Recency Biased Causal Attention for Time-series Forecasting

Kareem Hegazy
UC Berkeley and ICSI

Michael W. Mahoney
UC Berkeley and ICSI and LBNL

N. Benjamin Erichson
ICSI and LBNL

Abstract

Recency bias is a useful inductive prior for sequential modeling: it emphasizes nearby observations and can still allow longer-range dependencies. Standard Transformer attention lacks this property, relying on all-to-all interactions that overlook the causal and often local structure of temporal data. We propose a simple mechanism to introduce recency bias by reweighting attention scores with a smooth heavy-tailed decay. This adjustment strengthens local temporal dependencies without sacrificing the flexibility to capture broader and data-specific correlations. We show that recency-biased attention consistently improves sequential modeling, aligning Transformer more closely with the read–ignore–write operations of RNNs. Finally, we demonstrate that our approach achieves competitive and often superior performance on challenging time-series forecasting benchmarks.

1 INTRODUCTION

Recency bias is a useful inductive prior for sequential modeling: it emphasizes nearby information, while still allowing longer-range dependencies. Recurrent neural networks (RNNs) naturally embody this bias through their local and causal structure, which enables them to excel at tasks that require selective reading, ignoring, or writing of information. For example, in the flip-flop task (Liu et al., 2023), RNNs easily learn to store and discard information as needed, whereas Transformers often fail despite their greater expressivity. This limitation highlights a fundamental gap: while RNNs leverage temporal locality, standard Transformer attention does not.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

Transformers have nevertheless revolutionized sequential modeling, achieving state-of-the-art results across language and time-series benchmarks. At the core of their success lies the multihead attention (MHA) mechanism, which integrates information from all positions through pairwise similarity. This all-to-all formulation is powerful for natural language processing (NLP), where dependencies between words can span long distances and do not depend strongly on order. Yet the same property can be a liability for time-series forecasting and structured sequential tasks, where local and causal structure is essential, and less relevant distant interactions can obscure the signal.

Recent work has shown that introducing recency biases can improve Transformer performance on weakly local and even noncausal language tasks (Press et al., 2022; Raffel et al., 2023; Liu et al., 2025). These results suggest that models benefit from temporal locality even when the underlying data does not demand it. Surprisingly, however, the role of recency bias in time-series forecasting remains underexplored, despite the fact that time-series data is causal and highly local.

In time-series forecasting, each embedding corresponds to a causal temporal measurement, and the strength of dependencies typically decays with temporal separation. Time-series forecasting problems are inherently causal; the input and target sequences do not overlap. However, attention methods in most state-of-the-art models (Nie et al., 2023; Zhou et al., 2022, 2023, 2021; Wu et al., 2021) do **not** apply a causal mask. Thus, the temporal embeddings are non-causally updated with other input sequence embeddings from later times. This mismatch between the all-to-all nature of Transformer attention and the local-causal structure of time-series data raises a central question: how well suited is standard attention for time-series forecasting?

We develop the Recency Biased Causal Attention (RBCA) framework to generally study the effects of causal and recency biases on Transformers as they are applied to local and causal datasets. We investigate the flexibility of recency biases to simultaneously perturb the attention correlation structure while suffi-

ciently capturing pairwise correlations unique to each dataset. To do so, we leverage insights from time-series forecasting problems to broadly improve upon Transformer's fundamental capabilities. Specifically, we note that attention weights are a normalized similarity metric similar to correlations. We are further motivated by the observation that many physical systems exhibit heavy-tailed autocorrelations, e.g., the pairwise correlation strength may decay as a power-law as the time delay grows (Clauset et al., 2009).

We develop power-law recency biases to impose a natural correlation structure on the attention weight distribution and systematically compare their effects across tasks. Our study evaluates recency biases in a principled way, highlighting both their strengths and their nuanced differences across causal and locally structured domains in NLP and time-series forecasting. We begin by testing whether recency biases equip Transformers with the basic read, ignore, and write capabilities characteristic of RNNs. Next, we apply RBCA to a standard encoder-decoder Transformer and assess its performance on causal and locally correlated time-series datasets. Finally, we investigate the effects of RBCA on modern time-series forecasting architectures by developing *Powerformer*, a PatchTST (Nie et al., 2023) variant with RBCA.

Our main contributions are the following.

- We propose Recency-Biased Causal Attention (RBCA), a general framework for introducing recency bias into Transformers. With RBCA, we develop power-law-inspired recency biases that induce a heavy-tailed attention distribution while allowing attention to learn the unique data-dependent correlation structure.

- We evaluate recency biases on the flip-flop task to test core sequential capabilities such as read, write, and ignore. We find that NLP-derived recency biases degrade performance, while our power-law biases enable Transformers to achieve better accuracy.

- We assess RBCA in time-series forecasting. First, we apply it to a standard Transformer; and then we introduce *Powerformer*, a simple encoder-only variant that amplifies recency effects. Our power-law biases consistently improve forecasting accuracy across benchmarks. Notably, *Powerformer* further strengthens the recency bias despite its regularizing effect, thus highlighting the fundamental importance of recency biases for time-series prediction.

2 RELATED WORK

The success of Transformers (Vaswani et al., 2017) and LLMs inspired Transformer-based time-series models. For example, Time-GPT (Garza et al., 2024) and One-Fits-All (Zhou et al., 2023), which fine-tunes a pre-trained GPT-2 (Radford et al., 2019) with temporal embedding. Many other models aimed to alter the attention mechanism or the input embeddings.

Many early time-series Transformer models altered the attention mechanism to improve alignment with time-series properties. Fourier representations separate fast and slow varying dynamics, simplifying attention representations (Zhou et al., 2022; Woo et al., 2022). AutoFormer (Wu et al., 2021) replaces the similarity metric with autocorrelations. FlowFormer (Wu et al., 2022) achieves linear complexity attention without a recency bias. Informer (Zhou et al., 2021) selects high attention weight values, increasing prediction capacity. iTransformer (Liu et al., 2024) inverts the temporal and embedding dimensions during MHA.

Some works alter the input embedding to enrich the input or better align it with NLP tasks. PatchTST patches the input sequence, outperforming more complicated methods (Nie et al., 2023). Chronos (Ansari et al., 2024) tokenizes the continuous input and trains on large datasets. TOTEM (Talukder et al., 2024) also discretizes the input time-series, but uses VQVAE (van den Oord et al., 2017).

To impose a locality bias, recency biases have been used in both NLP (Liu et al., 2025) and time-series tasks. For NLP, ALiBi (Press et al., 2022) employs smooth exponential decay, while block (Lu et al., 2025) and sliding-window (Beltagy et al., 2020) attentions employ a cutoff scheme. Previous recency-biased time-series models include Reformer (Kitaev et al., 2020), which uses a locality-sensitive hashing mechanism, and ETSformer (Woo et al., 2022), which employs a non-learnable exponential decay in one of their attention mechanisms. Earlier time-series CNN-based works also employed temporal dilations to bias towards early data (Li et al., 2019; van den Oord et al., 2016; Franceschi et al., 2019).

3 METHOD

We propose Recency-Biased Causal Attention (RBCA) as a framework for recency biases. In this section, we review standard Multihead Attention (MHA) and demonstrate how we add causal and recency biases to it. Figures 1 and 2 illustrate RBCA and its effect on attention weights.

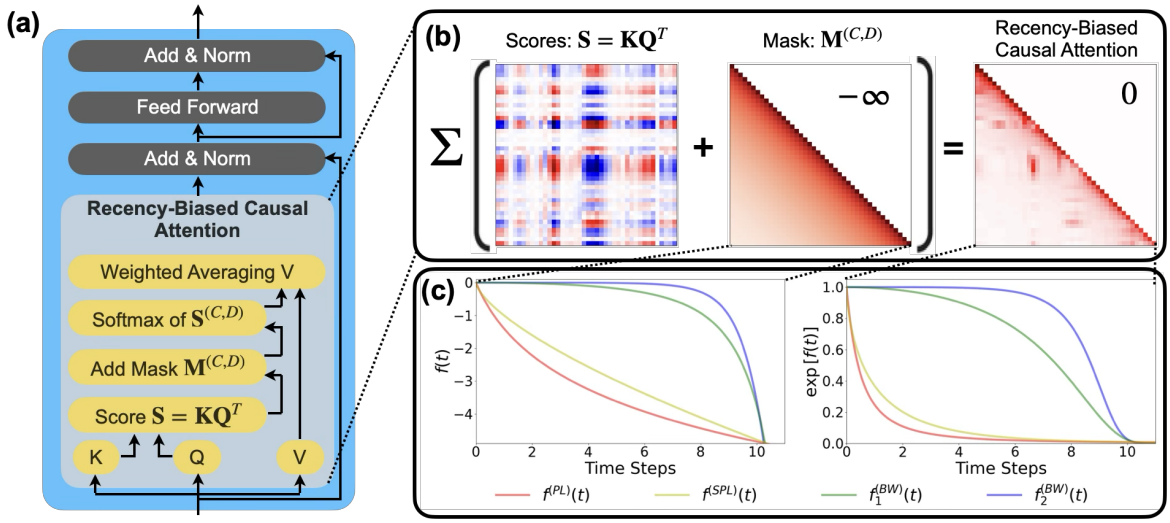

Figure 1: Illustration of Recency Biased Causal Attention (RBCA) and how it fits into the Transformer layer and adapts the attention. Panel (a) shows how RBCA replaces MHA in the standard Transformer block. Panel (b) shows a local-causal mask and subsequent weights, where Σ corresponds to the softmax function, red and blue indicate positive and negative values, respectively, and white is labeled when ambiguous. Panel (c) demonstrates the 4 different masking functions we apply to the score (left) and their effects on the weights (right).

3.1 Standard Multihead Attention

Transformers typically leverage MHA to compute new embeddings from weighted sums of input embeddings. Let $\mathbf{X} \in \mathbb{R}^{T\times D}$ be a sequence of T input vectors, each of dimension D. For all H attention heads, indexed by h, MHA first projects $\mathbf{X}$ into *query*, *key*, and *value* matrices, respectively denoted by $\mathbf{Q}_h$, $\mathbf{K}_h$, and $\mathbf{V}_h$: $\mathbf{Q}_h = \mathbf{X}\,\mathbf{W}_h^{(\mathrm{Q})}$, $\mathbf{K}_h = \mathbf{X}\,\mathbf{W}_h^{(\mathrm{K})}$, and $\mathbf{V}_h = \mathbf{X}\,\mathbf{W}_h^{(\mathrm{V})}$. Each projection matrix $\mathbf{W}_h^{(\cdot)} \in \mathbb{R}^{D\times D_k}$ is learnable, with head size D_k. The parameters $\mathbf{Q}_h$, $\mathbf{K}_h$, and $\mathbf{V}_h$ thus each have the shape $T \times D_k$.

The *unnormalized attention similarity scores* measure how each query interacts with all keys

$$\mathbf{S}_h = \frac{\mathbf{K}_h\,\mathbf{Q}_h^\top}{\sqrt{D_k}}, \tag{1}$$

where the factor of $1/\sqrt{D_k}$ helps stabilize the gradients by normalizing the dot products according to the head dimension. In turn, the *normalized attention weights* $\mathbf{C}_h = \mathrm{Softmax}\big(\mathbf{S}_h\big)$ has each row summing to 1. Finally, each head's output $\tilde{\mathbf{X}}_h = \mathbf{C}_h\,\mathbf{V}_h$ is computed as a weighted sum of the values, concatenated with the other heads, and transformed by a linear projection

$$\tilde{\mathbf{X}} \;=\; [\tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_H]\,\mathbf{W}^{(\mathrm{A})}, \tag{2}$$

where $\mathbf{W}^{(\mathrm{A})} \in \mathbb{R}^{(H\,D_k)\times D}$ is a learnable matrix. The result $\tilde{\mathbf{X}}$ has the same dimensionality as the input $\mathbf{X}$ (i.e., $T \times D$), allowing it to be passed into subsequent Transformer layers or decoded.

Enforcing Causality The standard causal mask prohibits future embeddings from updating past ones:

$$\mathbf{M}^{(\mathrm{C})}_{t,t'} = \begin{cases} -\infty & t' > t, \\ 0 & t' \le t, \end{cases} \tag{3}$$
$$\mathbf{S}_h^{(\mathrm{C})} = \mathbf{S}_h + \mathbf{M}^{(\mathrm{C})}, \quad \mathbf{C}_h^{(\mathrm{C})} = \mathrm{Softmax}\big(\mathbf{S}_h^{(\mathrm{C})}\big).$$

The attention weights $\mathbf{C}^{(C)}_{t,t'} = 0$ for all $t' > t$, ensuring a causal structure for the time-series data.

3.2 Recency-Biased Causal Attention

The recency bias is a mask $\mathbf{M}^{(\mathrm{L})}$ defined by some non-positive function $f(\Delta t) \le 0$ that is dependent on the relative time difference (Δt) between embeddings:

$$\mathbf{M}^{(\mathrm{L})}_{i,j} = \begin{cases} 0 & j > i, \\ f(t_i - t_j) & j \le i. \end{cases} \tag{4}$$

RBCA combines the recency bias and causal mask:

$$\mathbf{M}^{(\mathrm{C,L})} = \mathbf{M}^{(\mathrm{C})} + \mathbf{M}^{(\mathrm{L})}, \quad \mathbf{S}_h^{(\mathrm{C,L})} = \mathbf{S}_h + \mathbf{M}^{(\mathrm{C,L})} \tag{5}$$
$$\mathbf{C}_h^{(\mathrm{C,L})} = \mathrm{Softmax}\big(\mathbf{S}_h^{(\mathrm{C,L})}\big) \tag{6}$$
$$C^{(\mathrm{C,L})}_{h,i,j} \propto \begin{cases} 0 & j > i, \\ \exp\big[f(t_i - t_j)\big] & j \le i. \end{cases} \tag{7}$$

By combining causal masking and smoothly decaying weights, RBCA captures temporal ordering and localized dependencies inherent to real-world processes.

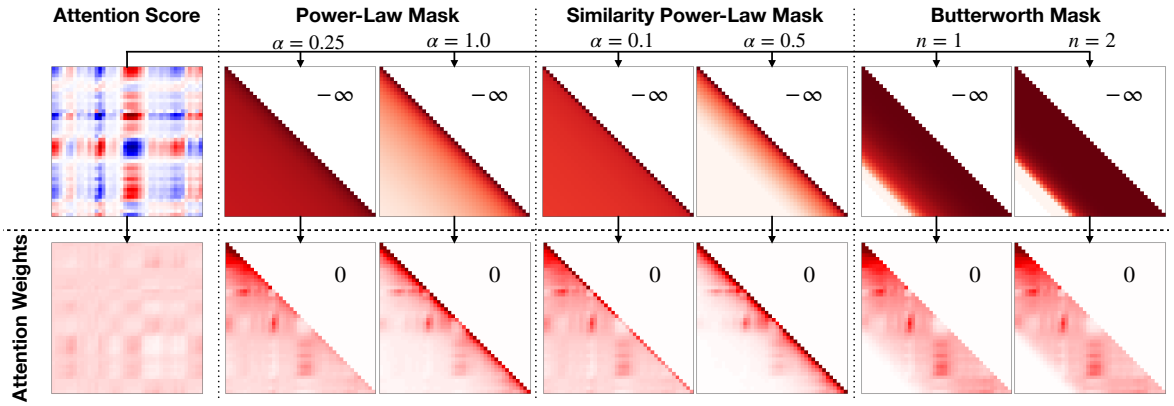

Figure 2: A comparison between standard non-causal and nonlocal attention (bottom left) and examples of RBCA with varying masks (top row) applied to the same attention score (top left). In the top row, red is maximized at 0 and minimized (white) at $-\infty$. In the bottom row, red is maximized at 1 and minimized (white) at 0. For the attention score red and blue indicate positive and negative.

3.3 Recency Bias Functions

Generally, previous NLP works consider sliding windows (Beltagy et al., 2020) and exponentially decaying recency biases (ALiBi (Press et al., 2022)). For RBCA, these are given by

$$f_n^{(\mathrm{BW})}(t) = \left(1 + \left(\frac{z(t)}{t_c}\right)^{2n}\right)^{-1/2}, \quad f^{(\mathrm{E})}(t) = -\alpha t,$$

respectively, where α is the decay time constant, and $f_n^{(\mathrm{BW})}(t)$ is the Butterworth filter that mimics a smoothed step function (Figs. 1c and 2). The Butterworth filter (Section S1.2) depends on the cutoff time t_c, the decay order n, and the digital filter gain $z(t)$.

Here, we leverage the fact that pairwise correlations in many physical systems follow a power-law distribution (Clauset et al., 2009). We note that attention weights are analogous to pairwise correlations, as they measure a pairwise similarity. Consequently, we make two power-law recency biases

$$f^{(\mathrm{PL})}(t) = -\alpha \log(t), \quad f^{(\mathrm{SPL})}(t) = -(t)^{-\alpha}.$$

$f^{(\mathrm{PL})}(t)$ induces a power-law envelope onto the attention weights, and $f^{(\mathrm{SPL})}(t)$ adds a power-law bias to the attention score that results in an exponential of a power-law after the softmax.

3.4 Regularization and Reduced Complexity

The recency bias is a regularizer on the input data, as it limits MHA's ability to attend to inputs far away. Intuitively, the recency bias acts as a hard-constraint that forces the method to attend to recent measurements. Some recency biases, depending on the decay rate α, sufficiently attenuate the weights to 0 at some time τ. Figures 1c and 2 show such instances. In these cases, the attention complexity can be reduced from quadratic in input length T to linear: $\mathrm{O}(\tau T)$.

3.5 Interpreting Power-Law Biases

Power-law recency biases strike a balance between sampling sinusoidal and slowly developing signals over long periods of time with sampling nearby quickly varying signals. To develop this intuition, we consider two limits of signal types: periodic systems (which benefit from long-time windows) and systems with fast exponentially decaying pairwise correlations (which benefit from a quickly decaying recency bias). Sinusoidal signals benefit from long and equally weighted windows, while short exponentially decaying signals benefit from swiftly decaying biases. The power-law biases are a combination of the benefits from the exponential bias and the long-time window. At recent times, the mask decays the fastest: analogous to the exponential decay that amplifies the most recent information. At longer times, the heavy tail allows the model to sample many more periods of an oscillation than other biases (e.g. exponential). The heavy tail filters for long-term signals since higher-frequency oscillations and fast variations will be averaged out, but long-term oscillations and slowly developing signals on the order of the tail's length will remain. By attending to attenuated distant tokens, the power law enables RBCA to capture (or remember) long-term signals, providing a more expressive class of inductive biases that outperform exponential decay in practice.

4 EXPERIMENTS

We investigate the general effects of recency biases (RBCA) on Transformers for causal and local tasks. This includes both NLP and time-series tasks. First, we test if the recency bias endows Transformer with capabilities similar to RNN's read, ignore, and write. We see that some biases do give Transformer these capabilities. Second, we build upon this insight by applying RBCA to the original encoder-decoder Transformer and testing on standard time-series tasks. Lastly, we continue investigating the benefits of RBCA by applying it to a simple state-of-the-art time-series model (PatchTST (Nie et al., 2023)) and find that it further improves performance and outperforms other models that claim to beat PatchTST.

4.1 Addressing Transformer Glitching

Prior work (Liu et al., 2023) showed that Transformers struggle to reliably perform simple, yet important, RNN tasks together: write information to a state, ignore input information, and read from the stored state. These tasks are important for filtering and recalling important information in long sequences. RNNs can easily perform this task, likely due to their causal structure and implicit recency bias. Here, we investigate whether the causal and recency biases from RBCA endow Transformer with similar capabilities.

Evaluation To test these capabilities, (Liu et al., 2023) developed the flip-flop test: a string of alternating characters $c \in \{i, w, r\}$ and numbers $n \in \{0, 1\}$. Each character instructs the model to ignore (i) the following number, write (w) the following number to memory, or read (r) the latest number that was written. For the example $w0i1i0w1i0r$, the model reads 1 since it was the last written number. The out-of-distribution (OOD) test contains many ignores (i) after the last write (w) and tests Transformer's ability to generalize its read, ignore, and write capabilities to longer sequences. We train and test a 2-layer encoder-only Transformer with and without RBCA. Each attention head has a different α. Section S5.1 provides details on the model, evaluation, and results.

Attention Analysis We first investigate the effects of RBCA on the attention weight distributions of ALiBi (one of the most prominent NLP recency biases) and our physically-inspired power-law bias: $f^{(\mathrm{E})}(t)$ and $f^{(\mathrm{PL})}(t)$ respectively. Figure 3 shows these attention weight distributions for the best attention heads, full results are in Section S5.1. Most importantly, the attention weight between the 'read' and correct number pair needs to be high, while the weight between the 'read' and the incorrect numbers should be close to 0. For ALiBi ($f^{(\mathrm{E})}(t)$), the amount of weights equal to 0 and 1 for the correct number pair are similar (Fig. 3a). Conversely, for $f^{(\mathrm{PL})}(t)$, all weights corresponding to the correct number are nonzero, with almost 90% equal to 1 (Fig. 3c). When identifying the correct 'write' command, almost all the weights corresponding to the correct 'write' are 0 for ALiBi (Fig. 3b), indicating that it is generally unable to find it. For $f^{(\mathrm{PL})}(t)$, we see a distinct shift between the correct and incorrect 'write', with all weights greater than 0 for the correct 'read' and 'write' pair: (Fig. 3d). After observing these fundamental changes to the attention weight structure, we now explore these biases' performances.

