Learning to Embed Time Series Patches Independently

Seunghan Lee, Taeyoung Park, Kibok Lee
Department of Statistics and Data Science, Yonsei University
{seunghan9613,tpark,kibok}@yonsei.ac.kr

Abstract

Conventional masked time series modeling patchify and partially mask out time series (TS), and then train Transformers to capture the dependencies between patches by predicting masked patches from unmasked patches. However, we argue that capturing such dependencies might not be an optimal strategy for TS representation learning; rather, embedding patches independently results in better representations. Specifically, we propose to use 1) the patch reconstruction task, autoencoding each patch without looking at other patches, and 2) the MLP that embeds each patch independently. In addition, we introduce complementary contrastive learning to hierarchically capture adjacent TS information efficiently. Our proposed method improves various tasks compared to state-of-the-art Transformer-based models, while it is more efficient in terms of the number of parameters and training time.

1 Introduction

Masked time series modeling (MTM) has been studied as a pretext task that patchifies and partially masks out TS and predicts the masked parts from the unmasked parts using encoders capturing dependencies among the patches [34, 21]. However, we argue that learning such dependencies among patches might not be necessary for TS representation learning.

To this end, we introduce the concept of *patch independence* which does not consider the interaction between TS patches when embedding them, which is realized through two aspects: 1) the pretraining task and 2) the model architecture. Firstly, we propose a patch reconstruction task that reconstructs the unmasked patches in contrast to the conventional MTM task that predicts the masked patches. We refer to these tasks as the patch-independent (PI) and patch-dependent (PD) tasks, respectively, as the former does not require information of other patches while the latter does. Toy example in Figure 1 indicates that Transformer pretrained on the PI task is more robust to distribution shift than the model trained on the PD task [21]. Secondly, we propose to use the PI architecture (e.g., MLP) instead of the PD architecture (e.g., Transformer), which is not only more efficient but also performs better.

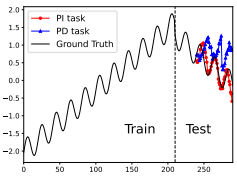

Figure 1: PI vs. PD.

In this paper, we propose **P**atch **I**ndependence for **T**ime **S**eries (*PITS*), which utilizes unmasked patch reconstruction as the PI pretraining task and MLP as the PI architecture. On top of that, we introduce complementary contrastive learning (CL) to efficiently capture adjacent TS information. The main contributions are summarized as follows:

- We argue that *learning to embed time series patches independently* is superior to learning them dependently for TS representation learning. To achieve PI, we propose PITS, incorporating two major modifications on the MTM: 1) to make the task patch-independent, reconstructing the unmasked patches instead of predicting the masked ones, and 2) to make the encoder patch-independent, removing the attention mechanism to ignore correlation between the patches.
- We introduce complementary contrastive learning to hierarchically capture adjacent TS information efficiently, where positive pairs are made by complementary random masking.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

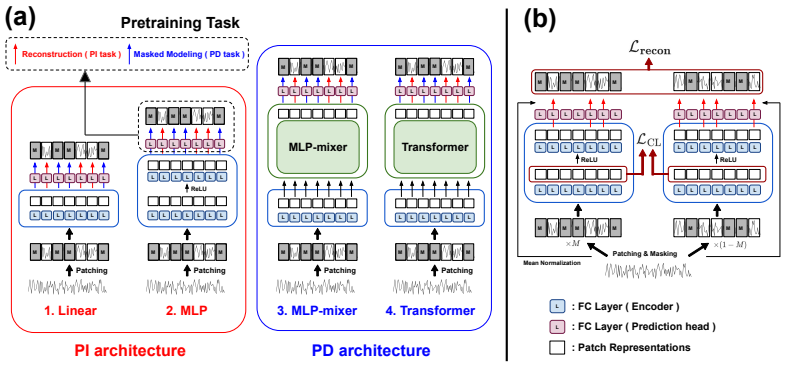

Figure 2: **Patch-independent strategy of PITS.** (a) illustrates the pretraining tasks and encoder architectures in terms of PI and PD. (b) demonstrates the proposed PITS, which utilizes a PI task with a PI architecture. TS is divided into patches and augmented with complementary masking.

- Extensive experiments demonstrate that our method improves SOTA performance on both forecasting and classification. We also discover that PI tasks outperforms PD tasks in managing distribution shifts, and that PI architecture is more interpretable and robust to patch size than PD architecture.

2 Methods

We address the task of learning an embedding function $f_\theta : \boldsymbol{x}_p^{(i,c,n)} \rightarrow \boldsymbol{z}^{(i,c,n)}$ for a TS patch where $\boldsymbol{x}_p = \left\{\boldsymbol{x}_p^{(i,c,n)}\right\}$, $\boldsymbol{z} = \left\{\boldsymbol{z}^{(i,c,n)}\right\}$, and $i = 1, \ldots, B$, $c = 1, \ldots, C$, $n = 1, \ldots, N$. Here, B, C, N are the number of TS, number of channels in a TS, and number of patches in a channel. The input and the output dimension, which are the patch length and patch embedding dimension, are denoted as P and D, respectively, i.e., $\boldsymbol{x}_p^{(i,c,n)} \in \mathbb{R}^P$ and $\boldsymbol{z}^{(i,c,n)} \in \mathbb{R}^D$. For f_θ, we adopt channel-independent [21] and patch-independent architecture, i.e, f_θ is independent to c and n respectively.

2.1 Patch-Independent Task: Patch Reconstruction

Unlike the conventional PD task that predicts masked patches using unmasked ones, we propose the patch reconstruction task (i.e., PI task) that autoencodes each patch without looking at the other patches, as depicted in Figure 2(a). TS patch can be reconstructed in two different ways: 1) reconstruction at once by a FC layer processing the concatenation of representations, and 2) patch-wise reconstruction by a FC layer processing each representation. Similar to PatchTST [21], we employ the patch-wise reconstruction. On top of that, we propose mean normalization, subtracting the average value of unmasked parts for each TS before encoding and adding it back to the prediction.

2.2 Patch-Independent Architecture: MLP

While MTM has been usually studied with Transformers for capturing dependencies between patches, we argue that learning to embed patches independently is better. Following this idea, we propose to use the simple PI architecture (e.g., MLP) to solely focuses on extracting patch-wise representations. Figure 2(a) shows the examples of PI/PD tasks and architectures, where MLP-Mixer consists of a single FC layer for time-mixing followed by a two-layer MLP for patch-mixing.

To show the efficiency introduced by PI architectures, we compare PatchTST [21] and PITS in terms of the number of parameters and training time on the ETTm1 dataset, where PatchTST uses the PD task and PD architecture while PITS uses the PI task and PI architecture. As shown in Table 1, PITS requires much less parameters and trains faster.

Table 1: Time/parameter efficiency.

	PatchTST	PITS	Gain
# params.	406,028	5,772	x 70.3
Training time	46	15	x 3.06

2.3 Complementary Contrastive Learning

To further boost performance of learned representations, we propose complementary CL to hierarchically capture adjacent TS information. CL requires two views to generate positive pairs, and we achieve this by a complementary masking: for a TS $\boldsymbol{x}$ and a mask $\boldsymbol{m}$ with the same length, we consider $\boldsymbol{m} \odot \boldsymbol{x}$ and $(1 - \boldsymbol{m}) \odot \boldsymbol{x}$ as two views for CL. Note that the purpose of masking is to generate two views for CL; it does not affect the PI task, and it does not require an additional forward pass when using the proposed PI architectures, such that the additional computational cost is negligible.

Figure 3 illustrates an example of complementary CL, where we perform CL hierarchically [32] by max-pooling on the representations along the temporal axis, and compute and aggregate losses computed at each level. Then, the model learns to find missing patch information in one view, by contrasting the similarity with another view and the others, so that the model can capture adjacent TS information hierarchically.

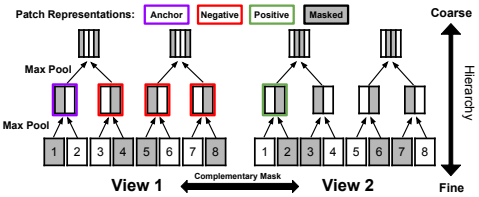

Figure 3: Complementary CL.

2.4 Objective Function

Reconstruction loss. As illustrated in Figure 2(b), we perform CL at the first layer and reconstruction by an additional projection head on top of the second layer, where we denote representations obtained from the two layers as $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, respectively. We feed $\boldsymbol{z}_2$ into the patch-wise linear projection head to get a reconstructed result: $\widehat{\boldsymbol{x}}_p = W\boldsymbol{z}_2$. Then, the reconstruction loss can be written as:

$$\begin{aligned}\mathcal{L}_{\text{recon}} &= \sum_{i=1}^{B}\sum_{c=1}^{C}\sum_{n=1}^{N}\left\|\boldsymbol{m}^{(i,c,n)}\odot\left(\boldsymbol{x}_p^{(i,c,n)}-\widehat{\boldsymbol{x}}_p^{(i,c,n)}\right)\right\|_2^2+\left\|(1-\boldsymbol{m}^{(i,c,n)})\odot\left(\boldsymbol{x}_p^{(i,c,n)}-\widehat{\boldsymbol{x}}_p^{(i,c,n)}\right)\right\|_2^2 \\ &= \sum_{i=1}^{B}\sum_{i=1}^{C}\sum_{n=1}^{N}\left\|\boldsymbol{x}_p^{(i,c,n)}-\widehat{\boldsymbol{x}}_p^{(i,c,n)}\right\|_2^2, \end{aligned} \tag{1}$$

where $\boldsymbol{m}^{(i,c,n)} = 0$ if the first view $\boldsymbol{x}_p^{(i,c,n)}$ is masked, and 1 otherwise. As derived in Eq. 1, the reconstruction task is not affected by complementary masking.

