Enhancing Masked Time-Series Modeling via Dropping Patches

Tianyu Qiu[1], Yi Xie[1*], Hao Niu[1], Yun Xiong[1*], Xiaofeng Gao[2]

[1]Shanghai Key Lab of Data Science, School of Computer Science, Fudan University, Shanghai, China
[2]MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai, China
{tyqiu22, yixie18, hniu18, yunx}@fudan.edu.cn
gao-xf@cs.sjtu.edu.cn

Abstract

This paper explores how to enhance existing masked time-series modeling by randomly dropping sub-sequence level patches of time series. On this basis, a simple yet effective method named DropPatch is proposed, which has two remarkable advantages: 1) It improves the pre-training efficiency by a square-level advantage; 2) It provides additional advantages for modeling in scenarios such as in-domain, cross-domain, few-shot learning and cold start. This paper conducts comprehensive experiments to verify the effectiveness of the method and analyze its internal mechanism. Empirically, DropPatch strengthens the attention mechanism, reduces information redundancy and serves as an efficient means of data augmentation. Theoretically, it is proved that DropPatch slows down the rate at which the Transformer representations collapse into the rank-1 linear subspace by randomly dropping patches, thus optimizing the quality of the learned representations.

Extended version — https://arxiv.org/abs/2412.15315

Introduction

In recent years, masked modeling has emerged as a prevalent self-supervised method in various fields, including natural language processing (Devlin et al. 2018; Liu et al. 2019) and computer vision (Baevski et al. 2022; He et al. 2022; Bao et al. 2021). This technique improves representation learning by reconstructing masked content based on unmasked parts. Masked modeling has also been adapted for time-series analysis. A notable advancement involves segmenting time-series into patches (sub-sequence) and applying a patch-level masking strategy, which has received considerable attention since its inception (Nie et al. 2022). This method not only shows promising performance in transfer learning, but also significantly enhances supervised forecasting by employing self-supervised pre-training to initialize model parameters, consistent with recent findings (Amos, Berant, and Gupta 2023). Building upon the patching technique, numerous time-series foundation model works have emerged and achieve significant performance in time-series forecasting (Goswami et al. 2024; Woo et al. 2024).

*Corresponding author.

Despite its potential, we observed that masked time-series modeling, represented by PatchTST (Nie et al. 2022), faces a dilemma. A relatively low mask ratio reduces effectiveness in learning useful features (He et al. 2022; Zhang, Wang, and Wang 2022). Given the characteristic of periodicity and repetitive pattern of time-series data, the masked patch can be recovered with little high-level understanding of the underlying patterns, leading to superficial learning and over-fitting as shown in Figure 1 (A). A natural idea is to increase the mask ratio, but another issue emerges: the presence of an excessive number of masked patches can further dilute the attention mechanism's capacity to concentrate on the relevant and informative parts of data, termed as scattered attention as shown in Figure 1 (C). It can lead to the degradation of downstream task performance as the representations gradually lose their distinctiveness (Noci et al. 2022; Dong, Cordonnier, and Loukas 2021; Zhai et al. 2023).

We introduce a simple yet effective strategy, **DropPatch**, to encourage learning useful features and improve the overall performance. Building on foundational time-series pre-training techniques (Nie et al. 2022), DropPatch randomly removes a predefined proportion of patches. The remaining patches are subsequently processed for masking and reconstruction. It is crucial to distinguish between dropping and masking in the context of pre-training. For a given time-series sample, the dropping operation is applied prior to masking and reconstruction. Removed patches are entirely excluded from all training steps during the current epoch. In contrast, masked patches, represented as zero tensors overlaid with positional encoding, remain part of the training process throughout the epoch.

In our empirical study, DropPatch demonstrates clear advantages in mitigating over-fitting (Figure 1 (B)), enhancing attention focus (Figure 1 (C)), and improving forecasting performance (Figure 1 (D)). The reduction in the number of patches due to the dropping operation leads to significant improvements in computational efficiency and reduced memory consumption.

Extensive experiments validate the effectiveness of DropPatch. Through detailed experimental analysis, we uncover the underlying mechanisms driving these improvements. The DropPatch strategy enhances the attention mechanism by enabling a sharper focus on multi-scale and diverse information. It strengthens the model's ability to capture crit-