Results When comparing the RBCA accuracy on the OOD test set against standard MHA, both power-law biases ($f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$) consistently outperform, while both NLP biases ($f^{(\mathrm{E})}(t)$ and $f_n^{(\mathrm{BW})}(t)$) underperform. As noted by (Press et al., 2022), the OOD accuracy varies during training, but $f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$ consistently hover around 100%, half the time being exactly 100%. Both NLP-inspired recency biases result in **worse** accuracy compared to vanilla MHA. MHA hovers around 85%, ALiBi ($f^{(\mathrm{E})}(t)$) hovers around 75%, and sliding window ($f_n^{(\mathrm{BW})}(t)$) performs very poorly. Section S5.1 provides full OOD test results. These results indicate that recency biases significantly affect sequential read, write, and ignore capabilities. The correct recency bias can be sufficient to endow the Transformer with these capabilities.

4.2 RBCA in Vanilla Transformer

Given the success of power-law-biased RBCA on flip-flop, we further evaluate RBCA's capabilities on more challenging causal and local tasks, namely time-series forecasting. Here, we take a principled approach by first testing the effects of RBCA on a standard encoder-decoder Transformer. This is important, because adding more features to the model will convolute RBCA's effects and potentially confuse the results.

Datasets Our time-series tasks use 7 standard benchmark datasets (Zhou et al., 2021; Wu et al., 2021): ETT (4 subsets), Weather, Electricity, and Traffic. The ETT subsets measure the electricity transformer oil temperature at 2 different time scales (every 15 minutes and hourly) for two regions. These are particularly interesting for testing the RBCA's sensitivity to sampling changes and distribution shifts. Section S2 provides a detailed description of these datasets, available in (Wu et al., 2021). We note that Weather, Electricity, and Traffic have much larger timescales, spanning from one to multiple years (Ta-

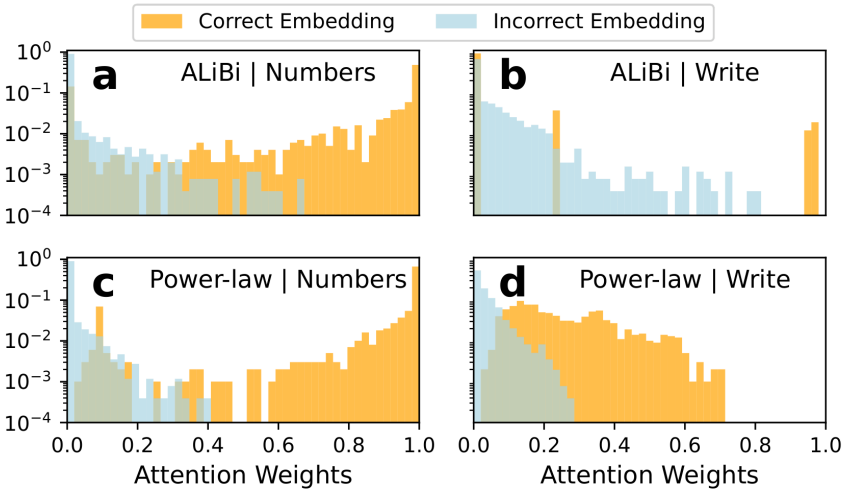

Figure 3: The attention weights corresponding to the flip flop 'read' command and the correct/incorrect 'write' command and number. We show these distributions for the ALiBi ($f^{(\mathrm{E})}(t)$) and the $f^{(\mathrm{PL})}(t)$ biases.

ble S1). These 7 datasets are large enough to capture slowly decaying temporal correlations (Appendix S2).

Evaluation Following previous works (Nie et al., 2023; Wu et al., 2021; Zhou et al., 2022; Woo et al., 2022; Liu et al., 2024; Talukder et al., 2024; Kitaev et al., 2020; Zhou et al., 2021), we train for multiple prediction lengths (96, 192, 336, and 720) and input lengths (336 and 512). Each model is trained three times for each setting with different seeds. For each dataset, we treat the recency bias, decay rate, and input sequence length as hyperparameters. We select one input sequence for the entire dataset, and a recency bias and decay rate for each prediction length. Sections S5.2 and S3.4 provide details on our experimental process and hyperparameters, respectively. All benchmarks are trained with varying input sequences, and the optimal model is selected in the same way.

For each dataset, we evaluate the $f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$ biases. We evaluate $f_n^{(\mathrm{BW})}(t)$ on the ETT and weather datasets only, due to its poor performance on these datasets and the Transformer glitching benchmark. We do not evaluate $f^{(\mathrm{E})}(t)$ due to its similarity to $f^{(\mathrm{SPL})}(t)$ and inferior performance on the Transformer glitching benchmark.

Results In the vast majority of cases, RBCA outperforms standard MHA, especially for datasets with slowly decaying long-range dependencies. When RBCA outperforms MHA, it does so by much larger margins than when it underperforms. For example, on the ETTh2 dataset RBCA and MHA outperform each other the same number of times, however, RBCA improves MSE and MAE by 16% and underperforms by 2%. Table S4 shows the full MSE and MAE results.

4.3 *Powerformer*

To further investigate RBCA's effects on time-series forecasting, we add it to modern time-series architectures to determine how RBCA may affect state-of-the-art models. Similar to our previous tests, our goal is to understand the effects of recency biases on attention. For this reason, we develop a **simple** time-series model, allowing us to isolate the recency bias effects. Adding many features convolutes the recency bias' effects and confuses the source of the outcome.

Model We introduce *Powerformer*, a time-series Transformer variant built atop PatchTST (Nie et al., 2023) by replacing MHA with RBCA. *Powerformer* integrates the encoder-only architecture, RBCA, and PatchTST's (Nie et al., 2023) successful patching framework. PatchTST follows common time-series architecture designs: an encoder-only Transformer with a linear readout head. We select PatchTST because it does not alter the Transformer, but instead only makes changes to the input embeddings by patching. The lack of alterations to the attention mechanism in PatchTST will isolate the effects of RBCA on both attention weight distributions and forecasting performance. While there have been more advances in time-series forecasting since PatchTST, such advancements make larger changes to the Transformer and attention architecture. We believe that many of these changes will potentially confuse the results by making it challenging to determine where the improvement came from. Appendix S3.3 provides further details on *Powerformer*'s architecture.

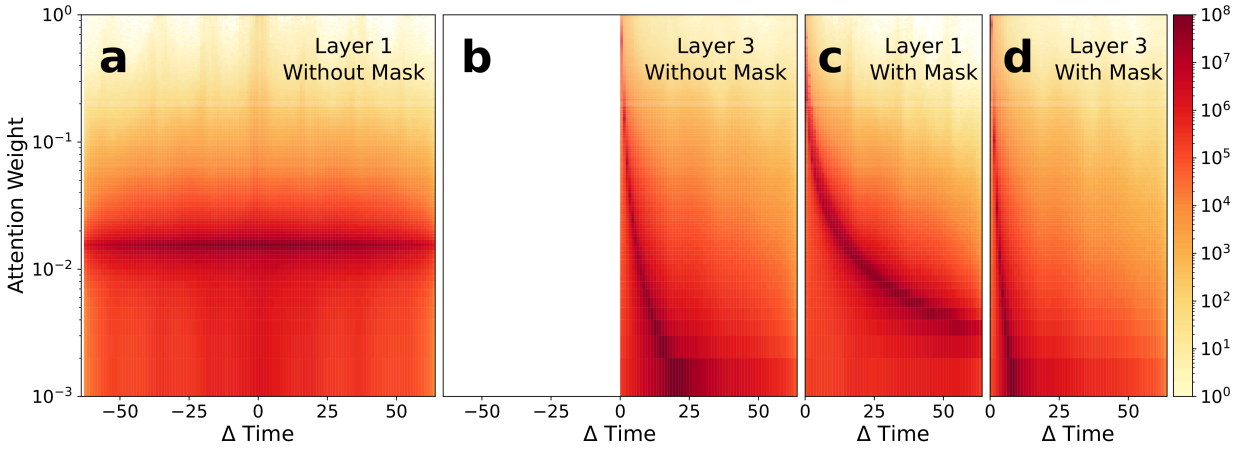

Figure 4: After training *Powerformer*, we show the attention weights without the mask $\mathbf{M}^{(\mathrm{C,L})}$ (a and b) and with the mask (c and d) for the first (a and c) and last (b and d) encoder layers. The linear trend illustrates the power law. These results are from the Weather dataset with $f^{(\mathrm{PL})}(t)$, $\alpha = 1.0$, and input/prediction lengths 512/96.

Evaluation We evaluate *Powerformer* on the same datasets and in the same manner as done in Section 4.2. Sections S5.2 and S3.4 provide further details on our experimental process and hyperparameters, respectively. We benchmark *Powerformer* against models that focus on modifying the attention mechanism and models that use recency biases. These models take a similar approach to *Powerformer* and are thus a fair comparison. We do not compare against all state-of-the-art models, as they contain techniques *Powerformer* purposely lacks, but can be added to *Powerformer*, or RBCA could be added to other models. Either expanding *Powerformer* or adding RBCA to other models is an exciting direction for future work.

Attention Analysis Interestingly, *Powerformer* strengthens our causal and recency biases beyond $\mathbf{M}^{(\mathrm{C,R})}$, which is a key insight into the importance of the causal and recency biases. At first glance, this is counterintuitive because the causal and recency biases act as regularizers that remove (and diminish contributions from) parts of the input sequence. The removed data could otherwise be used to help the model memorize the output and overfit. Moreover, since the recency bias mask ($\mathbf{M}^{(\mathrm{R})}$) is constant, *Powerformer* has to capability to mitigate it by learning to counter the mask's regularization effects.

Rather than learning to remove the recency bias regularization, *Powerformer* strengthens the recency bias. To demonstrate this behavior, we look at the attention weight distribution with and without $\mathbf{M}^{(\mathrm{C,R})}$on *Powerformer* trained with $f^{(\mathrm{PL})}(t)$ ($\alpha = 1$) and $\mathbf{M}^{(\mathrm{C,R})}$. Figure 4 shows the attention weight distribution as a function of time delay between embeddings. The first encoder layer without the mask (Fig. 4a) is relatively invariant to causality and locality (unbiased). When $\mathbf{M}^{(\mathrm{C,R})}$ is applied to the first layer (Fig. 4c) the non-causal weights vanish and the power-law structure emerges. This behavior is expected, unlike the behavior in the last layer. In the last layer, the attention weight distribution **without** $\mathbf{M}^{(\mathrm{C,R})}$ (Fig. 4b) is entirely causal with a strong recency bias. That is, even without applying the causal and recency bias, *Powerformer* is enforcing causality and the power-law recency bias, rather than trying to remove this regularization. Therefore, once $\mathbf{M}^{(\mathrm{C,R})}$ is applied to the last layer (Fig. 4d), the recency bias is further increased, as seen by the steeper slope. Therefore, *Powerformer* learns embeddings that further strengthen the effects of the recency bias, which is strong evidence that these biases are crucial in time-series forecasting.

Results Compared to our benchmarks (Table 1), *Powerformer* achieves the best performance against more complicated (FEDformer (Zhou et al., 2022), ETSformer (Woo et al., 2022), One-Fits-All (Zhou et al., 2023)) models. *Powerformer* achieves the best performance in 46 forecasting tasks compared to the next best model (PatchTST) which achieves 9. We note that *Powerformer* outperforms ETSformer, which employs an exponential temporal decay without attention. This result reinforces the importance of correctly balancing RBCA's recency bias with attentions ability to learn data-dependent correlations. Notably, *Powerformer* outperforms One-Fits-All (Zhou et al., 2023), which is a pre-trained GPT-2 model more than twice *Powerformer's* size.

Table 1: *Powerformer's* performance on standard publicly-available time-series datasets compared to similar state-of-the-art Transformer models. The best and second best results are bolded and underlined, respectively.

		Powerformer		PatchTST		iTransformer		One-Fits-All		FEDformer		ETSformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.361**	**0.390**	0.370	0.400	0.401	0.417	0.376	0.397	0.376	0.415	0.494	0.479
	192	**0.395**	**0.410**	0.413	0.429	0.440	0.445	0.416	0.418	0.423	0.446	0.538	0.504
	336	**0.406**	**0.420**	0.422	0.440	0.455	0.454	0.442	0.433	0.444	0.462	0.574	0.521
	720	**0.434**	**0.455**	0.447	0.468	0.520	0.511	0.477	0.456	0.469	0.492	0.562	0.535
ETTh2	96	**0.269**	**0.334**	0.274	0.336	0.302	0.358	0.285	0.342	0.332	0.374	0.340	0.391
	192	**0.330**	**0.375**	0.339	0.379	0.369	0.400	0.354	0.389	0.407	0.446	0.430	0.439
	336	**0.323**	**0.380**	0.331	**0.380**	0.407	0.430	0.373	0.407	0.400	0.447	0.485	0.479
	720	**0.370**	**0.417**	0.379	0.422	0.436	0.462	0.406	0.441	0.412	0.469	0.500	0.497
ETTm1	96	**0.285**	**0.342**	0.290	**0.342**	0.301	0.355	0.292	0.346	0.326	0.390	0.375	0.398
	192	**0.328**	**0.368**	0.332	0.369	0.342	0.380	0.332	0.372	0.365	0.415	0.408	0.410
	336	**0.357**	**0.385**	0.366	0.392	0.376	0.401	0.366	0.394	0.392	0.425	0.435	0.428
	720	**0.410**	**0.414**	0.420	0.424	0.440	0.438	0.417	0.421	0.446	0.458	0.499	0.462
ETTm2	96	**0.163**	**0.251**	0.165	0.255	0.179	0.271	0.173	0.262	0.180	0.271	0.189	0.280
	192	**0.218**	**0.290**	0.220	0.292	0.236	0.310	0.229	0.301	0.252	0.318	0.253	0.319
	336	**0.273**	**0.326**	0.278	0.329	0.284	0.342	0.286	0.341	0.324	0.364	0.314	0.357
	720	**0.358**	**0.380**	0.367	0.385	0.370	0.394	0.378	0.401	0.410	0.420	0.414	0.413
Weather	96	**0.147**	**0.197**	0.149	0.198	0.163	0.211	0.162	0.212	0.238	0.314	0.197	0.281
	192	**0.191**	**0.239**	0.194	0.241	0.204	0.253	0.204	0.248	0.275	0.329	0.237	0.312
	336	**0.243**	**0.279**	0.245	0.282	0.256	0.291	0.254	0.286	0.339	0.377	0.298	0.353
	720	**0.310**	**0.329**	0.314	0.334	0.326	0.337	0.326	0.337	0.389	0.409	0.352	0.388
Electricity	96	**0.129**	0.223	**0.129**	**0.222**	0.131	0.226	0.139	0.238	0.186	0.302	0.187	0.304
	192	**0.145**	**0.240**	0.147	**0.240**	0.152	0.247	0.153	0.251	0.197	0.311	0.199	0.315
	336	**0.162**	**0.257**	0.163	0.259	0.168	0.263	0.169	0.266	0.213	0.328	0.212	0.329
	720	0.198	0.289	0.197	0.290	**0.191**	**0.284**	0.206	0.297	0.233	0.344	0.233	0.345
Traffic	96	0.365	0.252	0.360	**0.249**	**0.352**	0.258	0.388	0.282	0.576	0.359	0.607	0.392
	192	0.382	0.258	0.379	**0.256**	**0.368**	0.267	0.407	0.290	0.610	0.380	0.621	0.399
	336	0.391	0.265	0.392	**0.264**	**0.384**	0.273	0.412	0.294	0.608	0.375	0.622	0.396
	720	0.430	**0.286**	0.432	**0.286**	**0.417**	0.290	0.450	0.312	0.621	0.375	0.632	0.396
$1^{st}/2^{nd}$		**46 / 7**		9 / 38		6 / 3		0 / 10		0 / 0		0 / 0	

Qualitatively, *Powerformer* outperforms the next best model (PatchTST) by capturing high-frequency variability. *Powerformer* better models sharper changes in oscillations and deviations from a generic sinusoidal signal, as demonstrated in Appendix S4. This behavior can be described in the context of the power-law's heavy tail, which induces a multi-scale attention structure. During attention, the embeddings in the long tail will be summed together with relatively similar weights since the heavy tail and the unbiased attention weights (Fig. 2) are relatively constant. This summation removes short-lived input sequence frequency components with periods less than the tail's length. This is because the summation covers multiple, if not many, periods of these frequency components, causing them to cancel out. However, long-term signals with periods greater than the tail length will be captured. Therefore, the tail highlights longer-term dynamics in the input sequence. Conversely, the quickly varying power-law near 0 amplifies a shorter context window. This shorter window highlights local higher-frequency components by reducing the period at which the signal is washed out in the summation. Again, striking a balance between attending to long-term sinusoidal signals and quickly decaying exponential pair-wise distributions. Therefore, the power-law recency bias endows *Powerformer* with an implicit temporal multi-scale structure: the tail focuses on low-frequency signal while high-frequency components come from the most recent signal.

Powerformer underperforms for the Traffic dataset and outperforms by a smaller margin in the Electricity dataset due to these datasets' highly periodic nature. As highlighted, *Powerformer* can capture short and long-term sinusoidal signals, but long and

constant attention windows – used in all the other benchmarks – are better suited to highly-periodic signals. In cases where the data is highly periodic, like Traffic (Fig. S5) and Electricity (Figs. S6 and S9), these systems can often be effectively modeled by simpler Fourier or statistical methods without requiring the much larger training and evaluation overhead of Transformers. *Powerformer* is built as a general-purpose time-series method to handle challenging non-stationary dynamics.

Powerformer is sensitive to the recency bias and aligns $f(t)$ with the dataset's specific pairwise correlation. To test this, we evaluated the first 5 datasets on $f_n^{(\mathrm{BW})}(t)$, which induces a flat pairwise correlation structure. The $f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$ biases consistently outperform $f_n^{(\mathrm{BW})}(t)$, see Tables S5, S6, S7, and S8 for the full evaluation results of $f^{(\mathrm{PL})}(t)$, $f^{(\mathrm{SPL})}(t)$, $f_1^{(\mathrm{BW})}(t)$, and $f_2^{(\mathrm{BW})}(t)$, respectively. The superior performance of $f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$ over $f_n^{(\mathrm{BW})}(t)$ demonstrates *Powerformer's* sensitivity to the dataset's correlation structure.

In addition to distinguishing between biases that naturally align with the data distribution, *Powerformer* is also sensitive to the dataset's correlation time scale. The ETTh and ETTm datasets test *Powerformer's* ability to adapt the timescale α of $\mathbf{M}^{(\mathrm{R})}$ for datasets sampled from the same distribution but with different temporal resolution. Since ETTh and ETTm are sampled on the hour and minute timescales, respectively, ETTm requires a slowly decaying mask to access the same information as a quickly decaying mask for ETTh. As expected, *Powerformer* with $f^{(\mathrm{SPL})}(t)$ (Table S6) favors fast decaying masks on ETTh and slowly decaying masks on ETTm. This illustrates *Powerformer's* sensitivity to adjust its timescale to cover the same information content in each dataset.

4.4 Ablation

We ablate *Powerformer* to determine the effects of the causal mask ($\mathbf{M}^{(\mathrm{C})}$) and the addition of the recency bias ($\mathbf{M}^{(\mathrm{R})}$). Table S9 compares the performance of *Powerformer* with MHA (without $\mathbf{M}^{(\mathrm{C,R})}$), with only the causal mask ($\mathbf{M}^{(\mathrm{C})}$), and with RBCA ($\mathbf{M}^{(\mathrm{C,R})}$). We first observe that causal-only attention ($\mathbf{M}^{(\mathrm{C})}$) outperforms MHA, validating the importance of our causal inductive bias. In all but one case, RBCA ties or outperforms causal-only attention.

To learn the temporal information timescale, we make α learnable instead of treating it as a hyperparameter. Section S8.1 outlines the experimental details and full results. During training, α drops consistently and monotonically. This is expected as a small magnitude α facilitates overfitting by providing the model with extraneous (long-term) information. To keep α from converging to 0, we initialize α with the hyperparameter tuning values and quickly decay the learning rate. The learned α values decrease in magnitude, but we observe very little difference between the learnable and constant α (Table S10). This indicates that RBCA regularizes the model, removing superfluous information to avoid overfitting and improve generalization.

5 CONCLUSION

We develop RBCA to test the importance of causal and recency biases in Transformers, as they are applied to causal and locally correlated NLP and time-series tasks. Some recency biases ($f^{(\mathrm{PL})}(t)$) are sufficient requirements to imbue Transformer with advantageous RNN capabilities: read and write to memory, and ignore input information. However, common NLP recency biases ($f^{(\mathrm{E})}(t)$ and $f_n^{(\mathrm{BW})}(t)$) diminish these capabilities. This indicates that locality biases are important for Transformers to gain important capabilities, but the bias' implementation matters.

We develop *Powerformer* to impose and investigate the importance of causal and recency biases for Transformer-based time-series models. *Powerformer*, with its simple architecture, outperforms larger and more complex models. Notably, *Powerformer* strengthens the causal and recency biases by enforcing a power-law decay before the $\mathbf{M}^{(\mathrm{C,R})}$ is applied. This is strong evidence that the causal and recency biases are important for time-series problems, as *Powerformer* is essentially removing information at training time that could be used to overfit.