Contrastive loss. Inspired by the cross-entropy loss-like formulation of the contrastive loss in i-Mix [17], we establish a softmax probability for the relative similarity among all the similarities considered when computing contrastive loss. For conciseness, let $\boldsymbol{z}_1^{(i,c,n)} = \boldsymbol{z}^{(i,c,n+2N)}$ and $\boldsymbol{z}^{(i,c,n+N)}$ be the two views of $\boldsymbol{x}^{(i,c,n)}$. Then, the softmax probability for a pair of patch indices (n, n') is defined as:

$$p(i,c,(n,n')) = \frac{\exp(\boldsymbol{z}^{(i,c,n)}\circ\boldsymbol{z}^{(i,c,n')})}{\sum_{s=1,s\neq n}^{2N}\exp(\boldsymbol{z}^{(i,c,n)}\circ\boldsymbol{z}^{(i,c,s)})}, \tag{2}$$

where $\circ$ is the dot product. Then, the total contrastive loss can be written as:

$$\mathcal{L}_{\text{CL}} = \frac{1}{2BCN}\sum_{i=1}^{B}\sum_{i=1}^{C}\sum_{n=1}^{2N}-\log p(i,c,(n,n+N)), \tag{3}$$

where we compute the hierarchical losses by max-pooling $\boldsymbol{z}^{(i,c,n)}$'s along with the dimension n repeatedly. The final loss is the sum of the reconstruction loss $\mathcal{L}_{\text{recon}}$ and contrastive loss $\mathcal{L}_{\text{CL}}$.

3 Experiments

Time series forecasting (TSF). For forecasting tasks, we experiment seven datasets, including four ETT datasets, Weather, Traffic, and Electricity [30], with a prediction horizon of $H \in \{96, 192, 336, 720\}$. For the baseline methods, we consider Transformer-based models including PatchTST [21, 8, 36, 36, 30] and MLP models including TSMixer [4, 33]. We follow the experimental setups and baseline results from PatchTST [21], SimMTM [8], and TSMixer [4]. Table 2 shows the results of the average MSE across four horizons, demonstrating that PITS is competitive to SOTA methods in both settings.

Table 2: Results of multivariate TSF.

Models	Self-supervised						Supervised													
	PITS		PatchTST*		SimMTM†		PITS		PatchTST		SimMTM†		DLinear		TSMixer		FEDformer		Autoformer	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.401**	**0.421**	0.424	0.439	0.404	0.428	**0.409**	**0.426**	0.417	0.431	0.431	0.443	0.423	0.437	0.412	0.428	0.428	0.454	0.473	0.477
ETTh2	**0.331**	**0.382**	0.373	0.404	0.348	0.391	0.337	0.386	**0.331**	**0.379**	0.395	0.427	0.431	0.447	0.355	0.401	0.388	0.434	0.422	0.443
ETTm1	**0.341**	**0.380**	0.349	**0.380**	**0.341**	**0.380**	0.350	0.381	0.352	0.382	0.356	0.362	0.355	0.379	**0.347**	**0.375**	0.382	0.422	0.515	0.493
ETTm2	**0.244**	**0.310**	0.264	0.323	0.260	0.318	**0.247**	**0.314**	0.258	0.315	0.279	0.336	0.267	0.332	0.267	0.322	0.292	0.343	0.310	0.357
Weather	**0.225**	**0.262**	0.226	**0.262**	0.235	0.280	**0.225**	**0.263**	0.230	0.265	0.239	0.275	0.246	0.300	**0.225**	0.264	0.310	0.357	0.335	0.379
Traffic	0.399	0.271	0.400	0.274	**0.392**	**0.264**	0.401	0.276	**0.396**	**0.266**	0.490	0.316	0.434	0.295	0.408	0.281	0.604	0.372	0.617	0.384
Electricity	**0.157**	0.253	0.159	**0.251**	0.162	0.256	**0.160**	0.255	0.162	**0.254**	0.212	0.300	0.166	0.264	**0.160**	0.257	0.207	0.321	0.214	0.327
Average	**0.300**	**0.326**	0.314	0.333	0.306	0.331	**0.304**	0.329	0.307	**0.327**	0.343	0.355	0.332	0.351	0.311	0.333	0.373	0.386	0.412	0.409

* We used the official code to replicate the results. † SimMTM is a concurrent work to ours.

Time series classification (TSC). For classification tasks, we use five datasets, SleepEEG [16], Epilepsy [1], FD-B [18], Gesture [20], and EMG [12]. For the baseline methods, we employ TS2Vec

Table 3: TSC.

	ACC.	F_1
TS2Vec	92.17	93.84
CoST	88.07	69.11
LaST	92.11	85.74
TF-C	93.96	89.46
TST	80.21	44.51
TimeMAE	80.34	45.20
SimMTM	94.75	91.41
PITS w/o CL	95.27	95.30
PITS	**95.67**	**95.64**

Table 4: TSC with transfer learning.

	In-domain		Cross-domain					
	→ Epilepsy		→ FD-B		→ Gesture		→ EMG	
	ACC.	F_1	ACC.	F_1	ACC.	F_1	ACC.	F_1
TS2Vec	93.95	90.45	47.90	43.89	69.17	65.70	78.54	67.66
CoST	88.40	76.88	47.06	34.79	68.33	66.42	53.65	35.27
LaST	86.46	70.67	46.67	45.17	64.17	58.76	66.34	72.55
TF-C	94.95	91.49	69.38	74.87	76.42	75.72	81.71	76.83
TST	80.21	44.51	46.40	41.34	69.17	66.01	46.34	21.11
TimeMAE	89.71	68.55	70.88	66.56	71.88	68.37	69.99	70.89
SimMTM	95.49	92.81	69.40	75.11	80.00	78.67	97.56	98.14
PITS	**95.71**	**95.70**	**87.70**	**87.68**	**92.50**	**92.48**	**100.0**	**100.0**

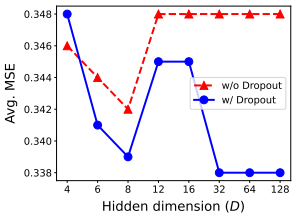

Figure 4: MSE by dropout.

Table 5: Effectiveness of PI.

	PI acrhitecture				PD architecture			
	Linear		MLP		MLP-Mixer		Transformer	
Task	PD	PI	PD	PI	PD	PI	PD	PI
ETTh1	0.408	0.408	0.418	**0.407**	0.420	**0.409**	0.425	**0.415**
ETTh2	0.343	**0.338**	0.361	**0.334**	0.365	**0.341**	0.353	**0.342**
ETTm1	0.359	**0.358**	0.356	**0.355**	0.354	**0.352**	0.350	0.350
ETTm2	0.254	**0.243**	0.258	**0.253**	0.259	**0.253**	0.274	**0.256**
Avg.	0.342	**0.340**	0.348	**0.337**	0.350	**0.339**	0.351	**0.341**

Table 6: Effect of complementary CL.

PI task		ETTh1	ETTh2	ETTm1	ETTm2	Avg.
Transformer		0.425	0.353	0.350	0.274	0.351
MLP	w/o CL	0.407	0.334	0.357	0.253	0.338
	w/ non-hier. CL	0.405	0.333	0.353	0.252	0.336
	w/ hier. CL	**0.401**	**0.331**	**0.341**	**0.244**	**0.329**

[32], CoST [29], LaST [26], TF-C [35], TST [34], TimeMAE [6] and SimMTM [8]. Table 3 demonstrates that PITS outperforms all SOTA methods on the Epilepsy dataset. We also conduct experiments with transfer learning in both in- and cross-domain transfer settings, using SleepEEG as the source dataset for both settings, where the settings are defined in SimMTM. Table 4 demonstrates that our PITS outperforms SOTA methods in all scenarios.

Effect of PI/PD tasks/architectures. To assess the effect of our proposed PI pretraining task and PI encoder architecture, we conduct an ablation study in Table 5. As shown in Table 5 which shows average MSE across four horizons, PI pretraining results in better TSF performance than PD pretraining regardless of the choice of the architecture.

Hierarchical design of complementary CL. The proposed complementary CL is structured hierarchically to capture both coarse and fine-grained information in TS. To evaluate the effect of this hierarchical design, we consider three different options: 1) without CL, 2) with non-hierarchical CL, and 3) with hierarchical CL. Table 6 presents the average MSE across four horizons, highlighting the performance gain by the hierarchical design.

MLP is more robust to patch size than Transformer. To assess the robustness of encoder architectures to patch size, we compare MLP and Transformer using ETTh1 with different patch sizes. Figure 5 illustrates the results, indicating that MLP is more robust for both the PI and PD tasks, resulting in consistently better forecasting performance across various patch sizes.

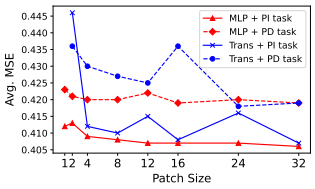

Figure 5: MSE by patch size.