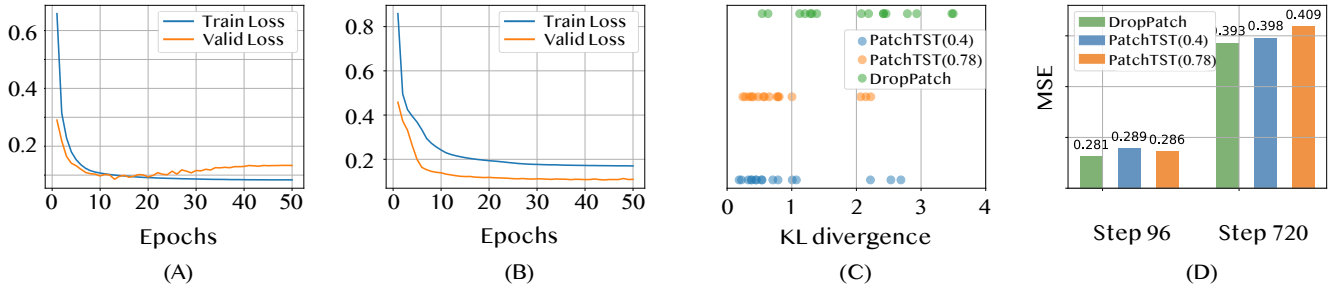

Figure 1: (A) The loss curve of PatchTST with mask ratio 0.4 (official implementation); (B) The loss curve of DropPatch (unless otherwise stated, the drop ratio and mask ratio is 0.6 and 0.4 throughout this paper); (C) The Kullback-Leibler (KL) divergence between the attention coefficients of the final encoder layer and a uniform distribution, where each dot represents an individual attention head. A larger KL divergence indicates that this set of attention distributions is farther from a uniform distribution and thus more focused. PatchTST(0.78) refers to the PatchTST configured with a mask ratio of 0.78, matching the number of visible patches in DropPatch. (D) Comparison of MSE metrics between PatchTST and DropPatch with forecasting steps $T \in \{96, 720\}$ on ETTm1.

ical patterns while reducing redundancy in representation. Furthermore, our theoretical findings indicate that the random dropping of patches effectively slows the convergence of the Transformer's representations toward a rank-1 linear subspace, thereby promoting the feature diversity.

Overall, our contributions can be summarized as follows:

- We introduce DropPatch, a simple yet effective strategy that enhances masked time-series modeling.
- Extensive experiments demonstrate that the DropPatch strategy improves pre-training efficiency and delivers substantial performance gains across diverse downstream tasks. Additionally, we compile comprehensive synthesized datasets to evaluate its role as a core component in foundational models for time-series analysis.
- Through rigorous empirical and theoretical analysis, we validate the effectiveness of DropPatch and provide insights into the mechanisms driving these improvements.

Method

In this section, we describe the details of our proposed pre-training method, DropPatch, as shown in Fig. 2. It is worth noting that DropPatch is a strategy applied during the pre-training stage, and the model does not perform the dropping operation during the fine-tuning stage.

Patching and Channel-Independence

For each sample of multivariate time-series $\mathbf{X} \in \mathbb{R}^{L \times C}$, where L represents the length of time-series, and C denotes the number of channels (variates). We first split the entire time-series sample into non-overlapping subseries-level patches, which are served as input tokens to Transformer, like PatchTST (Nie et al. 2022). We permute the original data of time-series into $\mathcal{X} \in \mathbb{R}^{C \times P \times L_P}$, where L_P denotes the length of each subseries-level patch, and P denotes the total number of patches.

Dropping Patches

After the patching operation, we will first conduct the positional encoding for these patches. The positional encoding process is designed to preserve the positional information during the self-attention computation and following the dropping operation. It should be noted that the positional encoding of each token is computed prior to dropping operation, ensuring that the original sequence position of each token is maintained after the removal.

We randomly drop patches in the patched time-series, which is the core idea of our proposed DropPatch. Let r denotes the ratio of dropping with condition $0 \leq r \leq 1$, implying that only $(1-r)P$ patches remains for further training and others will be directly absent in the subsequent operations. Formally, the remained patches and positional encoding will be denoted as $\bar{\mathcal{X}} \in \mathbb{R}^{C \times (1-r)P \times L_P}$, $\overline{PE} \in \mathbb{R}^{C \times (1-r)P \times d_{model}}$.

Representation Learning

Subsequently, a patch-level random masking strategy is applied to generate masked data, the resultant masked data can be expressed as $\bar{\mathcal{X}}_{masked} \in \mathbb{R}^{C \times (1-r)P \times L_P}$. Given a mask ratio $m \in [0, 1]$, we denote that the number of masked patches is $(1-r)\,mP$.

The masked data is then embedded, and the previously dropped positional encodings are added back to these embeddings to formulate the encoder input $\mathbf{E}$. After the encoder, we can obtain the representation $\mathbf{Z}$ of the input series which can be formalized as:

$$\mathbf{E} = Embed(\bar{\mathcal{X}}_{masked}) + \overline{PE}, \tag{1}$$

$$\mathbf{Z} = Encoder(\mathbf{E}), \tag{2}$$

where $\mathbf{E}, \mathbf{Z} \in \mathbb{R}^{C \times (1-r)P \times d_{model}}$. Finally, the representation $\mathbf{Z}$ is fed into a reconstruction head to obtain the reconstruction results $\hat{X} \in \mathbb{R}^{C \times (1-r)P \times L_P}$. In the implementation, we simply adopt a linear layer as the head. We choose