Acknowledgements

The authors thank Dongwei Lyu for her help in running the flip flop experiments to test the RBCA Transformer glitching. NBE would like to acknowledge the Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory under U.S. Department of Energy Contract No. DE-AC02-05CH11231,, and the Scientific Discovery through Advanced Computing (SciDAC) program, under Contract Number DE-AC02-05CH11231 at Berkeley Lab,. and NERSC (DE-AC02-05CH11231) for providing compute resources. MWM acknowledges DARPA, NSF, the DOE Competitive Portfolios grant, and the DOE SciGPT grant.

References

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Arango, S. P., Kapoor, S., Zschiegner, J., Maddix, D. C., Wang, H., Mahoney, M. W., Torkkola, K., Wilson, A. G., Bohlke-Schneider, M., and Wang, Y. (2024). Chronos: Learning the language of time series.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer.

Butterworth, S. (1930). On the theory of filter amplifiers. *Experimental Wireless and the Wireless Engineer*, 7:536,541.

Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2009). Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703.

Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. (2019). Unsupervised scalable representation learning for multivariate time series. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Garza, A., Challu, C., and Mergenthaler-Canseco, M. (2024). Timegpt-1.

Kitaev, N., Kaiser, L., and Levskaya, A. (2020). Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E. A., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, pages 5244–5254.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. (2023). Exposing attention glitches with flip-flop language modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Liu, J., Zhu, D., Bai, Z., He, Y., Liao, H., Que, H., Wang, Z., Zhang, C., Zhang, G., Zhang, J., Zhang, Y., Chen, Z., Guo, H., Li, S., Liu, Z., Shan, Y., Song, Y., Tian, J., Wu, W., Zhou, Z., Zhu, R., Feng, J., Gao, Y., He, S., Li, Z., Liu, T., Meng, F., Su, W., Tan, Y., Wang, Z., Yang, J., Ye, W., Zheng, B., Zhou, W., Huang, W., Li, S., and Zhang, Z. (2025). A comprehensive survey on long context language modeling.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. (2024). itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*.

Lu, E., Jiang, Z., Liu, J., Du, Y., Jiang, T., Hong, C., Liu, S., He, W., Yuan, E., Wang, Y., Huang, Z., Yuan, H., Xu, S., Xu, X., Lai, G., Chen, Y., Zheng, H., Yan, J., Su, J., Wu, Y., Zhang, N. Y., Yang, Z., Zhou, X., Zhang, M., and Qiu, J. (2025). Moba: Mixture of block attention for long-context llms.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*.

Press, O., Smith, N., and Lewis, M. (2022). Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2023). Exploring the limits of transfer learning with a unified text-to-text transformer.

Talukder, S., Yue, Y., and Gkioxari, G. (2024). Totem: Tokenized time series embeddings for general time series analysis.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. In *Arxiv*.

van den Oord, A., Vinyals, O., and kavukcuoglu, k. (2017). Neural discrete representation learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Woo, G., Liu, C., Sahoo, D., Kumar, A., and Hoi, S. (2022). Etsformer: Exponential smoothing transformers for time-series forecasting.

Wu, H., Wu, J., Xu, J., Wang, J., and Long, M. (2022). Flowformer: Linearizing transformers with conservation flows. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Confer-*

ence on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pages 24226–24242. PMLR.

Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with autocorrelation for long-term series forecasting. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. (2022). FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 27268–27286. PMLR.

Zhou, T., Niu, P., Wang, X., Sun, L., and Jin, R. (2023). One fits all: Power general time series analysis by pretrained LM. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Materials

S1 LOCALITY INDUCING DECAY FUNCTIONS

S1.1 Power-Law Functions

The two power-law-based masks $f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$ apply a power-law decay to the attention weights and scores, respectively. Here, we provide intuition for each mask type a well as illustrate the masks used in our evaluation.

We consider inducing a multiplicative power-law decay to the attention weights ($f^{(\mathrm{PL})}(t)$) to be the most natural way of inducing locality:

$$f^{(\mathrm{PL})}(t) = -\alpha \log(t). \tag{S1}$$

The attention weights are analogous to correlation coefficients. Therefore, since power-law distributions show up in correlation structures, we would like the attention weights to follow a similar power-law distribution. When training Transformer and *Powerformer* with $f^{(\mathrm{PL})}(t)$ we used decay times of $\alpha \in [0.1, 0.25, 0.5, 0.75, 1.0]$. Figure S1 shows the additive attention mask and the resulting multiplicative effects on the attention weights.

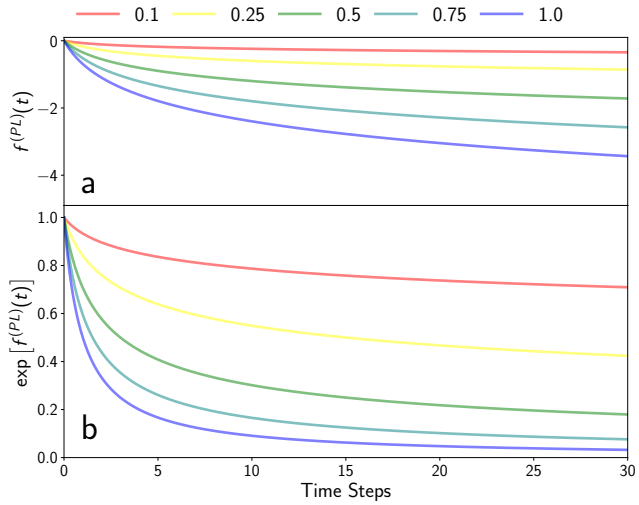

Figure S1: The power-law masks ($f^{(\mathrm{PL})}(t)$) used when evaluating RBCA on Transformer and when evaluating *Powerformer*. Panel (a) shows the additive mask, (b) shows the multiplicative effects on the attention weights.

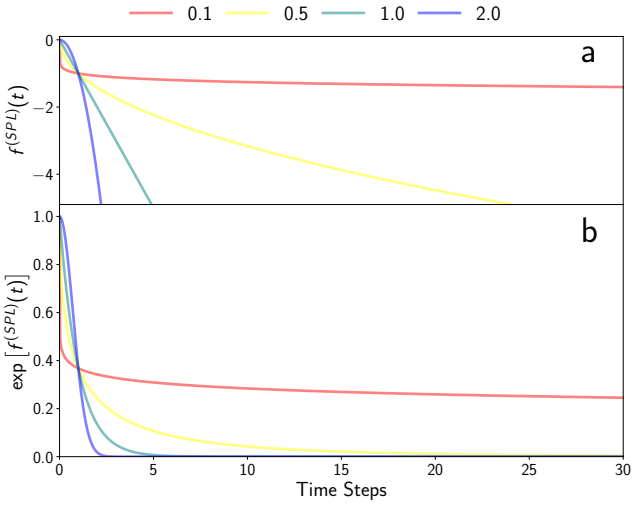

Figure S2: The similarity power-law masks ($f^{(\mathrm{SPL})}(t)$) used when evaluating RBCA on Transformer and when evaluating *Powerformer*. Panel (a) shows the additive mask, (b) shows the multiplicative effects on the attention weights.

We additionally apply the power-law directly to the similarity score:

$$f^{(\mathrm{SPL})}(t) = -(t)^{-\alpha}. \tag{S2}$$

During the attention calculation, we get an exponential of this power-law, which results in steep initial decay and a long tail. When training Transformer and *Powerformer* with $f^{(\mathrm{SPL})}(t)$ we used decay times of $\alpha \in [0.1, 0.5, 1.0, 2.0]$. Figure S2 shows the additive attention mask and the resulting multiplicative effects on the attention weights.

Combining the $f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$ masks, we cover decays starting from a relatively flat mask ($f^{(\mathrm{PL})}(t)$ with $\alpha = 0.5$) to a sharply decaying ($f^{(\mathrm{SPL})}(t)$ with $\alpha = 2.0$). To illustrate our coverage we plot $f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$ together in Fig. S3.

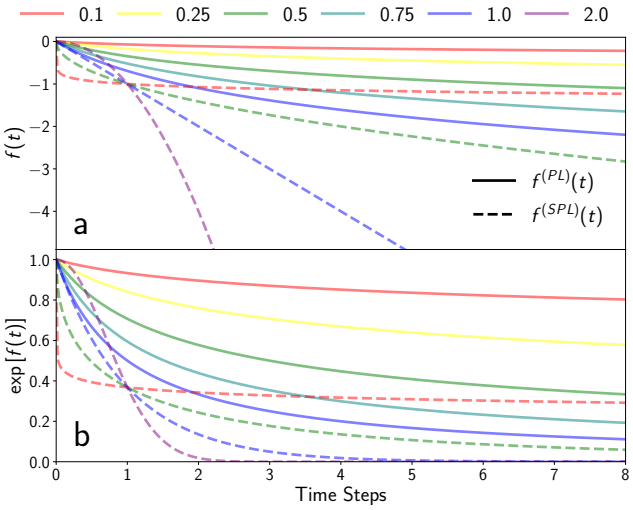

Figure S3: The $f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$ mask (a) and their effects on the weights (b) are shown to illustrate the coverage of decays used when assessing *Powerformer*.

S1.2 Butterworth Filter

The Butterworth filter (Butterworth, 1930) is often used in signal processing for low-, high-, and band-pass filters. It is designed to be optimally flat within the passband and sometimes referred to as the "maximally flat filter." The motivation behind it's derivation is to have equal sensitivity to frequency modes within the passband. We leveraged the Butterworth filter to equally weight temporal measurements within the lookback window. The analog Butterworth filter gain is given by

$$f_n^{(\mathrm{BW})}(z) = \frac{-1}{\sqrt{1 + \left(\frac{z}{t_c}\right)^{2n}}}, \tag{S3}$$

where t_c is the critical time that sets the width of the filter and the order n determines how fast the gain decays after t_c. In this work we use the digital Butterworth filter gain. Figure S4 shows the $f_1^{(\mathrm{BW})}(t)$ and $f_2^{(\mathrm{BW})}(t)$ contributions on the attention score and weights for varying decay constants.

S1.2.1 Digital Filter Gain

To transform the analog Butterworth filter into a digital filter, the filter design must be discretized. This discretization is often done using the bilinear transform

$$z = \frac{2}{T}\frac{z-1}{z+1}, \tag{S4}$$

where T is the numerical integration step size used in the trapezoidal rule.

S1.2.2 Code to Calculate the Gain

For reproducibility, we provide the code used to calculate the digital Butterworth filter gains. We multiply the gain by 5 to scale with the attention key-query overlap values.

```
import numpy as np
import scipy as sp
```

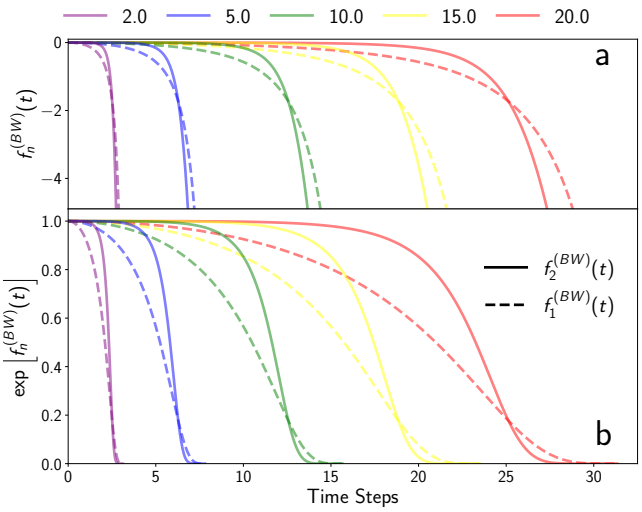

Figure S4: We show the effects of the Butterworth filter masks $\left(f_n^{(\mathrm{BW})}(t)\right)$ for orders 1 (dashed lines) and 2 (solid lines), while varying the critical time. Panel (a) shows the effects on the attention scores and Panel (b) shows the corresponding effects on the attention weights after applying Softmax to $f_n^{(\mathrm{BW})}(t)$.

```
def butterworth_filter(scale, order, times):
    b, a = sp.signal.butter(order, 0.8, 'lowpass', analog=False)
    t, decay = sp.signal.freqz(b, a)
    t = scale*t/2
    dc = 5*np.log(np.abs(decay))
    decay_interp = sp.interpolate.interp1d(t, dc)
    return decay_interp(times)
```

S2 DATASETS

We evaluate *Powerformer* on 7 real-world datasets that have become standard public benchmarks for time-series forecast tasks. The datasets can be found and downloaded in Ref. (Wu et al., 2021), or individually at references below.

Table S1: We provide the number of variables and the number of timesteps for each dataset.

Datasets	Illness	ETTh*	ETTm*	Weather	Electricity	Traffic
Features	7	7	7	21	321	862
Timesteps	966	17420	69680	52696	26304	17544

ETT[1] (Zhou et al., 2021) provides 7 measurements of the electrical transformer power load (including oil temperature) between July 2016 and July 2018. This is done over 2 regions in China at 2 different sampling rates (hourly and every 15 minutes), resulting in 4 separate datasets (ETTh1, ETTh2, ETTm1, ETTm2). We show the pairwise correlation dependence in Fig. S8.

Weather[2] (Wu et al., 2021) provides 21 meteorological measurements (air temperature, air pressure, humidity, precipitation, etc.) collected in Germany over the whole of 2020. These measurements are recorded every 10 minutes. We show the pairwise correlation dependence in Fig. S7.

[1] https://github.com/zhouhaoyi/ETDataset
[2] https://www.bg-ena.mpg.de/wetter/

Electricity[3] (Wu et al., 2021) provides the electricity consumption (kWh) of 321 consumers from 2012 to 2014. These measurements are sampled every hour. We show the pairwise correlation dependence in Fig. S6.

Traffic[4] (Wu et al., 2021) provides occupancy rates on San Francisco Bay Area freeways from 826 sensors. This data comes from the California Department of Transportation and is sampled hourly. We show the pairwise correlation dependence in Fig. S5.

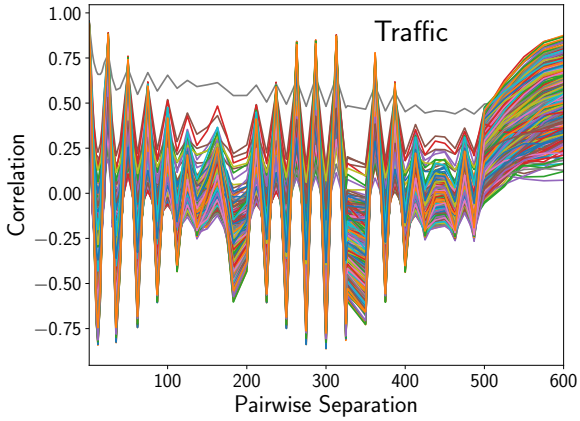

Figure S5: We show the Weather dataset's pairwise correlations as a pairwise separation function. Each colored line represents a different variable in the dataset.

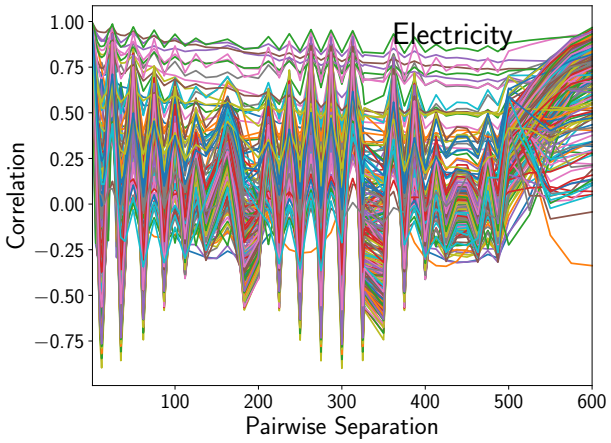

Figure S6: We show the Electricity dataset's pairwise correlations as a pairwise separation function. Each colored line represents a different variable in the dataset.

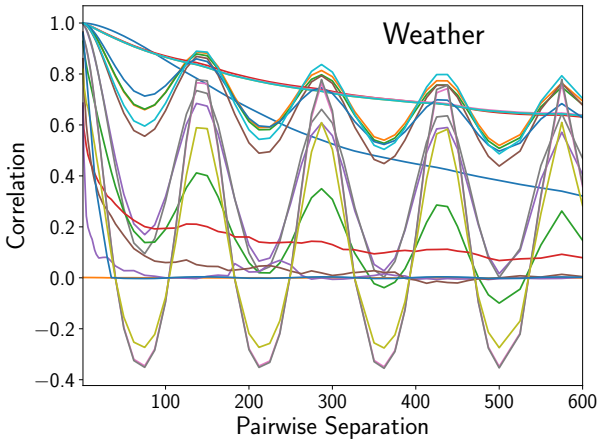

Figure S7: We show the Weather dataset's pairwise correlations as a pairwise separation function. Each colored line represents a different variable in the dataset.

[3] https://archive.ics.uci.edu/ dataset/321/electricityloaddiagrams20112014
[4] http://pems.dot.ca.gov

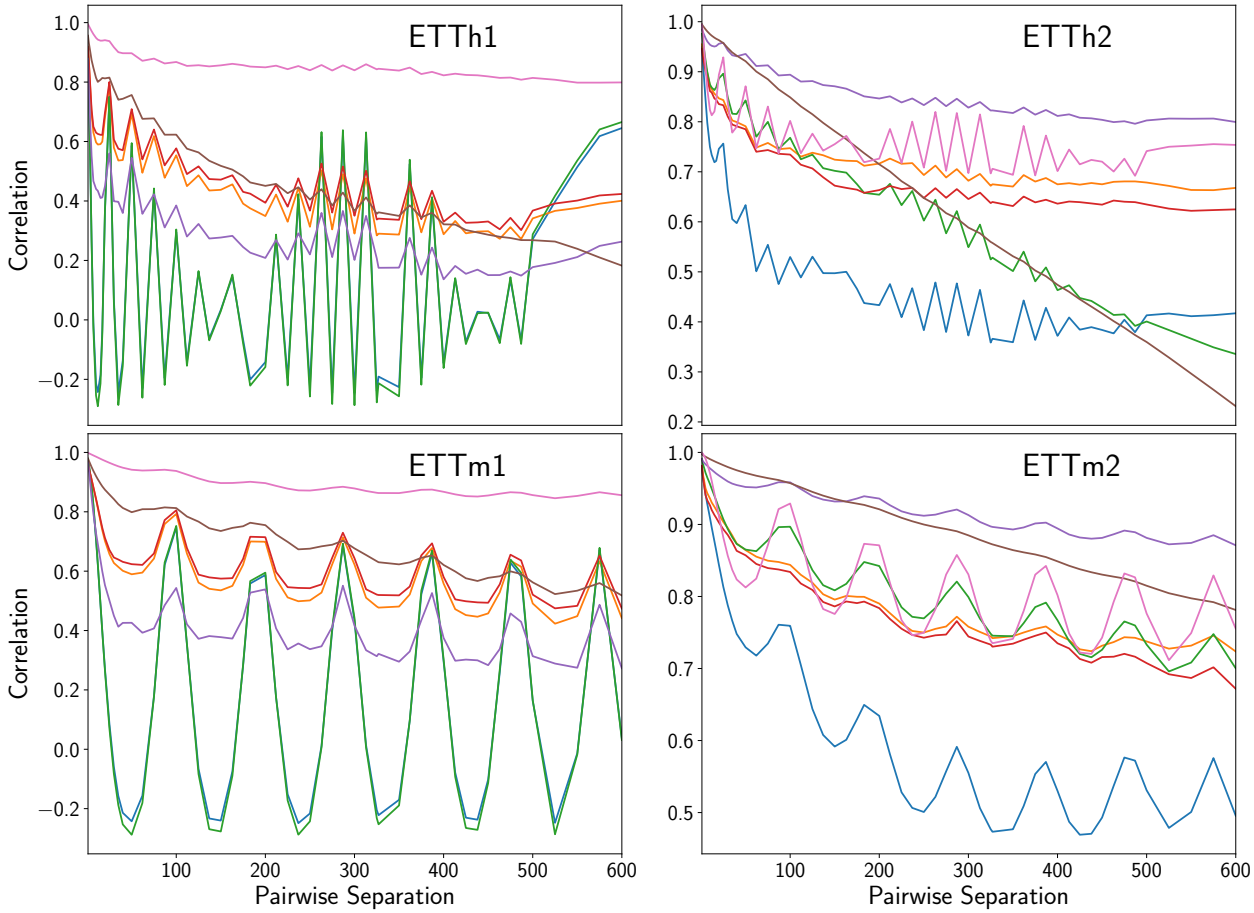

Figure S8: We show the ETT datasets' pairwise correlations as a pairwise separation function. Each colored line represents a different variable in the dataset.

S3 ARCHITECTURE DETAILS

S3.1 Transformer

We use a vanilla encoder-decoder Transformer architecture for all Transformer experiments. The decoder input is the last 48 time-steps in the sequence concatenated by an array of zeros with a length of the prediction window. For our baseline experiment, the decoder self-attention has a causal mask, which is common in Transformer architectures, while the encoder self-attention and decoder cross-attention do not have masks. Our RBCA variant of the Transformer applies RBCA to both the encoder and decoder self-attention, but not the decoder cross-attention. We do not apply RBCA to the decoder cross-attention because this process is already causal as it forecasts future points based on the given sequence. We present the Transformer hyperparameters used in this work in Table S2.

