Are Self-Attentions Effective for Time Series Forecasting?

Dongbin Kim[1], Jinseong Park[1], Jaewook Lee[1*], Hoki Kim[2*]

[1]Seoul National University   [2]Chung-Ang University

{dongbin413,jinseong,jaewook}@snu.ac.kr, hokikim@cau.ac.kr

Abstract

Time series forecasting is crucial for applications across multiple domains and various scenarios. Although Transformers have dramatically advanced the landscape of forecasting, their effectiveness remains debated. Recent findings have indicated that simpler linear models might outperform complex Transformer-based approaches, highlighting the potential for more streamlined architectures. In this paper, we shift the focus from evaluating the overall Transformer architecture to specifically examining the effectiveness of self-attention for time series forecasting. To this end, we introduce a new architecture, Cross-Attention-only Time Series transformer (CATS), that rethinks the traditional transformer framework by eliminating self-attention and leveraging cross-attention mechanisms instead. By establishing future horizon-dependent parameters as queries and enhanced parameter sharing, our model not only improves long-term forecasting accuracy but also reduces the number of parameters and memory usage. Extensive experiment across various datasets demonstrates that our model achieves superior performance with the lowest mean squared error and uses fewer parameters compared to existing models. The implementation of our model is available at: https://github.com/dongbeank/CATS.

1   Introduction

Time series forecasting plays a critical role within the machine learning society, given its applications ranging from financial forecasting to medical diagnostics. To improve the accuracy of predictions, researchers have extensively explored and developed various models. These range from traditional statistical methods to modern deep learning techniques. Most notably, Transformer [19] has brought about a paradigm shift in time series forecasting, resulting in numerous high-performance models, such as Informer [29], Autoformer [23], Pyraformer [11], FEDformer [31], and Crossformer [28]. This line of work establishes new benchmarks for high performance in time series forecasting.

However, Zeng et al. [26] have raised questions about the effectiveness of Transformer-based time series forecasting models especially for long term time series forecasting. Specifically, their experiments demonstrated that simple linear models could outperform these Transformer-based approaches, thereby opening new avenues for research into simpler architectural frameworks. Indeed, the following studies [12, 6] have further validated that these linear models can be enhanced by incorporating additional features.

Despite these developments, the effectiveness of each components in Transformer architecture in time series forecasting remains a subject of debate. Nie et al. [14] introduced an encoder-only Transformer that utilizes patching rather than point-wise input tokens, which exhibited improved performance compared to linear models. Zeng et al. [26] also highlighted potential shortcomings in simpler linear

---

*Corresponding authors

38th Conference on Neural Information Processing Systems (NeurIPS 2024).
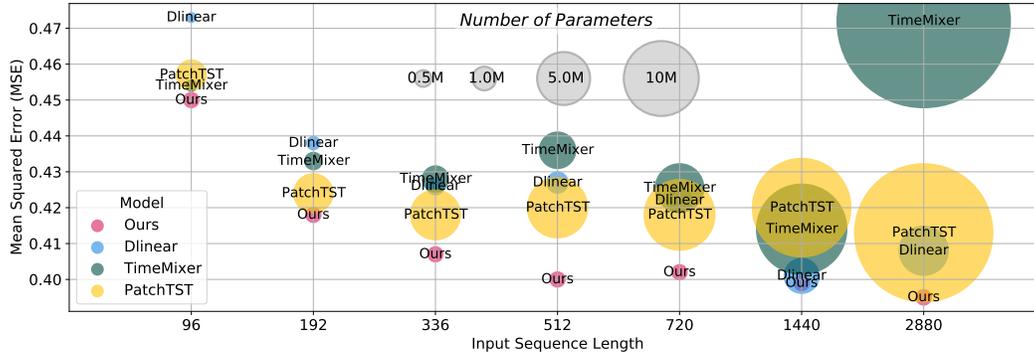
Figure 1: Experimental results illustrating the mean squared error (MSE) and the number of parameters with varying input sequence lengths on ETTm1. Each bubble represents a different model, with the bubble size indicating the number of parameters in millions—larger bubbles denote models with more parameters. Our model consistently shows the lowest MSE (i.e., best performance) with fewer parameters even for longer input sequences. The detailed results can be found in Table 5.

networks, such as their inability to capture temporal dynamics at change points [18] compared to Transformers. Consequently, while a streamlined architecture may be beneficial, it is imperative to critically evaluate which elements of the Transformer are necessary and which are not for time series modeling.

In light of these considerations, our study shifts focus from the overall architecture of the Transformer to a more specific question: **Are self-attentions effective for time series forecasting?** While this question is also noted in [26], their analysis was limited to substituting attention layers with linear layers, leaving substantial room for potential model design when focusing on Transformers. Furthermore, the issue of *temporal information loss* (i.e., permutation-invariant and anti-order characteristics of self-attention) is predominantly caused by the use of self-attention rather than the Transformer architecture itself. Therefore, we aim to resolve the issues of self-attention and therefore propose a new forecasting architecture that achieves higher performance with a more efficient structure.

In this paper, we introduce a novel forecasting architecture named Cross-Attention-only Time Series transformer (CATS) that simplifies the original Transformer architecture by eliminating all self-attentions and focusing on the potential of cross-attentions. Specifically, our model establishes future horizon-dependent parameters as queries and treats past time series data as key and value pairs. This allows us to enhance parameter sharing and improve long-term forecasting performance. As shown in Figure 1, our model shows the lowest mean squared error (i.e., better forecasting performance) even for longer input sequences and with fewer parameters than existing models. Moreover, we demonstrate that this simplified architecture can provide a clearer understanding of how future predictions are derived from past data with individual attention maps for the specific forecasting horizon. Finally, through extensive experiments, we show that our proposed model not only achieves state-of-the-art performance but also requires fewer parameters and less memory consumption compared to previous Transformer models across various time series datasets.

2 Related Work

Time Series Transformers   Transformer models [19] have shown effective in various domains [5, 4, 15], with a novel encoder-decoder structure with self-attention, masked self-attention, and cross-attention. The self-attention mechanism is a key component for extracting semantic correlations between paired elements, even with identical input elements; however, autoregressive inference with self-attention requires quadratic time and memory complexity. Therefore, Informer [29] proposed directly predicting multi-steps, and a line of work, such as Autoformer [23], FEDformer [31], and Pyraformer [11], investigated the complexity issue in time series transformers. Simultaneously, unique properties of time series, such as stationarity [12], decomposition [23], frequency features [31], or cross-dimensional properties [28] were employed to modify the attention layer for forecasting tasks. Recently, researchers have investigated the essential architecture in Transformers to capture long-term

2

dependencies. PatchTST [14] became a de-facto standard Transformer model by patching the time series input in a channel-independence manner, which is widely used in following Transformer-based forecasting models [13, 7]. On the other hand, Das et al. [3] emphasized the importance of decoder-only forecasting models, while they focused on zero-shot using pre-trained language models. However, none of them have investigated the importance of cross-attention for time series forecasting.

Temporal Information Encoding   Fixed temporal order in time series is the distinct property of time series, in contrast to the language domain where semantic information does not heavily depend on the word ordering [4]. Thus, some researchers have used learnable positional encoding in Transformers to embed time-dependent properties [9, 23]. However, Zeng et al. [26] first argued that self-attention is not suitable for time series due to its permutation invariant and anti-order properties. While they focus on building complex representations, they are inefficient in maintaining the original context of historical and future values. They rather proposed linear models without any embedding layer and demonstrated that it can achieve better performance than Transformer models, particularly showing robust performance to long input sequences. Recent linear time series models outperformed previous Transformer models with simple architectures by focusing on pre-processing and frequency-based properties [10, 2, 21]. On the other hand, Woo et al. [22] investigated the new line of works of time-index models, which try to model the underlying dynamics with given time stamps. These related works imply that preserving the order of time series sequences plays a crucial role in time series forecasting.

3   Revisiting Self-Attention in Time Series Forecasting

Motivation of Self-Attention Removal   Following the concerns about the effectiveness of self-attention on temporal information preservation [26], we conduct an experiment using PatchTST [14]. We consider three variations of the PatchTST model: the original PatchTST with overlapping patches with length 16 and stride 8 (Fig. 2a); a modified PatchTST with non-overlapping patches with length 24 (Fig. 2b); and a version where self-attention is replaced by a linear embedding layer, using non-overlapping patches with length 24 (Fig. 2c). This setup allows us to isolate the effects of self-attention on temporal information preservation, while controlling for the impact of patch overlap.

| (a) Original PatchTST | (b) PatchTST w/ non-overlapping | (c) PatchTST w/o self-attn |
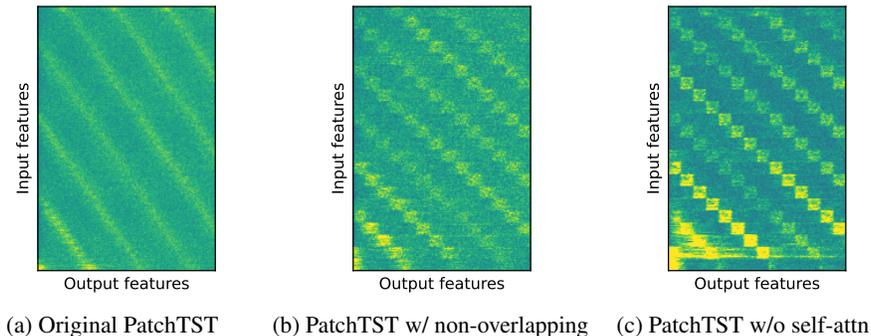
Figure 2: Absolute values of weights in the final linear layer for different PatchTST variations. The distinct patterns reveal how each model captures temporal information.

Fig. 2 illustrates the absolute values of the weights in the final linear layer for these model variations. Compared to the original PatchTST (Fig. 2a), both non-overlapping versions (Fig. 2b and Fig. 2c) show more vivid patterns. The version with linear embedding (Fig. 2c) demonstrates the clearest capture of temporal information, suggesting that the self-attention mechanism itself may not be necessary for capturing temporal information.

In Table 1, we summarize the forecasting performance of the original PatchTST (Fig. 2a) and PatchTST without self-attention (Fig. 2c). PatchTST without self-attention consistently improves or maintains performance across all forecasting horizons. Specifically, the original version with self-attention shows lower performance for longer forecast horizons. This result suggests that self-attention may not only be unnecessary for effective time series forecasting but could even hinder

3

performance in certain cases. Therefore, better performance of w/o self-attention challenges the conventional belief regarding the importance of self-attention mechanisms in Transformer-based models for time series forecasting tasks.

Our findings offer new insights into the role of self-attention in time series forecasting. As shown in Fig. 2 and Table 1, replacing self-attention with a linear layer not only captures clear temporal patterns but also results in significant performance improvements, particularly for longer forecast horizons. These results highlight potential areas for enhancing the handling of temporal information, beyond addressing the well-known concerns regarding computational complexity.

Table 1: Effect of self-attention in PatchTST on forecasting performance (MSE) on ETTm1.

