machine learning & knowledge extraction

MDPI

Article

WinStat: A Family of Trainable Positional Encodings for Transformers in Time Series Forecasting

Cristhian Moya-Mota [1], Ignacio Aguilera-Martos [1,2], Diego García-Gil [1,3] and Julián Luengo [1,2,*]

[1] Andalusian Institute of Data Science and Computational Intelligence (DaSCI), University of Granada, 18012 Granada, Spain; cris190402@correo.ugr.es (C.M.-M.); nacheteam@ugr.es (I.A.-M.); djgarcia@ugr.es (D.G.-G.)
[2] Department of Computer Science and Artificial Intelligence, University of Granada, 18012 Granada, Spain
[3] Department of Software Engineering, University of Granada, 18012 Granada, Spain
* Correspondence: julianlm@decsai.ugr.es

Abstract

Transformers for time series forecasting rely on positional encoding to inject temporal order into the permutation-invariant self-attention mechanism. Classical sinusoidal absolute encodings are fixed and purely geometric; learnable absolute encodings often overfit and fail to extrapolate, while relative or advanced schemes can impose substantial computational overhead without being sufficiently tailored to temporal data. This work introduces a family of window-statistics positional encodings that explicitly incorporate local temporal semantics into the representation of each timestamp. The base variant (WinStat) augments inputs with statistics computed over a sliding window; WinStatLag adds explicit lag-difference features; and hybrid variants (WinStatFlex, WinStatTPE, WinStatSPE) learn soft mixtures of window statistics with absolute, learnable, and semantic positional signals, preserving the simplicity of additive encodings while adapting to local structure and informative lags. We evaluate proposed encodings on four heterogeneous benchmarks against state-of-the-art proposals: Electricity Transformer Temperature (hourly variants), Individual Household Electric Power Consumption, New York City Yellow Taxi Trip Records, and a large-scale industrial time series from heavy machinery. All experiments use a controlled Transformer backbone with full self-attention to isolate the effect of positional information. Across datasets, the proposed methods consistently reduce mean squared error and mean absolute error relative to a strong Transformer baseline with sinusoidal positional encoding and state-of-the-art encodings for time series, with WinStatFlex and WinStatTPE emerging as the most effective variants. Ablation studies that randomly shuffle decoder inputs markedly degrade the proposed methods, supporting the conclusion that their gains arise from learned order-aware locality and semantic structure rather than incidental artifacts. A simple and reproducible heuristic for setting the sliding-window length—roughly one quarter to one third of the input sequence length—provides robust performance without the need for exhaustive tuning.

Keywords: time series; transformer; positional encoding; embedding; local context

Academic Editor: Mehmed Kantardzic

Received: 9 October 2025
Revised: 5 December 2025
Accepted: 25 December 2025
Published: 29 December 2025

1. Introduction

The proliferation of sensors, Internet of Things (IoT) devices, and large-scale digital infrastructures has exponentially increased the availability of multivariate and high-frequency time series, thereby broadening the potential and the challenges of Time Series Forecasting (TSF). TSF has become a cornerstone of contemporary artificial intelligence research, with a wide spectrum of applications in domains such as energy demand management [1], climate

modeling [2], financial prediction [3], healthcare monitoring [4], and intelligent transportation systems [5] of TSF [6–8]. Consequently, the field is evolving beyond classical statistical techniques like AutoRegressive Integrated Moving Average (ARIMA) [9], favoring machine learning and deep learning architectures that excel at modeling nonlinear dependencies and complex temporal dynamics [7,8].

TSF methods have evolved over the past several decades from classical statistical models to machine learning and, more recently, deep learning approaches [10]. Statistical methods, such as ARIMA, modeled linear temporal dynamics under stationarity assumptions, while subsequent machine learning techniques (e.g., gradient boosting trees) enabled more flexible nonlinear pattern modeling. The advent of deep learning introduced sequence-based models, notably recurrent neural networks (RNNs) and long short-term memory (LSTM) networks. These kinds of methods equipped TSF with long-term memory that can be learned, while convolutional neural networks (CNNs) captured local temporal patterns. However, CNNs cannot effectively capture global or long-range temporal dependencies, and RNN/LSTM-based models often struggle to model very long sequences due to vanishing gradients and computational constraints. To address these limitations, attention-based Transformer models have emerged as a powerful alternative, leveraging self-attention mechanisms to model long-term dependencies across time steps [11].