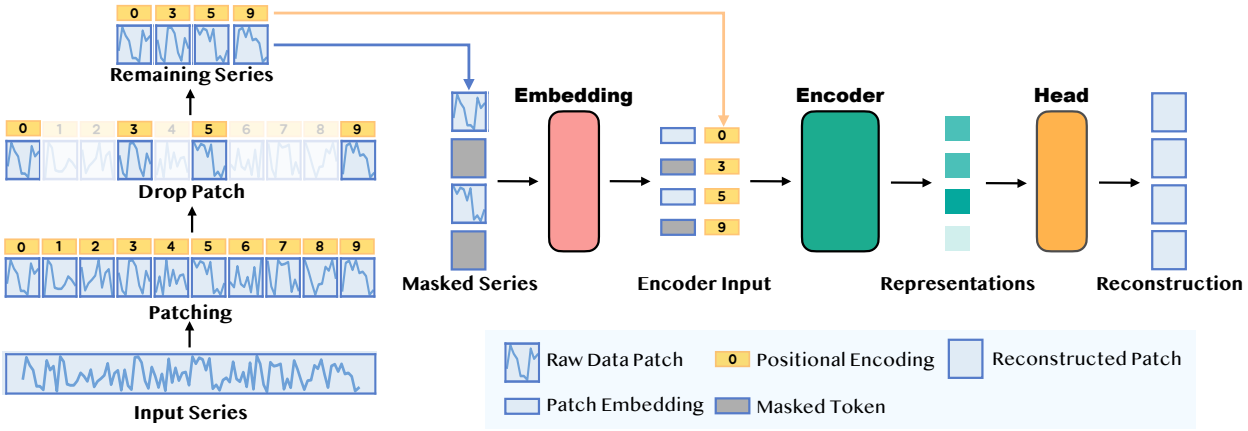

Figure 2: The overall pre-training framework of DropPatch.

to use the Mean Squared Error (MSE) loss to mesure the reconstruction and the ground truth. Only the reconstructions on the masked patches are considered in the loss.

Here, we present a corollary to describe from the perspective of representation space why DropPatch is effective, which will be validated through both experimental and theoretical approaches in the following text. The theoretical analysis is provided in the Appendix.

Lemma 1. *Let* SAN *denote a self-attention layer, and consider stacking L such layers. Then, under certain conditions, the representations within the stacked self-attention layers will converge to a rank-1 matrix as $L \to \infty$.*

Corollary 1. *The DropPatch strategy effectively slows down the rate at which the representation matrix of a Transformer degenerates into a rank-1 matrix.*

Experiments

We evaluate the effectiveness of our proposed method on time-series forecasting tasks under various setups, including in-domain, cross-domain, few-shot, cold-start, and multi-dataset pre-training scenarios. It is worth noting that we maintain consistent drop ratio and mask ratio to be fixed across various tasks and datasets, demonstrating the effectiveness and robustness of our approach. Our proposed DropPatch[1] exhibits significant improvement over other established strong baselines in various time-series forecasting scenarios, while enjoying the computational efficiency and reduced memory usage.

Datasets We evaluate performance of our proposed method DropPatch on 12 popular datasets. For in-domain, cross-domain and few-shot experiments, Weather, ECL, Traffic and 4 ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2) are included. In addition, we incorporate Exchange and PEMS dataset for cold start scenario in cross-domain transfer learning. All datasets are available on (Wu et al. 2021) (Liu et al. 2022). Moreover, we compile two synthesized datasets to conduct multi-dataset pre-training (Goswami et al. 2024), demonstrating the potential of DropPatch strategy in time-series foundation model.

Implementation We choose seven competitive self-supervised baseline methods, including the masked modeling method: PatchTST (Nie et al. 2022), SimMTM (Dong et al. 2024), Ti-MAE (Cheng et al. 2023), TST (Zerveas et al. 2021), the contrastive learning methods: LaST (Wang et al. 2022), CoST (Woo et al. 2022), TS2Vec (Yue et al. 2022). We also include supervised methods iTransoformer (Liu et al. 2023), DLinear (Zeng et al. 2023) and FEDformer (Zhou et al. 2022) in comparison with the cross-domain transfer results of DropPatch and PatchTST. We denote that *PatchTST* refer to the self-supervised version PatchTST. We conduct experiments in both in-domain and cross-domain settings. For the in-domain setting, we pre-train and fine-tune the model using the same dataset. In the cross-domain setting, we pre-train the model on one dataset and then fine-tune it on other target datasets to evaluate its adaptability and generality across diverse scenarios. Unless otherwise stated, the input sequence length of DropPatch is set to 512, and the patch length is fixed at 12 following the self-supervised PatchTST (Nie et al. 2022). Implementation details are provided in the Appendix.

In-Domain Forecasting

We conduct time-series forecasting experiments under an in-domain setting, where models are pre-trained and fine-tuned on the same datasets. The results are summarized in Table 1. Full results are presented in the Appendix.

In-domain experiments show that our DropPatch strategy surpasses existing methods in 13 out of 14 metrics across 7 datasets. Each metric demonstrates significant superiority in comparison with other baselines. PatchTST is noted as a strong baseline. Nevertheless, by simply applying the DropPatch strategy, performance is further improved in both MSE and MAE, with only half the time consumption and memory usage in pre-training stage.