S3.2 PatchTST

Here, we review PatchTST's patching mechanism, as PatchTST is otherwise similar to a traditional Transformer with a linear readout head that simultaneously predicts all future points. For a detailed description of PatchTST, see Ref. (Nie et al., 2023). Let us consider an input time-series $\mathbf{X}$ with dimensions $[B, T_{\text{seq}}, D]$, where B is batch size, T_{seq} is the time interval, and D is the number of input dimensions. The multivariate input is first normalized to zero mean and unit variance along the time axis and flattened into D univariate inputs, resulting in a dimensionality of $[B \times D, T_{\text{seq}}]$. PatchTST batches these time series via a convolution with N filters, a stride s, and patch size p, resulting in a dimensionality of $[B \times D, P, N]$. Here, $P = (T_{\text{seq}} - p)//s + 1$ is the

number of patches, and // is integer division. At this point, the patched data is fed into the Transformer where a positional embedding is added and the patched embeddings are further contextualized by vanilla MHA (Vaswani et al., 2017). The linear output head takes the $[B \times D, P, N]$ output of the Transformer, flattens the last two dimensions, and applies a linear matrix multiplication (of shape $[P \times N, T_{\text{pred}}]$) to predict the output time series of length T_{pred}. The linear head returns a matrix of shape $[B \times D, T_{\text{pred}}]$. This matrix is reshaped into the original multivariate dimensionality ($[B, T_{\text{pred}}, D]$) and renormalized. We note that the patching mechanism is performed before applying the Transformer and is independent of it.

S3.3 Powerformer

Powerformer forecasts univariate time-series by combining RBCA, the commonly used encoder-only Transformer architecture, and patching (Nie et al., 2023). We selected PatchTST as the base of *Powerformer* since PatchTST already includes decomposes multivariate time-series into univariate channels and performs the patching. Additionally, similar to other modern time-series Transformer-based models (Zhou et al., 2023; Liu et al., 2024; Talukder et al., 2024), PatchTST (Nie et al., 2023) uses an encoder-only architecture with a linear readout head. *Powerformer* uses a standard Transformer encoder architecture with the primary difference being the replacement of the vanilla MHA by RBCA. We additionally remove the dropout applied to the attention weights because our implicit biases enforce a deterministic temporal dependency on pairwise correlations. Dropout enforces redundancy between pairwise correlations and is therefore in conflict with our implicit biases. The input data comes in with D variates and shape $[B, D, T_{\text{seq}}]$, where B is the batch size and T_{seq} is the input sequence length. The PatchTST patching mechanism (described in Appendix S3.2) returns the patched and normalized univariate embeddings with shape $[B \times D, P, N]$, where P is the number of patches and N is the size of the embeddings. These embeddings are encoded through multiple RBCA Transformer encoder layers (see Appendix S3.4 for hyperparameter details). The encoder output (of shape $[B \times D, P, N]$) is flattened along the last two dimensions ($[B \times D, P \times N]$) in preparation for the linear readout head. The linear readout matrix has the shape $[P \times N, T_{\text{pred}}]$ and after acting on the flattened encoder output produces the univariate forecasts of shape $[B \times D, T_{\text{pred}}]$. Finally, these univariate forecasts are renormalized and reshaped into the multivariate input structure $[B, D, T_{\text{pred}}]$ and returned to the user. Consequently, *Powerformer* is a simple Transformer-based model with causal and local masks added to the attention scores $\mathbf{S}$, and patching applied to the input data. We present the *Powerformer* hyperparameters used in this work in Table S2.

S3.4 Hyperparameter Tuning and Selection

In addition to tuning standard Transformer hyperparameters, *Powerformer* introduces two patching and two recency-bias hyperparameters. The patching mechanism is determined by the patch length and the patch stride. The patch length is the number of input tokens that are averaged together. The patch stride is the stride of the patching mechanism that helps reduce the size of the input data. For the recency-bias mask, one can tune the type of mask functional and it's decay time. As previously described, and illustrated in Fig. 2, we test four masks: power-law mask ($f^{(\text{PL})}(t)$), similarity power-law mask ($f^{(\text{SPL})}(t)$), and two Butterworth masks for $n \in 1, 2$ ($f_n^{(\text{BW})}(t)$). For each mask, one may try different decay constants α that determine how fast the mask decays. In this works we tried the following decay constants for each mask: for $f^{(\text{PL})}(t)$ we evaluate $\alpha \in \{0.1, 0.25, 0.5, 0.75, 1.0\}$; for $f^{(\text{SPL})}(t)$ we evaluate $\alpha \in \{0.1, 0.5, 1.0, 2.0\}$; and for $f_n^{(\text{BW})}(t)$ we evaluate $\alpha \in \{2, 5, 10, 15, 20\}$. Although *Powerformer* introduces two more hyperparameters over PatchTST, we observe in Tables S5, S6, S7, and S8 that many masks are not very sensitive to changes in α within a certain range. For this reason, tuning α will likely be very fast as it does not require much fine-tuning after finding this large range.

We note that in addition to the hyperparameters previously discussed, we try two different input sequence lengths (336, 512). There is no standard measurement rate for general time-series datasets, and in general, their dynamics are quite different and happen on different time scales. Thus, the amount of variation within some number of time steps depends on both the dynamics being measured and the measurement rate. The input sequence length must therefore change to encompass the same amount of information (or variation) for datasets with differing measurement rates and temporal variability.

To reproduce our results, we list our hyperparameters used on each dataset for *Powerformer* (Table S2) and Transformer (Table S3). For some models, we apply weight decay to the encoder only, leaving the forecasting head without regularization. For each dataset and model, we trained with the following 3 random seeds: 2021,

1776, and 1953.

Table S2: We provide *Powerformer* architecture and training parameters used for each dataset.

Parameters	**ETTh1**	**ETTh2**	**ETTm1**	**ETTm2**	**Weather**	**Electricity**	**Traffic**
Sequence Length	336 / 512	336 / 512	336 / 512	336 / 512	336 / 512	336 / 512	336 / 512
Patch Length	16	16	16	16	16	16	16
Patch Stride	8	8	8	8	8	8	8
Encoder Layers	3	3	3	3	3	3	3
Embedding Size	16	16	128	128	128	128	128
MHA Heads	4	4	16	16	16	16	16
MHA Feed Forward	128	128	256	256	256	256	256
Dropout %	30	30	20	20	20	20	20
Linear Dropout %	30	30	20	20	20	20	20
Train Epochs	100	100	100	100	100	100	100
Patience	–	–	20	20	20	10	10
Learning Rate	10^{-3}	10^{-4}	10^{-5}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Weight Decay	1.0	1.0	–	0.05	–	–	–
Batch Size	128	128	128	128	128	32	24

Table S3: We provide Transformer architecture and training parameters used for each dataset.

Parameters	**ETTh1**	**ETTh2**	**ETTm1**	**ETTm2**	**Weather**	**Electricity**	**Traffic**
Sequence Length	256	256	256	256	256	256	256
Decoder Input Length	48	48	48	48	48	48	48
Encoder Layers	2	2	2	2	2	2	2
Decoder Layers	1	1	1	1	1	1	1
Embedding Size	512	512	512	512	512	512	512
MHA Heads	8	8	8	8	8	8	8
MHA Feed Forward	2048	2048	2048	2048	2048	2048	2048
Dropout %	5	5	5	5	5	5	5
Linear Dropout %	5	5	5	5	5	5	5
Train Epochs	100	100	100	100	100	100	100
Patience	–	–	–	–	–	–	–
Learning Rate	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Batch Size	128	128	128	128	128	128	128

S4 FORECASTING VISUALIZATION

Here, we show *Powerformer* forecasting visualizations and compare various decay time constants against ground truth and PatcTST (Nie et al., 2023). We also highlight plots where both *Powerformer* and PatchTST struggle. In these cases, *Powerformer* generally outperforms by better capturing faster-moving features, non-periodic features, and is better able to alter periodic predictions. We believe this is due to *Powerformer's* ability to focus on and amplify local high-frequency features while dampening or removing long-time high-frequency components. RBCA achieves this through the power-law masks $f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$, which amplify local correlations and dampen long-time correlations.

S5 EXPERIMENTS

S5.1 Flip Flop Experiment Details

We generate the flip-flop dataset by producing a string of 512 characters, alternating between instructional characters $c \in \{w, r, i\}$ and numbers $n \int \{0, 0\}$ (Press et al., 2022). The character commands dictate the model to do the following: write (w) the following number to memory (replacing the previous one), ignore (i) the following number and do not commit it to memory, and read (r) the number saved in memory. The first character of the string is a w, the last character is r, and the character following r is space holder '-' that is meant to be replaced by the predicted number. During training, there can be multiple r commands where the most recently written number is predicted. We also generate and test the models on an out-of-distribution (OOD) dataset, where there are many ignore (i) commands between the last write and the last read commands. When evaluating the OOD task, we only record results from the last read.

Due to the recurrent neural network (RNNs) architecture, these write-ignore-read capabilities are built in; however, this is not the case for Transformers (Vaswani et al., 2017). To test this, we use a 2-layer Transformer with embedding size 512, max sequence length of 512, absolute positional embeddings, and 4 attention heads. In our experiments, we either varied or removed the causal mask. We trained each model with a learning rate of $10^{-}5$ for 11000 epochs with a batch size of 32 and a weight decay scale of 0.1.

The OOD evaluation as a function of training, Fig. S12, show that the power-law recency biases outperform. In general, the power-law bias $f^{(\mathrm{PL})}(t)$ consistently outperforms regular MHA and other recency biases. Interestingly, the ALiBi bias $f^{(\mathrm{E})}(t)$ does significantly worse than MHA even though it is introducing a recency bias.

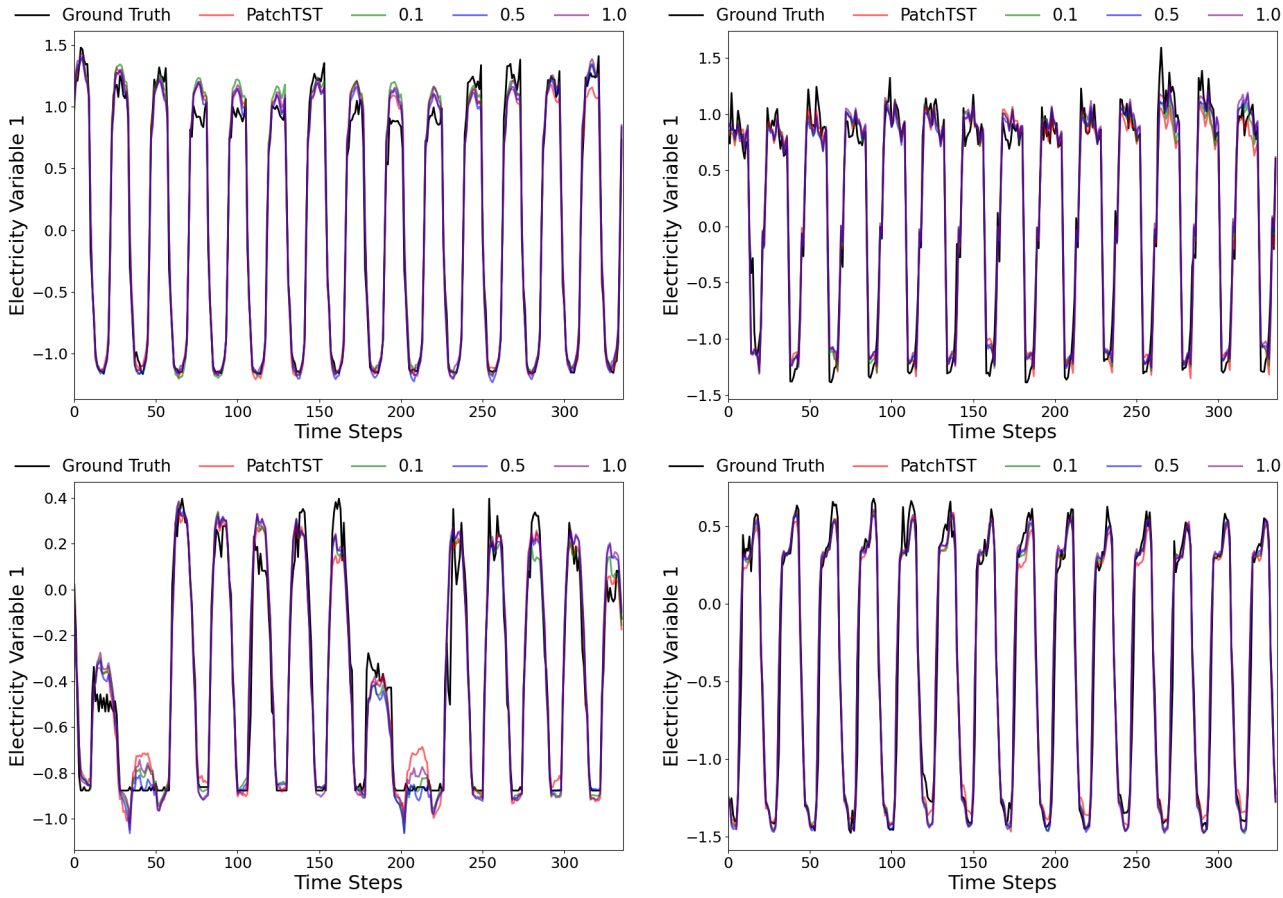

Figure S9: We show forecasting results on the Electricity dataset for *Powerformer* with $f^{(\mathrm{PL})}(t)$ and PatchTST. The colored lines represent PatchTST and different $f^{(\mathrm{PL})}(t)$ decay constants. For these forecasts, the sequence length is 512 and the prediction length is 336.

The similarity power-law $f^{(\mathrm{SPL})}(t)$ performs similarly to MHA, likely because this recency bias is relatively flat.

S5.2 Time-Series Experiment Details

All experiments are evaluated 3 times with different initialization (S3.4). For each dataset, we select a single input sequence based on the best-performing input sequence length across all prediction lengths. We choose a single input sequence length for each dataset because the strength of the correlations between time steps is independent of the prediction length, but the required input sequence length to capture the information necessary to forecast is unique for each dataset. For each prediction length, we treat the attention mask as a hyperparameter and select the best mask and time constant pair. We select attention masks for each prediction length because the forecasting timescale will likely emphasize short or long-term correlations for shorter or longer forecasting windows. We make our selections based on the MSE error. Section S3.4 provides more details on the hyperparameter tuning and the hyperparameters used in this work.

S5.3 Powerformer

We evaluate the weight power-law ($f^{(\mathrm{PL})}(t)$) and similarity power-law ($f^{(\mathrm{SPL})}(t)$) filters on all 7 datasets. For $f^{(\mathrm{PL})}(t)$ we evaluate $\alpha \in \{0.1, 0.25, 0.5, 0.75, 1.0\}$ and for $f^{(\mathrm{SPL})}(t)$ we evaluate $\alpha \in \{0.1, 0.5, 1.0, 2.0\}$. Similar to PatchTST (Nie et al., 2023) we evaluate the power-law filters with two input lengths: 336 and 512.

We evaluate both Butterworth filters $\left(f_1^{(\mathrm{BW})}(t) \text{ and } f_2^{(\mathrm{BW})}(t)\right)$ on all the ETT datasets and the Weather datasets. For each Butterworth filter, we evaluated $\alpha \in \{2, 5, 10, 15, 20\}$ with an input length of 336. Due to

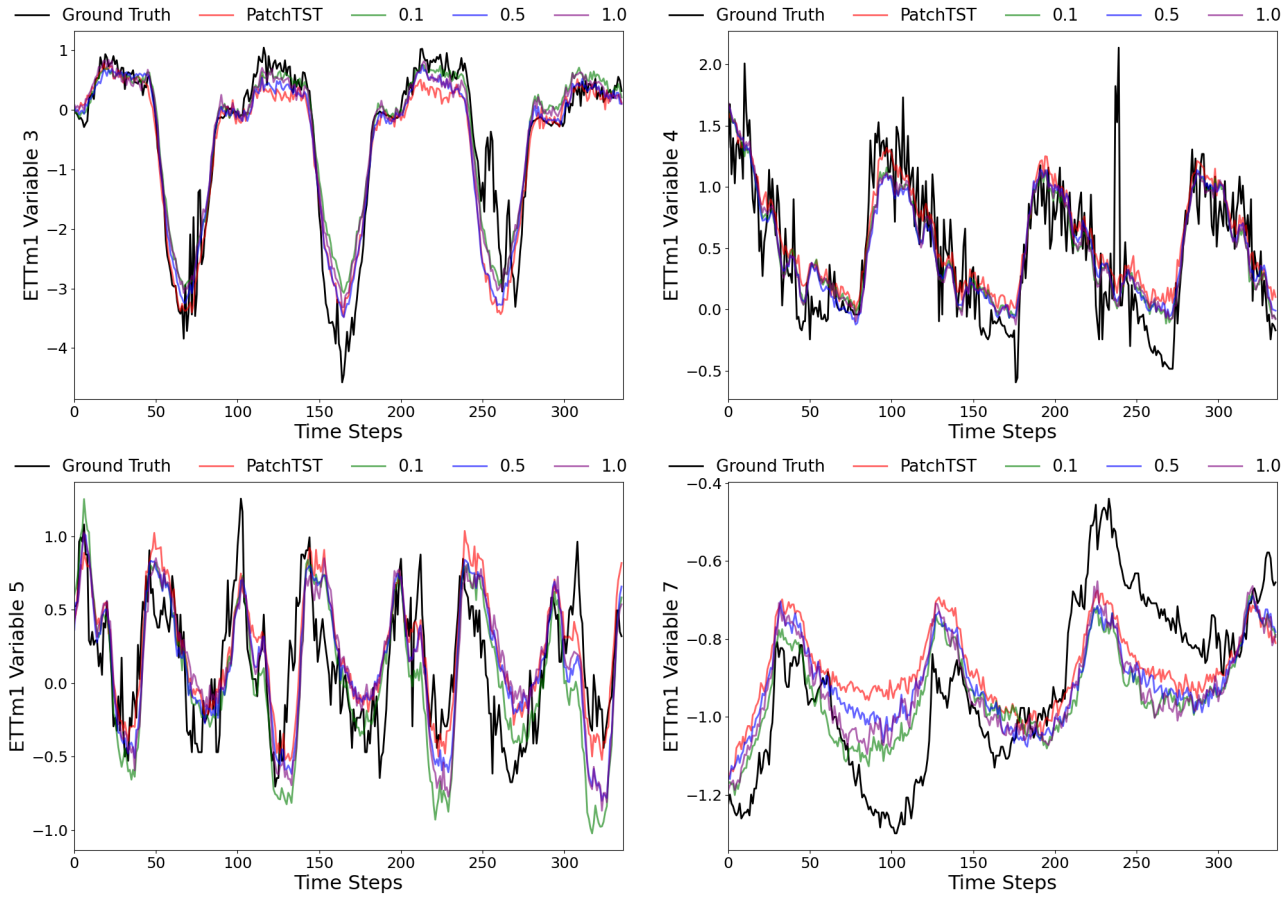

Figure S10: We show forecasting results on the ETTm1 dataset for *Powerformer* with $f^{(\mathrm{PL})}(t)$ and PatchTST. The colored lines represent PatchTST and different $f^{(\mathrm{PL})}(t)$ decay constants. For these forecasts, the sequence length is 512 and the prediction length is 336.

poor performance, we did not evaluate the Butterworth filters on the larger Electricity and Traffic datasets, nor with the longer 512 input length.

S5.4 Transformer

We train a standard encoder-decoder Transformer architecture both with and without RBCA. Generic Transformer models often incorporate causality through masking. To account for this, our Transformer benchmark uses masking for the decoder MHA but not for the encoder MHA nor the decoder cross-attention. We investigate RBCA's effects by replacing the encoder and decoder self-attention, but not the decoder cross-attention as it is already causal. Section S3.1 provides more architectural details.

We train Transformer models, with and without RBCA, using the weight power-law ($f^{(\mathrm{PL})}(t)$) filter on all 7 datasets. We explore the following decay constants: $\alpha \in [0.1, 0.25, 0.5, 0.75, 1.0]$ and provide further experimental detail in sections 4 and S5.2.