Transformers have achieved state-of-the-art performance in TSF, particularly for long-horizon forecasting tasks, by capturing global temporal relationships that eluded earlier methods. Unlike RNNs, the Transformer's self-attention is permutation-invariant and does not inherently encode temporal order, so Transformer-based TSF models rely heavily on positional encoding schemes to inject sequence ordering information. Recent research underscores that the design of these positional encoding strategies is both crucial and challenging for time series data [12]. Likewise, the authors in [7] observed that the efficacy of different positional encoding methods in time series remains an open question, with debates over the best approaches (absolute vs. relative encoding). Thus, despite the success of Transformer models in TSF due to their ability to model long-term dependencies, they introduce new challenges stemming from their heavy reliance on positional encoding strategies for temporal-order modeling.

Numerous efforts have been devoted to refining and enhancing positional encoding to preserve the sequential coherence inherent in time series data. Several variants have been proposed to improve adaptability to the data and strengthen locality and semantic characteristics. For example, time Absolute Positional Encoding (tAPE) [7] adapts the encoding frequencies to the length of the input sequence to preserve distance-awareness, while Relative Positional Encoding (RPE) [13] seeks to capture correspondences between different points in the series. Nonetheless, these methods remain heavily dependent on a static geometric component. To overcome this limitation, alternatives such as Temporal Positional Encoding (TPE) [8] —which introduces a semantic dimension—or learnable positional encodings (LPEs) [14] have been explored. Although these methods promise performance gains, their outcomes are often unstable due to increased computational demands, difficulties in constraining values within a normalized range, and sensitivity to initialization.

In light of these issues, the development of novel positional encoding mechanisms specifically designed for Time Series Forecasting (TSF) remains a critical open problem. This paper aims to address this gap, driven by a twofold motivation: first, to overcome the limitations of fixed encodings, which often fail to capture the dynamic and nonlinear temporal relationships inherent in multivariate and high-frequency series; and second, to enable Transformer-based models to learn the best positional representation during training for the given task. Representing a novel contribution unrelated to traditional fixed baselines, the WinStat family is designed to adjust to the underlying temporal structure

of each dataset. This approach aims to improve the model's capacity to capture both local and long-range dependencies, thereby improving forecasting accuracy and robustness. The effectiveness of these learnable positional encodings will be tackled through comprehensive experimental evaluations on widely used TSF benchmarks, comparing their performance with established absolute and relative encoding strategies.

These experiments demonstrate that, across diverse forecasting benchmarks—from household power consumption and transformer load monitoring to urban taxi demand and large-scale industrial sensor streams—the proposed WinStat positional encodings consistently outperform both classic sinusoidal and purely learnable schemes. In particular, WinStatFlex and WinStatTPE yield average reductions in mean squared error of 10–30% and exhibit up to 90% improvement in high-precision settings relative to a strong Informer baseline. Notably, in certain cases, the WinStat base variant also surpasses the baseline, emerging as a competitive third alternative within the family. Moreover, ablation analyses with decoder-input shuffling confirm that these gains arise from meaningful order-aware locality and semantic context rather than incidental artifacts. Such results validate our hypothesis that integrating local window statistics and adaptive positional signals offers a robust and effective approach to TSF with Transformers.