Horizon	original	w/o self-attn
96	**0.290**	**0.290**
192	0.332	**0.328**
336	0.366	**0.359**
720	0.416	**0.414**

| (a) Transformer | (b) Encoder | (c) Linear model | (d) Ours |
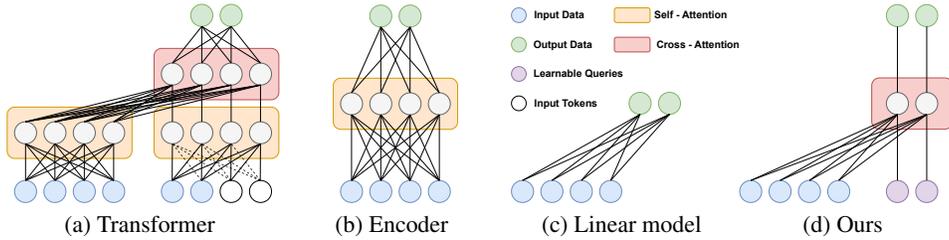
Figure 3: Illustration of existing time series forecasting architectures and the proposed architecture.

Rethinking Transformer Design   Given the challenges associated with self-attention in time series forecasting, we propose a fundamental rethinking of the Transformer architecture for this task. Fig. 3 illustrates the differences between existing architectures and our proposed approach. Traditional Transformer architectures (Fig. 3a) and encoder-only models (Fig. 3b) rely heavily on self-attention mechanisms, which may lead to temporal information loss. In contrast, Zeng et al. [26] proposed a simplified linear model, DLinear (Fig. 3c), which removes all Transformer-based components. While this approach reduces computational load and potentially avoids some temporal information loss, it may struggle to capture complex temporal dependencies.

To address these challenges while preserving the advantages of Transformer architectures, we propose the Cross-Attention-only Time Series transformer (CATS), depicted in Fig. 3d. Our approach removes all self-attention layers and focuses solely on cross-attention, aiming to better capture temporal dependencies while maintaining the structural advantages of the transformer architecture. In the following section, we will introduce our CATS model in detail, explaining our key innovations including a novel use of cross-attention, efficient parameter sharing, and adaptive masking techniques.

4   Proposed Methodology

4.1   Problem Definition and Notations

A multivariate time series forecasting task aims to predict future values $\tilde{X} = \{\mathbf{x}_{L+1}, \ldots, \mathbf{x}_{L+T}\} \in \mathbb{R}^{M \times T}$ with the prediction $\hat{X} = \{\hat{\mathbf{x}}_{L+1}, \ldots, \hat{\mathbf{x}}_{L+T}\} \in \mathbb{R}^{M \times T}$ based on past datasets $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\} \in \mathbb{R}^{M \times L}$. Here, T represents the forecasting horizon, L denotes the input sequence length, and M represents the dimension of time series data.

In traditional time series transformers, we feed the historical multivariate time series X to embedding layers, resulting in the historical embedding $H \in \mathbb{R}^{D \times L}$. Here, D is the embedding size. Note that, in channel-independence manners, the multivariate input is considered to separate univariate time series $\mathbf{x} \in \mathbb{R}^{1 \times L}$. With patching [14], univariate time series $\mathbf{x}$ transforms into patches $\mathbf{p} = \mathtt{Patch}(\mathbf{x}) \in \mathbb{R}^{P \times N_L}$ where P is the size of each patch and N_L is the number of input patches. Similar to non-patching situations, patches are fed to embedding layers $P = \mathtt{Embedding}(\mathbf{p}) \in \mathbb{R}^{D \times N_L}$.

4
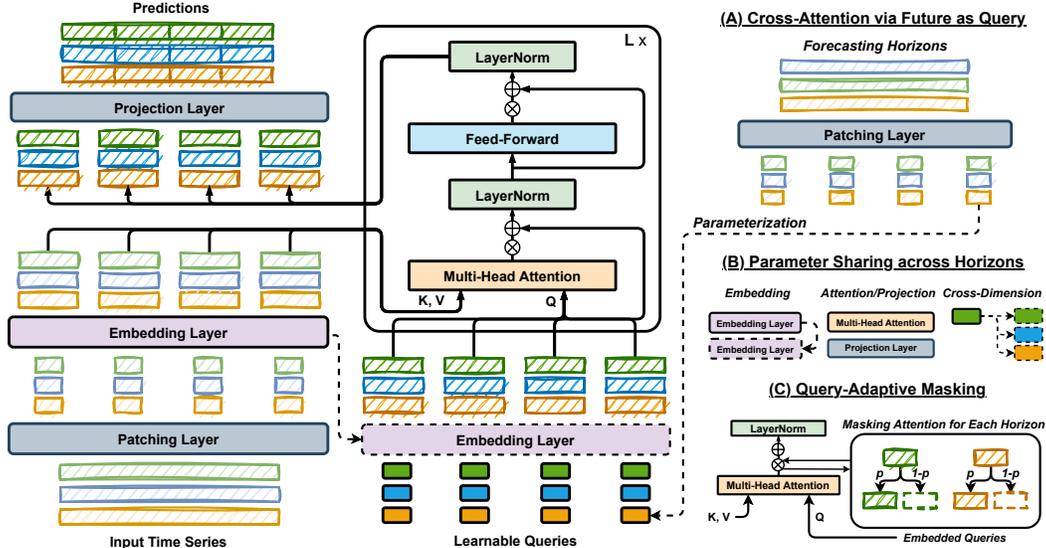
Figure 4: Illustration on the proposed model architecture. Our model removes all self-attentions from the original Transformer structure and focuses on cross-attentions. To fully utilize the cross-attention, we conceptualize the future horizon as queries and use the input time series (i.e., past time series) as keys and values (Fig. A). This simplified structure enables us to enhance the parameter sharing across forecasting horizons (Fig. B) and make use of query-adaptive masking (Fig. C) for performance.

4.2 Model Structure

Our proposed architecture consists of three key components: (A) *Cross-Attention with Future as Query*, (B) *Parameter Sharing across Horizons*, and (C) *Query-Adaptive Masking*. Fig. 4 illustrates how our model modifies the traditional transformer structure.

By focusing solely on cross-attention, our approach allows us to maintain the periodic properties of time series, which self-attention, with its permutation-invariant and anti-order characteristics, struggles to capture. Furthermore, we leverage advanced architectural designs of time series transformers, such as patching [14], which linear models cannot utilize. The following subsections provide detailed descriptions of each component of our CATS model, explaining how these elements work together to address the challenges of time series forecasting.

Cross-Attention via Future as Query   Similar to self-attention, the cross-attention mechanism employs three elements: key, query, and value. The distinctive feature of cross-attention is that the query originates from a different source than the key or value. Generally, the query component aims to identify the most relevant information among the keys and uses it to extract crucial data from the values [1, 27]. In the realm of time series forecasting, where predictions are often made for a specific target horizon—such as forecasting 10 steps ahead. Therefore, within this concept of forecasting, we argue that each future horizon should be regarded as a question, i.e., an independent query.

To implement this, we establish horizon-dependent parameters as learnable queries. As shown in Fig. 4, we begin by creating parameters for the specified forecasting horizon. For each of these virtualized parameters, we assign a fixed number of parameters to represent the corresponding horizon as learnable queries $\mathbf{q}$. For example, $\mathbf{q}_i$ is a horizon-dependent query at $L+i$. When patching is applied, these queries are then processed independently; each learnable query $\mathbf{q} \in \mathbb{R}^P$ is first fed into the embedding layer, and then fed into the multi-head attention with the embedded input time series patches as the key and value.

Based on these new query parameters, we can utilize a cross-attention-only structure in the decoder, resulting in an advantage in efficiency. In Table 2, we summarize the time complexity of recent Transformer models and ours. The results indicate that our method only requires the time complexity of $\mathcal{O}(LT/P^2)$, where most of the Transformer-based models require $\mathcal{O}(L^2)$ except FEDformer and

5

Table 2: Time complexity of Transformer-based models to calculate attention outputs. Time refers to the inference time obtained by averaging 10 runs under $L = 96$ and $T = 720$ on Electricity.

Method	Encoder	Decoder	Time	Method	Encoder	Decoder	Time
Transformer [13]	$\mathcal{O}(L^2)$	$\mathcal{O}(T(T+L))$	10.4ms	Informer [29]	$\mathcal{O}(L\log L)$	$\mathcal{O}(T(T+\log L))$	13.5ms
Autoformer [23]	$\mathcal{O}(L\log L)$	$\mathcal{O}((L/2+H)\log(L/2+T))$	24.1ms	Pyraformer [11]	$\mathcal{O}(L)$	$\mathcal{O}(T(T+L))$	11.2ms
FEDformer [31]	$\mathcal{O}(L)$	$\mathcal{O}(L/2+H)$	69.3ms	Crossformer [28]	$\mathcal{O}\left(ML^2/P^2\right)$	$\mathcal{O}\left(MT(T+L)/P^2\right)$	30.6ms
PatchTST [14]	$\mathcal{O}\left(L^2/P^2\right)$	-	7.6ms	CATS (Ours)	-	$\mathcal{O}\left(LT/P^2\right)$	7.0ms

Pyraformer. However, since these two models have an encoder-decoder and a relatively huge amount of parameters, they require 10x and 4x computational times than ours, respectively.

Parameter Sharing across Horizons   One of the strongest benefits of cross-attention via future horizon as a query $\mathbf{q}$ is that each cross-attention is only calculated on the values from a single forecasting horizon and the input time series. Mathematically, for a prediction of future value $\hat{\mathbf{x}}_{L+i}$ can be expressed as a function solely dependent on the past samples $\boldsymbol{X} = [\mathbf{x}_1, ..., \mathbf{x}_L]$ and $\mathbf{q}_i$, independent of $\mathbf{q}_j$ for all $i \neq j$ or i and j are not in the same patch.

This independent forecasting mechanism offers a notable advantage; a higher level of parameter sharing. As demonstrated in [14], significant reductions in the required number of parameters can be achieved in time series forecasting through parameter sharing between inputs or patches, enhancing computational efficiency. Regarding this, we propose parameter sharing across all possible layers — the embedding layer, multi-head attention, and projection layer — for every horizon-dependent query $\mathbf{q}$. In other words, all horizon queries $\mathbf{q}_1, \ldots, \mathbf{q}_T$ or $\mathbf{q}_1, \ldots, \mathbf{q}_{N_T}$ share the same embedding layer used for the input time series $\mathbf{x}_1, \ldots, \mathbf{x}_L$ or patches $\mathbf{p}_1, \ldots, \mathbf{p}_{N_L}$ before proceeding to the cross-attention layer, respectively. Furthermore, to maximize the parameter sharing, we also propose cross-dimension sharing that uses the same query parameters for all dimensions.

For the multi-head attention and projection layers, we apply the same algorithm across horizons. Notably, unlike the approach in PatchTST [14], we also share the projection layer for each prediction. Specifically, PatchTST, being an encoder-only model, employs a fully connected layer as the projection layer for the encoder outputs $\boldsymbol{P} \in \mathbb{R}^{D \times N_L}$, resulting in $(D \times N_L) \times T$ parameters. In contrast, our model first processes raw queries $\mathbf{q} = [\mathbf{q}_1, \ldots, \mathbf{q}_{N_T}] \in \mathbb{R}^{P \times N_T}$. These queries are then embedded through the cross-attention mechanism, resulting in $\boldsymbol{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{N_T}] \in \mathbb{R}^{D \times N_T}$. The final projection uses shared parameters $W \in \mathbb{R}^{P \times D}$, producing an output $W\boldsymbol{Q} \in \mathbb{R}^{P \times N_T}$. Thus, our number of parameters for this projection becomes $P \times D$, which is not proportionally increasing to T. This approach significantly reduces time complexity during both the training and inference phases.