The forecasting performance of PatchTST, SimMTM,

[1]In this section, we refer to DropPatch as DropPatch strategy implemented on top of the PatchTST backbone

Models	**DropPatch**		PatchTST		SimMTM		Ti-MAE		TST		LaST		CoST		TS2Vec	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.336**	**0.378**	0.341	0.379	0.340	0.379	0.682	0.532	0.494	0.471	0.383	0.399	0.477	0.486	0.664	0.689
ETTm2	**0.254**	**0.315**	0.258	0.318	0.260	0.318	0.392	0.417	0.425	0.371	0.389	0.394	0.825	0.651	0.359	0.420
ETTh1	**0.400**	0.429	0.430	0.445	0.404	**0.428**	0.721	0.591	0.624	0.562	0.571	0.532	0.710	0.627	0.643	0.728
ETTh2	**0.347**	**0.390**	0.355	0.394	0.348	0.391	0.482	0.488	0.429	0.458	0.499	0.497	1.664	0.999	0.801	0.856
Weather	**0.220**	**0.259**	0.225	0.261	0.235	0.280	0.324	0.343	0.419	0.448	0.237	0.268	1.111	0.801	0.658	0.751
ECL	**0.157**	**0.249**	**0.157**	0.252	0.162	0.356	0.561	0.554	0.310	0.353	0.186	0.274	0.228	0.335	0.354	0.427
Traffic	**0.378**	**0.257**	0.382	0.259	0.392	0.264	0.916	0.423	0.611	0.503	0.713	0.397	0.760	0.428	0.501	0.375

Table 1: In-domain time-series forecasting results, averaged from all forecasting steps $T \in \{96, 192, 336, 720\}$.

and DropPatch is significantly superior to other baselines. The commonality among these three methods is the use of channel-independent masked time-series modeling.

Compared to PatchTST, the DropPatch strategy offers further improvements in this task. This is primarily because the masked time-series modeling task can be done with a little understanding of underlying patterns in the time-series, which can lead to superficial learning and over-fitting. Random dropping introduces a significant amount of randomness to each sample, thus acting as a data augmentation method that helps mitigate the over-fitting issue. In the meanwhile, the challenging pre-training task requires a comprehensive understanding of underlying patterns and thus encourages the learning of useful representation.

Cross-Domain Forecasting

In this section, we explore multiple scenarios in cross-domain transfer learning. We perform fine-tuning on target datasets using all available training samples. Specifically, we conduct experiments with 1) ECL as the fixed source dataset, following the setup in (Nie et al. 2022), and 2) ETTm1 as the fixed target dataset. The results are summarized in Table 2 3. Full results are presented in the Appendix. Notably, when the source dataset has a mismatch in the number of channels compared to the target dataset, some baseline models are unable to perform the transfer. Although SimMTM is capable of transferring under conditions of channel mismatch, we encountered an out-of-memory (OOM) issue when pre-training SimMTM on the ECL dataset, even with a batch size of 1. Therefore, we also include supervised models for comparison when using ECL as the source dataset.

From the comparison, we observe that DropPatch significantly surpasses the other baselines. Notably, while PatchTST falls behind some supervised methods, DropPatch consistently outperforms these supervised methods. The improved performance stems from the prevention of severe over-fitting in the source dataset, ensuring the model's robustness and generalization capability when applied to unseen target datasets. In contrast, over-fitting can hinder PatchTST's ability to generalize effectively to new patterns.

Evaluations on Synthesized Dataset

In the cross-domain experiments mentioned above, the models are initially pre-trained on a single source dataset and then fine-tuned on a target dataset. For the purpose of developing time-series foundation models (Goswami et al. 2024; Woo et al. 2024; Liu et al. 2024), the source dataset could be a mixed dataset. In the mixed dataset, time-series samples are from different domains, exhibiting varying frequencies, and containing diverse semantic information. This setup aims to enhance the model's robustness and ability to generalize across different scenarios, while also posing a challenge for models to handle diverse data.

We compile two synthesized datasets to facilitate multi-dataset pre-training for evaluation. This section primarily focuses on exploring the potential of applying DropPatch to time-series foundation models, without the concern with pushing state-of-the-art results.

Specifically, we merge 10 datasets to compile a synthesized time-series dataset, named STS66M, which has a total file size of over 66 MB and consists of more than 3.76 million data points. The models are pre-trained on STS66M and subsequently fine-tuned on other target datasets. The results are presented in Table 4. DropPatch significantly outperforms PatchTST, demonstrating its superior adaptability to diverse pre-training data and its ability to learn more robust and general representations for downstream tasks.