The rest of this paper is organized as follows. Section 2 reviews the evolution of time series forecasting methods and the role of positional encodings in Transformer architectures. Section 3 introduces our WinStat family of window-statistics-based positional encodings, detailing the base WinStat model and its Lag, Flex, TPE, and SPE variants. Section 4 describes the datasets, model configuration, data preprocessing, and evaluation protocols used to ensure fair comparison across encoding schemes. Section 5 presents a comprehensive experimental evaluation using four diverse forecasting benchmark datasets. Section 5.3 includes ablation studies with shuffled inputs to validate the semantic contribution of each encoding. Finally, Section 6 offers concluding remarks, highlights the key findings, and outlines promising directions for future research.

2. Background

In this section, we present an overview of the methodological advances that have shaped the field, tracing the progression from classical statistical models and machine learning approaches to the recent dominance of deep learning and Transformer-based architectures. We focus particularly on the role of positional encoding mechanisms, which are crucial for adapting permutation-invariant self-attention to the inherently ordered nature of time series, setting the stage for our proposed enhancements in later sections.

2.1. Time Series Forecasting

TSF has long been a fundamental problem in statistics, econometrics, and, more recently, artificial intelligence. Its goal is to estimate the future evolution of a temporal signal based on historical data, enabling informed decision-making in various domains, including energy demand management, finance, traffic prediction, healthcare, and stock price and crash prediction [15,16]. Classical approaches, such as ARIMA [9] and exponential smoothing methods such as Prophet [17], are grounded in linear modeling assumptions and stationarity. These models provide interpretable components—trend, seasonality, and residuals—and remain valuable baselines. However, they often fail in the presence of strong nonlinearity, abrupt changes, and complex multivariate dependencies.

With the increasing availability of high-frequency and multivariate data, machine learning approaches such as support vector regression and gradient boosting machines [18] gained popularity due to their ability to approximate nonlinear functions without strict assumptions. These models improved robustness across heterogeneous datasets, but are

limited by their reliance on manually designed features and their difficulty in capturing sequential dependencies across long time horizons.

The deep learning era brought about a paradigm shift in TSF. Sequence models such as RNNs and their gated variants—LSTMs and Gated Recurring Units (GRUs)—became the dominant architectures for sequential data [19]. These models utilize recurrent connections to maintain a hidden state over time, allowing for the modeling of long-term dependencies. Concurrently, CNNs demonstrated remarkable efficacy in processing temporal data, exploiting their inductive bias to capture local patterns and hierarchical representations [20]. Despite these advantages, such architectures present inherent limitations. CNNs are restricted by fixed receptive fields, whereas RNNs suffer from the vanishing gradient problem and scalability issues regarding extended sequences. These deficiencies [21] become particularly pronounced in long-term forecasting scenarios, where error propagation and restricted temporal context significantly compromise predictive accuracy.

2.2. Transformers for TSF

Transformers, introduced by Vaswani et al. [22], represent a breakthrough in sequence modeling by replacing recurrence and convolutions with the self-attention mechanism. This design enables parallelization during training, efficient handling of long dependencies, and the ability to model pairwise interactions between all elements in a sequence. The success of Transformers in natural language processing and computer vision has inspired their adaptation to TSF, where capturing both short- and long-range temporal dependencies is crucial.

Transformer-based TSF varies its approach from sparsity-driven efficiency to structure and variable-aware designs. LogTrans [23] first relaxed the quadratic self-attention bottleneck by introducing LogSparse patterns, with $\Theta(L(\log L)^2)$ memory efficiency, together with convolutional self-attention to inject locality. Informer [24] then proposed ProbSparse attention to reach $\Theta(L \log L)$ time and memory complexity, added self-attention distilling to handle extreme input lengths, and used a generative-style decoder to predict the full horizon in one forward pass. Moving beyond sparsity, Autoformer [25] embedded seasonal-trend decomposition and replaced point-wise attention with an auto-correlation mechanism (computed via Fast Fourier Transform) to capture period-based dependencies with $\Theta(L \log L)$ complexity. In parallel, Pyraformer [26] adopted a pyramidal multiresolution attention graph that shortens the maximum signal path to $\Theta(1)$ complexity while keeping the time and memory complexity of $\Theta(L)$. Building on decomposition-based designs, FedFormer [27] brought frequency-domain blocks (Fourier and Wavelet) and mixture-of-experts decomposition, achieving linear complexity by selecting a fixed number of frequency modes. Triformer [28] targeted multivariate settings with a triangular, patch-based attention of linear complexity and variable-specific parametrization, improving the modeling of important variable dynamics over long horizons. More recently, Ister [29] advanced decomposition-based approaches by inverting the seasonal trend framework and employing a dual transformer with dot-attention, improving interpretability and efficiency while capturing multi-periodic patterns. In parallel, Gateformer [30] focused on multivariate forecasting through a dual-stage design that encodes temporal dynamics within each variable and applies gated cross-variate attention, yielding significant performance gains across diverse datasets.