In Table 3, we outline the impact of parameter sharing across different forecasting horizons. In contrast to the model without parameter sharing, which exhibits a rapid increase in parameters as the forecasting horizon extends, our model, which shares all layers including the projection layer, maintains a nearly consistent number of parameters.

Additionally, all operations, including embedding and multi-head attention, are performed independently for each learnable query. This implies that the forecast for a specific horizon does not depend on other horizons. Such an

Table 3: Effect of parameter sharing across horizons on the number of parameters for different forecasting horizons on ETTh1.

Horizon	w/ sharing	w/o sharing
96	355,320	404,672
192	355,416	552,320
336	355,560	958,112
720	355,944	3,121,568

approach allows us to generate distinct attention maps for each forecasting horizon, providing a clear understanding of how each prediction is derived. Please refer to Section 5.5.

Query-Adaptive Masking   Parameter sharing across horizons enhances the efficiency of our proposed architecture and simplifies the model. However, we observed that a high degree of parameter sharing could lead to overfitting to the keys and values (i.e., past time series data), rather than the queries (i.e., forecasting horizon). Specifically, the model may converge to generate similar or identical predictions, $\hat{\mathbf{x}}_{L+i}$ and $\hat{\mathbf{x}}_{L+j}$, despite receiving different horizon queries, $\mathbf{q}_i$ and $\mathbf{q}_j$ (i.e., the target horizons differ).

Therefore, to ensure the model focuses on each horizon-dependent query $\mathbf{q}$, we introduce a new technique that masks the attention outputs. As illustrated in the right-bottom figure of Fig. 4, for each horizon, we apply a mask to the direct connection from Multi-Head Attention to LayerNorm with a

6

probability p. This mask prevents access to the input time series information, resulting in only the query to influence future value predictions. This selective disconnection, rather than the application of dropout in the residual connections, helps the layers to concentrate more effectively on the forecasting queries. We note that this approach can be related to stochastic depth in residual networks [8]. The stochastic depth technique has proven effective across various tasks, such as vision tasks [17, 25]. To the best of our knowledge, this is the first application of stochastic depth in Transformers for time series forecasting. A detailed analysis of query-adaptive masking can be found in Appendix.

In summary, the framework described in this section, including cross-attention via future as query, parameter sharing across horizons, and query-adaptive masking, is named the **Cross-Attention-only Time Series transformer (CATS).**

5 Experiments

In this section, we provide extensive experiments to provide the benefits of our proposed framework, CATS, including forecasting performance and computational efficiency. To this end, we use 7 different real-world datasets and 9 baseline models. For datasets, we use Electricity, ETT (ETTh1, ETTh2, ETTm1, and ETTm2), Weather, Traffic, and M4. These datasets are provided in [23] and [24] for time series forecasting benchmark, detailed in Appendix.

For baselines, we utilize a wide range of various baselines, including the state-of-the-art long-term time series forecasting model TimeMixer [21], PatchTST [14], Timesnet [24], Crossformer [28], MICN [20], FiLM [30], DLinear [26], Autoformer [23], and Informer [29]. For both long-term and short-term time series forecasting results, we report performance of our model alongside the results of other models as presented in TimeMixer [21], ensuring a consistent comparison across all baselines. We used 4 NVIDIA RTX 4090 24GB GPUs with 2 Intel(R) Xeon(R) Gold 5218R CPUs @ 2.10GHz for all experiments.

5.1 Long-term Time Series Forecasting Results

To ease comparison, we follow the settings of [21] for long-term forecasting, using various forecast horizons with a fixed 96 input sequence length. Detailed settings are provided in the Appendix. Table 11 summarizes the forecasting performance across all datasets and baselines. Our proposed model, CATS, demonstrates superior performance in multivariate long-term forecasting tasks across multiple datasets. CATS consistently achieves the lowest Mean Squared Error (MSE) and Mean Absolute Error (MAE) on the Traffic dataset for all forecast horizons, outperforming all other models. For the Weather, Electricity, and ETT datasets, CATS shows competitive performance, achieving the best results on most forecast horizons. This indicates that CATS effectively captures underlying patterns in diverse time series data, highlighting its capability to handle complex temporal dependencies. Additional experiments with longer input sequence lengths of 512 are provided in the Appendix.

Table 4: Multivariate long-term forecasting results with recent forecasting models and ours for unified hyperparameter settings. The best results are in **bold** and the second best are underlined. Full results are provided in Appendix.

Models		CATS		TimeMixer		PatchTST		Timesnet		Crossformer		MICN		FiLM		DLinear		Autoformer		Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.161**	**0.207**	0.163	0.209	0.186	0.227	0.172	0.220	0.195	0.271	0.198	0.261	0.195	0.236	0.195	0.252	0.266	0.336	0.300	0.384
	192	**0.208**	**0.250**	**0.208**	**0.250**	0.234	0.265	0.219	0.261	0.209	0.277	0.239	0.299	0.239	0.271	0.237	0.295	0.307	0.367	0.598	0.544
	336	0.264	0.290	0.251	**0.287**	0.284	0.301	**0.246**	0.337	0.273	0.332	0.285	0.336	0.289	0.306	0.282	0.331	0.359	0.395	0.578	0.523
	720	0.342	**0.341**	**0.339**	**0.341**	0.356	0.349	0.365	0.359	0.379	0.401	0.351	0.388	0.361	0.351	0.345	0.382	0.419	0.428	1.059	0.741
Electricity	96	**0.149**	**0.237**	0.153	0.247	0.190	0.296	0.168	0.272	0.219	0.314	0.180	0.293	0.198	0.274	0.210	0.302	0.201	0.317	0.274	0.368
	192	**0.163**	**0.250**	0.166	0.256	0.199	0.304	0.184	0.322	0.231	0.322	0.189	0.302	0.198	0.278	0.210	0.305	0.222	0.334	0.296	0.386
	336	**0.180**	**0.268**	0.185	0.277	0.217	0.319	0.198	0.300	0.246	0.337	0.198	0.312	0.217	0.300	0.223	0.319	0.231	0.443	0.300	0.394
	720	0.219	**0.302**	0.225	0.310	0.258	0.352	0.220	0.320	0.280	0.363	**0.217**	0.330	0.278	0.356	0.258	0.350	0.254	0.361	0.373	0.439
Traffic	96	**0.421**	**0.270**	0.462	0.285	0.526	0.347	0.593	0.321	0.644	0.429	0.577	0.350	0.647	0.384	0.650	0.396	0.613	0.388	0.719	0.391
	192	**0.436**	**0.275**	0.473	0.296	0.522	0.332	0.617	0.336	0.665	0.431	0.589	0.356	0.600	0.361	0.598	0.370	0.616	0.382	0.696	0.379
	336	**0.453**	**0.284**	0.498	0.296	0.517	0.334	0.629	0.336	0.674	0.420	0.594	0.358	0.610	0.367	0.605	0.373	0.622	0.337	0.777	0.420
	720	**0.484**	**0.303**	0.506	0.313	0.552	0.352	0.640	0.350	0.683	0.424	0.613	0.361	0.691	0.425	0.645	0.394	0.660	0.408	0.864	0.472
ETT (Avg)	96	**0.289**	**0.339**	0.290	0.339	0.326	0.362	0.312	0.355	0.465	0.456	0.340	0.388	0.324	0.358	0.319	0.368	0.389	0.415	1.414	0.816
	192	**0.348**	0.374	0.350	0.373	0.388	0.397	0.365	0.385	0.553	0.518	0.408	0.431	0.384	0.393	0.399	0.418	0.448	0.443	1.985	0.989
	336	**0.376**	**0.395**	0.390	0.404	0.426	0.423	0.455	0.421	0.686	0.584	0.479	0.476	0.428	0.423	0.469	0.463	0.491	0.473	2.101	1.101
	720	**0.434**	**0.433**	0.439	0.438	0.464	0.455	0.467	0.455	1.038	0.754	0.597	0.541	0.481	0.459	0.596	0.537	0.533	0.504	2.343	1.163

7

Table 5: Comparison of models with the number of parameters, GPU memory consumption, and MSE across different input sequence lengths on ETTm1. Full results with more diverse input lengths are provided in Appendix.

	Parameters				GPU Memory				MSE			
Input Length	336	720	1440	2880	336	720	1440	2880	336	720	1440	2880
PatchTST	4.3M	8.7M (2.0x)	17.0M (4.0x)	33.6M (7.9x)	3.5GB	7.4GB (2.1x)	22.0GB (6.3x)	58.6GB (16.9x)	0.418	0.418	0.420	0.412
TimeMixer	1.1M	4.1M (3.6x)	14.2M (12.6x)	52.9M (46.8x)	2.9GB	3.9GB (1.3x)	5.9GB (2.0x)	10.3GB (3.6x)	0.428	0.425	0.414	0.472
DLinear	0.5M	1.0M (2.1x)	2.1M (4.2x)	4.2M (8.5x)	1.1GB	1.1GB (1.0x)	1.2GB (1.0x)	1.2GB (1.1x)	0.426	0.422	0.401	0.408
CATS	0.4M	0.4M (1.0x)	0.4M (1.0x)	0.4M (1.1x)	1.9GB	2.1GB (1.1x)	2.7GB (1.4x)	3.8GB (2.0x)	0.407	0.402	0.399	0.395

5.2 Efficient and Robust Forecasting for Long Input Sequences

Zeng et al. [26] observed that many models experience a decline in performance when using long input sequences for time series forecasting. To address this, some approaches have been developed to capture long-term dependencies. For instance, TimeMixer [21] employs linear models with mixed scale, and PatchTST [14] utilizes an encoder network to encode long-term information. However, these models still have computational issues, particularly in terms of escalating memory and parameter requirements. Thus, in this subsection, we provide a comparison between previous models and ours in terms of efficient and robust forecasting for long input sequences.

First of all, to provide a fair comparison, we summarize the number of parameters, GPU memory consumption, and forecasting performance of comparison models with varying input lengths. As summarized in Table 5, existing complex models, such as PatchTST and TimeMixer, suffer from increased parameters and computational burdens when performing forecasting with long input lengths. Although DLinear uses fewer parameters and less GPU memory, its performance is limited due to its linear structure in capturing non-linearity patterns. Considering both performance and efficiency, the proposed model demonstrates robust performance improvement even with longer input lengths. In Appendix, we provide additional experimental results supporting these findings.

(a) Model Performance   (b) Parameter Efficiency   (c) Memory Efficiency   (d) Running Time Efficiency
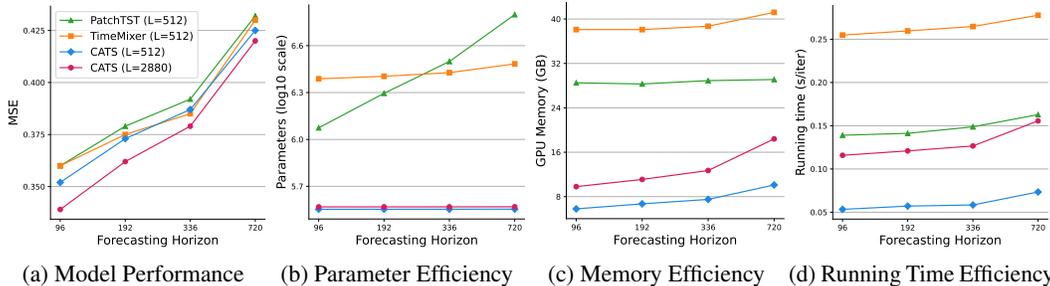
Figure 5: Efficiency and performance analysis for time series forecasting models. We summarize the forecasting performance, number of parameters, GPU memory consumption, and running time with varying forecasting horizon lengths on Traffic. The running time is averaged from 300 iterations.