An important application of pre-trained models is to provide priori knowledge for downstream datasets, particularly in scenarios with limited fine-tuning data availability, commonly referred to as few-shot learning. This capability is crucial for the fast adaptation of deep models, which has been demonstrated remarkable performance in NLP (Brown et al. 2020; Achiam et al. 2023). To further explore this, we expand the size of our synthesized time-series dataset by including ECL and PEMS07. The expanded dataset has a file size over 162MB, named STS162M, consisting of 32.5 million data points. We then conduct few-shot learning experiments using models pre-trained on STS162M. The results are presented in Table 5. For each unseen target dataset, we employ only the headmost 100, 300, and 500 training samples to evaluate DropPatch and PatchTST. DropPatch can generalize well and achieve improved performance.

Models	**DropPatch**		PatchTST		iTransformer		DLinear		FEDformer	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL→ETTm1	0.349	0.383	**0.346**	0.383	0.371	0.400	0.357	**0.379**	0.382	0.422
ECL→ETTm2	0.258	0.321	**0.257**	**0.318**	0.272	0.333	0.267	0.332	0.292	0.343
ECL→ETTh1	**0.395**	**0.426**	0.434	0.448	0.451	0.462	0.423	0.437	0.428	0.454
ECL→ETTh2	**0.350**	**0.392**	0.354	0.395	0.387	0.418	0.431	0.447	0.388	0.434
ECL→Weather	**0.222**	**0.260**	0.226	0.264	0.246	0.279	0.246	0.300	0.310	0.357
ECL→Traffic	**0.379**	**0.257**	0.411	0.285	0.380	0.271	0.434	0.295	0.604	0.372

Table 2: Cross-domain time-series forecasting results. ECL→ETTm1 denotes the models are pre-trained on ECL and then are fine-tuned on ETTm1. iTransformer, DLinear, and FEDformer are trained directly on the target dataset using supervised learning. Results are averaged from all forecasting steps $T \in \{96, 192, 336, 720\}$.

Models	**DropPatch**		PatchTST		SimMTM		Ti-MAE		TST		LaST		TF-C		CoST		TS2Vec	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1→ETTm1	0.352	0.386	0.352	0.386	**0.346**	**0.384**	0.666	0.529	0.482	0.444	0.353	0.390	0.746	0.562	0.359	0.407	0.697	0.616
ETTh2→ETTm1	**0.361**	0.390	0.364	0.391	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
ETTm2→ETTm1	**0.343**	**0.382**	0.353	0.390	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
Weather→ETTm1	**0.348**	**0.385**	0.359	0.390	0.358	0.388	-		-		-		-		-		-	

Table 3: Cross-domain time-series forecasting results. ETTh1→ETTm1 denotes the models are pre-trained on ETTh1 and then are fine-tuned on ETTm1. Results are averaged from all forecasting steps $T \in \{96, 192, 336, 720\}$. Notation "−" means transfer learning is not feasible due to the mismatch in the number of channels.

Cold Start

This task aims to forecast in target datasets where lookback length L_{ft} is relatively short, providing limited historical information for fine-tuning. The experimental setup was first introduced in time-series forecasting by (Jin et al. 2022). In our experiments, the lookback length is fixed at $L_{ft} = 96$, which is shorter than the lookback length $L_{pt} = 512$ on the pre-training stage. We perform experiments on Exchange and four PEMS(PEMS03, PEMS04, PEMS07, PEMS08) as the target datasets. The source dataset is fixed as ECL. Forecasting steps $T \in \{96, 192, 336, 720\}$ for Exchange and $T \in \{12, 24, 48, 96\}$ for the PEMS datasets. Under cold start scenario, the pre-trained models are expected to leverage the limited historical information for future forecasting. In Table 6, we present the averaged results across the target dataset. Full results are presented in the Appendix.

Model Efficiency

We compared the training speed and memory usage during the pre-training stage, results are presented in Table 7. All experiments are conducted on a single NVIDIA Tesla V100-SXM2-32GB GPU. In comparison with the other two leading masked time-series modeling methods, DropPatch significantly reduces the memory usage and training time consumption by a large margin. This computational efficiency makes it feasible to scale up and potentially improve model performance by exposing the model to a larger dataset.

Discussion

Since its inception, the self-supervised PatchTST, which employs a patch-level masking pre-training paradigm, has consistently achieved state-of-the-art performance. Our proposed method DropPatch improves upon this by dropping a certain proportion of patches prior to applying the patch-level masking strategy, resulting in superior performance in both in-domain and cross-domain scenarios. This raises several questions: How does DropPatch strategy differ from PatchTST, and what drives its enhanced performance?

We will provide a brief description and present the findings for each empirical study. Similar results are observed across various datasets; results on ETTm1 are displayed here as a representative example. Unless otherwise specified, the experiments are conducted in an in-domain scenario using the ETTm1 dataset.

Normalized Attention Distance

Firstly, we analyze the averaged attention distances before and after applying the DropPatch strategy. Specifically, following previous work (Xie et al. 2023), we define *distance* as the absolute position difference between two patches, and *normalized attention distance* as the product of these attention distances with the attention weights. Intuitively, a larger normalized attention distance indicate a focus on global information, while a smaller one reflect attention to local information. The results for each head in all layers are shown in Figure 3 (A).