Hidden dimension and dropout. The PI task may raise a concern on the trivial solution: when the hidden dimension D is larger than the input dimension P, the identity mapping perfectly reconstructs the input. This can be addressed by introducing dropout, where we add a dropout layer before the linear projection head. Figure 4 displays the average MSE on four ETT datasets across four horizons under various D in MLP, without dropout or with the dropout rate of 0.2. Note that for this experiment, the patch length P is 12, and a trivial solution can occur if $D \geq 12$. The results confirm that using dropout is necessary to learn high dimensional representations, leading to better performance.

4 Conclusion

This paper revisits masked modeling in time series analysis, focusing on two key aspects: 1) the pretraining task and 2) the model architecture. In contrast to previous works that primarily emphasize dependencies between TS patches, we advocate a patch-independent approach on two fronts: 1) by introducing a patch reconstruction task and 2) employing patch-wise MLP. Our results demonstrate that the proposed PI approach is more robust to distribution shifts and patch size compared to the PD approach, resulting in superior performance while more efficient in both forecasting and classification tasks. We hope that our work sheds light on the effectiveness of self-supervised learning through simple pretraining tasks and model architectures in various domains.

References

[1] 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.

[2] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *ICML*, 2022.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[4] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*, 2023.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

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

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2018.

[8] Jiaxiang Dong, Haixu Wu, Haoran Zhang, Li Zhang, Jianmin Wang, and Mingsheng Long. Simmtm: A simple pre-training framework for masked time-series modeling. *arXiv preprint arXiv:2302.00861*, 2023.

[9] 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*, 2021.

[10] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP*, 2021.

[11] Spyros Gidaris and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

[12] 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.

[13] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

[14] Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *arXiv preprint arXiv:2304.05206*, 2023.

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[16] 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.

[17] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-mix: A domain-agnostic strategy for contrastive representation learning. In *ICLR*, 2021.

[18] 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. PHM Society, 2016.

[19] Jianglin Liang and Ruifang Liu. Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network. In *CISP*, 2015.

[20] Jun Liu, Lin Zhong, Jehan Wickramasuriya, and Vijay Vasudevan. Uwave: accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[21] Yushan Nie, Nam H Nguyen, Pattarawat Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *ICLR*, 2023.

[22] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[23] Pengxiang Shi, Wenwen Ye, and Zheng Qin. Self-supervised pre-training for time series classification. In *IJCNN*, 2021.

[24] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(11), 2008.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[26] Zhiyuan Wang, Xovee Xu, Weifeng Zhang, Goce Trajcevski, Ting Zhong, and Fan Zhou. Learning latent seasonal-trend representations for time series forecasting. In *NeurIPS*, 2022.

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

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

[29] 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.

[30] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, 2021.

[31] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *CVPR*, 2022.

[32] Zhanwei Yue, Yiqun Wang, Jinghua Duan, Tao Yang, Chen Huang, Yunhai Tong, and Bo Xu. Ts2vec: Towards universal representation of time series. In *AAAI*, 2022.

[33] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI*, 2023.

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

[35] Xiaotian Zhang, Zeyu Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. Self-supervised contrastive pre-training for time series via time-frequency consistency. In *NeurIPS*, 2022.

[36] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*, 2022.

A Dataset Description

A.1 Time Series Forecasting

For time series forecasting, we assess the effectiveness of our proposed PITS using seven datasets, including four ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2), Weather, Traffic, and Electricity. These datasets have been widely employed for benchmarking and are publicly accessible [30]. The statistics of these datasets are summarized in Table A.1.

Datasets	ETTh1	ETTh2	ETTm1	ETTm2	Weather	Traffic	Electricity
Features	7	7	7	7	21	862	321
Timesteps	17420	17420	69680	69680	52696	17544	26304

Table A.1: Statistics of datasets for forecasting.

A.2 Time Series Classification

For time series classification, we use five datasets of different characteristics, as described in Table A.2. Note that both SleepEEG and Epilepsy datasets belong to the same domain, characterized by being EEG datasets. For transfer learning tasks, we define them as being part of the same domain.

Dataset	# Samples	# Channels	# Classes	Length	Freq (Hz)
SleepEEG	371,055	1	5	200	100
Epilepsy	60 / 20 / 11,420	1	2	178	174
FD-B	60 / 21 / 13,559	1	3	5,120	64,000
Gesture	320 / 120 / 120	3	8	315	100
EMG	122 / 41 / 41	1	3	1,500	4,000

Table A.2: Statistics of datasets for classification.

B Related Works

Self-supervised learning. In recent years, self-supervised learning (SSL) has gained attention for learning powerful representations from unlabeled data across various domains. The success of SSL comes from active research on pretext tasks that predict a certain aspect of data without supervision. Next token prediction [3] and masked token prediction [7] are commonly used in natural language processing, and jigsaw puzzles [22] and rotation prediction [11] are commonly used in computer vision.

Recently, contrastive learning (CL) [13] has emerged as an effective pretext task. The key principle of CL is to maximize similarities between positive pairs while minimizing similarities between negative pairs [10, 5, 32]. Another promising technique is masked modeling, which trains the models to reconstruct masked patches based on the unmasked part. For instance, in natural language processing, models predict masked words within a sentence [7], while in computer vision, they predict masked patches in images [2, 15, 31] within their respective domains.

		CL for TS*	TST (KDD 2021)	TS2Vec (AAAI 2022)	FEDFormer (ICML 2022)	DLinear (AAAI 2023)	PatchTST (ICLR 2023)	TimeMAE (arXiv 2023)	SimMTM (arXiv 2023)	PITS
Pretrain	CL	✓		✓						✓
	MTM		✓				✓	✓	✓	✓
	No (SL)		✓		✓	✓	✓		✓	✓
Task	Forecasting		✓	✓	✓	✓	✓		✓	✓
	Classification	✓	✓	✓				✓	✓	✓

* T-Loss (NeurIPS 2019), Self-Time (arXiv 2020), TS-SD (IJCNN 2021), TS-TCC (IJCAI 2021), TNC (arXiv 2021), Mixing-up (PR Letters 2022), TF-C (NeurIPS 2022), TimeCLR (KBS 2022), CA-TCC (TPAMI 2023).

Table B.1: Comparison table of SOTA methods in TS.

Masked time series modeling. Besides CL, masked modeling has gained attention as a pretext task for SSL in time series. This task involves masking a portion of the TS and predicting the missing values, known as masked time series modeling (MTM). While CL has shown impressive performance in high-level classification tasks, masked modeling has excelled in low-level forecasting tasks [32, 21].

TST [34] applies the masked modeling paradigm to TS, aiming to reconstruct masked timestamps. PatchTST [21] focuses on predicting masked subseries-level patches to capture local semantic information and reduce memory usage. SimMTM [8] reconstructs the original TS from multiple masked TS. TimeMAE [6] trains a transformer-based encoder using two pretext tasks, masked codeword classification and masked representation regression. Table B.1 compares various methods in TS including ours in terms of two criterions: pretraining methods and downstream tasks.

Different from recent MTM works, we propose to reconstruct unmasked patches through autoencoding. A primary concern on autoencoding is the trivial solution of identity mapping, such that the dimension of hidden layers should be smaller than the input. To alleviate this, we introduce dropout after intermediate fully-connected (FC) layers, which is similar to the case of stacked denoising autoencoders [19], where the ablation study can be found in Figure 4.

Linear models for time series forecasting. Transformer [25] is a popular sequence modeling architecture that has prompted a surge in Transformer-based solutions for time series analysis [27]. Transformers derive their primary strength from the multi-head self-attention mechanism, excelling at extracting semantic correlations within extensive sequences. Nevertheless, recent work [33] shows that simple linear models can still extract such information captured by Transformer-based methods. Motivated by this work, we propose to use a simple MLP architecture that does not encode interaction between time series patches.

C More on Experiments Section

C.1 Experimental Settings

Tasks and evaluation metrics. We demonstrate the effectiveness of the proposed PITS on two downstream tasks: time series forecasting (TSF) and classification (TSC) tasks. For evaluation, we mainly follow the standard SSL framework that pretrains and fine-tunes the model on the same dataset, but we also consider in-domain and cross-domain transfer learning settings in some experiments. As evaluation metrics, we use the mean squared error (MSE) and mean absolute error (MAE) for TSF, and accuracy, precision, recall, and the F_1 score for TSC.

C.2 Time Series Forecasting

Datasets and baseline methods. For forecasting tasks, we experiment seven datasets, including four ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2), Weather, Traffic, and Electricity [30], with a prediction horizon of $H \in \{96, 192, 336, 720\}$. For the baseline methods, we consider Transformer-based models, including PatchTST [21], SimMTM [8], FEDformer [36], and Autoformer [30], and MLP models, including DLinear [33] and TSMixer [4]. We also compare PITS and PatchTST without self-supervised pretraining[1], which essentially compares PI and PD architectures. We follow the experimental setups and baseline results from PatchTST, SimMTM, and TSMixer.

Standard setting. Table C.1 shows the comprehensive results on the multivariate TSF task, demonstrating that our proposed PITS is competitive to PatchTST in both settings, which is the SOTA Transformer-based method, while PITS is much more efficient than PatchTST. SimMTM is a concurrent work showing similar performance to ours in SSL while significantly worse in supervised learning. Table C.2 compares PITS and PatchTST under three different scenarios: fine-tuning (FT), linear probing (LP), and supervised learning without self-supervised pretraining (Sup.), where we present the average MSE across four horizons. As shown in Table C.2, PITS outperforms PatchTST for all scenarios on average, and the margin is significant for the linear probing scenario.