S5.5 Baseline Selection

We compare *Powerformer* against multiple Transformer-based state-of-the-art methods. We populate Table 1 with values taken from the models' respective papers. All models except TOTEM (Talukder et al., 2024) and iTransformer (Liu et al., 2024) treat the input length as a hyperparameter and report the best results. PatchTST (Nie et al., 2023) provides results for two input sequence lengths (336 and 512). In Table 1 we report the best result between these input sequence lengths for each dataset based on the MSE. We employ the same

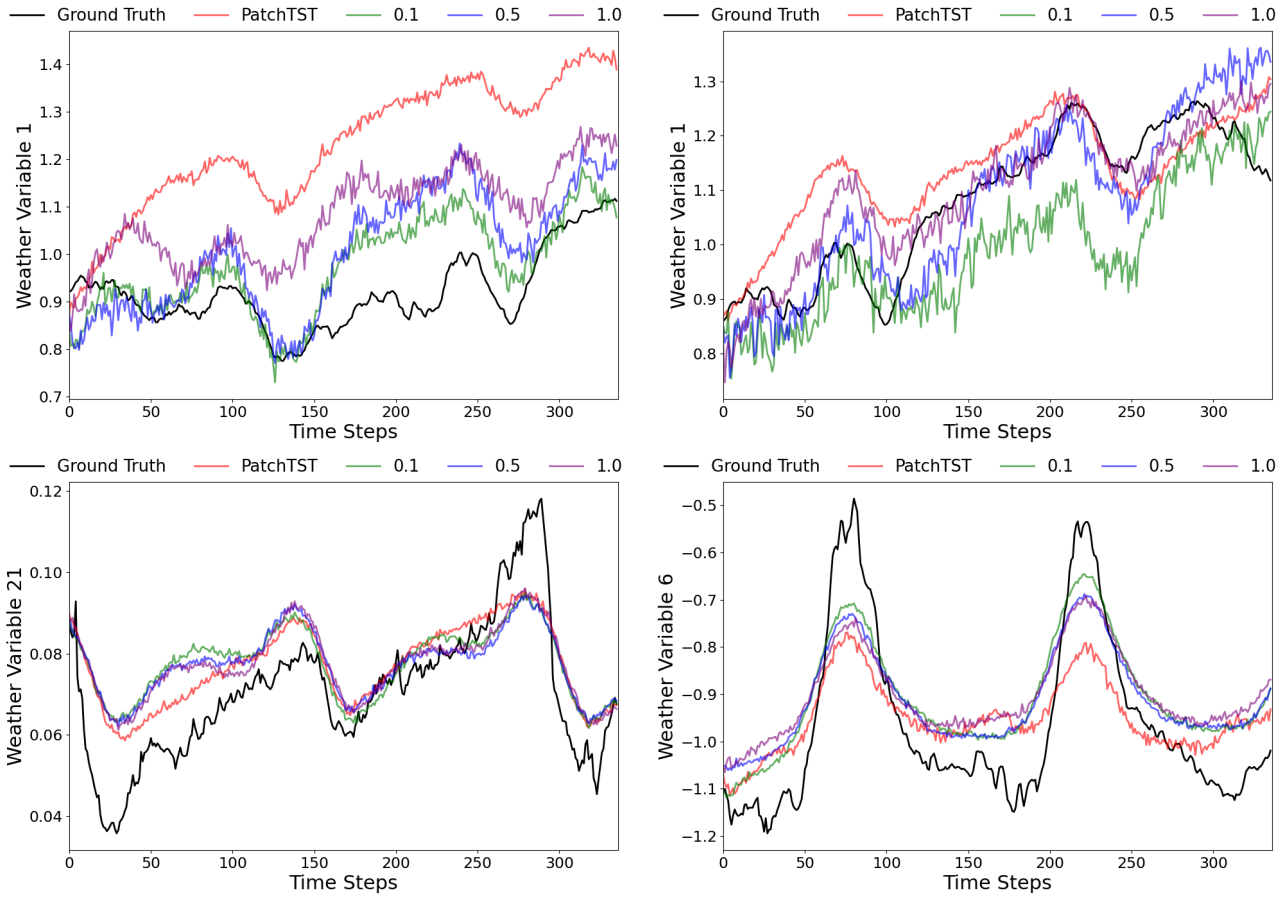

Figure S11: We show forecasting results on the Weather dataset for *Powerformer* with $f^{(\mathrm{PL})}(t)$ and PatchTST. The colored lines represent PatchTST and different $f^{(\mathrm{PL})}(t)$ decay constants. For these forecasts, the sequence length is 512 and the prediction length is 336.

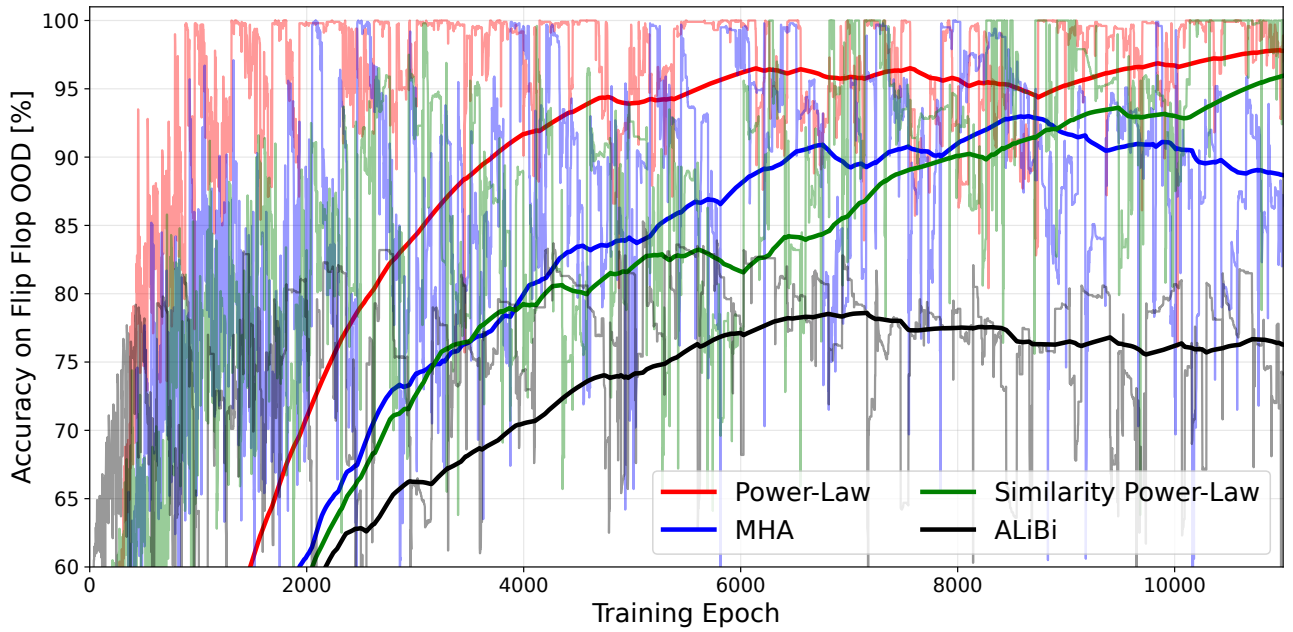

Figure S12: The flip-flip OOD evaluation for various recency biases and MHA. The transparent lines are the accuracies, while the solid lines are an exponential moving average.

input sequence length selection for *Powerformer*.

S5.6 Computing Resources

These experiments were run on NVIDIA A40 and NVIDIA TITAN RTX GPUs. Compute time and resources are highly variable between datasets; the smallest (ETTh1) takes roughly 20 minutes on a single A40, while the largest (Traffic) takes roughly 15 hours on 4 A40s.

S6 TRANSFORMER RESULTS

Here, we present the full results from evaluating Transformer with RBCA. The architecture is described in Appendix S3.1 and the experimental details are provided in Appendices S5.2 and S5.4.

Table S4 provides the aggregate Transformer performance without RBCA and with RBCA for varying decay times (α). RBCA decisively outperforms MHA on 4 or the 7 datasets and ties with the number of best results on ETTh2. Looking further into ETTh2 we see that when RBCA outperforms MHA it does so by much larger margins (16%) than when it underperforms (2%). Moreover, we observe the same larger margins of overperformance across all of the datasets. This indicates that when the natural pairwise distribution is found we can see significant improvements in MSE and MAE, but the effects of imposing the wrong pairwise distribution are much less significant.

In Figs. S13-S16 we present the attention score and weight distributions with and without applying the local-causal mask. The distributions can have significant deviations between datasets. In general, the RBCA decoder self-attention shows the smallest deviation from the MHA base case. We note that the MHA base case already has a causal mask. The last RBCA encoder self-attention layer often shows the largest difference between RBCA and MHA distributions. This makes sense since the MHA encoder attention is not causal. We do not alter the decoder cross-attention, but we still observe shifts in the distribution, which differ between datasets. Interestingly, we do not observe the same bimodal distribution in attention weights as we do for *Powerformer*. This may be due to the Transformer's encoder-decoder structure, whereas *Powerformer* only has an encoder.

Table S4: We provide the forecasting MSE and MAE on the test sets for Transformer with MHA (base case) and for Transformer with RBCA and the weight power-law mask $f^{(\mathrm{PL})}(t)$. The top row indicates the model or decay constant (α), the bold numbers are the best (lowest) performance, and the underlined numbers are the second best.

		Base Case		0.1		0.25		0.5		0.75		1	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.936	0.741	0.955	0.752	0.944	0.745	0.921	0.729	0.896	0.711	**0.874**	**0.693**
	192	0.988	0.769	0.916	0.759	0.951	0.789	0.887	0.749	0.895	**0.745**	**0.876**	0.749
	336	1.061	0.815	1.067	0.835	1.063	0.814	1.054	0.809	**1.038**	0.836	1.044	**0.802**
	720	1.113	0.881	**1.047**	**0.842**	1.061	0.855	1.180	0.893	1.185	0.895	1.259	0.899
ETTh2	96	1.585	1.038	**1.344**	**0.918**	1.349	0.920	1.392	0.936	1.496	1.000	1.511	1.005
	192	**1.889**	**1.131**	1.899	1.133	3.025	1.459	2.916	1.423	2.800	1.384	2.761	1.365
	336	**2.668**	**1.384**	2.875	1.393	2.872	1.386	2.879	1.412	2.860	1.409	2.951	1.442
	720	3.143	1.528	3.389	1.570	3.455	1.582	3.533	1.621	2.635	1.331	**2.488**	**1.272**
ETTm1	96	0.560	0.545	0.566	0.533	0.592	0.545	0.563	0.541	0.535	0.522	**0.487**	**0.492**
	192	0.874	0.729	0.907	0.724	0.838	0.711	0.830	0.698	0.774	0.686	**0.718**	**0.645**
	336	0.960	0.786	0.948	0.786	0.938	0.779	0.956	0.786	1.029	0.800	**0.888**	**0.752**
	720	0.900	0.746	**0.864**	**0.710**	0.886	0.729	0.902	0.749	0.892	0.729	1.005	0.788
ETTm2	96	**0.597**	**0.614**	0.619	0.628	0.632	0.636	0.660	0.653	0.697	0.674	0.700	0.638
	192	**0.849**	**0.749**	0.876	0.761	0.893	0.769	0.927	0.787	0.970	0.808	1.011	0.827
	336	**1.227**	**0.909**	1.259	0.920	1.271	0.925	1.295	0.936	1.320	0.947	1.333	0.955
	720	**2.176**	**1.237**	2.214	1.241	2.213	1.241	2.216	1.252	2.209	1.252	2.195	1.250
Weather	96	0.250	0.333	0.211	0.297	0.209	0.302	**0.207**	**0.291**	0.216	0.305	0.228	0.316
	192	0.344	0.414	0.295	0.373	**0.287**	**0.367**	0.297	0.375	0.307	0.383	0.313	0.390
	336	0.435	0.467	0.403	0.446	0.416	0.455	0.420	0.458	0.410	0.452	**0.384**	**0.436**
	720	0.463	0.489	0.420	0.460	0.435	0.471	0.424	0.464	0.426	0.468	**0.403**	**0.454**
Electricity	96	0.265	0.360	0.271	0.364	**0.264**	**0.355**	0.266	0.359	0.272	0.367	0.280	0.370
	192	0.284	0.378	0.278	0.372	0.286	0.378	**0.270**	**0.366**	0.276	0.369	0.291	0.386
	336	0.280	0.371	0.289	0.381	0.290	0.381	0.288	0.381	0.281	0.369	**0.277**	**0.366**
	720	0.300	0.382	0.318	0.395	0.308	0.389	0.296	0.378	0.302	0.383	**0.289**	**0.373**
Traffic	96	0.662	0.365	0.656	0.359	0.653	0.357	0.652	**0.355**	0.669	0.364	**0.649**	0.358
	192	**0.655**	**0.350**	0.675	0.368	0.669	0.361	0.669	0.365	0.660	0.356	**0.655**	0.355
	336	**0.651**	**0.350**	0.652	0.353	0.657	0.357	0.670	0.364	0.667	0.361	0.658	0.358
	720	**0.660**	**0.354**	0.669	0.363	0.676	0.360	0.667	0.355	0.668	0.361	0.689	0.370

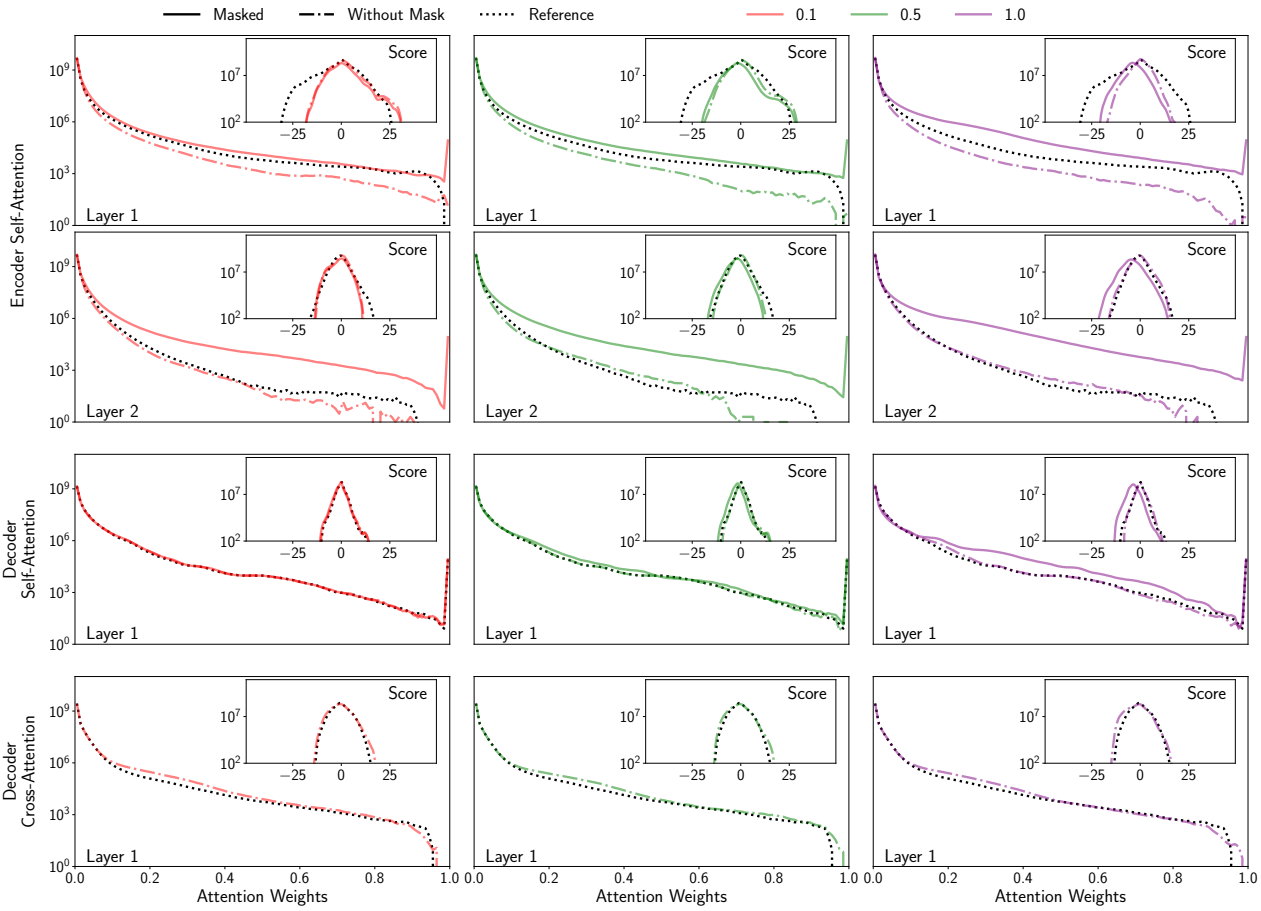

Figure S13: We present the Transformer attention score (inset) and weight distributions on the Weather dataset with a forecasting length of 96. The dotted line represents the reference MHA Transformer results, the dashed-dotted line represents RBCA with $f^{(\mathrm{PL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{PL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

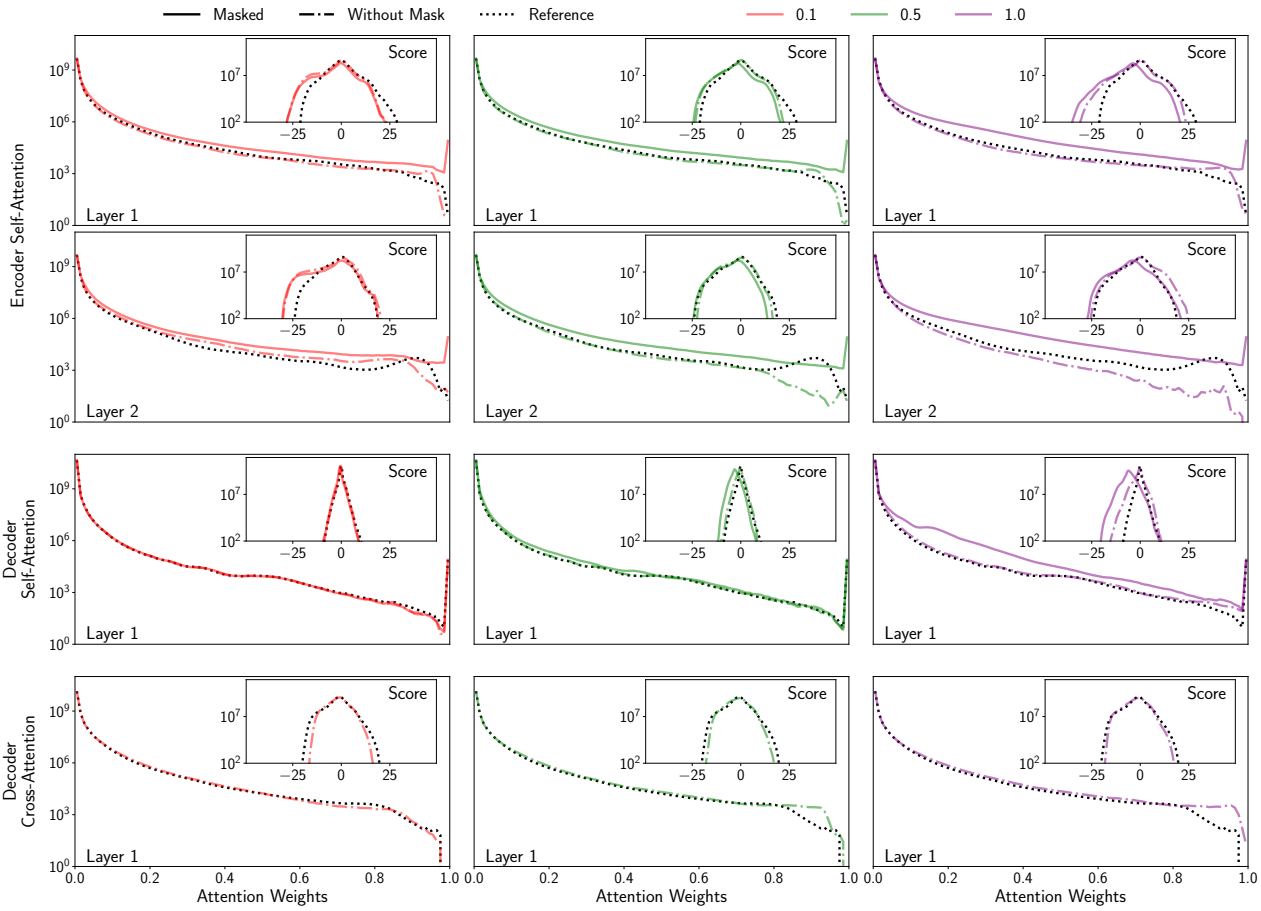

Figure S14: We present the Transformer attention score (inset) and weight distributions on the Weather dataset with a forecasting length of 720. The dotted line represents the reference MHA Transformer results, the dashed-dotted line represents RBCA with $f^{(\mathrm{PL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{PL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