	Datasets	Weather		ETTh1		ETTh2		ETTm1		ETTm2		ECL		Traffic	
Models	S	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
DropPatch	96	**0.142**	**0.190**	**0.374**	**0.409**	**0.288**	**0.346**	**0.289**	**0.345**	0.171	0.261	**0.129**	**0.221**	**0.361**	**0.255**
	192	**0.186**	**0.234**	**0.401**	**0.427**	**0.352**	**0.385**	**0.334**	**0.373**	**0.229**	**0.301**	**0.148**	**0.239**	**0.378**	**0.262**
	336	**0.238**	**0.274**	**0.406**	0.437	**0.360**	**0.401**	**0.361**	**0.394**	0.282	0.337	**0.165**	**0.258**	**0.389**	**0.268**
	720	**0.312**	**0.330**	0.446	0.469	**0.384**	**0.426**	**0.408**	**0.426**	**0.365**	0.389	**0.201**	**0.290**	**0.427**	**0.289**
	AVG	**0.220**	**0.257**	**0.407**	0.436	**0.346**	**0.390**	**0.348**	**0.385**	**0.262**	**0.322**	**0.161**	**0.252**	**0.389**	**0.269**
PatchTST	96	0.144	0.193	0.381	0.412	0.303	0.355	0.293	0.346	**0.170**	**0.262**	0.131	0.224	0.372	0.266
	192	0.191	0.240	0.407	0.430	0.367	0.390	0.336	0.375	0.235	0.309	**0.148**	0.240	0.389	0.272
	336	0.244	0.281	0.411	**0.435**	0.366	0.403	0.364	**0.394**	**0.280**	**0.334**	**0.165**	**0.258**	0.396	0.273
	720	0.317	0.334	**0.443**	**0.464**	0.395	0.431	0.412	0.428	0.366	**0.387**	0.203	0.291	0.434	0.293
	AVG	0.224	0.262	0.411	**0.435**	0.358	0.395	0.351	0.386	0.263	0.323	0.162	0.253	0.398	0.276

Table 4: Cross-domain fine-tuning results. Models are pre-trained on STS66M, then fine-tuned on other unseen datasets. Forecasting steps $T \in \{96, 192, 336, 720\}$.

	Datasets	Weather		ETTh1		ETTh2		ETTm1		ETTm2		Traffic	
Models	# Samples	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
DropPatch	100	**0.242**	**0.290**	**0.626**	**0.525**	**0.372**	0.411	**0.502**	**0.465**	**0.277**	**0.343**	**0.447**	**0.309**
	300	0.223	0.273	**0.506**	**0.488**	**0.312**	**0.366**	0.531	0.478	**0.237**	**0.311**	**0.399**	**0.275**
	500	**0.212**	**0.263**	**0.474**	**0.461**	**0.317**	**0.366**	0.518	0.476	**0.210**	**0.292**	**0.395**	**0.275**
PatchTST	100	0.247	0.294	0.666	0.552	0.381	**0.401**	0.521	0.474	0.282	0.347	0.450	0.313
	300	**0.222**	**0.271**	0.520	0.503	0.319	0.375	**0.508**	**0.469**	0.257	0.327	**0.399**	0.276
	500	0.225	0.274	0.483	0.481	0.323	0.372	**0.493**	**0.461**	0.214	0.298	0.396	**0.275**

Table 5: Few-shot learning results. Models are pre-trained on STS162M, then fine-tuned on other unseen datasets using limited training samples. Forecasting steps are fixed at 96.

Finding 1 : By comparing normalized attention distances, we found that the DropPatch strategy enables each attention head in the model to focus on information at varying scales. Specifically, this strategy enhancing the model's ability to capture both short-term and long-term dependencies, empowering the model with a more comprehensive understanding of the time-series.

Attention Coefficients Distribution

We then analyze the distributions of attention coefficients across different heads and layers. Uniform attention coefficients lead to a loss of distinctiveness, effectively diminishing the model's ability to capture unique patterns. In contrast, distributions with sharper focus and higher distinctiveness are regarded as more effective (Zhou et al. 2021; Chen et al. 2022; Vyas, Katharopoulos, and Fleuret 2020; Choromanski et al. 2020). In our empirical study, we quantify the distinctiveness of these distributions by computing the Kullback-Leibler (KL) divergence between the uniform distribution and the attention distributions. A larger KL divergence indicates a greater deviation from the uniform distribution, reflecting sharper and more distinctive attention patterns. The results are shown in Figure 3 (B).

Finding 2 : The results indicate that applying the DropPatch strategy sharpens the focus of attention heads, facilitating the identification of more valuable information and underlying patterns.

Attention Coefficients Difference

The previous two subsections reveal that attention heads in DropPatch exhibit greater diversity in behavior. In this subsection, we further investigate whether different attention heads capture diverse information. Specifically, we conduct an analysis of the attention distribution across different heads by calculating the KL divergence between attention heads in the same layer. This comparison highlights the distributional differences among attention heads. A higher KL divergence indicates greater differences, suggesting that each head has learned distinct information, thereby reducing redundancy in the information captured by different heads. As shown in Figure 4, attention heads in DropPatch exhibit higher KL divergence compared to those in PatchTST.