Models		Self-supervised						Supervised													
		PITS		PatchTST*		SimMTM†		PITS		PatchTST		SimMTM†		DLinear		TSMixer		FEDformer		Autoformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.364**	**0.392**	0.379	0.408	0.367	0.402	0.369	0.397	0.375	0.399	0.380	0.412	0.375	0.399	**0.361**	**0.392**	0.376	0.415	0.435	0.446
	192	**0.398**	**0.414**	0.414	0.428	0.403	0.425	**0.402**	**0.416**	0.414	0.421	0.416	0.434	0.405	**0.416**	0.404	0.418	0.423	0.446	0.456	0.457
	336	**0.415**	**0.427**	0.435	0.446	**0.415**	0.430	**0.409**	**0.425**	0.431	0.436	0.448	0.458	0.439	0.443	0.420	0.431	0.444	0.462	0.486	0.487
	720	**0.425**	**0.451**	0.468	0.474	0.430	0.453	0.457	**0.465**	**0.449**	0.466	0.481	0.469	0.472	0.490	0.463	0.472	0.469	0.492	0.515	0.517
ETTh2	96	**0.269**	**0.332**	0.306	0.351	0.288	0.347	0.281	0.343	**0.274**	**0.336**	0.325	0.374	0.289	0.353	**0.274**	0.341	0.332	0.374	0.332	0.368
	192	**0.326**	**0.372**	0.361	0.392	0.346	0.385	0.345	0.384	**0.339**	**0.379**	0.400	0.424	0.383	0.418	0.339	0.385	0.407	0.446	0.426	0.434
	336	**0.354**	**0.396**	0.405	0.427	0.363	0.401	0.343	0.389	**0.331**	**0.380**	0.405	0.433	0.448	0.465	0.361	0.406	0.400	0.447	0.477	0.479
	720	**0.378**	**0.425**	0.419	0.446	0.396	0.431	0.388	0.430	**0.379**	**0.422**	0.451	0.475	0.605	0.551	0.445	0.470	0.412	0.469	0.453	0.490
ETTm1	96	0.296	0.355	0.294	0.345	**0.289**	**0.343**	0.295	0.346	0.290	0.342	0.295	0.346	0.299	0.343	**0.285**	**0.339**	0.326	0.390	0.510	0.492
	192	**0.321**	**0.368**	0.327	0.369	0.323	0.369	**0.331**	0.369	0.332	0.369	0.333	0.374	0.335	0.365	0.327	0.365	0.365	0.415	0.514	0.495
	336	0.353	0.388	**0.364**	**0.390**	0.349	0.385	0.360	0.388	0.366	0.392	0.370	0.398	0.369	0.386	**0.356**	**0.382**	0.392	0.425	0.510	0.492
	720	**0.395**	**0.412**	0.409	0.415	0.399	0.418	**0.416**	0.420	0.420	0.424	0.427	0.431	0.425	0.421	0.419	**0.414**	0.446	0.458	0.527	0.493
ETTm2	96	**0.163**	**0.255**	0.167	0.256	0.166	0.257	0.163	0.255	0.165	0.255	0.175	0.268	0.167	0.260	0.163	**0.252**	0.180	0.271	0.205	0.293
	192	**0.213**	**0.289**	0.232	0.302	0.223	0.295	**0.215**	0.293	0.220	0.292	0.240	0.312	0.224	0.303	0.216	**0.290**	0.252	0.318	0.278	0.336
	336	**0.263**	**0.324**	0.291	0.342	0.282	0.334	**0.266**	0.328	0.278	0.329	0.298	0.351	0.281	0.342	0.268	**0.324**	0.324	0.364	0.343	0.379
	720	**0.337**	**0.373**	0.368	0.390	0.370	0.385	**0.342**	**0.381**	0.367	0.385	0.403	0.413	0.397	0.421	0.420	0.422	0.410	0.420	0.414	0.419
Weather	96	0.149	0.201	**0.146**	**0.194**	0.151	0.202	0.153	0.202	0.152	0.199	0.166	0.216	0.176	0.237	**0.145**	**0.198**	0.238	0.314	0.249	0.329
	192	0.195	0.241	**0.192**	**0.238**	0.223	0.295	**0.191**	**0.242**	0.197	0.243	0.208	0.254	0.220	0.282	**0.191**	**0.242**	0.275	0.329	0.325	0.370
	336	**0.244**	**0.280**	0.245	**0.280**	0.246	0.283	0.245	**0.280**	0.249	0.283	0.257	0.290	0.265	0.319	**0.242**	**0.280**	0.339	0.377	0.351	0.391
	720	**0.312**	**0.328**	0.320	0.336	0.320	0.338	**0.310**	0.329	0.320	0.335	0.326	0.338	0.323	0.362	0.320	0.336	0.389	0.409	0.415	0.426
Traffic	96	0.373	**0.257**	0.393	0.275	**0.368**	0.262	0.375	0.264	**0.367**	**0.251**	0.471	0.309	0.410	0.282	0.376	0.264	0.576	0.359	0.597	0.371
	192	0.388	0.266	0.376	0.254	**0.373**	**0.251**	0.389	0.270	**0.385**	**0.259**	0.475	0.308	0.423	0.287	0.397	0.264	0.610	0.380	0.607	0.382
	336	0.401	0.271	0.384	0.259	**0.395**	**0.254**	0.401	0.277	**0.398**	**0.265**	0.490	0.315	0.436	0.296	0.413	0.290	0.608	0.375	0.623	0.387
	720	0.436	**0.290**	0.446	0.306	**0.432**	**0.290**	0.437	0.294	**0.434**	**0.287**	0.524	0.332	0.466	0.315	0.444	0.306	0.621	0.375	0.639	0.395
Electricity	96	0.129	0.225	**0.126**	**0.221**	0.133	0.223	0.131	0.228	**0.130**	**0.222**	0.190	0.279	0.140	0.237	0.131	0.229	0.186	0.302	0.196	0.313
	192	**0.144**	0.240	0.145	0.238	0.147	**0.237**	**0.147**	0.242	0.148	**0.240**	0.195	0.285	0.153	0.249	0.151	0.246	0.197	0.311	0.211	0.324
	336	**0.160**	**0.256**	0.164	**0.256**	0.166	0.265	0.162	**0.260**	0.167	0.261	0.211	0.301	0.169	0.267	**0.161**	0.261	0.213	0.328	0.214	0.327
	720	**0.197**	**0.290**	0.200	**0.290**	0.203	0.297	0.199	**0.290**	0.202	0.291	0.253	0.333	0.203	0.301	**0.197**	0.293	0.233	0.344	0.236	0.342
Average		**0.300**	**0.326**	0.314	0.333	0.306	0.331	**0.304**	0.329	0.307	**0.327**	0.343	0.355	0.332	0.351	0.311	0.333	0.373	0.386	0.412	0.409

* We used the official code to replicate the results. † SimMTM is a concurrent work to ours.

Table C.1: **Results of multivariate TSF.** We compare both the supervised and self-supervised versions of PatchTST and our method. The best results are in bold and the second best are underlined.

Transfer learning. In in-domain transfer, we experiment datasets with the same frequency for the source and target datasets, whereas in cross-domain transfer, datasets with different frequencies are utilized for the source and target datasets. Table C.3 shows the results of the average MSE across four horizons, which demonstrates that our proposed PITS surpasses the SOTA methods in most cases.

[1]For PITS and PatchTST supervised learning, patches are overlapped following PatchTST [21].

Metric: MSE	PITS			PatchTST		
	FT	LP	Sup.	FT	LP	Sup.
ETTh1	**0.401**	0.403	0.409	0.424	0.434	0.417
ETTh1	**0.331**	0.334	0.337	0.373	0.364	**0.331**
ETTm1	**0.341**	0.356	0.350	0.349	0.355	0.352
ETTm2	**0.244**	**0.244**	0.247	0.264	0.264	0.258
Weather	**0.225**	0.239	**0.225**	0.226	0.233	0.230
Traffic	0.399	0.406	0.401	0.400	0.424	**0.396**
Electricity	**0.157**	0.161	0.160	0.159	0.168	0.162
Average	**0.300**	0.306	0.304	0.314	0.320	0.307

Table C.2: PITS vs. PatchTST.

	Source	Target	PITS		PatchTST		SimMTM	TimeMAE	TST	LaST	TF-C	CoST
			FT	LP	FT	LP						
In-domain	ETTh2	ETTh1	0.404	**0.403**	0.423	0.464	0.415	0.728	0.645	0.443	0.635	0.584
	ETTm2	ETTm1	**0.345**	0.354	0.348	0.411	0.351	0.682	0.480	0.414	0.758	0.354
	Average		**0.375**	0.378	0.386	0.406	0.383	0.705	0.563	0.429	0.697	0.469
Cross-domain	ETTm2	ETTh1	0.407	**0.405**	0.433	0.421	0.428	0.724	0.632	0.503	1.091	0.582
	ETTh2	ETTm1	**0.350**	0.357	0.363	0.378	0.365	0.688	0.472	0.475	0.750	0.377
	ETTm1	ETTh1	**0.406**	0.407	0.447	0.432	0.422	0.726	0.645	0.426	0.700	0.750
	ETTh1	ETTm1	0.353	0.357	0.348	0.374	**0.346**	0.666	0.482	0.353	0.746	0.359
	Weather	ETTh1	**0.407**	**0.407**	0.437	0.423	0.456	-	-	-	-	-
	Weather	ETTm1	0.351	0.356	**0.348**	0.355	0.358	-	-	-	-	-
	Average		**0.379**	0.382	0.396	0.397	0.396	-	-	-	-	-

Table C.3: Results of TSF with transfer learning.