Despite these architectural innovations, the effectiveness of Transformers for long-term forecasting has been questioned. Zeng et al. [6] demonstrated that embarrassingly simple linear models (LTSF-Linear) outperform sophisticated Transformer variants across multiple benchmarks, suggesting that many proposed designs may not truly capture temporal relations but rather exploit dataset-specific properties. This debate highlights that while Transformers offer powerful modeling capabilities, their application to TSF requires

careful design, including the integration of inductive biases such as decomposition, spectral analysis, or position-awareness, to ensure robust temporal modeling.

2.3. Positional Encoding Approaches in Transformers for TSF

A central challenge in adapting Transformers to time series lies in their permutation-invariant self-attention mechanism. Since the model lacks an inherent notion of order, positional encodings (PEs) are crucial for injecting temporal information into the input. Early approaches used absolute sinusoidal encodings, while recent research has explored learnable, relative, and hybrid methods, each with advantages and limitations [8]. We summarize the main contributions in the state-of-the-art as follows, whilst Table 1 summarizes the main positional encoding strategies proposed in the literature for time series models, highlighting whether they are fixed or learnable, how they inject positional information, and their relative advantages. Complementing this overview, Figure 1 illustrates the standard data processing pipeline, specifically depicting the stage where the positional embedding is integrated into the input sequence.

Table 1. Summary of positional encoding approaches in Transformer-based time series models.

Method	Type	Injection	Attributes
Absolute Positional Encoding (APE) [22]	Absolute, Fixed	Additive to embeddings	Parameter-free, extrapolates to unseen lengths, based on sin/cos functions.
Learnable PE (LPE) [14]	Absolute, Learnable	Additive to embeddings	Trainable parameters per position; flexible but lacks extrapolation.
time Absolute PE (tAPE) [7]	Absolute, Fixed	Additive to embeddings	Rescales sinusoidal frequencies using sequence length L to preserve distance-awareness.
Relative PE (RPE) [13]	Relative, Learnable	Modifies attention scores	Encodes pairwise offsets $i - j$ in attention weights; translation invariant but memory heavy.
efficient RPE (eRPE) [7]	Relative, Learnable	Modifies attention scores	Memory-efficient relative PE variant; scalable to long sequences.
TUPE [31]	Hybrid (Abs + Rel), Learnable	Attention-level	Untied projections for query, key, and position embeddings; richer interactions between content and position.
RoPE [32]	Relative, Fixed	Rotation in embedding space	Encodes positions via complex rotations; preserves relative distances multiplicatively.
Temporal PE (T-PE) [8]	Hybrid (Abs + Semantic)	Additive + attention-level	Combines sinusoidal encoding with a Gaussian similarity kernel over inputs; couples global periodicity and local semantics.
Representative/Global Attention [33]	Relative, Fixed	Modifies attention scores	Variants tailored for anomaly detection and TSF: Representative focuses on local lags, Global captures long-range dependencies.