Finding 3 : The analysis of attention distributions demonstrates that the DropPatch strategy enables attention heads to capture distinct information, thereby reducing redundancy and enhancing the model's representation capabilities.

Central Kernel Alignment Analysis

We use CKA (Central Kernel Alignment) values (Kornblith et al. 2019) to compare the similarity of representations in a pre-trained model and a fine-tuned model. Specifically, we calculate CKA similarity using the last layer representations between the pre-trained model and the fine-tuned model. Models are pre-trained on ECL.

Models	**DropPatch**		PatchTST	
Metrics	MSE	MAE	MSE	MAE
Exchange	**0.348**	**0.396**	0.354	0.400
PEMS03	**0.198**	**0.293**	0.205	0.296
PEMS04	**0.264**	**0.339**	0.273	0.343
PEMS07	**0.214**	**0.312**	0.219	0.323
PEMS08	**0.225**	**0.300**	0.233	0.305

Table 6: Results of cold start setup. The look-back length L_{ft} is fixed at 96. Results are averaged from all forecasting steps.

Models	**DropPatch**		PatchTST		SimMTM	
Metrics	Mem.	T.C.	Mem.	T.C.	Mem.	T.C.
ETTm1	**1404**	**32.2**	1722	44.5	29090	823.3
Weather	**2094**	**42.1**	3914	75.1	OOM	-
ECL	**4256**	**306.7**	11050	528.5	OOM	-

Table 7: Model efficiency comparison. *Mem.* denotes the memory usage, measured in megabytes (MB). *T.C.* denotes the time consumption per epoch in seconds.

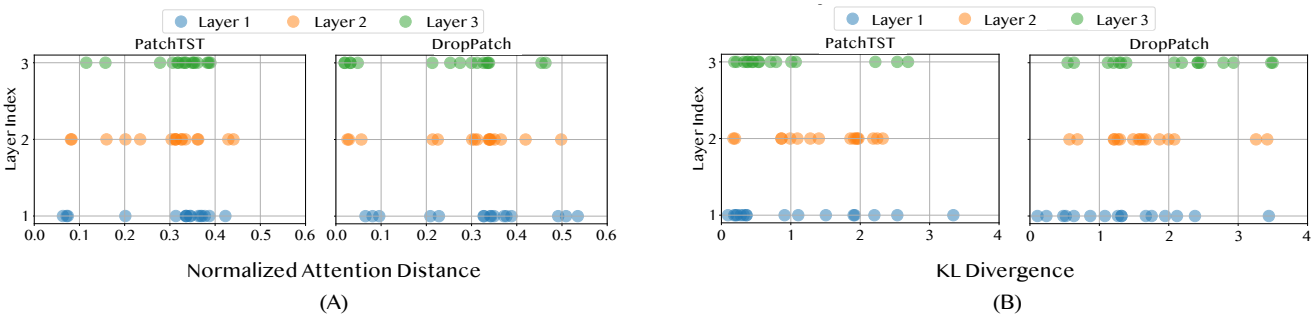

Figure 3: Analysis of (A) normalized distance, and (B) KL divergence between attention distributions and uniform distribution for each head across all layers. Each dot represents an individual attention head, while different colors indicate different layers.

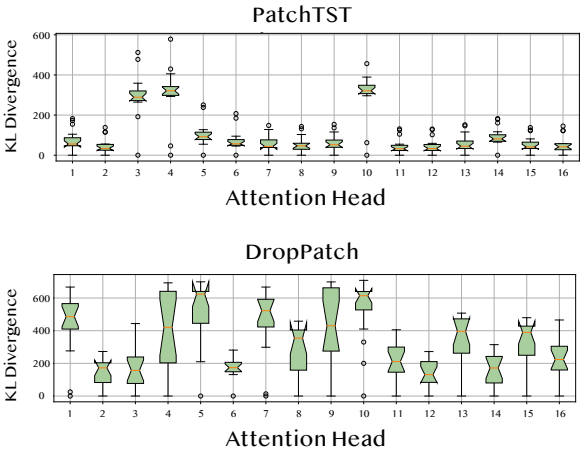

Figure 4: Attention distribution difference.

Finding 4 : As presented in Table 8, DropPatch significantly enhances the representation ability. For in-domain tasks, DropPatch achieves high CKA similarity, indicating that the model better learns the underlying patterns of the dataset. For cross-domain tasks, DropPatch exhibits reduced CKA similarity, which we attribute to the model's improved ability to handle domain shifts and adapt to unseen distributions after applying the DropPatch strategy.

	ETTh1	Weather	ECL
DropPatch	**0.941**	**0.883**	0.869
PatchTST	0.911	0.835	**0.916**

Table 8: CKA similarity results.