Furthermore, we conduct a deeper comparison between Transformer-based models. Especially, TimeMixer [21] argues that their model outperforms PatchTST [14] in the setting of long input sequences. Regarding this setting, we also conduct an experiment on $L = 512$. We summarize the results in Fig. 5. Among these Transformer-based models, our model achieves the lowest MSE for most forecasting horizons. Moreover, our model requires even a lower number of parameters, GPU memory, and running time. Especially, for parameter efficiency, CATS shows significant differences even on a log scale due to its efficient parameter-sharing. Fig. 5c highlights GPU memory usage across different forecasting horizons. While PatchTST and TimeMixer consume significantly more memory, CATS maintains a low and stable memory consumption, demonstrating superior memory efficiency. In Fig. 5d CATS also consistently achieves lower running times compared to PatchTST and TimeMixer.

Additionally, we also compare the same factors when we use a longer input length $L = 2880$. As more input length is used, the forecasting performance of our model outperforms all other models. Most importantly, while the computational complexity increases as input length increases, our model

8

achieves a faster running time, compared to other models with a 512 input sequence length. Overall, these results emphasize the efficiency and performance advantages of our model, particularly in terms of parameter count, memory usage, and running time.

5.3 Short-term Time Series Forecasting Results

Table 6 summarizes the averaged results for the M4 dataset, comparing the performance of various models. Our CATS model consistently achieved the best results across all metrics. Notably, CATS reduced MASE by 19.94% compared to PatchTST, a self-attention-only model that suffers from temporal information loss as mentioned in Section 3. CATS, with its cross-attention-only structure, effectively mitigated this issue and captured temporal dependencies more efficiently than previous models. Although TimeMixer, a state-of-the-art linear model, performed well, CATS surpassed it across all metrics. This demonstrates that CATS excels at capturing short-term temporal dependencies, providing superior performance in short-term forecasting tasks.

Table 6: Averaged univariate short-term forecasting results in the M4 dataset. The best results are in **bold** and the second best are underlined. Full results are presented in Appendix.

	Models	CATS	TimeMixer	Timesnet	PatchTST	MICN	FiLM	DLinear	Autoformer	Informer
Average	SMAPE	**11.701**	<u>11.723</u>	11.829	13.152	19.638	14.863	13.639	12.909	14.086
	MASE	**1.557**	<u>1.559</u>	1.585	1.945	5.947	2.207	2.095	1.771	2.718
	OWA	**0.838**	<u>0.840</u>	0.851	0.998	2.279	1.125	1.051	0.939	1.230

5.4 Replacement of Cross-Attention with Self-Attention

In our propose structure, we mainly use cross-attention rather than self-attention due to the forecasting-unfriendly properties of self-attention. To verify the effectiveness of cross-attention in the proposed structure, we replace the cross-attention layers with self-attention layers while maintaining other structures. In Table 7, we gradually replace the cross-attention with self-attention (SA) among a total of three cross-attention layers. To maintain the original Transformer structure, we set the maximum replacement as two. As shown in this table, we confirm the effectiveness of the cross-attention mechanism compared to using self-attention layers. Specifically, the zero SA, which is our model, shows better performance than using SA for almost all cases except only one case.

Table 7: Performance comparison on models with three attention layers. We replace one or more cross-attentions (CA) with self-attentions (SA) in our model. In total, there are three cross-attentions in all settngs and 'Zero SA' stands for our model. The best results are in **bold**.

Dataset		Electricity						ETTm1					
Case		Zero SA		One SA		Two SA		Zero SA		One SA		Two SA	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
96		**0.126**	**0.218**	0.128	0.220	0.133	0.225	**0.283**	0.340	0.284	**0.338**	0.284	0.340
192		**0.144**	**0.235**	0.150	0.238	0.153	0.245	**0.319**	**0.363**	0.331	0.373	0.324	0.368
336		**0.159**	**0.252**	0.167	0.257	0.169	0.263	**0.351**	**0.385**	0.376	0.401	0.369	0.400
720		**0.194**	**0.283**	0.205	0.293	0.210	0.300	**0.400**	**0.414**	0.429	0.437	0.442	0.445

5.5 Explaining Periodic Patterns through Cross-Attention

As noted in Section 4.2, in our proposed model, all operations including embedding and multi-head attention are performed independently for each learnable query. In other words, the forecast for a specific horizon does not depend on other horizons. This approach helps us better understand how each prediction is derived. Therefore, in this subsection, we visualize how the proposed model understands the periodic properties.

To provide an easy understanding, we here consider a simple time series forecasting task with data that consists of two independent signals as follows:

$$\mathbf{x}(t) = \{x_{(t \bmod \tau)}\}_{t=1}^{\infty}, \quad x_i \sim \mathcal{N}(0,1) \quad (i = 0, 1, \ldots, \tau - 1), \tag{1}$$

$$\boldsymbol{y}(t) = \begin{cases} +k & \text{if } t \equiv 0 \quad (\bmod S) \\ -k & \text{if } t \equiv \frac{1}{2}S \quad (\bmod S). \end{cases} \tag{2}$$

9

For prediction, we use an input sequence length $L = 48$ and a forecasting horizon $T = 72$ with signals $\mathbf{x}(t)$ and $\boldsymbol{y}(t)$ are defined with $\tau = 24$, $S = 8$, and $k = 5$. The patch length is set to 4 without overlapping to elucidate the distinct periodic components with 2 attention heads.

(a) Cross-attention score map of head 1   (b) Cross-attention score map of head 2
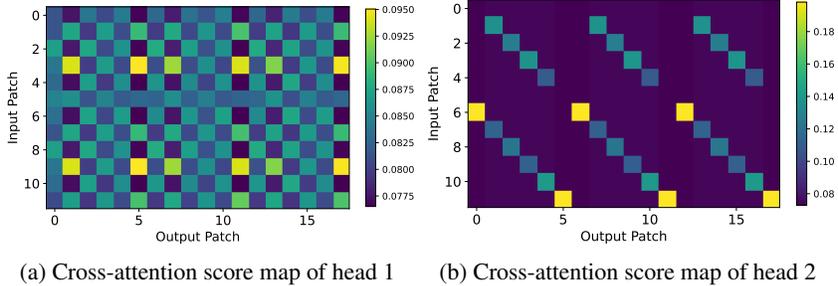
Figure 6: Score map of cross-attentions between input and output patches.

In Fig. 6, we illustrate a score map (12×18) of the cross-attention from the trained CATS. Since both patch length and stride are set to 4, each patch will contain exactly one shock value. We observe that the cross-attentions capture the shocks within the signal and the periodicity of the signal in Fig. 6a and Fig. 6b, respectively. Fig. 6a shows that patches an even number of steps before the current patch contain the shocks of the same direction, resulting in higher attention scores, while odd-numbered steps have lower scores. Moreover, the correlation over 24 steps is clearly demonstrated in patches spaced by multiples of 6 steps, as shown in Fig. 6b. This periodic pattern ensures that the attention mechanism effectively captures the periodicity in $\mathbf{x}(t)$, reflecting the model's ability to leverage this periodic information for more accurate predictions. In Appendix, we provide a detailed explanation.

(a) Forecasting results   (b) Averaged score   (c) L2-H2   (d) L4-H4
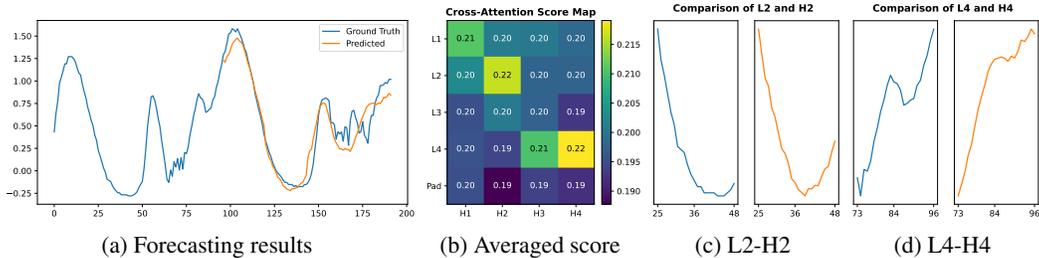
Figure 7: Illustration of (a) forecasting result, (b) averaged cross-attention score, and (c,d) patches with the highest score on ETTm1. The score map is averaged from all the heads across layers.

Fig. 7 illustrates (a) forecasting results, (b) a cross-attention score map (5×4) on the ETTm1 dataset, and (c, d) the two pairs with the highest attention scores. We predict 96 steps with input sequence length 96 on ETTm1. The input patches consist of four patches of 24 lengths and one padding patch. As shown in Fig. 7c and 7d, the patches with high attention scores exhibit similar temporal patterns, demonstrating the ability of CATS to detect sequential and periodic patterns.

6 Conclusion

Based on our study, we exploit the advantages of Transformer models in time series forecasting by removing self-attentions and developing a new cross-attention-based architecture. We believe that our model establishes a strong baseline for such forecasting tasks and offers further insights into the complexities of long-term forecasting problems. Our findings provide a reevaluation of self-attentions in this domain, and we hope that future research can critically assess the efficacy and efficiency across various time series analysis tasks. As a limitation, our proposed methods assume channel independence between variables based on the recent work [14]. As the time series data in the real-world are highly correlated, we hope future research can address cross-variate dependency with reduced computation complexity based on the proposed architecture.

10

7 Acknowledgements

This work was partly supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-II220984, Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation) and the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2024-00338859). This work was also supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2024-RS-2024-00438056) supervised by the IITP.

References

[1] Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 357–366, 2021.

[2] Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023.

[3] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*. PMLR, 2024.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Lee Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[6] Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 459–469, 2023.

[7] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

[8] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016.

[9] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[10] Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721*, 2023.

[11] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*, 2021.

[12] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, 2022.

11

[13] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=JePfAI8fah`.

[14] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=Jbdc0vTOcol`.

[15] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[16] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[17] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021.

[18] Gerrit JJ Van den Burg and Christopher KI Williams. An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*, 2020.

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

[20] Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*, 2022.

[21] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=7oLshfEIC2`.

[22] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Learning deep time-index models for time series forecasting. In *International Conference on Machine Learning*, pages 37217–37237. PMLR, 2023.

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

[24] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The eleventh international conference on learning representations*, 2022.

[25] Chenglin Yang, Yilin Wang, Jianming Zhang, He Zhang, Zijun Wei, Zhe Lin, and Alan Yuille. Lite vision transformer with enhanced self-attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11998–12008, 2022.

[26] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[27] Haokui Zhang, Wenze Hu, and Xiaoyu Wang. Fcaformer: Forward cross attention in hybrid vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6060–6069, 2023.

[28] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2022.

12

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

[30] Tian Zhou, Ziqing Ma, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, Rong Jin, et al. Film: Frequency improved legendre memory model for long-term time series forecasting. *Advances in Neural Information Processing Systems*, 35:12677–12690, 2022.

[31] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

13

A Experimental settings

A.1 Datasets

We evaluated the performance using seven datasets commonly used in long-term time series forecasting, including Weather, Traffic, Electricity, ETT (ETTh1, ETTh2, ETTm1, and ETTm2), and M4. These datasets capture a range of periodic characteristics and scenarios that are difficult to predict in the real world, making them highly suitable for tasks, such as long-term time series forecasting, generation, and imputation. Details of these datasets are described in Table 8. The M4 dataset are provided Wu et al. [24], while the remaining datasets are provided in Wu et al. [23].