C.3 Time Series Classification

Datasets and baseline methods. For classification tasks, we use five datasets, SleepEEG [16], Epilepsy [1], FD-B [18], Gesture [20], and EMG [12]. For the baseline methods, we employ TS-SD [23], TS2Vec [32], CoST [29], LaST [26], Mixing-Up [28], TS-TCC [9], TF-C [35], TST [34], TimeMAE [6] and SimMTM [8].

	ACC.	PRE.	REC.	F_1
TS2Vec	92.17	93.84	81.19	85.71
CoST	88.07	91.58	66.05	69.11
LaST	92.11	93.12	81.47	85.74
TF-C	93.96	94.87	85.82	89.46
TST	80.21	40.11	50.00	44.51
TimeMAE	80.34	90.16	50.33	45.20
SimMTM	94.75	95.60	89.93	91.41
PITS*	95.27	95.35	95.27	95.30
PITS	**95.67**	**95.63**	**95.67**	**95.64**

* PITS without CL.

Table C.4: Results of TSC.

	In-domain transfer learning				Cross-domain transfer learning											
	SleepEEG → Epilepsy				SleepEEG → FD-B				SleepEEG → Gesture				SleepEEG → EMG			
	ACC.	PRE.	REC.	F_1	ACC.	PRE.	REC.	F_1	ACC.	PRE.	REC.	F_1	ACC.	PRE.	REC.	F_1
TS-SD	89.52	80.18	76.47	77.67	55.66	57.10	60.54	57.03	69.22	66.98	68.67	66.56	46.06	15.45	33.33	21.11
TS2Vec	93.95	90.59	90.39	90.45	47.90	43.39	48.42	43.89	69.17	65.45	68.54	65.70	78.54	80.40	67.85	67.66
CoST	88.40	88.20	72.34	76.88	47.06	38.79	38.42	34.79	68.33	65.30	68.33	66.42	53.65	49.07	42.10	35.27
LaST	86.46	90.77	66.35	70.67	46.67	43.90	47.71	45.17	64.17	70.36	64.17	58.76	66.34	79.34	63.33	72.55
Mixing-Up	80.21	40.11	50.00	44.51	67.89	71.46	76.13	72.73	69.33	67.19	69.33	64.97	30.24	10.99	25.83	15.41
TS-TCC	92.53	94.51	81.81	86.33	54.99	52.79	63.96	54.18	71.88	71.35	71.67	69.84	78.89	58.51	63.10	59.04
TF-C	94.95	94.56	89.08	91.49	69.38	75.59	72.02	74.87	76.42	77.31	74.29	75.72	81.71	72.65	81.59	76.83
TST	80.21	40.11	50.00	44.51	46.40	41.58	45.50	41.34	69.17	66.60	69.17	66.01	46.34	15.45	33.33	21.11
TimeMAE	89.71	72.36	67.47	68.55	70.88	66.98	68.94	66.56	71.88	70.35	76.75	68.37	69.99	70.25	63.44	70.89
SimMTM	95.49	93.36	92.28	92.81	69.40	74.18	76.41	75.11	80.00	79.03	80.00	78.67	97.56	98.33	98.04	98.14
PITS	**95.71**	**95.69**	**95.71**	**95.70**	**87.70**	**87.94**	**87.70**	**87.68**	**92.50**	**93.32**	**92.50**	**92.48**	**100.0**	**100.0**	**100.0**	**100.0**

Table C.5: Results of TSC with transfer learning.

Standard setting. Table C.4 demonstrates that our proposed PITS outperforms all SOTA methods in all metrics on the SleepEEG dataset. This contrasts with the results in prior works that CL is superior to MTM for classification tasks [32]: while prior MTM methods such as TST and TimeMAE shows relatively low performance compared to CL methods such as TS2Vec and TF-C[2], the proposed PITS outperforms CL methods, even without complementary CL.

Transfer learning. For transfer learning, we conduct experiments in both in-domain and cross-domain transfer settings, using SleepEEG as the source dataset for both settings. For in-domain transfer, we use target datasets from the same domain as the source dataset, which share the characteristic of being EEG datasets, while we use target datasets from the different domain for cross-domain transfer. Table C.5 demonstrates that our PITS outperforms SOTA methods in all scenarios. In particular, the performance gain is significant in the challenging cross-domain transfer learning setting, implying that PITS would be more practical in real-world applications under domain shifts.

[2]An exception is SimMTM [8], which is not officially published at the time of submission.

C.4 Ablation Study

Pretrain Task		Trans-former	MLP	
Input	Output		w/o CL	w CL
X_u	X_u	**0.341**	**0.338**	**0.329**
X_u	X_m	0.351	0.348	0.364
0	X_u	0.342	0.348	0.348
0	**0**	0.343	0.345	0.345

Table C.6: Pretraining tasks.

Layer 1	-	-	-	PI	CL
Layer 2	CL	PI	CL+PI	CL	PI
ETTh1	0.720	0.409	0.417	0.442	**0.401**
ETTh2	0.394	0.336	0.366	0.371	**0.331**
ETTm1	0.711	0.355	0.356	0.358	**0.341**
ETTm2	0.381	0.247	0.254	0.265	**0.244**
Avg.	0.552	0.337	0.348	0.359	**0.329**

Table C.7: Effect of CL.

	z_1	z_2	z_2^*
96	0.371	**0.364**	0.369
192	**0.396**	0.398	0.403
336	**0.411**	0.415	0.428
720	0.448	**0.425**	0.460
Avg.	0.407	**0.401**	0.415

Table C.8: Representation for downstream tasks.

Performance of various pretrain tasks. In addition to the 1) PD task of reconstructing the masked patches (X_m) and 2) PI task of autoencoding the unmasked patches (X_u), we also employ two other basic tasks for comparison: 3) predicting X_u from zero-filled patches ($\mathbf{0}$) and 4) autoencoding $\mathbf{0}$. Table C.6 displays the average MSE on four ETT datasets across four horizons, highlighting that the model pretrained with the PD task performs even worse than the two basic tasks with $\mathbf{0}$ as inputs. This emphasizes the ineffectiveness of the PD task and the effectiveness of the proposed PI task.

Which representation to use for downstream tasks? In SSL, the boundary of the encoder and the task-specific projection head is often unclear. To determine location to extract representation for downstream tasks, we conduct experiments using representations from intermediate layers in MLP: 1) $\boldsymbol{z}_1$ from the first layer, 2) $\boldsymbol{z}_2$ from the second layer, and 3) $\boldsymbol{z}_2^*$ from the additional projection layer attached on top of the second layer. Table C.8 displays the MSE of ETTh1 across four horizons, indicating that the second layer $\boldsymbol{z}_2$ yields the best results.

Location of complementary CL. To assess the effect of complementary CL together with PI reconstruction, we conduct an ablation study on the choice of pretext tasks and their location in the MLP encoder: the contrastive and/or reconstruction loss is computed on the first or second layer, or neither. Table C.7 displays the average MSE on four ETT datasets across four horizons. We observe that the PI reconstruction task is essential, and CL is effective when it is considered in the first layer.

Comparison with PatchTST. PITS can be derived from PatchTST, by changing the pretraining task and encoder architecture. Table C.9 shows how each modification contributes to the performance improvement on the ETTh1 dataset. Note that we apply minor modifications including mask ratio of 50% and mean normalization to PatchTST, which does not affect the performance (marked with *).

1) Encoder Architecture		
Transformer	Linear	MLP
0.425*	0.408	0.418
2) PD task → PI task		
0.415	0.408	0.407
3) + Complementary CL		
-	-	**0.401**

Table C.9: PatchTST→PITS

C.5 Analysis

PI task is more robust to distribution shift than PD task. To assess the robustness of pretraining tasks to distribution shifts, which are commonly observed in real-world datasets [14], we generate toy examples with varying trends and seasonality, depicted in the left panel of Figure C.1. The right panel of Figure C.1 visualizes the performance gap between the models trained with PD task and PI task, where the x- and y-axis correspond to the slope and amplitude differences between the training and test phases, respectively. The result indicates that the model trained with PI task exhibits better robustness to distribution shifts, where the gap increases as the shift becomes more severe.

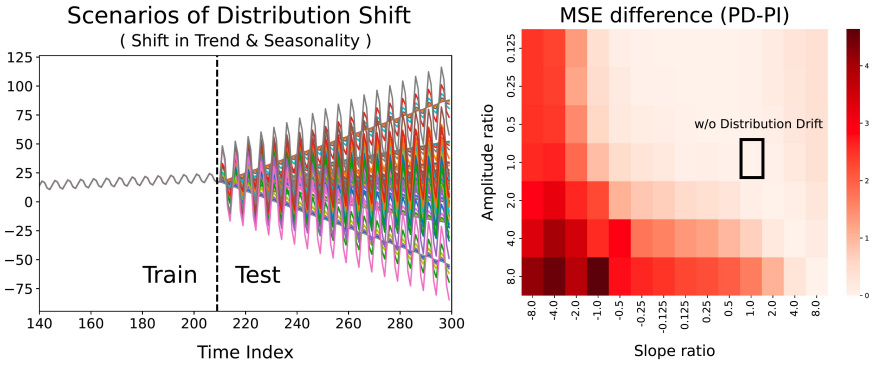

Figure C.1: PI vs. PD tasks under distribution shifts.