Conclusion

In this paper, we propose DropPatch, an enhancement to masked time-series modeling achieved by introducing the random dropping of sub-series patches. This approach yields significant improvements in pre-training efficiency and various downstream tasks. Extensive experiments validate the effectiveness, highlighting its ability to improve the attention mechanism by enabling a sharper focus on multi-scale and diverse information. Furthermore, out theoretical analysis reveals that this technique slows the degeneration of Transformer representations toward a rank-1 linear subspace, underlying its beneficial impact on model performance.

Acknowledgments

This work is funded in part by the Shanghai Science and Technology Development Fund No.22dz1200704, the National Key Research and Development Plan Project 2022YFC3600901. This work is supported by Ant Group.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Amos, I.; Berant, J.; and Gupta, A. 2023. Never Train from Scratch: Fair Comparison of Long-Sequence Models Requires Data-Driven Priors. *arXiv preprint arXiv:2310.02980*.

Baevski, A.; Hsu, W.-N.; Xu, Q.; Babu, A.; Gu, J.; and Auli, M. 2022. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*, 1298–1312. PMLR.

Bao, H.; Dong, L.; Piao, S.; and Wei, F. 2021. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, J.; Agarwal, A.; Abdelkarim, S.; Zhu, D.; and Elhoseiny, M. 2022. Reltransformer: A transformer-based long-tail visual relationship recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19507–19517.

Cheng, M.; Liu, Q.; Liu, Z.; Zhang, H.; Zhang, R.; and Chen, E. 2023. Timemae: Self-supervised representations of time series with decoupled masked autoencoders. *arXiv preprint arXiv:2303.00320*.

Choromanski, K.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J.; Mohiuddin, A.; Kaiser, L.; et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dong, J.; Wu, H.; Zhang, H.; Zhang, L.; Wang, J.; and Long, M. 2024. Simmtm: A simple pre-training framework for masked time-series modeling. *Advances in Neural Information Processing Systems*, 36.

Dong, Y.; Cordonnier, J.-B.; and Loukas, A. 2021. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, 2793–2803. PMLR.

Goswami, M.; Szafer, K.; Choudhry, A.; Cai, Y.; Li, S.; and Dubrawski, A. 2024. MOMENT: A Family of Open Time-series Foundation Models. *arXiv preprint arXiv:2402.03885*.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16000–16009.

Jin, X.; Park, Y.; Maddix, D.; Wang, H.; and Wang, Y. 2022. Domain adaptation for time series forecasting via attention sharing. In *International Conference on Machine Learning*, 10280–10297. PMLR.

Kornblith, S.; Norouzi, M.; Lee, H.; and Hinton, G. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, 3519–3529. PMLR.

Liu, M.; Zeng, A.; Chen, M.; Xu, Z.; Lai, Q.; Ma, L.; and Xu, Q. 2022. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35: 5816–5828.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2023. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Liu, Y.; Zhang, H.; Li, C.; Huang, X.; Wang, J.; and Long, M. 2024. Timer: Transformers for Time Series Analysis at Scale. *arXiv preprint arXiv:2402.02368*.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*.

Noci, L.; Anagnostidis, S.; Biggio, L.; Orvieto, A.; Singh, S. P.; and Lucchi, A. 2022. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 35: 27198–27211.

Vyas, A.; Katharopoulos, A.; and Fleuret, F. 2020. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems*, 33: 21665–21674.

Wang, Z.; Xu, X.; Zhang, W.; Trajcevski, G.; Zhong, T.; and Zhou, F. 2022. Learning latent seasonal-trend representations for time series forecasting. *Advances in Neural Information Processing Systems*, 35: 38775–38787.

Woo, G.; Liu, C.; Kumar, A.; Xiong, C.; Savarese, S.; and Sahoo, D. 2024. Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592*.

Woo, G.; Liu, C.; Sahoo, D.; Kumar, A.; and Hoi, S. 2022. Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. *arXiv preprint arXiv:2202.01575*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34: 22419–22430.

Xie, Z.; Geng, Z.; Hu, J.; Zhang, Z.; Hu, H.; and Cao, Y. 2023. Revealing the dark secrets of masked image modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 14475–14485.

Yue, Z.; Wang, Y.; Duan, J.; Yang, T.; Huang, C.; Tong, Y.; and Xu, B. 2022. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8980–8987.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 11121–11128.

Zerveas, G.; Jayaraman, S.; Patel, D.; Bhamidipaty, A.; and Eickhoff, C. 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.

Zhai, S.; Likhomanenko, T.; Littwin, E.; Busbridge, D.; Ramapuram, J.; Zhang, Y.; Gu, J.; and Susskind, J. M. 2023. Stabilizing transformer training by preventing attention entropy collapse. In *International Conference on Machine Learning*, 40770–40803. PMLR.

Zhang, Q.; Wang, Y.; and Wang, Y. 2022. How mask matters: Towards theoretical understandings of masked autoencoders. *Advances in Neural Information Processing Systems*, 35: 27127–27139.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, 27268–27286. PMLR.