Table 8: Details of 13 real-world datasets.

	Dimension	Frequency	Timesteps	Information	Forecasting Horizon
Weather	21	10-min	52,696	Weather	(96, 192, 336, 720)
Electricity	321	Hourly	17,544	Electricity	(96, 192, 336, 720)
Traffic	862	Hourly	26,304	Transportation	(96, 192, 336, 720)
ETTh1	7	Hourly	17,420	Temperature	(96, 192, 336, 720)
ETTh2	7	Hourly	17,420	Temperature	(96, 192, 336, 720)
ETTm1	7	15-min	69,680	Temperature	(96, 192, 336, 720)
ETTm2	7	15-min	69,680	Temperature	(96, 192, 336, 720)
M4-Quartely	1	Quartely	48000	Finance	8
M4-Monthly	1	Monthly	96000	Industry	18
M4-Yearly	1	Yearly	46000	Demographic	6
M4-Weekly	1	Weekly	718	Macro	13
M4-Daily	1	Daily	8454	Micro	14
M4-Hourly	1	Hourly	828	Other	48

A.2 Hyperparameter Settings

In every experiment in our paper, following [14], we fixed the random seed of 2021 to enhance the reproducibility of our experiments. Additionally, following numerous studies in the field of time series forecasting [14], we fixed the input sequence length $L = 96$. For the forecasting horizon T, we also used the widely accepted values, i.e., $[96, 192, 336, 720]$. For our model, in all configurations, we adopt the GeGLU activation function [16] between the two linear layers in the feed-forward network for our model. Additionally, we use learnable positional embedding parameters for the input data and omit positional embeddings for learnable queries to avoid redundant parameter learning.

For the experiments summarized in Table 4 and Table 11, our model uses three cross-attention layers with embedding size $D = 256$, number of attention heads $H = 32$. Specifically, to avoid overfitting on small datasets [14], we use patch length 48 on the ETTh1 and ETTh2 datasets. Further details on the hyperparameter settings for these experiments are provided in Table 9.

Table 9: Experimental settings used in Table 4 and Table 11.

Metric	Layers	Embedding Size	Query Sharing	Input Sequence Length	Batch Size	Epoch	Learning Rate
Weather	3	256	False	96	64	30	10^{-3}
Electricity	3	256	False	96	32	30	10^{-3}
Traffic	3	256	True	96	32	100	10^{-3}
ETTh1	3	256	False	96	256	10	10^{-3}
ETTh2	3	256	True	96	256	10	10^{-3}
ETTm1	3	256	False	96	128	30	10^{-3}
ETTm2	3	256	True	96	128	30	10^{-3}

Additionally, for the short-term forecasting experiments on the M4 dataset, we employed a slightly different configuration to better suit the nature of short-term time series. The hyperparameter settings for these experiments are detailed in Table 10. The complete results for these short-term experiments

14

are presented in Table 12, while the full results for the long-term forecasting experiments with a fixed 96 input sequence length are provided in the Appendix in Table 11.

Table 10: Experimental settings of the M4 dataset.

Dataset	Layers	Embedding Size	Input Sequence Length	Batch Size	Epoch	Patience	Learning Rate
M4-Quartely	3	64	16	32	30	10	10^{-3}
M4-Monthly	3	64	36	32	30	10	10^{-3}
M4-Yearly	3	64	12	32	30	10	10^{-3}
M4-Weekly	3	64	26	32	30	10	10^{-3}
M4-Daily	3	64	28	32	30	10	10^{-3}
M4-Hourly	3	128	96	32	30	10	10^{-3}

Table 11: Multivariate long-term forecasting results with recent forecasting models and ours for unified hyperparameter settings. The best results are in **bold** and the second best are underlined.

Dataset	Horizon	CATS MSE	CATS MAE	TimeMixer MSE	TimeMixer MAE	PatchTST MSE	PatchTST MAE	Timesnet MSE	Timesnet MAE	Crossformer MSE	Crossformer MAE	MICN MSE	MICN MAE	FiLM MSE	FiLM MAE	DLinear MSE	DLinear MAE	Autoformer MSE	Autoformer MAE	Informer MSE	Informer MAE
Weather	96	**0.161**	**0.207**	0.163	0.209	0.186	0.227	0.172	0.220	0.195	0.271	0.198	0.261	0.195	0.236	0.195	0.252	0.266	0.336	0.300	0.384
Weather	192	**0.208**	**0.250**	**0.208**	**0.250**	0.234	0.265	0.219	0.261	0.209	0.277	0.239	0.299	0.239	0.271	0.237	0.295	0.307	0.367	0.598	0.544
Weather	336	0.264	0.290	0.251	**0.287**	0.284	0.301	**0.246**	0.337	0.273	0.332	0.285	0.336	0.289	0.306	0.282	0.331	0.359	0.395	0.578	0.523
Weather	720	0.342	**0.341**	**0.339**	**0.341**	0.356	0.349	0.365	0.359	0.379	0.401	0.351	0.388	0.361	0.351	0.345	0.382	0.419	0.428	1.059	0.741
Electricity	96	**0.149**	**0.237**	0.153	0.247	0.190	0.296	0.168	0.272	0.219	0.314	0.180	0.293	0.198	0.274	0.210	0.302	0.201	0.317	0.274	0.368
Electricity	192	**0.163**	**0.250**	0.166	0.256	0.199	0.304	0.184	0.322	0.231	0.322	0.189	0.302	0.198	0.278	0.210	0.305	0.222	0.334	0.296	0.386
Electricity	336	**0.180**	**0.268**	0.185	0.277	0.217	0.319	0.198	0.300	0.246	0.337	0.198	0.312	0.217	0.300	0.223	0.319	0.231	0.443	0.300	0.394
Electricity	720	0.219	0.302	0.225	0.310	0.258	0.352	0.220	0.320	0.280	0.363	**0.217**	0.330	0.278	0.356	0.258	0.350	0.254	0.361	0.373	0.439
Traffic	96	**0.421**	**0.270**	0.462	0.285	0.526	0.347	0.593	0.321	0.644	0.429	0.577	0.350	0.647	0.384	0.650	0.396	0.613	0.388	0.719	0.391
Traffic	192	**0.436**	**0.275**	0.473	0.296	0.522	0.332	0.617	0.336	0.665	0.431	0.589	0.356	0.600	0.361	0.598	0.370	0.616	0.382	0.696	0.379
Traffic	336	**0.453**	**0.284**	0.498	0.296	0.517	0.334	0.629	0.336	0.674	0.420	0.594	0.358	0.610	0.367	0.605	0.373	0.622	0.337	0.777	0.420
Traffic	720	**0.484**	**0.303**	0.506	0.313	0.552	0.352	0.640	0.350	0.683	0.421	0.613	0.361	0.691	0.425	0.645	0.394	0.660	0.408	0.864	0.472
ETTm1	96	**0.318**	**0.357**	0.320	**0.357**	0.352	0.374	0.338	0.375	0.404	0.426	0.365	0.387	0.353	0.370	0.346	0.374	0.505	0.475	0.672	0.571
ETTm1	192	**0.357**	**0.377**	0.361	0.381	0.390	0.393	0.374	0.387	0.450	0.451	0.403	0.408	0.389	0.387	0.382	0.391	0.553	0.496	0.795	0.669
ETTm1	336	**0.387**	**0.401**	0.390	0.404	0.421	0.414	0.410	0.411	0.532	0.515	0.436	0.431	0.421	0.408	0.415	0.415	0.621	0.537	1.212	0.871
ETTm1	720	**0.448**	**0.437**	0.454	0.441	0.462	0.449	0.478	0.450	0.666	0.589	0.489	0.462	0.481	0.441	0.473	0.451	0.671	0.561	1.166	0.823
ETTm2	96	0.178	0.261	**0.175**	**0.258**	0.183	0.270	0.187	0.267	0.287	0.366	0.197	0.296	0.183	0.266	0.193	0.293	0.255	0.339	0.365	0.453
ETTm2	192	0.248	0.308	**0.237**	**0.299**	0.255	0.314	0.249	0.309	0.414	0.492	0.284	0.361	0.248	0.305	0.284	0.361	0.281	0.340	0.533	0.563
ETTm2	336	0.304	0.343	**0.298**	**0.340**	0.309	0.347	0.321	0.351	0.597	0.542	0.381	0.429	0.309	0.343	0.382	0.429	0.339	0.372	1.363	0.887
ETTm2	720	0.402	0.402	**0.391**	**0.396**	0.412	0.404	0.408	0.403	1.730	1.042	0.549	0.522	0.410	0.400	0.558	0.525	0.433	0.432	3.379	1.338
ETTh1	96	**0.371**	**0.395**	0.375	0.400	0.460	0.447	0.384	0.402	0.423	0.448	0.426	0.446	0.438	0.433	0.397	0.412	0.449	0.459	0.865	0.713
ETTh1	192	**0.426**	0.422	0.429	**0.421**	0.512	0.477	0.436	0.429	0.471	0.474	0.454	0.464	0.493	0.466	0.446	0.441	0.500	0.482	1.008	0.792
ETTh1	336	**0.437**	**0.432**	0.484	0.458	0.546	0.496	0.638	0.469	0.570	0.546	0.493	0.487	0.547	0.495	0.489	0.467	0.521	0.496	1.107	0.809
ETTh1	720	**0.474**	**0.461**	0.498	0.482	0.544	0.517	0.521	0.500	0.653	0.621	0.526	0.526	0.586	0.538	0.513	0.510	0.514	0.512	1.181	0.865
ETTh2	96	**0.287**	**0.341**	0.289	0.355	0.308	0.355	0.340	0.374	0.745	0.584	0.372	0.424	0.322	0.364	0.340	0.394	0.346	0.388	3.755	1.525
ETTh2	192	**0.361**	**0.388**	0.372	0.392	0.393	0.405	0.402	0.414	0.877	0.656	0.492	0.492	0.404	0.414	0.482	0.479	0.456	0.453	5.602	1.931
ETTh2	336	**0.374**	**0.403**	0.386	0.414	0.427	0.436	0.452	0.452	1.043	0.731	0.607	0.555	0.435	0.445	0.591	0.541	0.482	0.486	4.721	1.835
ETTh2	720	**0.412**	**0.433**	**0.412**	0.434	0.436	0.450	0.462	0.468	1.104	0.763	0.824	0.655	0.447	0.458	0.839	0.661	0.515	0.511	3.647	1.625

Table 12: Full results of univariate short-term forecasting in the M4 dataset. All forecasting horizons are in [6, 48]. The best results are in **bold** and the second best are underlined.