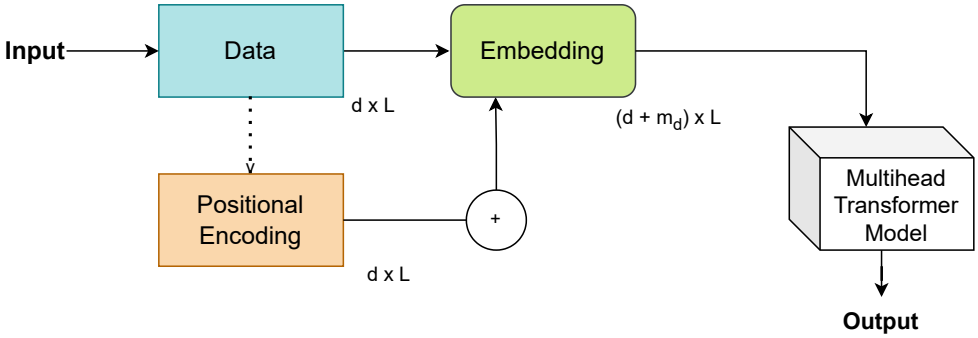

Figure 1. Schematic of the standard Transformer input pipeline, highlighting the injection of positional encodings into the sequence.

2.3.1. Positional Encodings

Here, we review both absolute and relative encodings, both fixed and learnable, that embed each time step with index-based signals added directly to input representations. Although these methods offer simplicity and efficient extrapolation, they often lack adaptability to the rich nonlinear dynamics present in real-world temporal data.

Absolute Positional Encoding (APE)

The original absolute encoding proposed by Vaswani et al. [22] defines position-dependent vectors using sinusoidal functions:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{1}$$

where pos denotes the position index and i the embedding dimension. These encodings are fixed and parameter-free, allowing extrapolation to unseen lengths. LPEs replace trigonometric functions with trainable parameters, granting the flexibility of the model but sacrificing extrapolation [14]. For time series, Foumani et al. [7] introduced the time Absolute Position Encoding (tAPE), which adjusts frequency terms to incorporate sequence length L:

$$\omega_k = 10000^{-2k/d_{model}}, \; \omega_k^{new} = \frac{\omega_k \times d_{model}}{L} \tag{2}$$

thereby preserving *distance awareness* even for low-dimensional embeddings.

Relative Positional Encoding (RPE)

RPE captures the distance between elements, rather than their absolute positions. Shaw et al. [13] formulated the attention score as

$$e_{ij} = \frac{(x_i W_Q)(x_j W_K + a^K_{i-j})^\top}{\sqrt{d_z}}, \tag{3}$$

where a^K_{i-j} encodes the relative offset between positions i and j. The output of the attention is defined as

$$z_i = \sum_j \alpha_{ij}(x_j W_V + a^V_{i-j}), \; where \; \alpha_{ij} = softmax_j(e_{ij}) \tag{4}$$

This provides translation invariance and improves generalization when relative lags are more important than absolute timestamps. Foumani et al. [7] further proposed an efficient Relative PE (eRPE), which reduces the memory cost of naive implementations.

2.3.2. Hybrid and Advanced Encodings

Although absolute and relative encodings capture complementary aspects of positional information, recent research has introduced *hybrid* and more *advanced* schemes that combine or extend their strengths.

Transformer with Untied Positional Encoding (TUPE)

Ke et al. [31] proposed TUPE, which modifies the attention formulation by incorporating both absolute and relative terms, but with *untied* parameters between the query, key, and positional embeddings. Instead of simply adding position vectors to the input embeddings, TUPE directly introduces them into the attention computation:

$$e_{ij} = \frac{(x_i W_Q)(x_j W_K)^\top}{\sqrt{d}} + \frac{(x_i W_Q)(p_j W_K^P)^\top}{\sqrt{d}} + \frac{(p_i W_Q^P)(x_j W_K)^\top}{\sqrt{d}}, \tag{5}$$

where p_i and p_j are positional embeddings, and W_Q^P, W_K^P are learnable position-specific projections. This untied design allows queries and keys to interact with both content and positional signals independently, providing more expressive control over position modeling.