MLP is more interpretable than Transformer. While PI architectures process each patch independently, PD architectures share information from all patches, leading to information leaks among patches. This makes MLP more interpretable than Transformer, as visualizing the weight matrix of the linear layer additionally introduced and learned for the downstream task shows each patch's contribution to predictions. Figure C.2 illustrates the seasonality of ETTm1 and the downstream weight matrix trained on ETTm1 for both architectures. While the weight matrix of the linear layer on top of Transformer is mostly uniform, that of MLP reveals seasonal patterns and emphasizes recent information, highlighting that MLP captures the seasonality better than Transformer.

t-SNE visualization. To evaluate the quality of TS representations obtained from PI and PD tasks, we utilize t-SNE [24] for visualization. For this analysis, we create toy examples with 10 classes, each exhibiting its own trend and seasonality patterns, as shown in the top panel of Figure C.3. The results, visualized in Figure C.3, demonstrate that representations learned from the PI task exhibit a better ability to distinguish between classes.

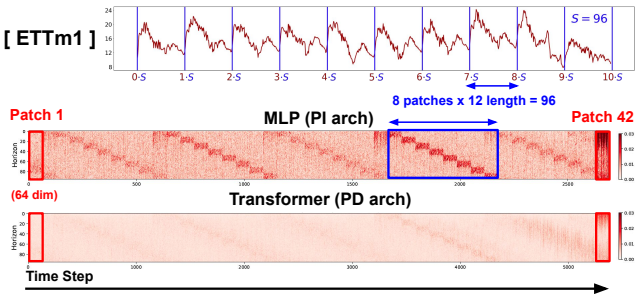

Figure C.2: Downstream task weight $W \in \mathbb{R}^{H \times N \cdot D}$.

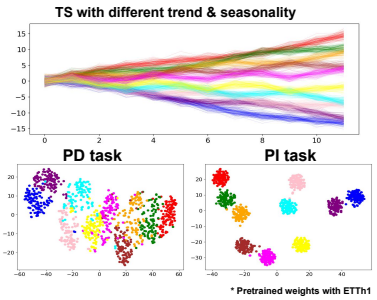

Figure C.3: t-SNE visualization.

D Transfer Learning

For time series forecasting under transfer learning, we consider both in-domain and cross-domain transfer learning settings, where we consider datasets with same frequency as in-domain. We perform transfer learning in both in-domain and cross-domain using five datasets: four ETT datasests and Weather. The full results are described in Table D.1, where missing values are not reported in literature.

Models			PITS						PatchTST						SimMTM		TimeMAE		TST		LaST		TF-C		CoST		TS2Vec	
			FT		LP		SL		FT		LP		SL															
source	target	horizon	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
In-domain	ETTh2 ↓ ETTh1	96	0.367	0.394	0.367	**0.392**	0.378	0.400	0.380	0.411	0.405	0.426	0.458	0.443	0.372	0.402	0.703	0.562	0.653	0.468	**0.362**	0.420	0.596	0.569	0.378	0.421	0.849	0.694
		192	**0.398**	**0.413**	**0.398**	**0.413**	0.419	0.420	0.419	0.436	0.433	0.443	0.514	0.472	0.414	0.425	0.715	0.567	0.658	0.502	0.426	0.478	0.614	0.621	0.424	0.451	0.909	0.738
		336	**0.416**	**0.427**	0.418	0.429	0.433	0.431	0.436	0.449	0.447	0.456	0.559	0.498	0.429	0.436	0.733	0.579	0.631	0.561	0.522	0.509	0.694	0.664	0.651	0.582	1.082	0.775
		720	0.435	0.455	**0.428**	**0.453**	0.459	0.464	0.457	0.474	0.572	0.540	0.507	0.490	0.446	0.458	0.762	0.622	0.638	0.608	0.460	0.478	0.635	0.683	0.883	0.701	0.934	0.769
		avg	0.404	0.423	**0.403**	**0.422**	0.422	0.429	0.423	0.443	0.464	0.466	0.510	0.476	0.415	0.430	0.728	0.583	0.645	0.535	0.443	0.471	0.635	0.634	0.584	0.539	0.944	0.744
	ETTm2 ↓ ETTm1	96	**0.292**	**0.348**	0.308	0.350	0.302	0.352	0.294	0.350	0.294	0.350	0.327	0.360	0.297	**0.348**	0.647	0.497	0.471	0.422	0.304	0.388	0.610	0.577	0.239	0.331	0.586	0.515
		192	**0.330**	**0.370**	0.339	**0.370**	0.342	0.377	0.333	0.371	**0.330**	0.372	0.393	0.398	0.332	**0.370**	0.597	0.508	0.495	0.442	0.429	0.494	0.725	0.657	0.339	0.371	0.624	0.562
		336	**0.355**	**0.384**	0.365	0.385	0.373	0.391	0.359	0.392	0.359	0.386	0.425	0.425	0.364	0.393	0.700	0.525	0.455	0.424	0.499	0.523	0.768	0.684	0.371	0.421	1.035	0.806
		720	**0.404**	0.411	0.406	**0.409**	0.422	0.420	0.407	0.414	0.406	0.415	0.500	0.473	0.410	0.431	0.786	0.596	0.498	0.532	0.422	0.450	0.927	0.759	0.467	0.481	0.780	0.669
		avg	**0.345**	**0.378**	0.354	0.379	0.359	0.386	0.348	0.382	0.347	0.381	0.411	0.414	0.351	0.383	0.682	0.531	0.480	0.455	0.414	0.464	0.758	0.669	0.354	0.401	0.756	0.638
Cross-domain	ETTm2 ↓ ETTh1	96	0.372	**0.398**	**0.370**	**0.398**	0.381	0.405	0.385	0.411	0.379	0.408	0.450	0.436	0.388	0.421	0.699	0.566	0.559	0.489	0.428	0.454	0.968	0.738	0.377	0.419	0.783	0.669
		192	0.407	0.425	**0.402**	**0.420**	0.417	0.429	0.425	0.439	0.414	0.430	0.504	0.466	0.419	0.423	0.722	0.573	0.600	0.579	0.427	0.497	1.080	0.801	0.422	0.450	0.828	0.691
		336	**0.417**	**0.442**	**0.417**	**0.442**	0.439	0.444	0.440	0.451	0.431	0.446	0.543	0.483	0.435	0.444	0.714	0.569	0.677	0.572	0.528	0.540	1.091	0.824	0.648	0.580	0.990	0.762
		720	**0.433**	**0.461**	**0.433**	**0.461**	0.480	0.488	0.482	0.488	0.460	0.476	0.523	0.502	0.468	0.474	0.760	0.611	0.694	0.664	0.527	0.537	1.226	0.893	0.880	0.699	0.985	0.783
		avg	0.407	0.431	**0.405**	**0.430**	0.429	0.441	0.433	0.447	0.421	0.440	0.505	0.472	0.428	0.441	0.724	0.580	0.632	0.576	0.503	0.507	1.091	0.814	0.582	0.537	0.896	0.726
	ETTh2 ↓ ETTm1	96	0.300	0.354	0.304	**0.346**	**0.294**	0.347	0.302	0.353	0.326	0.372	0.326	0.361	0.322	0.347	0.658	0.505	0.449	0.343	0.314	0.396	0.677	0.603	0.253	0.342	0.466	0.480
		192	0.335	0.372	0.335	**0.365**	**0.332**	0.367	0.342	0.375	0.354	0.386	0.371	0.392	**0.332**	0.375	0.594	0.511	0.477	0.407	0.587	0.545	0.718	0.638	0.367	0.392	0.557	0.532
		336	**0.361**	0.393	0.367	**0.383**	0.363	0.387	0.370	0.392	0.392	0.409	0.413	0.418	0.394	0.391	0.732	0.532	0.407	0.519	0.631	0.584	0.755	0.663	0.388	0.431	0.646	0.576
		720	**0.403**	0.417	0.423	**0.414**	0.420	0.419	0.439	0.426	0.440	0.434	0.486	0.460	0.411	0.424	0.768	0.592	0.557	0.523	0.468	0.429	0.848	0.712	0.498	0.488	0.752	0.638
		avg	**0.350**	0.384	0.357	**0.377**	0.352	0.380	0.363	0.387	0.378	0.400	0.399	0.408	0.365	0.384	0.688	0.535	0.472	0.448	0.475	0.489	0.750	0.654	0.377	0.413	0.606	0.556
	ETTm1 ↓ ETTh1	96	0.375	0.399	0.375	0.399	0.382	0.402	0.388	0.411	0.373	0.401	0.456	0.442	0.367	0.398	0.715	0.581	0.627	0.477	**0.360**	**0.374**	0.666	0.647	0.423	0.450	0.991	0.765
		192	0.411	0.421	0.409	0.421	0.417	0.421	0.422	0.431	0.408	0.423	0.520	0.482	0.396	0.421	0.729	0.587	0.628	0.500	**0.381**	**0.371**	0.672	0.653	0.641	0.578	0.829	0.699
		336	**0.415**	**0.432**	0.416	0.436	0.441	0.436	0.449	0.449	0.448	0.452	0.544	0.494	0.471	0.437	0.712	0.583	0.683	0.554	0.472	0.531	0.626	0.711	0.863	0.694	0.971	0.787
		720	**0.425**	**0.456**	0.428	0.459	0.451	0.461	0.530	0.513	0.499	0.492	0.532	0.507	0.454	0.463	0.747	0.627	0.642	0.600	0.490	0.488	0.835	0.797	1.071	0.805	1.037	0.820
		avg	**0.406**	**0.427**	0.407	0.428	0.422	0.430	0.447	0.451	0.432	0.442	0.513	0.481	0.422	0.430	0.726	0.595	0.645	0.533	0.426	0.441	0.700	0.702	0.750	0.632	0.957	0.768
	ETTh1 ↓ ETTm1	96	0.301	0.353	0.302	0.346	0.299	0.352	0.293	0.344	0.316	0.359	0.322	0.360	0.290	0.348	0.667	0.521	0.425	0.381	0.295	0.387	0.672	0.600	**0.248**	**0.332**	0.605	0.561
		192	0.341	0.377	0.334	**0.365**	0.334	0.371	**0.327**	0.366	0.351	0.378	0.388	0.399	**0.327**	0.372	0.561	0.479	0.495	0.478	0.335	0.379	0.721	0.639	0.336	0.391	0.615	0.561
		336	0.364	0.390	0.367	**0.384**	0.365	0.392	0.364	0.397	0.386	0.399	0.408	0.415	**0.357**	0.392	0.690	0.533	0.456	0.441	0.379	0.363	0.755	0.664	0.381	0.421	0.763	0.677
		720	**0.404**	0.417	0.424	**0.415**	0.424	0.419	0.409	0.417	0.441	0.430	0.491	0.464	0.409	0.423	0.744	0.583	0.554	0.477	0.403	0.431	0.837	0.705	0.469	0.482	0.805	0.664
		avg	0.353	0.384	0.357	**0.377**	0.356	0.384	0.348	0.381	0.374	0.392	0.402	0.410	**0.346**	0.384	0.666	0.529	0.482	0.444	0.353	0.390	0.746	0.652	0.359	0.407	0.697	0.616
	Weather ↓ ETTh1	96	**0.374**	**0.398**	**0.374**	**0.398**	0.379	0.401	0.386	0.409	0.384	0.401	0.469	0.444	0.477	0.444	-	-	-	-	-	-	-	-	-	-	-	-
		192	0.409	0.420	0.409	0.420	0.408	**0.419**	**0.405**	0.420	0.408	0.422	0.518	0.476	0.454	0.522	-	-	-	-	-	-	-	-	-	-	-	-
		336	**0.416**	**0.432**	**0.416**	**0.432**	0.421	0.436	0.448	0.454	0.421	0.438	0.551	0.497	0.424	0.434	-	-	-	-	-	-	-	-	-	-	-	-
		720	**0.427**	**0.456**	**0.427**	**0.456**	0.477	0.480	0.508	0.508	0.479	0.489	0.542	0.507	0.468	0.469	-	-	-	-	-	-	-	-	-	-	-	-
		avg	**0.407**	**0.427**	**0.407**	**0.427**	0.421	0.434	0.437	0.448	0.423	0.438	0.520	0.481	0.456	0.467	-	-	-	-	-	-	-	-	-	-	-	-
	Weather ↓ ETTm1	96	0.299	0.354	0.308	0.350	0.299	0.354	**0.292**	**0.347**	0.300	0.351	0.339	0.365	0.304	0.354	-	-	-	-	-	-	-	-	-	-	-	-
		192	0.336	0.376	0.336	**0.367**	0.342	0.384	**0.332**	0.373	0.336	0.372	0.381	0.395	0.338	0.375	-	-	-	-	-	-	-	-	-	-	-	-
		336	**0.358**	0.390	0.365	**0.384**	0.365	0.390	0.360	0.391	0.370	0.392	0.423	0.423	0.371	0.397	-	-	-	-	-	-	-	-	-	-	-	-
		720	0.412	0.420	0.414	**0.411**	0.418	0.421	**0.406**	0.421	0.413	0.425	0.506	0.466	0.417	0.426	-	-	-	-	-	-	-	-	-	-	-	-
		avg	0.351	0.386	0.356	**0.378**	0.356	0.388	**0.348**	0.383	0.355	0.385	0.412	0.412	0.358	0.388	-	-	-	-	-	-	-	-	-	-	-	-