	Models	CATS	TimeMixer	Timesnet	PatchTST	MICN	FiLM	DLinear	Autoformer	Informer
Quartely	SMAPE	**9.979**	9.996	10.100	10.644	15.214	12.925	12.145	11.338	11.360
Quartely	MASE	**1.164**	1.166	1.182	1.278	1.963	1.664	1.520	1.365	1.401
Quartely	OWA	0.878	**0.825**	0.890	0.949	1.407	1.193	1.106	1.012	1.027
Monthly	SMAPE	**12.557**	12.605	12.670	13.399	16.943	15.407	13.514	13.958	14.062
Monthly	MASE	**0.916**	0.919	0.933	1.031	1.442	1.298	1.037	1.103	1.141
Monthly	OWA	**0.866**	0.869	0.878	0.949	1.265	1.144	0.956	1.002	1.024
Yearly	SMAPE	13.263	**13.206**	13.387	16.463	25.022	17.431	16.965	13.974	14.727
Yearly	MASE	2.967	**2.916**	2.996	3.967	7.162	4.043	4.283	3.134	3.418
Yearly	OWA	0.779	**0.776**	0.786	1.003	1.667	1.042	1.058	0.822	0.881
Others	SMAPE	**4.560**	4.564	4.891	6.558	41.985	7.134	6.709	5.485	24.460
Others	MASE	**3.107**	3.115	3.302	4.511	62.734	5.09	4.953	3.865	20.960
Others	OWA	**0.970**	0.982	1.035	1.401	14.313	1.553	1.487	1.187	5.879
Average	SMAPE	**11.701**	11.723	11.829	13.152	19.638	14.863	13.639	12.909	14.086
Average	MASE	**1.557**	1.559	1.585	1.945	5.947	2.207	2.095	1.771	2.718
Average	OWA	**0.838**	0.840	0.851	0.998	2.279	1.125	1.051	0.939	1.230

15

B   Additional Experimental Results

B.1   Performance with Longer Input Sequences

In Table 5, we compared models with the number of parameters, GPU memory consumption, and MSE across different input lengths on ETTm1 with varying input sequence lengths. In this section, we provide comprehensive results on longer input sequence lengths $L = 512$. The detailed parameters can be found in Table 13 and the corresponding experimental results are summarized in Table 14. As with unified hyperparameter settings, we follow the settings of the most recent work [21] to ease comparison. Overall, the experimental results clearly illustrate the superiority of CATS over recent forecasting models across multiple datasets and prediction horizons. CATS consistently shows the lowest Mean Squared Error (MSE) and Mean Absolute Error (MAE) across a variety of datasets and forecast horizons. For instance, on Electricity, at the 96 forecast horizon, CATS achieves the best MSE score of 0.144 and the best MAE score of 0.189, underscoring its accuracy in predicting electrical demand.

Table 13: Experimental settings with an input sequence length of 512.

Metric	Layers	Embedding Size	Query Sharing	Input Sequence Length	Batch Size	Epoch	Learning Rate
Weather	3	128	False	512	128	30	10^{-3}
Electricity	3	128	False	512	32	30	10^{-3}
Traffic	3	128	True	512	32	100	10^{-3}
ETTh1	3	256	False	512	128	10	10^{-3}
ETTh2	3	256	True	512	256	10	10^{-3}
ETTm1	3	128	False	512	128	30	10^{-3}
ETTm2	3	256	True	512	128	30	10^{-3}

Table 14: Multivariate long-term forecasting results with an input sequence length of 512. The best results are in **bold** and the second best are underlined.

Models		CATS		TimeMixer		PatchTST		Timesnet		Crossformer		MICN		FiLM		DLinear		Autoformer		Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.144**	0.199	0.147	**0.197**	0.149	0.198	0.172	0.220	0.232	0.302	0.161	0.229	0.199	0.262	0.176	0.237	0.266	0.336	0.300	0.384
	192	**0.188**	0.240	0.189	**0.239**	0.194	0.241	0.219	0.261	0.371	0.410	0.220	0.281	0.228	0.288	0.220	0.282	0.307	0.367	0.598	0.544
	336	**0.238**	**0.280**	0.241	**0.280**	0.306	0.282	0.246	0.337	0.495	0.515	0.278	0.331	0.267	0.323	0.265	0.319	0.359	0.395	0.578	0.523
	720	**0.308**	**0.329**	0.310	0.330	0.314	0.334	0.365	0.359	0.526	0.542	0.311	0.356	0.319	0.361	0.323	0.362	0.419	0.428	0.590	0.741
Electricity	96	**0.126**	**0.218**	0.129	0.224	0.129	0.222	0.168	0.272	0.150	0.251	0.164	0.269	0.154	0.267	0.140	0.237	0.201	0.317	0.274	0.368
	192	0.144	0.235	**0.140**	**0.220**	0.147	0.240	0.184	0.322	0.161	0.260	0.177	0.285	0.164	0.258	0.153	0.249	0.222	0.334	0.296	0.386
	336	**0.159**	**0.252**	0.161	0.255	0.163	0.259	0.198	0.300	0.182	0.281	0.193	0.304	0.188	0.283	0.169	0.267	0.231	0.338	0.300	0.394
	720	**0.194**	**0.283**	0.194	0.287	0.197	0.290	0.220	0.320	0.251	0.339	0.212	0.321	0.236	0.332	0.203	0.301	0.254	0.361	0.373	0.439
Traffic	96	**0.352**	0.243	0.360	0.249	0.360	0.249	0.593	0.321	0.514	0.267	0.519	0.309	0.416	0.294	0.410	0.282	0.613	0.388	0.719	0.391
	192	**0.373**	0.253	0.375	0.250	0.379	0.256	0.617	0.336	0.549	0.252	0.537	0.315	0.408	0.288	0.423	0.287	0.616	0.382	0.696	0.379
	336	0.387	0.260	0.385	0.270	0.392	0.264	0.629	0.336	0.530	0.300	0.534	0.313	0.425	0.298	0.436	0.296	0.622	0.337	0.777	0.420
	720	**0.425**	**0.281**	0.430	0.281	0.432	0.286	0.640	0.350	0.573	0.313	0.577	0.325	0.520	0.353	0.466	0.315	0.660	0.408	0.864	0.472
ETTh1	96	0.373	0.401	**0.361**	**0.390**	0.370	0.400	0.384	0.402	0.418	0.438	0.421	0.431	0.422	0.432	0.375	0.399	0.449	0.459	0.865	0.713
	192	**0.401**	0.421	0.409	**0.414**	0.413	0.429	0.436	0.429	0.539	0.517	0.474	0.487	0.462	0.458	0.405	0.416	0.500	0.482	0.080	0.792
	336	**0.415**	0.434	0.430	0.429	0.422	0.440	0.638	0.469	0.709	0.638	0.569	0.551	0.501	0.483	0.439	0.443	0.521	0.496	0.107	0.809
	720	**0.435**	**0.446**	0.445	0.460	0.447	0.468	0.521	0.500	0.733	0.636	0.770	0.672	0.544	0.526	0.472	0.490	0.514	0.512	0.181	0.865
ETTh2	96	**0.256**	**0.328**	0.271	0.330	0.274	0.337	0.340	0.374	0.425	0.463	0.299	0.364	0.323	0.370	0.289	0.353	0.358	0.397	0.755	0.525
	192	0.311	0.366	0.317	0.402	0.314	0.382	**0.231**	**0.322**	0.473	0.500	0.441	0.454	0.391	0.415	0.383	0.418	0.456	0.453	0.602	0.931
	336	**0.319**	**0.382**	0.332	0.396	0.329	0.384	0.452	0.452	0.581	0.562	0.654	0.567	0.415	0.440	0.448	0.465	0.482	0.486	0.721	0.835
	720	0.395	0.438	**0.342**	**0.408**	0.379	0.422	0.462	0.468	0.775	0.665	0.956	0.716	0.441	0.459	0.605	0.551	0.515	0.511	0.647	0.625
ETTm1	96	**0.283**	**0.340**	0.291	0.340	0.293	0.346	0.338	0.375	0.361	0.403	0.316	0.362	0.302	0.345	0.299	0.343	0.505	0.475	0.672	0.571
	192	**0.319**	**0.363**	0.327	0.365	0.333	0.370	0.374	0.387	0.387	0.422	0.363	0.390	0.338	0.368	0.335	0.365	0.553	0.496	0.795	0.669
	336	**0.351**	0.385	0.360	0.381	0.369	0.392	0.410	0.411	0.605	0.572	0.408	0.426	0.373	0.388	0.369	0.386	0.621	0.537	0.212	0.871
	720	**0.400**	**0.414**	0.415	0.417	0.416	0.420	0.478	0.450	0.703	0.638	0.481	0.476	0.420	0.420	0.425	0.421	0.671	0.561	0.166	0.823
ETTm2	96	0.165	0.256	**0.164**	**0.254**	0.166	0.256	0.187	0.267	0.275	0.358	0.179	0.275	0.165	0.256	0.167	0.260	0.255	0.339	0.365	0.453
	192	**0.221**	0.297	0.223	**0.295**	0.223	0.296	0.249	0.309	0.345	0.400	0.307	0.376	0.222	0.296	0.224	0.303	0.281	0.340	0.533	0.563
	336	**0.274**	0.334	0.279	0.330	**0.274**	**0.329**	0.321	0.351	0.657	0.528	0.325	0.388	0.277	0.333	0.281	0.342	0.339	0.372	0.363	0.887
	720	0.362	0.390	**0.359**	**0.383**	0.362	0.385	0.408	0.403	0.208	0.753	0.502	0.490	0.371	0.389	0.397	0.421	0.422	0.419	0.379	0.388

B.2   Additional Results for Section 5.2

We provide additional experimental results to support the findings discussed in Section 5.2. Tables 15 and 16 summarize detailed comparisons of the number of parameters, GPU memory consumption,

16

Table 15: Comparison of models with the number of parameters, GPU memory consumption, and MSE across different input sequence lengths on ETTm1. Full results of Table 5.

Models	Parameters across different input lengths						
	96	192	336	512	720	1440	2880
PatchTST	1,506,384	2,612,304	4,271,184	6,298,704	8,694,864	16,989,264	33,578,064
TimeMixer	190,313	484,217	1,129,193	2,250,137	4,046,633	14,211,593	52,912,313
DLinear	139,680	277,920	485,280	738,720	1,038,240	2,075,040	4,148,640
CATS	360,264	360,776	361,544	362,440	363,592	367,432	375,112

Models	GPU Memory Consumption across different input lengths						
	96	192	336	512	720	1440	2880
PatchTST	2,234MB	2,650MB	3,484MB	4,914MB	7,368MB	2,1968MB	58,590MB
TimeMixer	2,204MB	2,522MB	2,914MB	3,414MB	3,888MB	5,876MB	10,324MB
DLinear	1,098MB	1,102MB	1,104MB	1,114MB	1,114MB	1,154MB	1,214MB
CATS	1,712MB	1,796MB	1,884MB	2,042MB	2,140MB	2,700MB	3,826MB

Models	MSE across different input lengths						
	96	192	336	512	720	1440	2880
PatchTST	0.457	0.424	0.418	0.420	0.418	0.420	0.413
TimeMixer	0.454	0.433	0.428	0.436	0.425	0.414	0.472
DLinear	0.473	0.438	0.426	0.427	0.422	0.401	0.408
CATS	**0.450**	**0.418**	**0.407**	**0.400**	**0.402**	**0.399**	**0.395**

Table 16: Comparison of models with the number of parameters, GPU memory consumption, and MSE across different input sequence lengths on Weather.