Rotary Position Embedding (RoPE)

Su et al. [32] introduced Rotary Position Embeddings (RoPE), which encode absolute positions through *rotation* in a complex plane. The idea is to apply a position-dependent rotation matrix R_θ to the query and key vectors before computing attention:

$$Q_i^{\text{RoPE}} = R_{\theta(i)} Q_i, \quad K_j^{\text{RoPE}} = R_{\theta(j)} K_j, \tag{6}$$

where $R_{\theta(k)}$ is a block-diagonal matrix performing 2D rotations with angle $\theta(k)$. The inner product then becomes

$$(Q_i^{\text{RoPE}})(K_j^{\text{RoPE}})^\top = (Q_i K_j^\top) \cdot \cos(\theta(i-j)) + (Q_i \otimes K_j) \cdot \sin(\theta(i-j)), \tag{7}$$

thus encoding relative positions implicitly through multiplicative phase shifts. RoPE naturally preserves distance-awareness and enables extrapolation.

Temporal Positional Encoding (T-PE)

Irani and Metsis [8] described a hybrid scheme (T-PE) that augments absolute sinusoidal encodings with a semantic similarity kernel across time steps. Specifically,

$$TPE(i) = APE(i) + \sum_{j=1}^{L} \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \tag{8}$$

where $APE(i)$ is the absolute positional encoding of index i, and the Gaussian kernel captures local similarity in the embedding space. This formulation allows the model to combine global periodic signals with local contextual information. Although powerful, it scales quadratically with the sequence length.

Representative and Global Attention

Alioghli and Okay [33] proposed two variants of Relative Positional Encoding, tailored for anomaly detection and time series tasks.

Representative Attention partitions the sequence into local neighborhoods and assigns a representative positional embedding r_g to each group g. The attention score between the token i and j becomes

$$e_{ij} = \frac{(x_i W_Q)(x_j W_K)^\top}{\sqrt{d}} + (x_i W_Q)(r_{g(j)} W_K^R)^\top, \tag{9}$$

where $r_{g(j)}$ is the representative embedding of the group containing position j. This formulation reduces sensitivity to small offsets and is particularly effective for short sequences.

Global Attention introduces a single global positional embedding g that interacts with all queries and keys, capturing long-range dependencies. The score is modified as

$$e_{ij} = \frac{(x_i W_Q)(x_j W_K)^\top}{\sqrt{d}} + (x_i W_Q)(g W_K^G)^\top, \tag{10}$$

where g acts as a learnable global anchor. This scheme emphasizes global coherence across the entire sequence, making it better suited for long horizons.

Both methods extend the relative encoding family by biasing attention scores toward either local (Representative) or global (Global) temporal contexts, thus adapting positional awareness to different forecasting regimes.

2.4. Limitations of Previous Positional Encoding Approaches for TSF

Despite the progress achieved by absolute, relative, and hybrid positional encodings, several limitations remain when applying them to TSF:

- *Fixed absolute encodings*, such as the sinusoidal functions introduced by Vaswani et al. [22], are attractive because they are parameter-free and allow extrapolation to unseen sequence lengths. However, they lack any adaptivity to the underlying temporal dynamics of the data. This rigidity becomes particularly problematic in TSF, where sampling frequencies and temporal dependencies vary greatly across domains. Moreover, at low embedding dimensions, sinusoidal encodings suffer from anisotropy, producing highly correlated vectors that diminish the model's ability to distinguish between positions [7].
- *Learnable Absolute Encodings (LPE)* [14] attempt to address fixed encoding limitations by allowing position vectors to be optimized during training. While this improves flexibility, it also sacrifices extrapolation capacity, since learned encodings are tied to the maximum sequence length observed during training. Consequently, they can overfit to specific horizons and degrade in generalization performance when deployed on longer forecasting tasks. Similarly, methods such as tAPE [7] adjust frequencies according to sequence length to improve distance-awareness, but they remain deterministic and do not fully capture dataset-specific temporal structures.
- *Relative Positional Encodings (RPEs)* [13] alleviate some of these issues by focusing on temporal offsets rather than absolute indices, improving translation invariance and often delivering better performance in tasks where relative lags are critical. However, standard implementations require $O(L^2 d)$ memory and computation, severely limiting scalability to long sequences, with L being the sequence length and d the dimension of the embedding. Efficient variants, such as eRPE [7], reduce this overhead but at the cost of architectural complexity and potential information loss.
- *Hybrid approaches*, including TUPE [31], RoPE [32], and T-PE [8], enrich positional information by combining absolute and relative signals or by encoding positions as rotations. Although these strategies offer improved expressiveness, they introduce significant additional computational costs and are not specifically designed to address

