	0.4607	0.4420	0.5040	0.5191	0.5196	0.5404	0.4918	0.4773	**0.3430**	**0.3818**
		192	0.6361	0.5932	0.5972	0.5853	0.5293	0.5438	0.4691	0.4511	0.5553	0.5498	0.5623	0.5614	0.5451	0.4940	**0.3714**	**0.4046**
		336	0.6866	0.6188	0.6890	0.6186	0.6131	0.5862	0.5024	0.4732	0.6207	0.5893	0.6280	0.5902	0.5660	0.5037	**0.4200**	**0.4180**
		720	0.7536	0.6526	0.7255	0.6416	0.7641	0.6732	0.5417	0.5102	0.7321	0.6513	0.7399	0.6556	0.6000	0.5263	**0.4766**	**0.4638**
	ETTm2	96	0.2230	0.3034	0.2566	0.3296	0.2523	0.3235	0.2301	0.3024	0.2358	0.3130	0.2250	0.3058	0.2490	0.3258	**0.2073**	**0.2967**
		192	0.2733	0.3335	0.2905	0.3493	0.2845	0.3427	0.3113	0.3566	0.2766	0.3369	0.2677	0.3313	0.2916	0.3534	**0.2586**	**0.3279**
		336	0.3310	0.3659	0.3304	0.3708	0.3232	0.3640	0.4208	0.4207	0.3229	0.3622	0.3154	0.3581	**0.3107**	**0.3591**	0.3166	0.3609
		720	0.4218	0.4171	0.4125	0.4174	0.4101	0.4142	0.5687	0.4918	0.4100	0.4122	0.4043	0.4103	0.4069	0.4135	**0.3890**	**0.4042**
	Weather	96	0.3743	0.3805	0.2473	0.3578	0.2771	0.3474	0.2054	0.2563	0.2438	0.3450	0.2500	0.3617	0.2506	0.3025	**0.2044**	**0.2581**
		192	0.3066	0.3926	0.2745	0.3739	0.3219	0.4061	0.2645	0.2991	0.2863	0.3767	0.3055	0.4011	0.2802	0.3203	**0.2563**	**0.2915**
		336	0.3385	0.4098	0.3093	0.3879	0.3635	0.4296	0.3702	0.3506	0.3255	0.4016	0.3479	0.4236	0.3145	0.3401	**0.3057**	**0.3300**
		720	0.3899	0.4432	0.3782	0.4383	0.4381	0.4856	0.4136	0.3831	0.3710	0.4317	0.3700	0.4275	0.3658	0.3754	**0.3513**	**0.3561**
	Exchange	96	0.2466	0.3708	0.2698	0.3859	0.3267	0.4238	**0.2232**	**0.3408**	0.2648	0.3814	0.2919	0.3997	0.3519	0.4365	0.2444	0.3674
		192	0.3735	0.4596	0.3897	0.4653	0.4429	0.4964	0.3438	0.4311	0.3835	0.4619	0.4132	0.4779	0.4575	0.5025	**0.3276**	**0.4171**
		336	0.5570	0.5735	0.5679	0.5703	0.6143	0.5958	**0.5148**	**0.5373**	0.5575	0.5657	0.5844	0.5779	0.6168	0.5953	0.6046	0.5830
		720	1.2109	0.8383	1.2040	0.8360	1.2320	0.8457	**1.1916**	**0.8286**	1.1942	0.8329	1.2100	0.8382	1.1896	0.8288	1.1940	0.8321
Linear Evaluation	ETTh1	96	0.6281	0.5395	0.5802	0.5168	0.6106	0.5294	0.7100	0.5713	0.5589	0.5076	0.6309	0.5385	0.7139	0.5761	**0.5230**	**0.5018**
		192	0.6410	0.5489	0.5937	0.5272	0.6243	0.5400	0.7200	0.5802	0.5765	0.5189	0.6436	0.5481	0.7225	0.5835	**0.5513**	**0.5150**
		336	0.6400	0.5547	0.5938	0.5359	0.6196	0.5463	0.7176	0.5841	0.5735	0.5339	0.6428	0.5549	0.7206	0.5900	**0.5852**	**0.5389**
		720	0.6508	0.5798	0.6108	0.5679	0.6326	0.5755	0.7357	0.6129	0.5823	0.5496	0.6621	0.5872	0.7446	0.6214	**0.6353**	**0.5836**
	ETTh2	96	0.3959	0.4257	0.3900	0.4300	0.3814	0.4196	0.3949	0.4236	0.3871	0.4214	0.3864	0.4255	0.3956	0.4261	**0.3813**	**0.4153**
		192	0.4137	0.4403	0.4086	0.4369	0.3993	0.4334	0.4562	0.4569	0.4035	0.4347	0.4083	0.4424	0.4110	0.4407	**0.4090**	**0.4403**
		336	0.4118	0.4444	0.4010	0.4358	0.3958	0.4343	0.4724	0.4688	0.4004	0.4368	0.4104	0.4463	0.4110	0.4436	**0.4156**	**0.4502**
		720	0.4606	0.4725	0.4460	0.4631	0.4470	0.4646	0.5055	0.4984	0.4502	0.4661	0.4748	0.4821	0.4575	0.4707	**0.4641**	**0.4762**