Models	Parameters across different input lengths						
	96	192	336	512	720	1440	2880
PatchTST	1,506,384	2,612,304	4,271,184	6,298,704	8,694,864	16,989,264	33,578,064
TimeMixer	219,249	595,305	1,465,569	3,028,185	5,582,529	20,343,969	77,423,049
DLinear	139,680	277,920	485,280	738,720	1,038,240	2,075,040	4,148,640
CATS	370,344	370,856	371,624	372,520	373,672	377,512	385,192

Models	GPU Memory Consumption across different input lengths						
	96	192	336	512	720	1440	2880
PatchTST	1,680MB	2,110MB	2,762MB	4,596MB	5,726MB	16,472MB	45,278MB
TimeMixer	1,894MB	2,154MB	2,728MB	3,414MB	4,356MB	8,358MB	20,624MB
DLinear	1,106MB	1,114MB	1,188MB	1,188MB	1,188MB	1,362MB	1,632MB
CATS	1,522MB	1,590MB	1,665MB	1,755MB	1,892MB	2,282MB	3,140MB

Models	MSE across different input lengths						
	96	192	336	512	720	1440	2880
PatchTST	0.351	0.336	0.320	0.315	0.309	0.308	0.312
TimeMixer	**0.339**	0.331	0.318	0.319	0.324	0.318	0.327
DLinear	0.346	0.334	0.325	0.320	0.316	0.311	0.309
CATS	0.342	**0.325**	**0.314**	**0.308**	**0.305**	**0.301**	**0.291**

and MSE across different input lengths for the ETTm1 and Weather datasets, respectively. The linear models, TimeMixer and DLinear, exhibit smaller parameters for shorter input lengths. Despite CATS having slightly more parameters than TimeMixer for smaller inputs, it outperforms in terms of memory usage and MSE. This suggests that CATS is more efficient and effective in handling shorter inputs. For PatchTST, the number of parameters does not increase within the actual Transformer backbone as the input length increases. However, due to the need to flatten and project all inputs

17

at the end, the parameters scale linearly with the input length. This highlights a limitation of the Encoder's architecture. On the other hand, TimeMixer's parameters grow almost quadratically as the input length doubles. Similarly, DLinear's parameters increase linearly with the input length. Our proposed model, CATS demonstrates significant efficiency through parameter sharing, where the parameters hardly increase with longer inputs. Notably, from an input length of 336, CATS has fewer parameters than DLinear, showcasing the deep learning model's advantage in detecting inherent patterns in the data.

Regarding GPU memory consumption, we observe that both PatchTST and TimeMixer require significantly more GPU memory as the input length increases. For example, PatchTST's GPU memory usage scales drastically, making it less feasible for long input sequences. TimeMixer also shows an increase in GPU memory consumption, although it is less severe than PatchTST. In contrast, DLinear maintains a relatively constant GPU memory usage, demonstrating its efficiency in terms of computational resources. However, CATS stands out by offering a balanced approach, with moderate GPU memory usage that scales more favorably compared to PatchTST and TimeMixer. This balance between memory efficiency and performance is crucial for practical applications requiring long-term time series forecasting.

Furthermore, when analyzing the MSE across different input lengths, CATS consistently shows the best performance. It maintains lower MSE compared to other models across all input lengths. This robustness in performance, combined with its efficient parameter and memory usage, highlights the superiority of CATS in long-term time series forecasting tasks. Overall, these results show the advantages of CATS in terms of parameter efficiency, GPU memory consumption, and forecasting accuracy. These findings support the proposed model's potential for practical and scalable time series forecasting solutions.

Table 17 presents the full results on the Traffic dataset. Here, we use the Traffic dataset with a batch size of 8. All GPU memory consumption was measured in a setting using four multi-GPUs. As shown in Table 17, CATS with a 2880 input sequence length consistently outperforms models with a 512 input sequence length, including PatchTST and TimeMixer. Specifically, CATS demonstrates fewer parameters, lower GPU memory consumption, and faster running speeds. These results highlight the efficiency of CATS with large input sizes. The Traffic dataset, characterized by high-dimensional data, shows a significant reduction in MSE when using longer input sequences.

Table 18 provides the full results on the Electricity dataset. Similar to the Traffic dataset, CATS shows superior efficiency in training, particularly with an input size of 2880, across all cases. Here, we use the Electricity dataset with a batch size of 32. All GPU memory consumption was measured in a setting using four multi-GPUs. In this experiment, CATS with a 512 input sequence length did not use parameter sharing for queries, while CATS with a 2880 input sequence length did. This demonstrates the effectiveness of query parameter sharing when utilizing large amounts of data for training. The results confirm that query sharing among dimensions leads to greater efficiency and improved performance.

18

Table 17: Comparison of models with the number of parameters, GPU memory consumption, running speed, and MSE across different forecasting horizon sizes on Traffic. Full results of Fig. 5.

Horizon	Models	Paramters	Gpu Memory	Running Time	MSE
96	PatchTST	1,186,272	28.54GB	0.1390s/iter	0.360
	TimeMixer	2,442,961	38.12GB	0.2548s/iter	0.360
	CATS ($L = 512$)	357,496	5.81GB	0.0533s/iter	0.352
	CATS ($L = 2880$)	370,168	9.79GB	0.1158s/iter	**0.339**
192	PatchTST	1,972,800	28.34GB	0.1412s/iter	0.379
	TimeMixer	2,535,505	38.13GB	0.2596s/iter	0.375
	CATS ($L = 512$)	357,592	6.73GB	0.0571s/iter	0.373
	CATS ($L = 2880$)	370,264	11.10GB	0.1209s/iter	**0.362**
336	PatchTST	3,152,592	28.91GB	0.1487s/iter	0.392
	TimeMixer	2,674,321	38.69GB	0.2647s/iter	0.385
	CATS ($L = 512$)	357,736	7.46GB	0.0584s/iter	0.387
	CATS ($L = 2880$)	370,408	12.72GB	0.1266s/iter	**0.379**
720	PatchTST	6,298,704	29.15GB	0.1628s/iter	0.432
	TimeMixer	3,044,497	41.17GB	0.2777s/iter	0.430
	CATS ($L = 512$)	358,120	10.10GB	0.0734s/iter	0.423
	CATS ($L = 2880$)	370,792	18.40GB	0.1556s/iter	**0.420**

Table 18: Comparison of models with the number of parameters, GPU memory consumption, running speed, and MSE across different forecasting horizon sizes on Electricity.

Horizon	Models	Paramters	Gpu Memory	Running Time	MSE
96	PatchTST	1,186,272	40.36GB	0.2021s/iter	0.129
	TimeMixer	2,429,049	33.80GB	0.2118s/iter	0.129
	CATS ($L = 512$)	388,216	6.89GB	0.0587s/iter	**0.126**
	CATS ($L = 2880$)	370,168	12.82GB	0.1653s/iter	**0.126**
192	PatchTST	1,972,800	40.39GB	0.2048s/iter	0.147
	TimeMixer	2,521,593	33.81GB	0.2212s/iter	0.140
	CATS ($L = 512$)	419,032	8.07GB	0.0636s/iter	0.144
	CATS ($L = 2880$)	370,264	14.70GB	0.1725s/iter	**0.139**
336	PatchTST	3,152,592	40.42GB	0.2070s/iter	0.163
	TimeMixer	2,660,409	34.24GB	0.2314s/iter	0.161
	CATS ($L = 512$)	465,256	9.15GB	0.0690s/iter	0.159
	CATS ($L = 2880$)	370,408	17.38GB	0.1839s/iter	**0.153**
720	PatchTST	6,298,704	41.40GB	0.2313s/iter	0.197
	TimeMixer	3,030,585	36.13GB	0.2478s/iter	0.194
	CATS ($L = 512$)	588,520	12.77GB	0.0964s/iter	0.194
	CATS ($L = 2880$)	370,792	25.86GB	0.2262s/iter	**0.183**

B.3 Ablation Study on Query-adaptive Masking

In this section, we demonstrate the effectiveness of query-adaptive masking compared to dropout, which is a widely adopted technique in Transformer-based forecasting models. We consider four different setups: using only dropout, using query-adaptive masking with fixed probabilities, query-adaptive masking with linearly increasing probabilities, and using both methods simultaneously. As shown in Fig. 8, the query-adaptive masking shows better forecasting performance and faster converge speed compared to dropout. Applying a gradually increasing masking probability based on the horizon predicted by the query shows slight performance improvements over using a fixed probability or combining with dropout. In contrast, using dropout alone shows noticeable differences in both convergence speed and overall performance. This demonstrates that when multiple inputs

19

with different forecasting horizons share a single model, probabilistic masking is more beneficial for model training than dropout.

(a) ETTm1 with $T = 720$ (b) Weather with $T = 720$
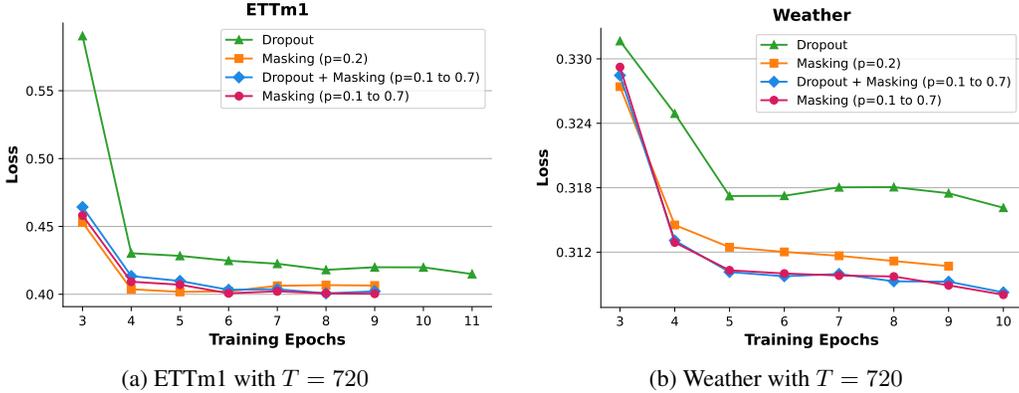
Figure 8: Comparison of performance with query-adaptive masking with two different probabilities, dropout, and using both query-adaptive masking and dropout. The results of $p = 0.1$ to 0.7 indicate a probability masking that is linearly increased proportionally to the horizon predicted by the query.

C Detailed Explanation of Section 5.5

In this section, we provide the detailed results of experiments in Section 5.5. We first restate the formulation of two independent signals used in Section 5.5 as follows:

$$\mathbf{x}(t) = \{x_{(t \bmod \tau)}\}_{t=1}^{\infty}, \quad x_i \sim \mathcal{N}(0,1) \quad (i = 0, 1, \ldots, \tau - 1),$$

$$\boldsymbol{y}(t) = \begin{cases} +k & \text{if } t \equiv 0 \pmod{S} \\ -k & \text{if } t \equiv \frac{1}{2}S \pmod{S} \end{cases},$$

We use the model parameters as follows: the patch length is 4 without overlapping, the decoder has 1 layer, and there are 2 attention heads. The signals $\mathbf{x}(t)$ and $\boldsymbol{y}(t)$ are defined with $\tau = 24$, $S = 8$, and $k = 5$. The visualization of synthetic data is shown in Fig. 9. We utilize an input sequence length $L = 48$ and a forecasting horizon $T = 72$. This setup allows us to generate time series data with distinct periodic components.

In the main paper, Fig. 6 displays a cross-attention score map between the input patch and the output patch derived from this experiment. The left figure presents the attention score of the first attention head, illustrating the model's ability to detect shocks within the signal. The right figure more clearly demonstrates the periodicity of the signal. Given that the patch length and stride are both set to 4, each patch will contain exactly one shock value, either -5 or +5. This is because the shocks occur every 4 steps, alternating between positive and negative shocks. Consequently, the patch immediately preceding the current patch will contain a different shock, leading to lower attention scores due to the differing shock values. In contrast, patches that are an even number of steps before the current patch will contain the same type of shock, resulting in higher attention scores. These points are well illustrated in Fig. 6a, where the varying attention scores correspond to the presence of alternating shocks. This pattern helps to highlight the alternating shock signal within the data.