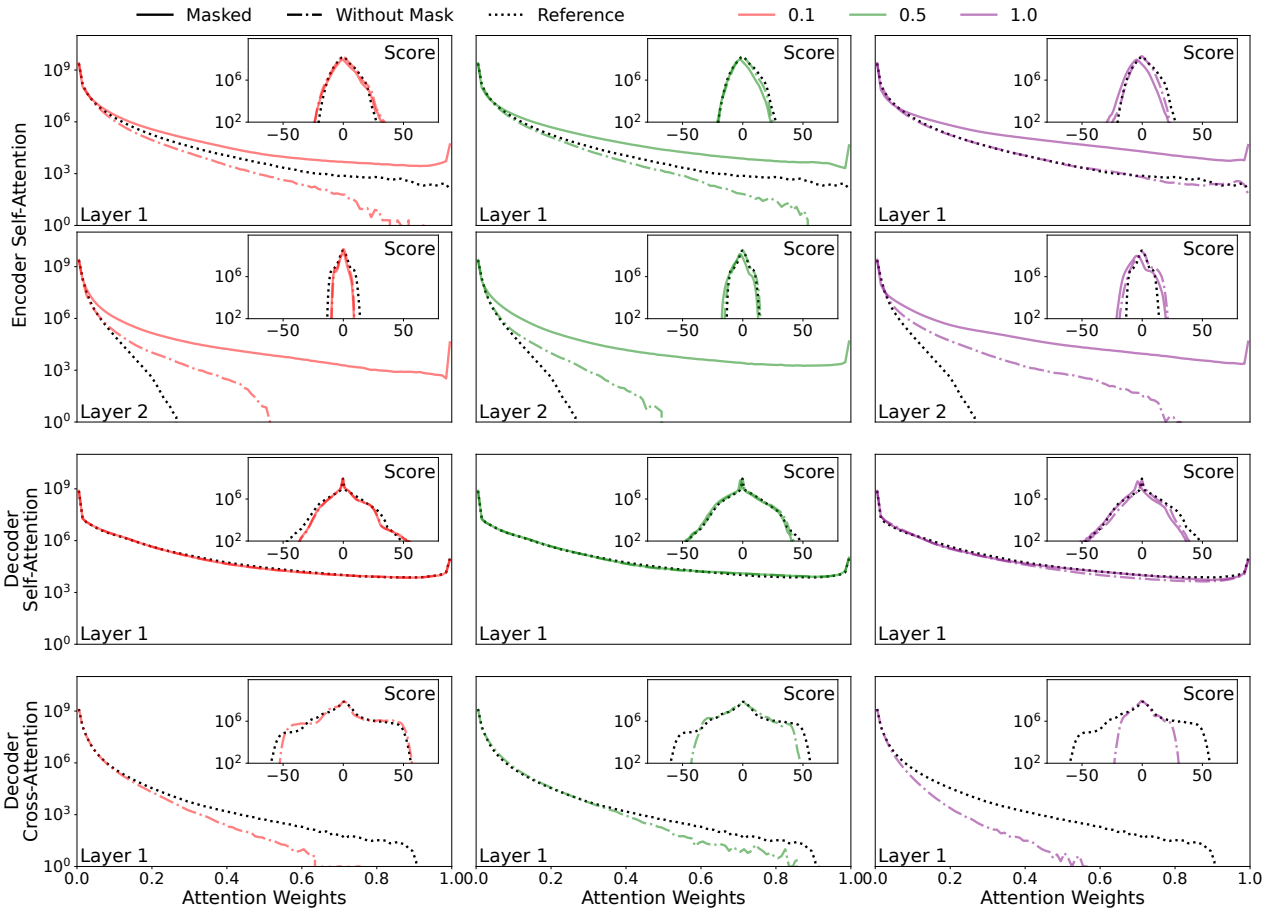

Figure S15: We present the Transformer attention score (inset) and weight distributions on the Electricity dataset with a forecasting length of 96. The dotted line represents the reference MHA Transformer results, the dashed-dotted line represents RBCA with $f^{(\mathrm{PL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{PL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

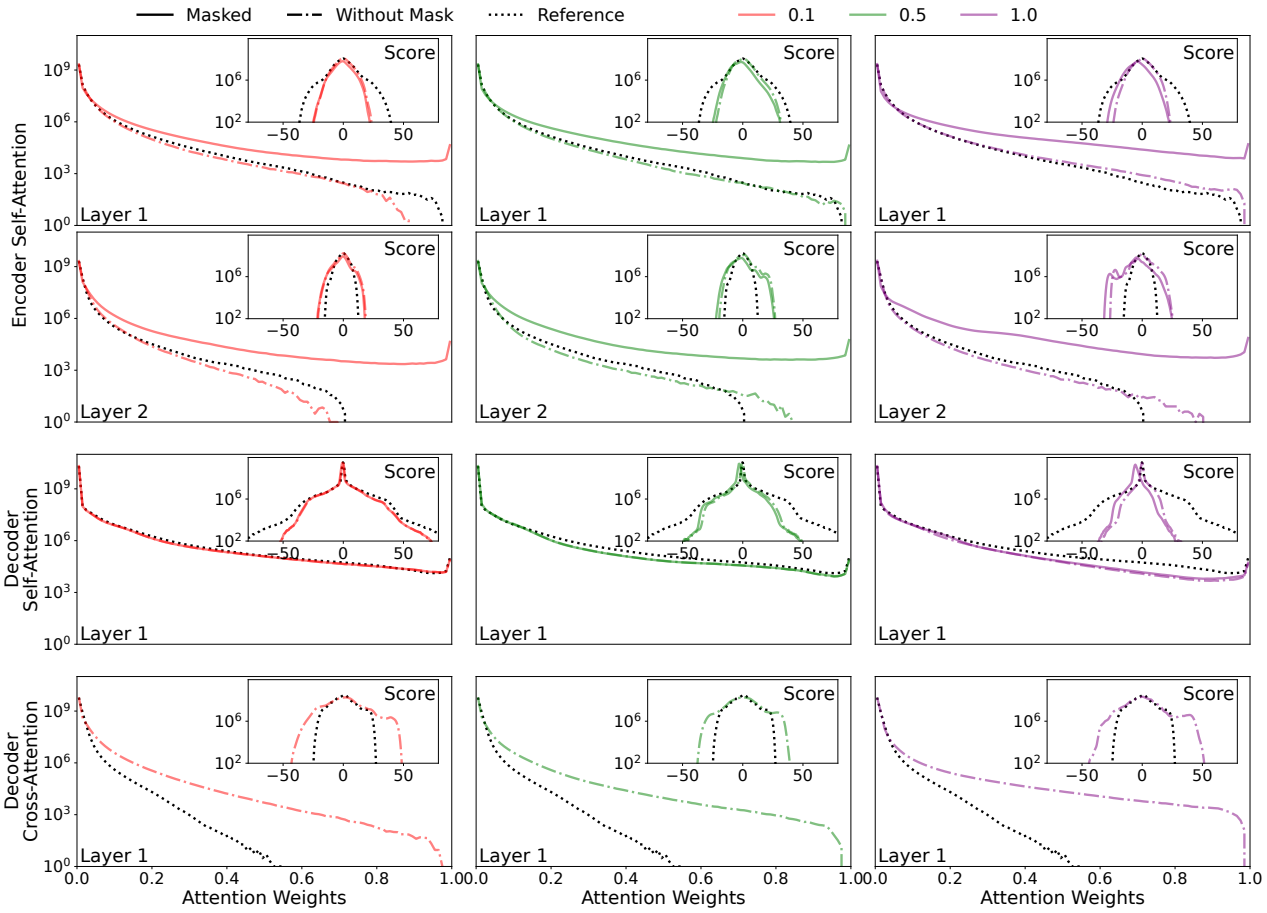

Figure S16: We present the Transformer attention score (inset) and weight distributions on the Electricity dataset with a forecasting length of 720. The dotted line represents the reference MHA Transformer results, the dashed-dotted line represents RBCA with $f^{(\mathrm{PL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{PL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

S7 POWERFORMER RESULTS

We present the full *Powerformer* results on all the evaluated masks: $f^{(\mathrm{PL})}(t)$, $f^{(\mathrm{SPL})}(t)$, $f_1^{(\mathrm{BW})}(t)$, and $f_2^{(\mathrm{BW})}(t)$. Architecture details can be found in Appendix S3.3, while Appendices S5.2 and S5.3 outline the experimental details.

S7.1 Weight Power-Law Mask

Table S5 presents the aggregate MSE and MAE results for weight power-law mask ($f^{(\mathrm{PL})}(t)$) with varying time decays (α), forecasting lengths, and input sequence lengths. The 512 input sequence generally outperforms the shorter 336 input length. As expected, the best performing α varies between datasets, but it also varies as a function of forecast length and input sequence length. Given that some of the best-performing decay lengths are $\alpha = 1$ with further experiments one may improve upon these values. However, our $f^{(\mathrm{SPL})}(t)$ mask includes faster-decaying weights, as shown in Fig. 2.

We compare all datasets' attention score and weight distributions both with MHA and RBCA in Figs S17-S20. We observe similar bimodal distributions for all datasets and note wider $\mathbf{C}^{(\mathrm{C,R})}$ distributions for ETTm1, ETTm2, and Weather.

Table S5: We compare *Powerformer's* performance with the weight power law mask $f^{(\mathrm{PL})}(t)$ on standard time-series datasets for varying decay lengths. The best results are bolded and second best are underlined.

Delay			0.1		0.25		0.5		0.75		1	
Metric			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	336	96	0.377	**0.401**	0.377	**0.401**	0.377	**0.401**	**0.376**	**0.401**	**0.376**	**0.401**
		192	**0.413**	**0.421**	**0.413**	**0.421**	**0.413**	**0.421**	**0.413**	**0.421**	0.414	**0.421**
		336	**0.424**	**0.430**	0.425	**0.430**	0.425	**0.430**	0.425	**0.430**	0.425	**0.430**
		720	**0.437**	**0.455**	**0.437**	**0.455**	**0.437**	**0.455**	**0.437**	0.456	0.438	0.456
	512	96	**0.369**	**0.399**	**0.369**	**0.399**	0.370	**0.399**	0.370	**0.399**	0.370	**0.399**
		192	0.404	0.420	0.404	0.420	**0.402**	**0.418**	0.403	**0.418**	0.403	**0.418**
		336	**0.414**	**0.428**	**0.414**	**0.428**	0.415	0.429	0.415	0.430	0.416	0.431
		720	**0.440**	**0.460**	**0.440**	**0.460**	**0.440**	**0.460**	0.443	0.462	0.443	**0.460**
ETTh2	336	96	**0.274**	**0.335**	**0.274**	0.336	0.275	0.336	0.275	**0.335**	0.275	**0.335**
		192	0.341	0.381	**0.338**	**0.379**	0.340	**0.379**	0.340	**0.379**	0.341	**0.379**
		336	**0.325**	**0.379**	0.326	0.380	0.331	0.384	0.333	0.383	0.334	0.383
		720	0.377	0.420	**0.376**	**0.419**	0.381	0.423	0.381	0.422	0.381	0.422
	512	96	**0.274**	**0.337**	**0.274**	**0.337**	**0.274**	**0.337**	0.275	**0.337**	0.275	0.338
		192	**0.340**	**0.381**	**0.340**	**0.381**	0.341	0.382	0.342	0.382	0.342	0.382
		336	**0.330**	**0.387**	0.331	**0.387**	0.333	0.388	0.334	0.388	0.334	0.388
		720	**0.383**	0.426	0.384	0.426	**0.383**	0.426	0.384	0.426	**0.383**	**0.425**
ETTm1	336	96	0.292	0.343	**0.290**	**0.342**	0.292	0.343	0.293	0.344	0.292	0.345
		192	0.333	0.371	0.334	0.372	0.332	0.371	0.332	0.370	**0.330**	**0.369**
		336	0.362	0.392	0.361	0.391	0.362	0.390	**0.359**	**0.389**	0.362	**0.389**
		720	0.413	0.423	0.413	0.422	0.413	**0.421**	0.413	**0.421**	**0.412**	0.423
	512	96	0.291	**0.345**	**0.290**	**0.345**	**0.290**	**0.345**	0.291	0.346	0.292	0.346
		192	0.331	0.370	0.330	0.370	**0.329**	**0.368**	0.332	0.370	0.332	0.372
		336	0.362	0.391	0.362	**0.390**	**0.361**	0.392	0.362	**0.390**	0.363	0.391
		720	0.417	0.428	**0.415**	0.425	0.417	**0.423**	0.420	0.426	0.416	0.430
ETTm2	336	96	**0.162**	**0.252**	0.165	0.255	0.163	0.253	0.164	0.254	0.164	0.254
		192	**0.222**	**0.292**	**0.222**	0.293	**0.222**	0.293	**0.222**	0.293	**0.222**	0.293
		336	**0.277**	**0.329**	**0.277**	**0.329**	**0.277**	**0.329**	**0.277**	**0.329**	0.278	0.330
		720	0.363	0.381	0.363	0.381	0.362	0.382	**0.359**	**0.380**	**0.359**	0.384
	512	96	0.165	**0.255**	0.165	**0.255**	0.165	0.256	0.165	0.257	**0.164**	**0.255**
		192	0.222	0.295	0.224	0.297	0.224	0.296	0.223	0.296	**0.221**	**0.294**
		336	0.274	0.329	0.274	0.329	0.274	0.329	**0.272**	**0.327**	0.273	**0.327**
		720	0.358	0.383	0.358	0.383	0.358	0.382	0.358	0.383	**0.356**	**0.381**
Weather	336	96	**0.151**	0.201	**0.151**	0.201	**0.151**	**0.200**	0.152	**0.200**	0.154	0.202
		192	**0.196**	0.243	**0.196**	0.242	**0.196**	**0.241**	**0.196**	**0.241**	0.197	0.242
		336	0.248	0.283	**0.247**	0.283	**0.247**	**0.281**	**0.247**	0.282	**0.247**	**0.281**
		720	0.318	0.334	0.318	0.334	0.318	0.334	**0.317**	**0.333**	**0.317**	**0.333**
	512	96	0.148	0.198	**0.147**	**0.197**	**0.147**	0.199	0.148	0.199	0.148	0.199
		192	0.193	0.241	0.193	0.241	0.193	0.241	**0.191**	**0.239**	0.193	0.241
		336	**0.243**	0.281	0.244	0.281	**0.243**	0.280	**0.243**	**0.279**	0.244	0.280
		720	**0.311**	0.330	**0.311**	0.330	0.314	0.331	0.312	0.330	**0.311**	**0.329**
Electricity	336	96	0.131	**0.224**	0.131	**0.224**	**0.130**	**0.224**	0.131	**0.224**	0.131	**0.224**
		192	**0.147**	0.241	**0.147**	**0.240**	**0.147**	**0.240**	**0.147**	0.241	0.148	0.241
		336	**0.164**	0.259	**0.164**	**0.257**	**0.164**	0.259	**0.164**	0.258	**0.164**	0.259
		720	**0.201**	**0.291**	**0.201**	0.292	0.202	0.292	**0.201**	**0.291**	0.202	0.293
	512	96	0.130	0.224	**0.129**	0.224	**0.129**	0.224	**0.129**	**0.223**	**0.129**	**0.223**
		192	0.147	0.241	**0.146**	0.241	0.147	**0.240**	**0.146**	**0.240**	0.147	0.241
		336	0.163	0.258	**0.162**	**0.257**	**0.162**	0.258	**0.162**	**0.257**	**0.162**	0.258
		720	0.199	0.290	**0.198**	0.290	**0.198**	0.290	**0.198**	**0.289**	**0.198**	0.290
Traffic	336	96	0.371	**0.253**	**0.370**	**0.253**	0.372	0.254	0.372	0.254	0.373	0.255
		192	**0.388**	**0.260**	**0.388**	**0.260**	0.390	0.261	0.390	0.261	0.390	0.261
		336	**0.400**	0.267	**0.400**	**0.266**	0.404	0.269	0.402	0.268	0.403	0.268
		720	0.435	0.287	**0.434**	**0.286**	0.435	**0.286**	**0.434**	**0.286**	**0.434**	**0.286**
	512	96	**0.365**	**0.252**	**0.365**	**0.252**	0.367	0.253	0.367	0.253	0.394	0.282
		192	**0.382**	**0.258**	0.391	0.269	0.383	0.259	**0.382**	0.260	0.407	0.286
		336	0.393	**0.265**	**0.391**	**0.265**	0.392	**0.265**	0.393	**0.265**	0.394	**0.265**
		720	0.435	0.289	0.432	0.288	0.432	0.287	0.431	0.287	**0.430**	**0.286**

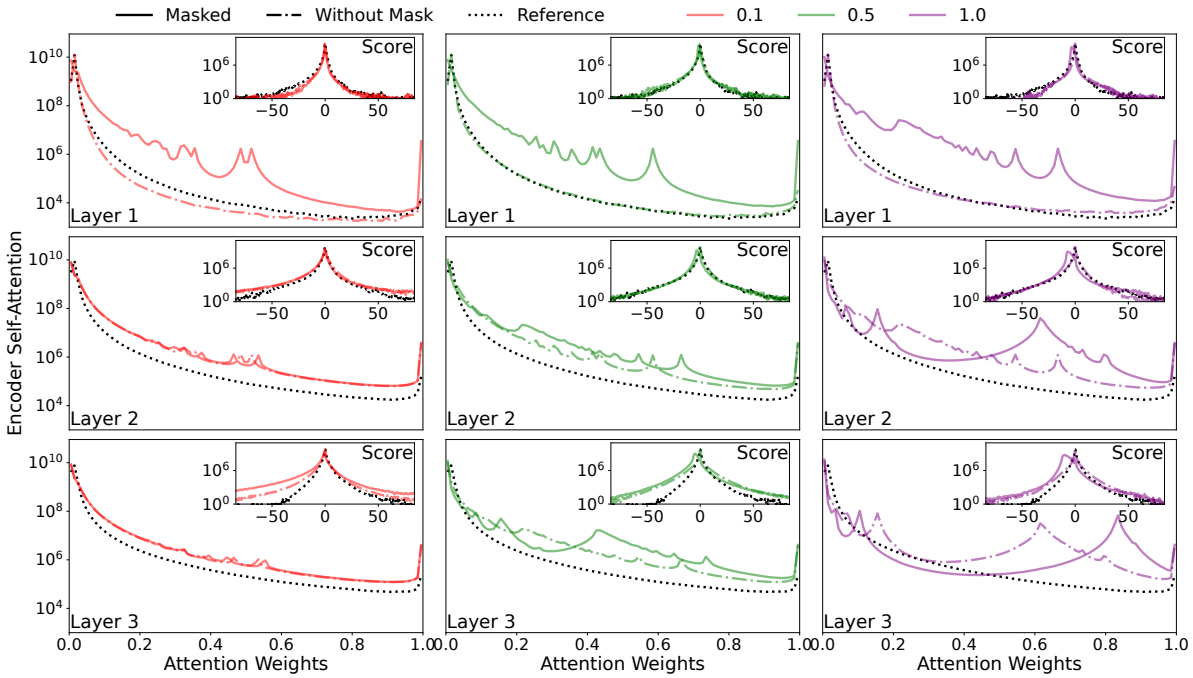

Figure S17: We present the *Powerformer* attention score (inset) and weight distributions on the Weather dataset with a forecast and input length of 96 and 512, respectively. The dotted line represents the reference MHA results, the dashed-dotted line represents RBCA with $f^{(\mathrm{PL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{PL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

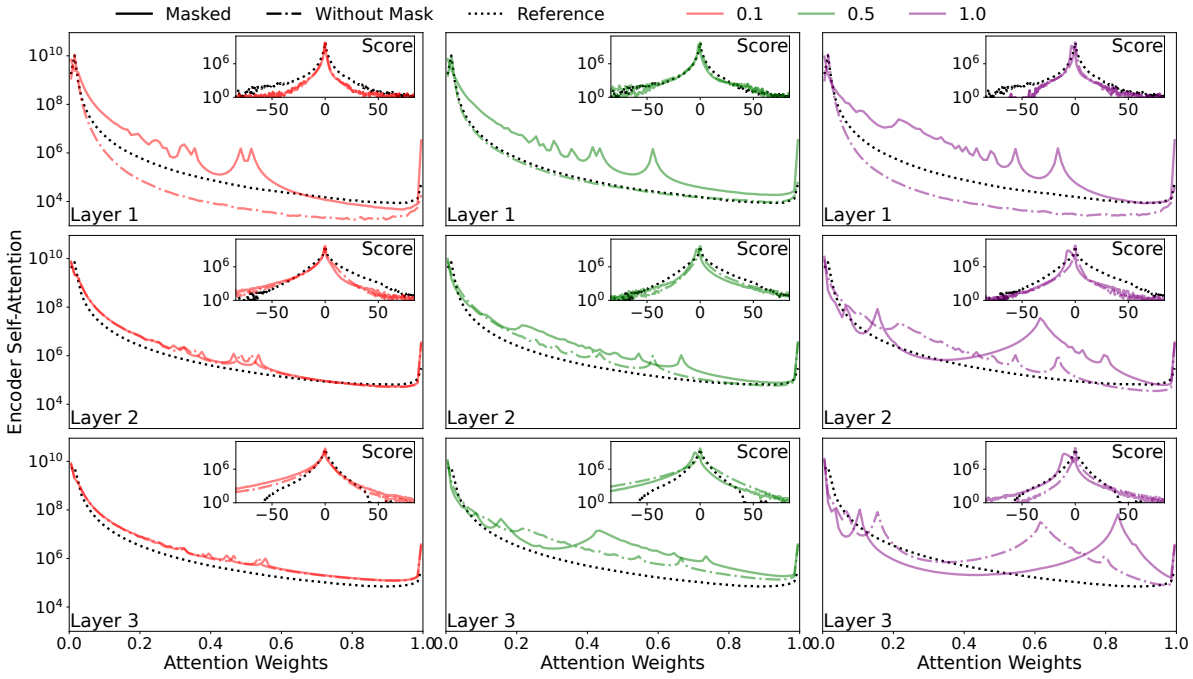

Figure S18: We present the *Powerformer* attention score (inset) and weight distributions on the Weather dataset with a forecast and input length of 720 and 512, respectively. The dotted line represents the reference MHA results, the dashed-dotted line represents RBCA with $f^{(\mathrm{PL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{PL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