Table D.1: **Results of multivariate TS forecasting with transfer learning.** We conduct experiments under two settings: (1) in-domain and (2) cross-domain transfer. The best results are in bold and the second best are underlined.

E Comparison with PatchTST

We compare our proposed method with PatchTST in three versions: 1) fine-tuning (FT), linear probing (LP), and supervised learning (SL). The results are described in Table E.1, which demonstrates that our proposed method outperforms PatchTST in every version in most of the datasets.

Models		PITS						PatchTST					
		FT		LP		SL		FT		LP		SL	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.364**	**0.392**	0.366	**0.392**	0.369	0.397	0.379	0.408	0.382	0.410	0.375	0.399
	192	**0.398**	**0.414**	**0.398**	**0.414**	0.402	0.416	0.414	0.428	0.433	0.441	0.414	0.421
	336	0.415	0.427	0.419	0.427	**0.409**	**0.425**	0.435	0.446	0.439	0.446	0.431	0.436
	720	**0.425**	**0.451**	0.430	0.454	0.457	0.465	0.468	0.474	0.482	0.482	0.449	0.466
	avg	**0.401**	**0.421**	0.403	0.422	0.409	0.426	0.424	0.439	0.434	0.445	0.417	0.431
ETTh2	96	**0.269**	**0.332**	**0.269**	0.333	0.281	0.343	0.306	0.351	0.299	0.350	0.274	0.336
	192	**0.326**	**0.372**	0.331	0.373	0.345	0.384	0.361	0.392	0.363	0.394	0.339	0.379
	336	0.354	0.396	0.352	0.395	0.343	0.389	0.405	0.427	0.386	0.417	**0.331**	**0.380**
	720	**0.378**	0.425	0.383	0.425	0.388	0.430	0.419	0.446	0.409	0.440	0.379	**0.422**
	avg	**0.331**	0.382	0.334	0.382	0.337	0.386	0.373	0.404	0.364	0.400	**0.331**	**0.379**
ETTm1	96	0.296	0.355	0.307	0.349	0.295	0.346	0.294	0.345	0.296	0.349	0.290	0.342
	192	**0.321**	**0.368**	0.337	**0.368**	0.331	0.369	0.327	0.369	0.333	0.370	0.332	0.369
	336	**0.353**	**0.388**	0.365	0.389	0.360	**0.388**	0.364	0.390	0.368	0.390	0.366	0.392
	720	**0.395**	**0.412**	0.415	**0.412**	0.416	0.420	0.409	0.415	0.422	0.418	0.420	0.424
	avg	**0.341**	0.381	0.356	**0.378**	0.350	0.381	0.349	0.380	0.355	0.382	0.352	0.382
ETTm2	96	0.163	0.255	**0.160**	**0.252**	0.163	0.255	0.167	0.256	0.168	0.257	0.165	0.255
	192	**0.213**	**0.289**	0.214	**0.289**	0.216	0.293	0.232	0.302	0.231	0.302	0.220	0.292
	336	**0.263**	**0.324**	**0.263**	**0.324**	0.267	0.328	0.291	0.342	0.290	0.341	0.278	0.329
	720	**0.337**	**0.373**	0.342	0.376	0.342	0.381	0.368	0.390	0.366	0.387	0.367	0.385
	avg	**0.244**	**0.310**	**0.244**	**0.310**	0.247	0.314	0.264	0.323	0.264	0.322	0.258	0.315
Weather	96	0.149	0.201	0.167	0.222	0.153	0.202	0.146	**0.194**	0.160	0.211	0.152	0.199
	192	0.195	0.241	0.211	0.259	**0.191**	0.242	0.192	**0.238**	0.203	0.248	0.197	0.243
	336	**0.244**	**0.280**	0.256	0.293	0.245	**0.280**	0.245	**0.280**	0.251	0.285	0.249	0.283
	720	0.312	**0.328**	0.319	0.338	**0.310**	0.329	0.320	0.336	0.319	0.334	0.320	0.335
	avg	**0.225**	**0.262**	0.239	0.278	**0.225**	0.263	0.226	**0.262**	0.233	0.269	0.230	0.265
Traffic	96	0.373	0.257	0.384	0.266	0.375	0.264	0.393	0.275	0.399	0.294	**0.367**	**0.251**
	192	0.388	0.266	0.395	0.270	0.389	0.270	**0.376**	**0.254**	0.412	0.298	0.385	0.259
	336	0.401	0.271	0.409	0.276	0.401	0.277	**0.384**	**0.259**	0.425	0.306	0.398	0.265
	720	0.436	0.290	0.438	0.295	0.437	0.294	0.446	0.306	0.460	0.323	**0.434**	**0.287**
	avg	0.399	0.271	0.406	0.277	0.401	0.276	0.400	0.274	0.424	0.305	**0.396**	**0.266**
Electricity	96	0.129	0.225	0.132	0.227	0.132	0.228	**0.126**	**0.221**	0.138	0.237	0.130	0.222
	192	**0.144**	0.240	0.147	0.241	0.147	0.242	0.145	**0.238**	0.156	0.252	0.148	0.240
	336	**0.160**	**0.256**	0.163	0.258	0.162	0.260	0.164	**0.256**	0.170	0.265	0.167	0.261
	720	**0.197**	**0.290**	0.201	**0.290**	0.199	**0.290**	0.200	**0.290**	0.208	0.297	0.202	0.291
	avg	**0.157**	0.253	0.161	0.252	0.160	0.255	0.159	**0.251**	0.168	0.263	0.162	0.254
Average		**0.300**	**0.326**	0.306	0.328	0.304	0.329	0.314	0.333	0.320	0.341	0.307	0.327

Table E.1: PITS vs. PatchTST in multivariate time series forecasting.