Additionally, if there is a correlation with the series preceding 24 steps, the patches that are 6 steps or multiples of 6 steps before the current patch will exhibit high attention scores due to the periodic nature of the signal $\mathbf{x}(t)$. The diagonal formation of the attention scores, which accurately follows a period of 24, is clearly depicted in Fig. 6b, highlighting the model's capability to utilize fixed-period input patches to predict future outcomes. This periodic pattern ensures that the attention mechanism effectively captures the 24-step periodicity in $\mathbf{x}(t)$, reflecting the model's ability to leverage this periodic information for more accurate predictions.

This experimental configuration provides a robust framework to evaluate how well our proposed model captures and interprets the underlying patterns in the data, specifically focusing on the alternating

20

(a) Two input signals   (b) Sum of two input signals
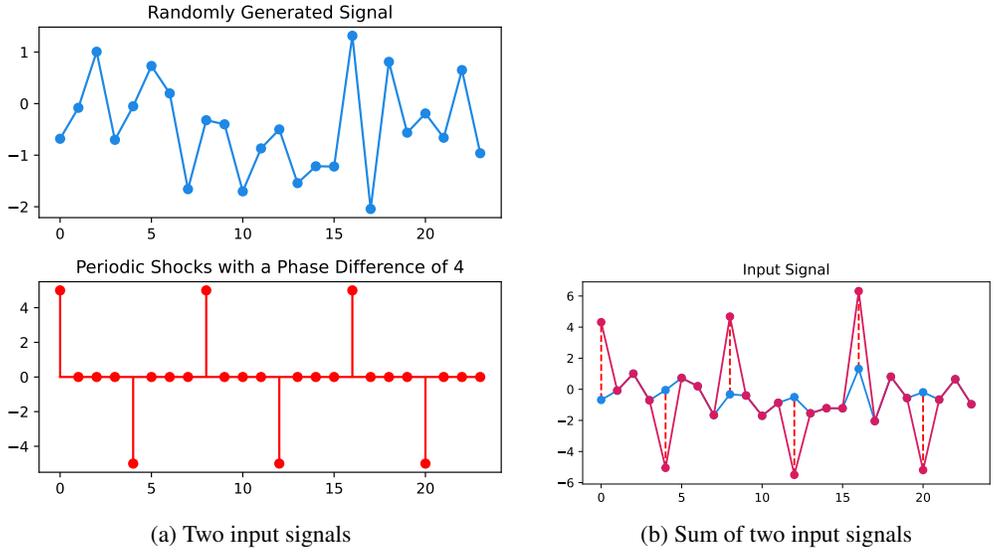
Figure 9: Visualization of input signals for toy experiment.

shock signal and the periodic nature of the normal signal. This dual emphasis on both the shock signal and the periodicity of the normal signal enhances the interpretability and predictive performance of the model, distinctly demonstrating how the model leverages periodic information to enhance prediction accuracy.

To push further, we reproduce the experiment of Fig. 7 with other datasets used in forecasting tasks. We illustrate the results of the Weather, Traffic, Electricity, ETTm2, ETTh1, and ETTh2 in Figures 10, 11, 12, 13, 14, and 15, respectively. For each figure, (a) represents the forecasting results, (b) shows the cross-attention score map, and (c) and (d) illustrate the two pairs with the highest attention scores. For all figures, our attention-based explanation successfully discovers similar periodic patterns. Therefore, we believe that our model has the potential to provide a clearer understanding of the mechanisms underlying forecasting predictions. We hope that future research will continue to explore and expand upon this foundation.

(a) Forecasting results   (b) Averaged score   (c) L1-H2   (d) L1-H3
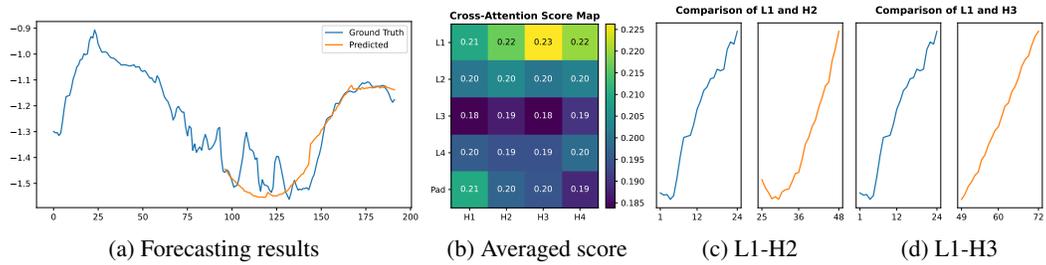
Figure 10: Illustration of (a) forecasting result, (b) averaged cross-attention score, and (c,d) patches with the highest score on Weather. The score map is averaged from all the heads across layers.

21
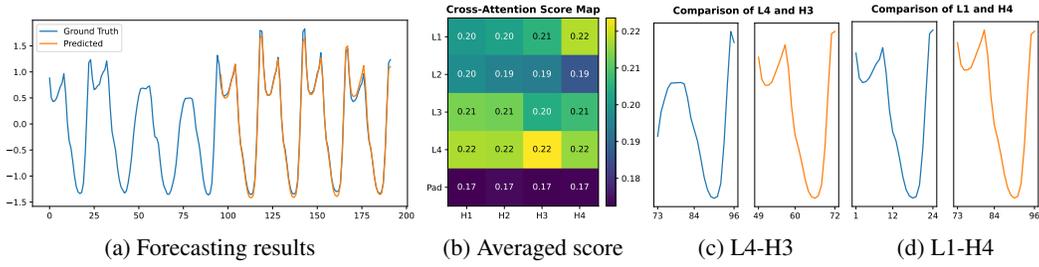
Figure 11: Illustration of (a) forecasting result, (b) averaged cross-attention score, and (c,d) patches with the highest score on Traffic. The score map is averaged from all the heads across layers.
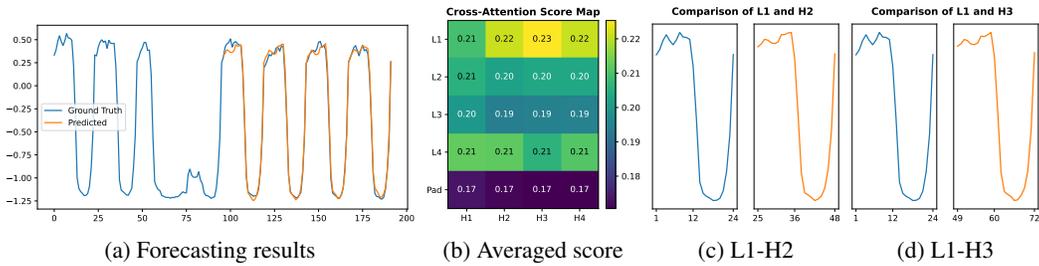
Figure 12: Illustration of (a) forecasting result, (b) averaged cross-attention score, and (c,d) patches with the highest score on Electricity. The score map is averaged from all the heads across layers.
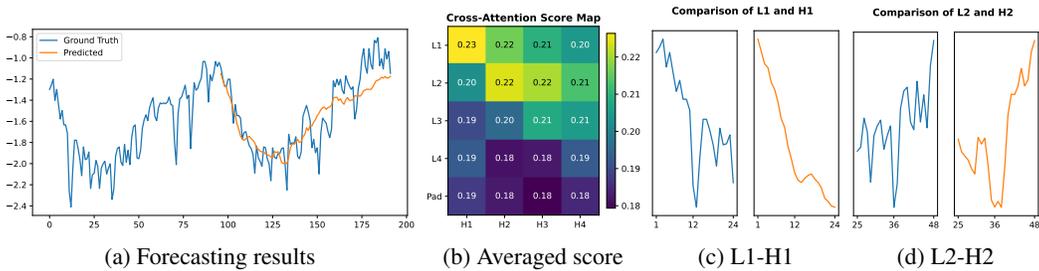
Figure 13: Illustration of (a) forecasting result, (b) averaged cross-attention score, and (c,d) patches with the highest score on ETTm2. The score map is averaged from all the heads across layers.
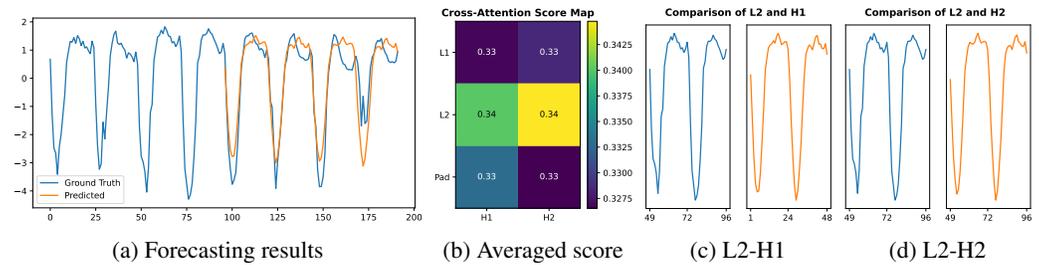
Figure 14: Illustration of (a) forecasting result, (b) averaged cross-attention score, and (c,d) patches with the highest score on ETTh1. The score map is averaged from all the heads across layers.

22

(a) Forecasting results   (b) Averaged score   (c) L1-H1   (d) L1-H2
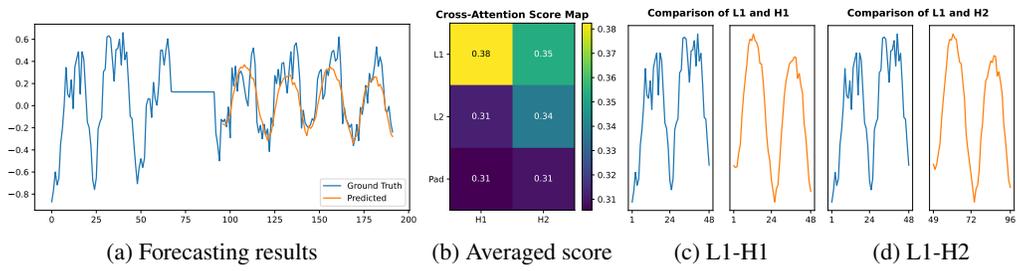
Figure 15: Illustration of (a) forecasting result, (b) averaged cross-attention score, and (c,d) patches with the highest score on ETTh2. The score map is averaged from all the heads across layers.

23

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in Appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist",**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We made the main claims in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

24

Justification: We discuss the limitations of the work in the Conclusion.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in Appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We uploaded the source code of our model and experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

25

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We uploaded the source code of our model and experiments.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

26

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results in Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in Appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Instead of error bars, following the numerous studies in the field of time series forecasting [14, 26], we fixed the random seed and provided the comparable experimental results in the main paper with generally accepted datasets and settings [21].

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In the main paper and the Appendix, we provide sufficient information on the used computer resources.

Guidelines:

27

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We confirm that the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: We confirm that there is no societal impact of the work performed.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

28

Answer: [NA]

Justification: We confirm that the paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We confirm that we have cited the original paper that produced the code package or dataset.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We confirm that that the paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

29

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We confirm that the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We confirm that the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

30