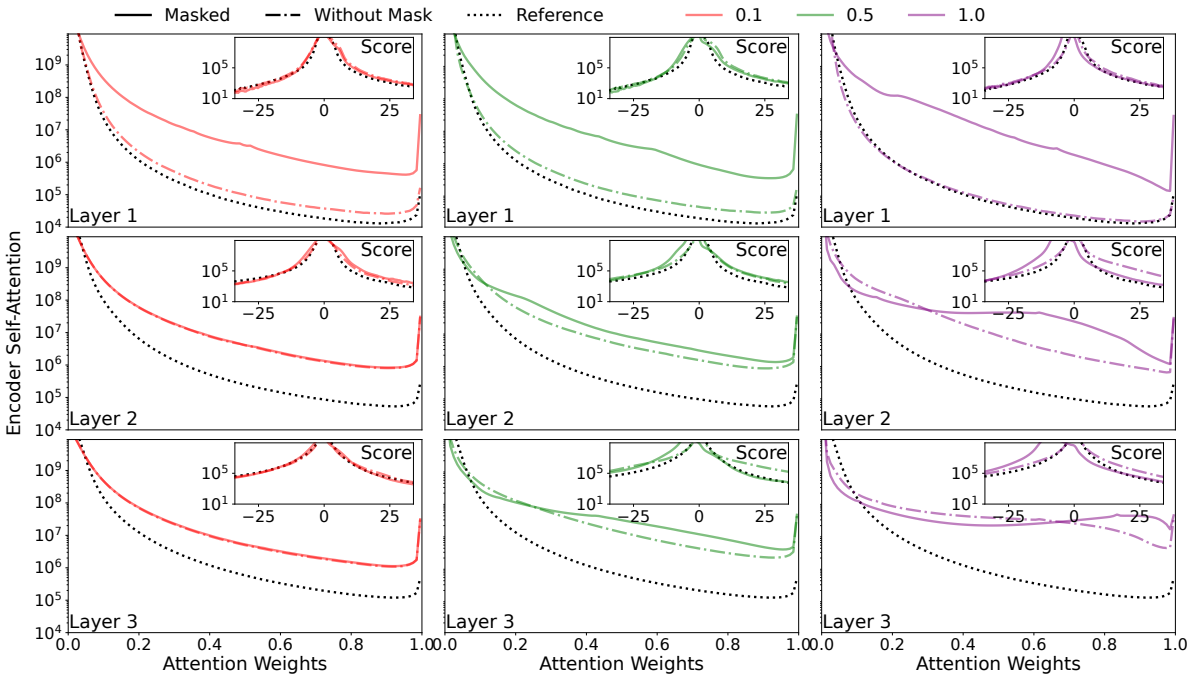

Figure S19: We present the *Powerformer* attention score (inset) and weight distributions on the Electricity dataset with a forecast and input length of 96 and 512, respectively. The dotted line represents the reference MHA results, the dashed-dotted line represents RBCA with $f^{(\mathrm{PL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{PL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

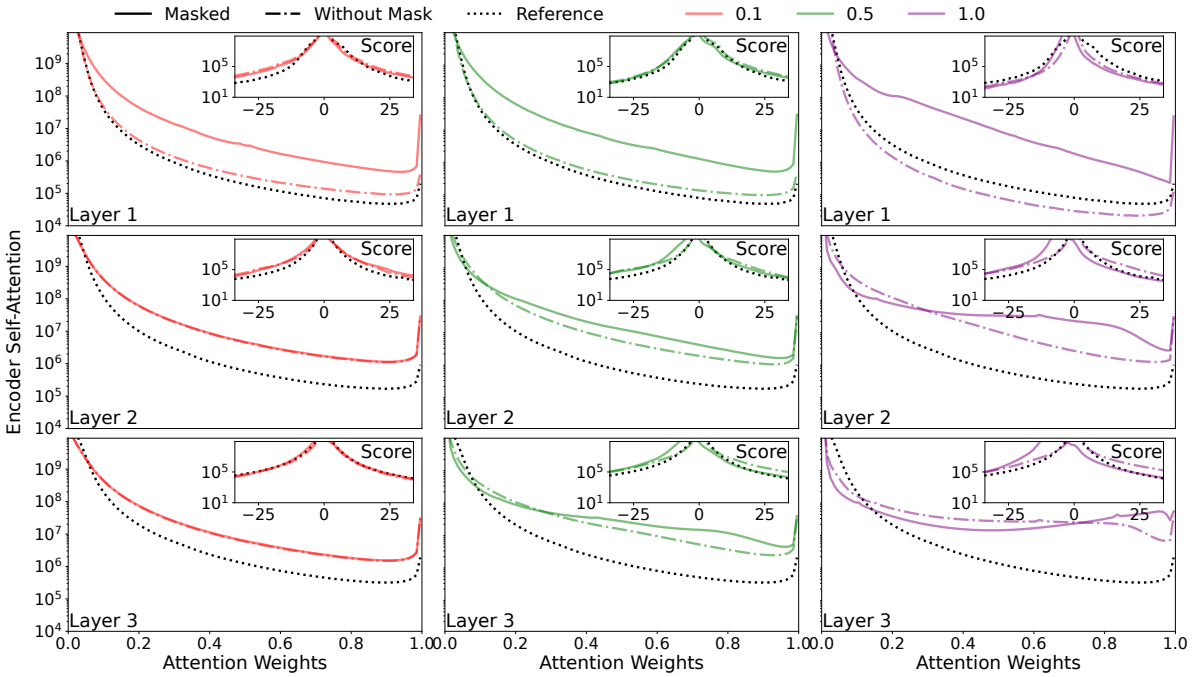

Figure S20: We present the *Powerformer* attention score (inset) and weight distributions on the Electricity dataset with a forecast and input length of 720 and 512, respectively. The dotted line represents the reference MHA results, the dashed-dotted line represents RBCA with $f^{(\mathrm{PL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{PL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

S7.2 Similarity Power-Law Mask

Table S6 presents the aggregate MSE and MAE results for the similarity power-law mask ($f^{(\mathrm{SPL})}(t)$) with varying time decays (α), forecast lengths, and input sequence lengths. The 512 input sequence generally outperforms the shorter 336 input length. Compared to $f^{(\mathrm{PL})}(t)$, there are more instances in which a 336 input sequence length outperforms. This is likely due to the much faster decay of $f^{(\mathrm{SPL})}(t)$ compared to $f^{(\mathrm{PL})}(t)$. As expected, the best performing α varies between datasets. However, there is much less variation as a function of forecast length and input sequence length when compared to $f^{(\mathrm{PL})}(t)$.

We compare all datasets' attention score and weight distributions for recency biased attention in Figs S21-S24. Due to the faster decay of $f^{(\mathrm{SPL})}(t)$ we observe stronger bimodal distributions for all datasets than for $f^{(\mathrm{PL})}(t)$, with wider $\mathbf{C}^{(\mathrm{C,R})}$ distributions for ETTm1, ETTm2, and Weather.

Table S6: We compare *Powerformer's* performance with the similarity power-law mask $f^{(\mathrm{SPL})}(t)$ on standard time-series datasets for varying decay lengths. The best results are bolded and the second best are underlined.

Delay			0.1		0.5		1		2	
Metric			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	336	96	**0.377**	**0.401**	0.378	0.402	0.378	0.402	0.379	0.403
		192	**0.413**	**0.421**	0.414	**0.421**	0.415	0.422	0.415	0.422
		336	**0.424**	**0.429**	0.425	**0.429**	0.425	**0.429**	0.425	**0.429**
		720	**0.439**	**0.457**	0.440	**0.457**	0.443	0.458	0.443	0.458
	512	96	**0.370**	**0.400**	0.371	**0.400**	0.371	**0.400**	0.371	**0.400**
		192	**0.404**	**0.420**	0.405	**0.420**	0.405	0.421	0.405	**0.420**
		336	0.418	0.433	0.417	0.429	**0.415**	**0.428**	**0.415**	**0.428**
		720	**0.439**	**0.459**	0.441	0.460	0.442	0.461	0.442	0.461
ETTh2	336	96	**0.274**	**0.336**	0.275	**0.336**	0.275	**0.336**	0.275	**0.336**
		192	**0.338**	**0.379**	0.340	**0.379**	**0.338**	**0.379**	**0.338**	**0.379**
		336	0.330	**0.384**	0.333	0.385	**0.329**	0.385	**0.329**	0.385
		720	0.380	**0.422**	0.381	**0.422**	**0.379**	**0.422**	**0.379**	**0.422**
	512	96	**0.274**	**0.337**	0.276	0.338	0.275	0.338	0.275	0.338
		192	**0.340**	**0.381**	0.342	0.382	0.341	**0.381**	0.341	**0.381**
		336	0.332	0.388	0.333	0.387	0.331	0.387	**0.330**	**0.386**
		720	0.386	0.428	0.381	**0.424**	0.381	**0.424**	**0.380**	**0.424**
ETTm1	336	96	0.295	0.345	**0.292**	**0.343**	0.293	0.344	0.294	0.345
		192	0.336	0.372	**0.329**	**0.369**	0.333	0.371	0.337	0.373
		336	0.363	0.392	**0.360**	**0.390**	0.363	0.393	0.362	0.392
		720	0.413	**0.423**	0.419	0.425	0.425	0.427	**0.412**	0.424
	512	96	0.291	**0.345**	**0.289**	**0.345**	0.293	0.346	0.291	**0.345**
		192	**0.333**	**0.372**	0.334	0.374	0.334	0.374	**0.333**	0.374
		336	**0.362**	**0.391**	0.365	**0.391**	**0.362**	0.396	**0.362**	0.396
		720	**0.419**	0.427	0.426	0.418	0.425	0.418	0.425	**0.417**
ETTm2	336	96	**0.163**	**0.253**	0.164	0.254	0.165	0.256	0.165	0.254
		192	**0.222**	**0.293**	**0.222**	0.294	0.223	0.294	0.223	0.294
		336	**0.277**	**0.329**	0.278	0.330	0.279	0.330	0.278	**0.329**
		720	0.363	0.381	**0.358**	0.383	**0.358**	**0.380**	0.361	0.384
	512	96	**0.164**	**0.255**	**0.164**	**0.255**	0.165	0.256	0.165	0.256
		192	**0.222**	**0.295**	0.223	0.296	0.224	0.296	0.224	0.296
		336	**0.274**	**0.329**	0.275	**0.329**	0.275	**0.329**	0.275	**0.329**
		720	0.354	**0.379**	0.354	0.380	**0.353**	0.380	**0.353**	**0.379**
Weather	336	96	**0.151**	**0.199**	0.154	0.202	0.154	0.202	0.156	0.204
		192	**0.195**	0.243	0.197	**0.242**	0.198	0.244	0.199	0.244
		336	**0.247**	0.283	0.248	**0.282**	0.249	**0.282**	0.248	0.283
		720	0.318	0.334	**0.317**	**0.332**	0.319	0.335	0.319	0.334
	512	96	**0.149**	**0.199**	0.150	0.200	0.152	0.202	0.154	0.204
		192	**0.192**	**0.240**	0.194	0.242	0.196	0.244	0.198	0.244
		336	**0.243**	**0.280**	0.244	**0.280**	0.246	0.283	0.246	0.281
		720	**0.310**	**0.329**	0.311	0.330	0.312	0.331	0.313	0.331
Electricity	336	96	**0.131**	**0.224**	**0.131**	0.225	**0.131**	0.225	**0.131**	0.225
		192	0.147	**0.240**	0.148	0.242	**0.146**	**0.240**	**0.146**	**0.240**
		336	0.164	**0.258**	0.163	**0.258**	**0.162**	**0.258**	0.163	**0.258**
		720	**0.201**	**0.291**	0.202	0.292	0.202	0.293	0.204	0.295
	512	96	**0.129**	**0.224**	0.130	**0.224**	**0.129**	**0.224**	**0.129**	**0.224**
		192	0.146	**0.240**	0.147	0.241	**0.145**	**0.240**	**0.145**	**0.240**
		336	**0.162**	**0.257**	0.163	0.259	0.163	0.259	0.163	0.259
		720	**0.198**	**0.290**	0.199	0.291	0.205	0.295	0.207	0.296
Traffic	336	96	**0.372**	**0.254**	0.374	0.256	0.375	0.255	0.376	0.255
		192	**0.390**	**0.261**	0.391	0.262	0.391	0.262	0.392	0.262
		336	**0.402**	**0.268**	0.403	0.269	0.403	**0.268**	0.404	0.269
		720	**0.435**	**0.286**	0.436	0.287	**0.435**	0.287	0.436	0.287
	512	96	**0.368**	**0.254**	0.394	0.281	0.394	0.281	0.395	0.282
		192	**0.383**	**0.260**	0.407	0.286	0.407	0.286	0.407	0.287
		336	**0.393**	**0.266**	0.395	**0.266**	0.395	**0.266**	0.395	**0.266**
		720	0.432	0.288	0.433	0.288	0.434	0.288	**0.431**	**0.287**

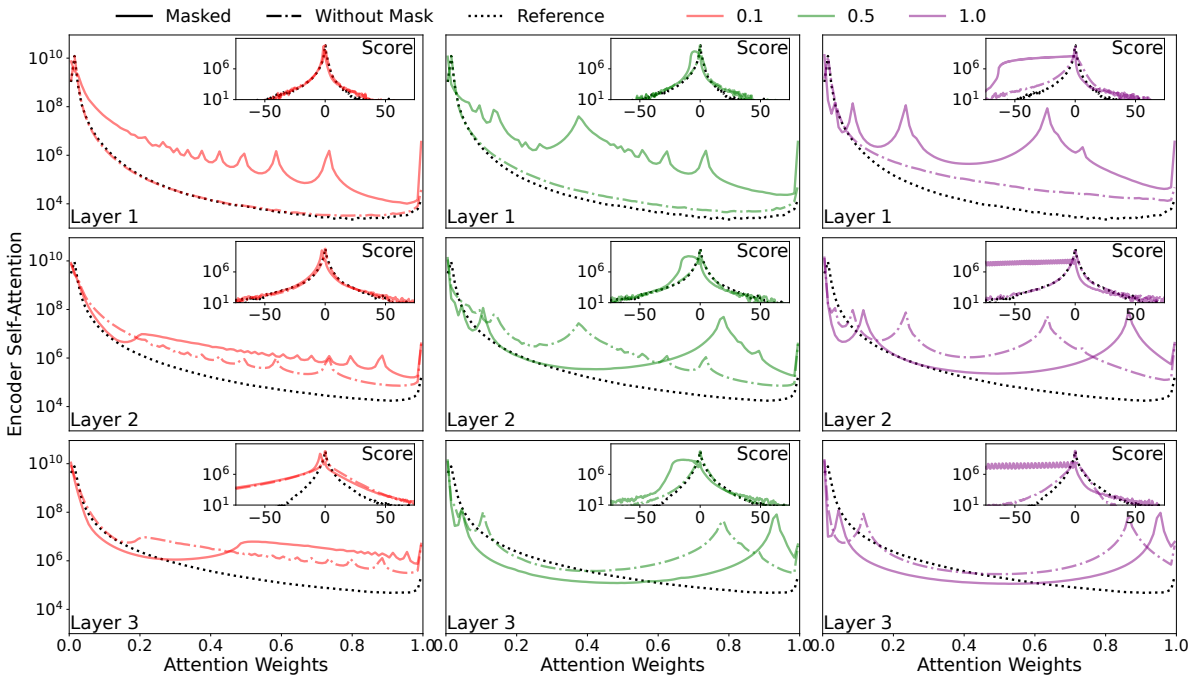

Figure S21: We present *Powerformer's* attention score and weight distributions on the Weather dataset with a forecast and input length of 96 and 512, respectively. The dotted line represents the reference MHA results, the dashed-dotted line represents RBCA with $f^{(\mathrm{SPL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{SPL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

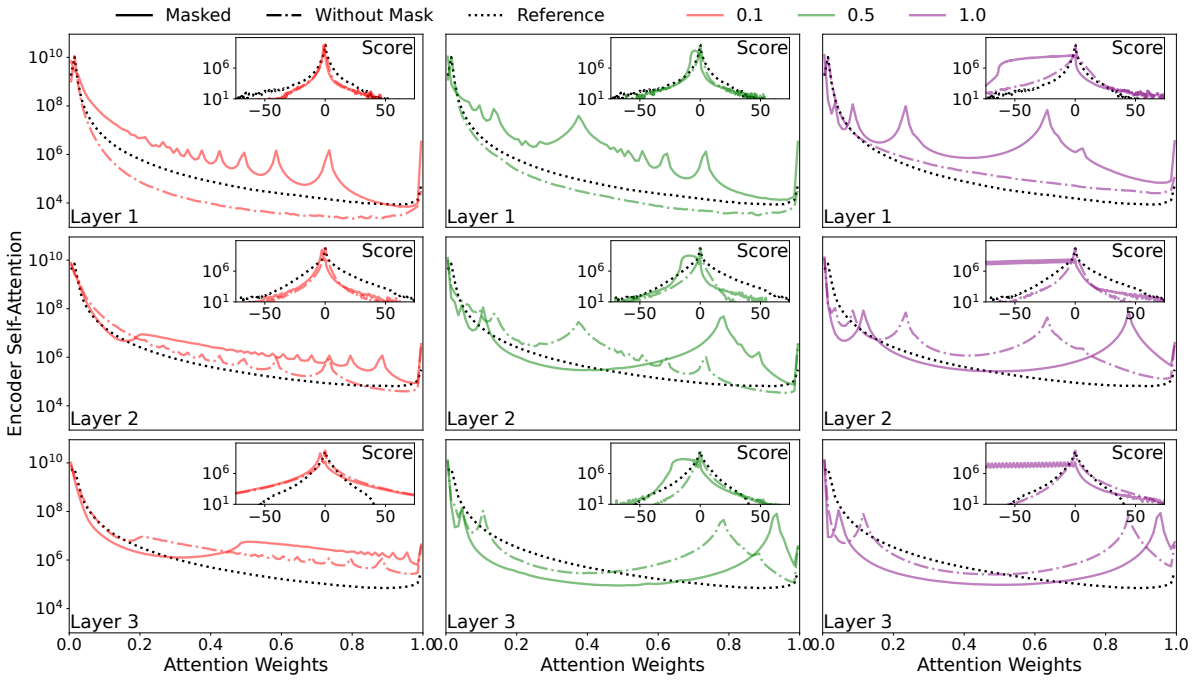

Figure S22: We present *Powerformer's* attention score and weight distributions on the Weather dataset with a forecast and input length of 720 and 512, respectively. The dotted line represents the reference MHA results, the dashed-dotted line represents RBCA with $f^{(\mathrm{SPL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{SPL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

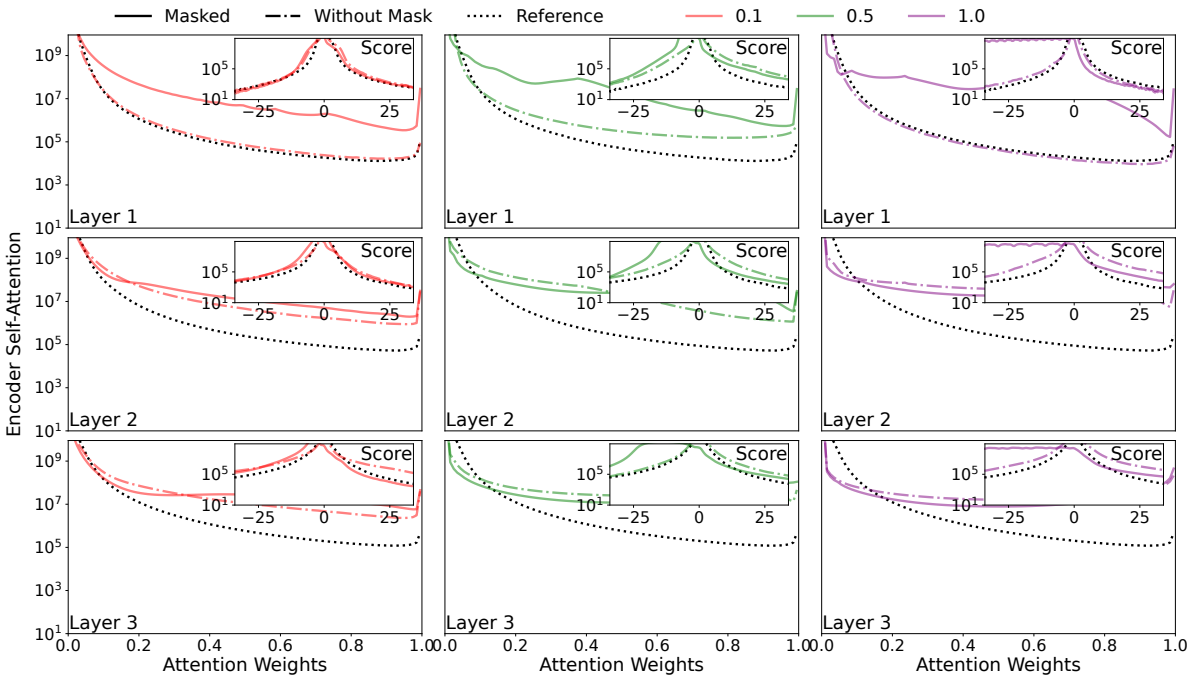

Figure S23: We present *Powerformer's* attention score and weight distributions on the Electricity dataset with a forecast and input length of 96 and 512, respectively. The dotted line represents the reference MHA results, the dashed-dotted line represents RBCA with $f^{(\mathrm{SPL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{SPL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

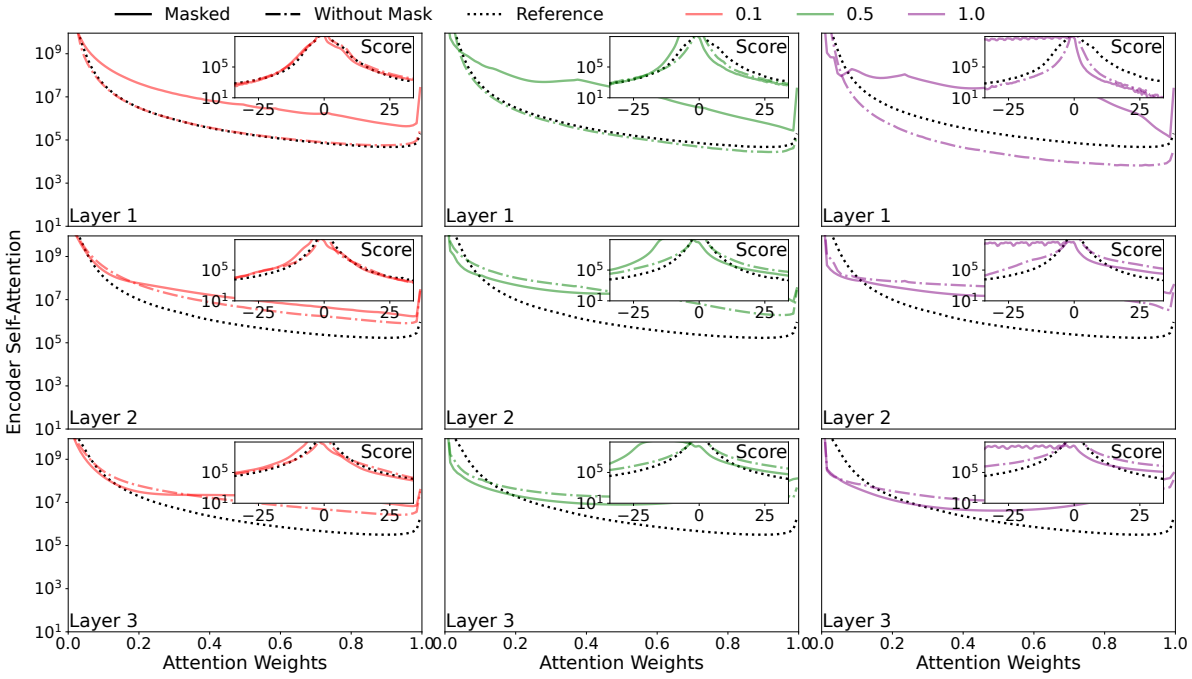

Figure S24: We present *Powerformer's* attention score and weight distributions on the Electricity dataset with a forecast and input length of 720 and 512, respectively. The dotted line represents the reference MHA results, the dashed-dotted line represents RBCA with $f^{(\mathrm{SPL})}(t)$ results before applying $\mathbf{M}^{(\mathrm{C,R})}$, and the solid lines represent RBCA with $f^{(\mathrm{SPL})}(t)$ results after applying $\mathbf{M}^{(\mathrm{C,R})}$.