F Effectiveness of PI Task and Contrastive Learning

To assess the effectiveness of the proposed patch reconstruction task and complementary contrastive learning, we conduct ablation studies in both time series forecasting and time series classification.

F.1 Time Series Forecasting

To examine the effect of PI task and CL on forecasting, we conduct an experiment using four ETT datasets. The results in Table F.1 demonstrate that performing CL with the representation obtained from the first layer and PI with the one from the second layer gives the best performance.

Layer 1		-	-	-	PI	CL
Layer 2		CL	PI	CL+PI	CL	PI
ETTh1	96	0.715	0.367	0.372	0.381	**0.364**
	192	0.720	0.400	0.409	0.416	**0.398**
	336	0.719	0.426	0.422	0.462	**0.415**
	720	0.727	0.443	0.465	0.509	**0.425**
	avg	0.720	0.409	0.417	0.442	**0.401**
ETTh2	96	0.373	0.270	0.307	0.303	**0.269**
	192	0.384	0.331	0.362	0.373	**0.326**
	336	0.386	0.361	0.387	0.391	**0.354**
	720	0.432	0.384	0.408	0.416	**0.378**
	avg	0.394	0.336	0.366	0.371	**0.331**
ETTm1	96	0.693	0.305	0.302	0.300	**0.296**
	192	0.702	0.335	0.337	0.336	**0.321**
	336	0.716	0.366	0.365	0.369	**0.353**
	720	0.731	0.413	0.413	0.426	**0.395**
	avg	0.711	0.355	0.356	0.358	**0.341**
ETTm2	96	0.346	**0.160**	0.167	0.171	0.163
	192	0.368	0.215	0.225	0.235	**0.213**
	336	0.397	0.266	0.274	0.278	**0.263**
	720	0.424	0.346	0.351	0.376	**0.337**
	avg	0.381	0.247	0.254	0.265	**0.244**
Total avg		0.552	0.337	0.348	0.359	**0.330**

Table F.1: Effect of PI task and CL on time series forecasting.

F.2 Time Series Classification

To evaluate the impact of employing CL and PI on classification, we conducted an experiment using the SleepEEG dataset. The results presented in Table F.2 demonstrate that as long as PI task is employed, the performance is robust to the design choices.

Layer 1		-	-	-	PI	CL
Layer 2		CL	PI	CL+PI	CL	PI
SleepEEG	ACC.	91.61	95.27	**95.67**	**95.67**	**95.67**
	PRE.	92.11	95.35	95.63	**95.70**	95.63
	REC..	91.61	95.27	95.66	95.66	**95.67**
	F1..	91.79	95.30	**95.68**	**95.68**	95.64

Table F.2: Effect of PI task and CL on time series classification.

G Effectiveness of PI Strategies

In this experiment, we investigate the impact of our proposed PI strategies from two perspectives: 1) the pretraining task and 2) the encoder architecture. The results, shown in Table G.1, encompass four ETT datasets with four different forecasting horizons. These results demonstrate that the PI task consistently outperforms the conventional PD task across all considered architectures.

Architecture		PI				PD			
		Linear		MLP		MLPMixer		Transformer	
Task		PD	PI	PD	PI	PD	PI	PD	PI
ETTh1	96	0.366	0.365	0.375	0.366	0.378	0.368	0.371	0.372
	192	0.398	0.398	0.407	0.397	0.414	0.399	0.410	0.404
	336	0.423	0.424	0.427	0.427	0.422	0.427	0.443	0.434
	720	0.444	0.444	0.463	0.440	0.465	0.440	0.475	0.452
	avg	**0.408**	**0.408**	0.418	**0.407**	0.420	**0.409**	0.425	**0.415**
ETTh2	96	0.272	0.270	0.290	0.270	0.301	0.276	0.283	0.271
	192	0.332	0.333	0.361	0.329	0.353	0.334	0.351	0.332
	336	0.370	0.364	0.373	0.353	0.394	0.363	0.378	0.369
	720	0.396	0.385	0.418	0.384	0.411	0.389	0.400	0.395
	avg	0.343	**0.338**	0.361	**0.334**	0.365	**0.341**	0.353	**0.342**
ETTm1	96	0.304	0.304	0.298	0.302	0.294	0.296	0.294	0.297
	192	0.337	0.338	0.341	0.337	0.332	0.334	0.335	0.336
	336	0.370	0.368	0.368	0.363	0.364	0.363	0.365	0.359
	720	0.423	0.423	0.416	0.420	0.418	0.416	0.405	0.403
	avg	0.359	**0.358**	0.356	**0.355**	0.354	**0.352**	**0.350**	**0.350**
ETTm2	96	0.163	0.163	0.169	0.164	0.170	0.164	0.172	0.172
	192	0.219	0.218	0.224	0.218	0.226	0.218	0.240	0.221
	336	0.272	0.271	0.275	0.271	0.276	0.272	0.300	0.274
	720	0.362	0.361	0.363	0.359	0.361	0.359	0.383	0.356
	avg	0.254	**0.253**	0.258	**0.253**	0.259	**0.253**	0.274	**0.256**
Total avg		0.341	**0.339**	0.348	**0.337**	0.350	**0.339**	0.351	**0.341**

Table G.1: Effectiveness of PI tasks and PI architectures.

H Robustness to Patch Size

To evaluate the robustness of encoder architectures to patch size, we compare MLP and Transformer with different patch sizes with ETTh2 and ETTm2. The left and the right panel of Figure H.1 illustrate the average MSE of four horizons of ETTh2 and ETTm2, respectively.

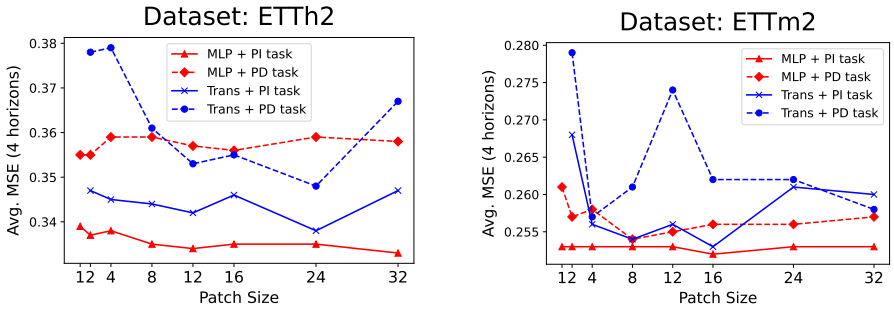

Figure H.1: Robustness of PI task to patch size.

I Performance by Dropout Rate

Figure I.1 displays the average MSE across four horizons, and Table I.1 lists all the MSE values for four ETT datasets trained with MLP of $D = 32$ at various dropout rates. These results emphasize

the importance of incorporating dropout during the pretraining phase of the reconstruction task, as it helps prevent trivial solutions when the hidden dimension is greater than the input dimension.

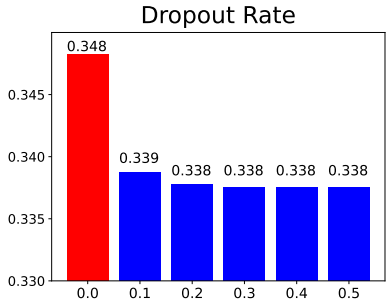

Figure I.1: Avg. MSE by dropout.

Dropout rate	ETTh1	ETTh2	ETTm1	ETTm2	Avg.
0.0	0.416	0.358	0.360	0.253	0.347
0.1	0.410	0.334	0.358	0.253	0.339
0.2	0.407	0.334	0.357	0.253	0.338
0.3	0.407	0.333	0.357	0.253	0.338
0.4	0.407	0.334	0.356	0.253	0.338
0.5	0.406	0.335	0.356	0.253	**0.337**

Table I.1: MSE by dropout.

J Performance of Various Pretrain Tasks

To see if the conventional PD task of reconstructing the masked patches (X_m) with the unmasked patches (X_u) is appropriate for TS representation learning, we employ two other simple pretraining tasks of 1) predicting X_u with zero-value patches ($\mathbf{0}$) and 2) reconstructing $\mathbf{0}$ with themselves. Table J.1 presents the results for four ETT datasets across three different architectures: Transformer, MLP without CL, and MLP with CL. These results underscore that models pretarined with PD task performs even worse than the two basic pretraining tasks with zero-value patch inputs, highlighting the ineffectiveness of the PI task and emphasizing the importance of the proposed PI task.

Pretrain Task		Transformer					MLP									
							w/o CL					w/ CL				
Input	Output	ETTh1	ETTh2	ETTm1	ETTm2	avg	ETTh1	ETTh2	ETTm1	ETTm2	avg	ETTh1	ETTh2	ETTm1	ETTm2	avg
X_u	X_u	0.415	0.342	0.350	0.256	**0.341**	0.407	0.334	0.357	0.253	**0.338**	0.401	0.331	0.341	0.244	**0.329**
X_u	X_m	0.425	0.353	0.350	0.274	0.351	0.418	0.361	0.356	0.258	0.348	0.457	0.376	0.362	0.261	0.364
$\mathbf{0}$	X_u	0.410	0.350	0.349	0.260	0.342	0.418	0.361	0.354	0.256	0.348	0.418	0.361	0.353	0.256	0.348
$\mathbf{0}$	$\mathbf{0}$	0.413	0.360	0.342	0.257	0.343	0.418	0.356	0.352	0.253	0.345	0.418	0.356	0.353	0.254	0.345

Table J.1: Performance of various pretrain tasks.