	ETTm1	96	0.4664	0.4545	0.5274	0.4792	0.4470	0.4446	0.3938	0.4146	0.4227	0.4287	0.4192	0.4259	0.6202	0.5217	**0.3602**	**0.4032**
		192	0.4882	0.4665	0.5405	0.4869	0.4676	0.4554	0.4095	0.4241	0.4440	0.4404	0.4401	0.4405	0.6512	0.5368	**0.3938**	**0.4199**
		336	0.5108	0.4792	0.5590	0.4958	0.4902	0.4676	0.4688	0.4591	0.4676	0.4536	0.4639	0.4528	0.6848	0.5503	**0.4238**	**0.4361**
		720	0.5437	0.4986	0.5860	0.5122	0.5227	0.4865	0.5123	0.4885	0.5032	0.4747	0.5023	0.4736	0.7106	0.5641	**0.4717**	**0.4632**
	ETTm2	96	0.2401	0.3161	0.2568	0.3296	0.2415	0.3170	0.2258	0.3049	0.2260	0.3057	0.2274	0.3074	0.2624	0.3361	**0.2192**	**0.3006**
		192	0.2859	0.3429	0.2905	0.3492	0.2772	0.3380	0.2827	0.3466	0.2704	0.3319	0.2694	0.3324	0.3050	0.3606	**0.2644**	**0.3305**
		336	0.3379	0.3713	0.3295	0.3700	0.3185	0.3608	0.3521	0.3886	0.3197	0.3592	0.3160	0.3585	0.3398	0.3797	**0.3055**	**0.3533**
		720	0.4264	0.4211	0.4116	0.4169	0.4060	0.4102	0.4820	0.4585	0.4081	0.4102	0.4047	0.4105	0.4220	0.4243	**0.3913**	**0.4027**
	Weather	96	0.2805	0.3746	0.2189	0.2752	0.2100	0.2716	0.2507	0.3059	0.2636	0.3687	0.2343	0.2878	0.2518	0.3034	**0.2239**	**0.2730**
		192	0.3182	0.3927	0.2541	0.2873	0.2449	0.2951	0.2793	0.3234	0.3020	0.3957	0.2725	0.3137	0.2807	0.3212	**0.2544**	**0.2971**
		336	0.3562	0.4200	0.2733	0.3097	0.2851	0.3201	0.3127	0.3433	0.3426	0.4190	0.3116	0.3383	0.3255	0.3498	**0.2940**	**0.3203**
		720	0.3830	0.3845	0.3380	0.3517	0.3406	0.3600	0.3646	0.3764	0.3700	0.3620	0.3691	0.3762	0.3790	0.3842	**0.3457**	**0.3553**
	Exchange	96	0.2846	0.3973	0.2718	0.3877	0.3063	0.4123	0.3027	0.4128	**0.2671**	**0.3823**	0.3059	0.4105	0.3380	0.4300	0.4194	0.4830
		192	0.4032	0.4765	0.3936	0.4678	0.4214	0.4858	0.4095	0.4811	**0.3829**	**0.4619**	0.4298	0.4887	0.4470	0.4983	0.5362	0.5470
		336	0.5810	0.5870	0.5725	0.5730	0.5949	0.5879	0.5728	0.5779	**0.5568**	**0.5654**	0.6040	0.5894	0.6121	0.5929	0.6818	0.6309
		720	1.2260	0.6430	1.2284	0.8429	1.2199	0.8428	1.1415	0.8132	**1.1889**	**0.8310**	1.2290	0.8458	1.1700	0.8208	1.2939	0.8658

Table 16: Full results of time series classification task evaluated by Accuracy and F1 Score.

Models	TNC		TS-TCC		TS2Vec		SimMTM		TST		TST-Zero		TST-Plus		CrossTimeNet	
Metric	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
HAR	0.8961	0.8951	0.8832	0.8815	0.8968	0.8957	0.9200	0.9220	0.9203	0.9203	0.9121	0.9120	0.8550	0.8520	**0.9335**	**0.9347**
EEG	0.7603	0.4457	0.7291	0.4347	0.7565	0.4449	0.8165	0.6123	0.8086	0.5516	0.7938	0.5211	0.7929	0.5426	**0.8541**	**0.6402**
ECG	0.2081	0.3310	0.1178	0.3780	0.1302	0.2064	0.2565	0.3562	0.2206	0.3317	0.1810	0.3861	0.2134	0.3246	**0.4378**	**0.6278**
HAR	0.8920	0.8912	0.7713	0.7652	0.7520	0.7504	0.7241	0.7861	0.8337	0.8300	0.7211	0.7120	0.7800	0.8520	**0.9146**	**0.9148**
EEG	0.1033	0.4643	0.7559	0.4643	0.7347	0.4172	0.5623	0.2281	0.6664	0.3553	0.5538	0.2236	0.6945	0.3762	**0.8381**	**0.6072**
ECG	0.1051	0.0810	0.0108	0.0140	N/A	N/A	N/A	N/A	0.1033	0.0234	N/A	N/A	0.0531	0.0818	**0.2134**	**0.3148**