S7.3 Butterworth Filter: Order 1

Here we evaluate the Butterworth order 1 mask ($f_1^{(\mathrm{BW})}(t)$) on the ETT and Weather datasets with a 336 look-back window. The $f_1^{(\mathrm{BW})}(t)$ decay is not as steep as $f^{(\mathrm{PL})}(t)$ but has a more gradual decay than $f_2^{(\mathrm{BW})}(t)$, as shown in Figs. S4 and 2. This mask offers a middle ground between the step function like $f_2^{(\mathrm{BW})}(t)$ and the faster decaying $f^{(\mathrm{PL})}(t)$. Table S7 presents the results for *Powerformer* with RBCA and $f_1^{(\mathrm{BW})}(t)$. We observe that $f_1^{(\mathrm{BW})}(t)$ underperforms $f^{(\mathrm{PL})}(t)$, $f^{(\mathrm{SPL})}(t)$ and $f_2^{(\mathrm{BW})}(t)$. The poor performance further supports our hypothesis that the functional form of $f(t)$ is important when imposing a locality bias.

Table S7: We compare *Powerformer's* performance with RBCA and the order 1 Butterworth Filter mask $f_1^{(\mathrm{BW})}(t)$ on standard time-series datasets for varying decay lengths. The best results are bolded and second best are underlined.

Delay		2		5		10		15		20	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.378	0.402	**0.377**	0.402	**0.377**	0.402	**0.377**	**0.401**	**0.377**	**0.401**
	192	0.415	0.422	**0.414**	**0.421**	**0.414**	**0.421**	**0.414**	**0.421**	**0.414**	**0.421**
	336	0.425	**0.429**	0.425	**0.429**	0.425	**0.429**	0.425	**0.429**	**0.424**	**0.429**
	720	0.442	0.459	**0.439**	0.458	**0.439**	**0.457**	0.440	0.458	**0.439**	0.458
ETTh2	96	0.275	**0.336**	0.275	**0.336**	**0.274**	**0.336**	**0.274**	**0.336**	**0.274**	**0.336**
	192	**0.338**	**0.379**	**0.338**	**0.379**	0.339	**0.379**	0.339	**0.379**	0.339	**0.379**
	336	0.331	0.386	0.331	0.386	0.331	**0.385**	0.331	**0.385**	**0.330**	**0.385**
	720	**0.380**	0.422	**0.380**	**0.421**	0.382	0.422	0.382	0.424	**0.380**	0.422
ETTm1	96	**0.291**	**0.343**	0.293	**0.343**	0.296	**0.343**	0.295	0.345	0.296	0.346
	192	**0.329**	**0.369**	0.333	0.371	0.337	0.372	0.338	0.373	0.336	0.373
	336	0.360	**0.389**	**0.359**	0.390	0.369	0.396	0.363	0.393	0.365	0.393
	720	**0.414**	0.426	0.425	0.429	0.418	0.425	0.417	**0.424**	0.416	0.425
ETTm2	96	0.164	**0.254**	**0.163**	**0.254**	0.165	0.255	0.166	0.256	0.166	0.255
	192	0.223	0.294	0.223	0.294	**0.222**	**0.293**	**0.222**	**0.293**	**0.222**	**0.293**
	336	0.279	0.330	0.278	0.330	**0.275**	**0.327**	0.277	0.331	0.277	0.329
	720	0.358	0.381	0.353	0.381	**0.351**	0.381	0.359	**0.380**	0.359	**0.380**
Weather	96	0.155	0.204	0.154	0.203	0.153	**0.202**	0.153	0.203	**0.152**	**0.202**
	192	0.199	**0.244**	0.198	**0.244**	0.198	**0.244**	0.198	0.245	**0.197**	0.245
	336	0.250	0.283	**0.248**	**0.282**	0.250	0.284	**0.248**	0.283	**0.248**	0.283
	720	0.320	0.334	**0.319**	**0.333**	0.325	0.338	0.322	0.336	**0.319**	0.335

S7.4 Butterworth Filter: Order 2

Here we evaluate the Butterworth order 2 mask ($f_2^{(\mathrm{BW})}(t)$) on the ETT and Weather datasets with a 336 look-back window. The $f_2^{(\mathrm{BW})}(t)$ decay is analogous to the step function, but has a smooth decay to 0 at the end of the window, as shown in Figs. S4. We use $f_2^{(\mathrm{BW})}(t)$ as an improved step function to avoid the sharp cutoff. Table S8 presents the results for *Powerformer* with RBCA and $f_2^{(\mathrm{BW})}(t)$. We observe that $f_2^{(\mathrm{BW})}(t)$ underperforms $f^{(\mathrm{PL})}(t)$ and $f^{(\mathrm{SPL})}(t)$, but outperforms $f_1^{(\mathrm{BW})}(t)$. The poor performance further supports our hypothesis that the functional form of $f(t)$ is important when imposing a locality bias.

Table S8: We compare *Powerformer's* performance with RBCA and the Butterworth filter mask $f_2^{(\mathrm{BW})}(t)$ on standard time-series datasets for varying decay lengths. The best results are bolded and second best are underlined.

Delay		2		5		10		15		20	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.378	0.402	**0.377**	0.402	**0.377**	**0.401**	**0.377**	**0.401**	**0.377**	**0.401**
	192	**0.414**	0.422	**0.414**	**0.421**	**0.414**	**0.421**	**0.414**	**0.421**	**0.414**	**0.421**
	336	0.425	**0.429**	0.425	**0.429**	0.425	**0.429**	**0.424**	**0.429**	**0.424**	**0.429**
	720	0.442	0.459	**0.439**	**0.457**	**0.439**	**0.457**	**0.439**	0.458	**0.439**	0.458
ETTh2	96	0.275	**0.336**	0.275	**0.336**	**0.274**	**0.336**	**0.274**	**0.336**	**0.274**	**0.336**
	192	**0.338**	**0.379**	**0.338**	**0.379**	**0.338**	**0.379**	0.339	**0.379**	0.339	**0.379**
	336	0.331	0.386	0.331	0.386	0.331	0.385	**0.330**	0.385	**0.330**	**0.384**
	720	**0.380**	0.422	**0.380**	**0.421**	0.381	**0.421**	0.381	0.423	**0.380**	0.422
ETTm1	96	**0.290**	**0.343**	0.293	0.344	0.295	0.345	0.295	0.346	0.297	0.344
	192	**0.329**	**0.370**	0.333	0.371	0.339	0.373	0.337	0.373	0.336	0.373
	336	**0.361**	0.391	**0.361**	0.391	0.362	0.393	0.365	0.395	0.363	**0.390**
	720	**0.413**	0.425	0.422	0.428	0.415	0.423	0.417	0.422	**0.413**	**0.421**
ETTm2	96	0.164	0.254	0.164	0.254	0.165	**0.253**	**0.163**	**0.253**	0.165	0.255
	192	0.223	0.294	0.223	0.294	**0.222**	**0.293**	**0.222**	**0.293**	**0.222**	**0.293**
	336	0.279	0.330	0.278	0.330	**0.276**	0.331	**0.276**	**0.329**	0.277	**0.329**
	720	0.358	0.381	0.354	0.381	**0.353**	0.382	0.359	**0.380**	0.360	**0.380**
Weather	96	0.156	0.204	0.154	0.203	0.154	0.203	0.153	**0.202**	**0.152**	0.203
	192	0.199	**0.244**	0.198	**0.244**	0.198	**0.244**	0.197	**0.244**	**0.196**	**0.244**
	336	0.250	0.283	**0.248**	**0.282**	0.249	0.284	**0.248**	0.283	**0.248**	0.284
	720	0.320	**0.334**	**0.318**	**0.334**	0.326	0.339	0.321	0.336	0.319	0.335

S8 ABLATION STUDIES

Here, we perform ablation experiments on *Powerformer*. To separate the effects of the causal mask and the recency bias, we train three *Powerformer* variants: with MHA, with the causal mask only, and with RBCA. We train and evaluate these variants on the ETT datasets and the Weather dataset for the 96 and 336 prediction lengths. Table S9 shows the corresponding results.

For both models, we examine the effects of different masks ($f^{(\mathrm{PL})}(t)$, $f^{(\mathrm{SPL})}(t)$, and $f_n^{(\mathrm{BW})}(t)$) and various time decays α. We do not perform the same amount of experiments for each model and mask pairing due to poor performance. In the remainder of this section, we provide detailed tables of the MSE and MAE scores, as well as plots of the attention scores and weights. In the last subsection (S8.1), we investigate the effects of making α a learnable parameter.

Table S9: Ablation results using vanilla non-causal MHA, causal MHA, and RBCA.

		MHA		Causal MHA		RBCA	
Metric		MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.370	0.400	**0.361**	**0.390**	**0.361**	**0.390**
	336	0.422	0.440	0.413	0.429	**0.406**	**0.420**
ETTh2	96	0.274	0.336	0.270	**0.334**	**0.269**	**0.334**
	336	0.331	**0.380**	**0.323**	**0.380**	**0.323**	**0.380**
ETTm1	96	0.290	**0.342**	0.298	0.344	**0.285**	**0.342**
	336	0.366	0.392	0.370	**0.384**	**0.357**	0.385
ETTm2	96	0.165	0.255	**0.163**	0.253	**0.163**	**0.251**
	336	0.278	0.329	**0.272**	**0.324**	0.273	0.326
Weather	96	0.149	0.198	**0.147**	**0.197**	**0.147**	**0.197**
	336	0.245	0.282	**0.243**	0.281	**0.243**	**0.279**

S8.1 Learnable Mask Time Decays

Although we treat the decay time α as a hyperparameter, one may consider learning the optimal decay time by making α learnable. We observe that α monotonically decreases and in some cases flips sign, changing our mask from a power-law decay to a power-law growth. To combat this continuous decrease we applied a learning rate scheduler to limit how far α can change from its initialization. To fairly compare with our hyperparameter results, we initialize α at the same times used in our hyperparameter tuning. We present the aggregate MSE and MAE results in Table S10.

We observe very minor differences between the best hyperparameter results and the best results from learning α. During training, we still observe a monotonically decreasing α that slows down only because of the learning rate scheduler. This makes sense since the training algorithm gains access to more information as α decreases and facilitates overfitting. This behavior further supports our claim that $\mathbf{M}^{(\mathrm{C,R})}$ and α act as regularizers.

Table S10: We train *Powerformer* with $f^{(\mathrm{PL})}(t)$ and learnable time constant and present the results here. The base cases are the best results from *Powerformer* with constant decay weights: Table S5. The remaining columns designate the initialized time decay.

Delay			Base Case		0.1		0.25		0.5		0.75		1	
Metric			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	336	96	**0.376**	**0.401**	0.377	**0.401**	0.377	**0.401**	0.377	**0.401**	**0.376**	**0.401**	**0.376**	**0.401**
		192	**0.413**	**0.421**	**0.413**	**0.421**	**0.413**	**0.421**	**0.413**	**0.421**	**0.413**	**0.421**	**0.413**	**0.421**
		336	**0.424**	**0.430**	0.425	**0.430**	0.425	**0.430**	0.425	**0.430**	0.425	**0.430**	0.425	**0.430**
		720	**0.437**	**0.455**	**0.437**	**0.455**	**0.437**	**0.455**	**0.437**	**0.455**	**0.437**	0.456	0.438	0.456
	512	96	**0.369**	**0.399**	**0.369**	**0.399**	**0.369**	**0.399**	0.370	**0.399**	0.370	**0.399**	0.370	**0.399**
		192	**0.402**	**0.418**	0.404	0.420	0.404	0.420	**0.402**	**0.418**	0.403	**0.418**	0.403	**0.418**
		336	**0.414**	**0.428**	**0.414**	**0.428**	**0.414**	**0.428**	0.415	0.429	0.415	0.430	0.416	0.431
		720	**0.440**	**0.460**	**0.440**	**0.460**	**0.440**	**0.460**	**0.440**	**0.460**	0.443	0.462	0.443	**0.460**
ETTh2	336	96	**0.274**	**0.335**	**0.274**	**0.335**	**0.274**	**0.335**	0.275	0.336	0.275	**0.335**	0.275	**0.335**
		192	**0.338**	**0.379**	0.341	0.381	**0.338**	**0.379**	0.340	**0.379**	0.340	**0.379**	0.341	**0.379**
		336	**0.325**	**0.379**	**0.325**	**0.379**	0.326	0.380	0.331	0.384	0.333	0.383	0.334	0.383
		720	**0.376**	**0.419**	0.377	0.420	0.379	0.421	0.381	0.423	0.381	0.422	0.381	0.422
	512	96	**0.274**	**0.337**	**0.274**	**0.337**	**0.274**	**0.337**	**0.274**	**0.337**	0.275	**0.337**	0.275	0.338
		192	**0.340**	**0.381**	**0.340**	**0.381**	**0.340**	**0.381**	0.341	0.382	0.342	0.382	0.342	0.382
		336	**0.330**	**0.387**	**0.330**	**0.387**	0.331	**0.387**	0.333	0.388	0.334	0.388	0.334	0.388
		720	**0.383**	**0.425**	**0.383**	0.426	0.384	0.426	**0.383**	0.426	0.384	0.426	**0.383**	**0.425**
ETTm1	336	96	**0.290**	**0.342**	0.292	0.343	**0.290**	**0.342**	0.292	0.343	0.293	0.344	0.292	0.345
		192	**0.330**	**0.369**	0.331	0.371	0.334	0.372	0.332	0.371	0.332	0.370	**0.330**	**0.369**
		336	**0.359**	**0.389**	0.362	0.392	0.361	0.391	0.362	0.390	**0.359**	**0.389**	0.361	0.390
		720	**0.412**	**0.421**	0.413	0.423	0.413	0.422	0.413	**0.421**	0.413	**0.421**	**0.412**	0.423
	512	96	**0.290**	**0.345**	0.291	**0.345**	**0.290**	**0.345**	**0.290**	**0.345**	0.291	0.346	0.292	0.346
		192	**0.329**	**0.368**	0.331	0.370	0.330	0.370	**0.329**	**0.368**	0.332	0.370	0.332	0.372
		336	**0.361**	**0.390**	0.362	0.391	0.362	**0.390**	**0.361**	0.392	0.362	**0.390**	0.363	**0.390**
		720	**0.415**	**0.423**	0.417	0.428	**0.415**	0.425	0.417	**0.423**	0.420	0.426	0.416	0.431
ETTm2	336	96	**0.162**	**0.252**	**0.162**	**0.252**	0.165	0.255	0.166	0.256	0.164	0.254	0.164	0.254
		192	**0.222**	**0.292**	**0.222**	**0.292**	**0.222**	0.293	**0.222**	0.293	**0.222**	0.293	**0.222**	0.293
		336	**0.277**	**0.329**	**0.277**	**0.329**	**0.277**	**0.329**	**0.277**	**0.329**	**0.277**	**0.329**	0.278	0.330
		720	**0.359**	**0.380**	0.363	0.381	0.363	0.381	0.362	0.382	**0.359**	**0.380**	**0.359**	0.384
	512	96	**0.164**	**0.255**	0.165	**0.255**	0.165	**0.255**	0.165	0.256	0.165	0.257	**0.164**	**0.255**
		192	**0.221**	**0.294**	0.222	0.295	0.224	0.297	0.224	0.296	0.223	0.296	**0.221**	**0.294**
		336	**0.272**	**0.327**	0.274	0.329	0.274	0.329	0.274	0.329	**0.272**	**0.327**	0.273	**0.327**
		720	**0.356**	**0.381**	0.358	0.383	0.358	0.383	0.358	0.382	0.358	0.383	**0.356**	**0.381**
Weather	336	96	**0.151**	**0.200**	**0.151**	0.201	**0.151**	**0.200**	**0.151**	**0.200**	0.153	0.201	0.154	0.202
		192	0.196	**0.241**	0.196	0.244	0.196	0.243	**0.195**	0.242	0.196	**0.241**	0.197	0.242
		336	0.247	**0.281**	0.248	0.283	0.247	0.284	0.247	0.282	**0.246**	**0.281**	0.248	0.282
		720	**0.317**	0.333	0.318	0.334	0.318	0.334	0.318	0.334	0.318	0.333	**0.317**	**0.332**
	512	96	**0.147**	**0.197**	0.148	0.198	0.148	0.199	**0.147**	0.198	0.148	0.198	0.149	0.199
		192	**0.191**	**0.239**	0.193	0.241	0.194	0.242	0.193	0.240	**0.191**	**0.239**	0.193	0.241
		336	**0.243**	**0.279**	0.244	0.281	**0.243**	0.281	**0.243**	0.280	**0.243**	**0.279**	0.244	0.280
		720	**0.311**	0.329	**0.311**	0.330	**0.311**	0.330	0.314	0.331	**0.311**	**0.328**	**0.311**	0.329
Electricity	336	96	**0.130**	**0.224**	**0.130**	**0.224**	0.131	**0.224**	0.131	**0.224**	0.131	**0.224**	0.131	**0.224**
		192	**0.147**	**0.240**	0.148	0.241	**0.147**	**0.240**	**0.147**	**0.240**	0.148	**0.240**	**0.147**	0.241
		336	**0.164**	**0.257**	**0.164**	0.258	**0.164**	0.258	**0.164**	0.258	**0.164**	0.259	**0.164**	0.258
		720	0.201	**0.291**	0.201	0.292	**0.200**	**0.291**	0.201	0.292	0.202	0.292	0.202	0.293
	512	96	**0.129**	**0.223**	0.130	**0.223**	**0.129**	0.224	**0.129**	0.224	**0.129**	**0.223**	0.130	**0.223**
		192	**0.146**	**0.240**	0.147	0.241	**0.146**	**0.240**	**0.146**	**0.240**	**0.146**	**0.240**	0.147	0.241
		336	**0.162**	**0.257**	**0.162**	0.258	**0.162**	0.258	**0.162**	0.258	**0.162**	**0.257**	**0.162**	0.258
		720	**0.198**	**0.289**	0.199	0.290	**0.198**	0.290	0.199	0.290	0.200	0.291	0.200	0.291
Traffic	336	96	**0.370**	**0.253**	**0.370**	**0.253**	**0.370**	**0.253**	0.372	0.254	0.373	0.255	0.374	0.256
		192	**0.388**	**0.260**	**0.388**	0.261	**0.388**	**0.260**	0.390	0.261	0.390	0.262	0.390	0.262
		336	**0.400**	**0.266**	**0.400**	0.267	**0.400**	**0.266**	0.404	0.269	0.404	0.270	0.405	0.270
		720	0.434	**0.286**	0.435	**0.286**	**0.433**	**0.286**	0.435	0.287	0.437	0.289	0.436	0.287
	512	96	**0.365**	0.252	**0.365**	**0.251**	0.366	0.252	0.366	0.253	0.367	0.254	0.394	0.282
		192	**0.382**	**0.258**	0.390	0.269	0.383	0.261	0.383	0.261	**0.382**	0.260	0.407	0.287
		336	**0.391**	0.265	**0.391**	**0.264**	0.392	0.265	0.392	0.265	0.393	0.265	0.394	0.265
		720	**0.430**	0.286	0.433	0.288	0.431	0.288	0.432	0.287	**0.430**	**0.285**	**0.430**	0.286