the challenges of forecasting multivariate high-frequency time series. Furthermore, recent empirical studies [6,33] have revealed that many positional encodings display inconsistent performance across benchmarks, with no single approach emerging as clearly superior.

Taken together, these weaknesses highlight a critical research gap: Current positional encoding schemes either lack adaptivity, impose excessive computational demands, or fail to generalize across forecasting horizons and datasets. For time series forecasting, which often involves complex nonlinear dependencies, irregular sampling, and diverse temporal patterns, there is a need for encodings that can (i) exploit local statistical context, (ii) capture temporal lag structures, and (iii) flexibly combine multiple sources of positional information. This motivates the family of proposals introduced in this work, *WinStat*, which aims to unify these requirements into a set of encodings that are learnable, efficient, and semantically meaningful for forecasting tasks.

3. WinStat Positional Encoding Family

In this section, we present a novel family of positional encoding strategies—collectively termed WinStat (short for Window-based Statistics)—that explicitly incorporates local temporal context into Transformer-based forecasting models. Building on the observation that classical sinusoidal encodings capture only global, index-based geometry, and that purely learnable or relative schemes either overfit or incur prohibitive overhead, WinStat instead augments each timestamp's embedding with summary statistics computed over a sliding window and, in extended variants, with short-term lag differences. We further introduce hybrid formulations that learn optimal mixtures of these window-based signals with traditional absolute and semantic similarity encodings, allowing the model to adaptively weight global periodicity, local variability, and context-dependent similarity. By maintaining the simplicity of additive encoding while injecting richer temporal semantics, our approach aims to enhance the Transformer's ability to model both local and long-range dependencies in diverse time series forecasting tasks.

3.1. WinStat Base

The first proposed method, *WinStat*, defines our most basic positional encoding method. *WinStat* enriches each input embedding by concatenating local statistics computed over a sliding window of size w. For each position t, we calculate

$$\begin{aligned} \mu_t &= \frac{1}{|W_t|} \sum_{x \in W_t} x, \\ \sigma_t &= \sqrt{\frac{1}{|W_t|} \sum_{x \in W_t} (x - \mu_t)^2}, \\ m_t^{\min} &= \min_{x \in W_t} x, \\ m_t^{\max} &= \max_{x \in W_t} x. \end{aligned}$$

where W_t is the neighborhood window centered at t.

The statistical vector obtained is

$$s_t = [\mu_t, \sigma_t, m_t^{\min}, m_t^{\max}] \in \mathbb{R}^4, \tag{11}$$

and the enriched embedding would be

$$\tilde{x}_t = [x_t \,\|\, s_t] \in \mathbb{R}^{d+4} \tag{12}$$

where the operation $\|$ notes the vector concatenation operation. The enriched sequence is noted as

$$\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_L] \in \mathbb{R}^{L\times(d+4)}. \tag{13}$$

3.2. WinStatLag

WinStatLag extends WinStat by incorporating explicit temporal lag differences. Given a set of lags $L = \{\ell_1, \ell_2, \ldots, \ell_p\}$, for each position t we compute

$$\delta_t^{(\ell_j)} = \begin{cases} |x_t - x_{t-\ell_j}|, & \text{if } t - \ell_j \geq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

The enriched embedding becomes

$$\tilde{x}_t = [x_t \,\|\, s_t \,\|\, \delta_t^{(\ell_{\mathcal{I}_1})} \,\|\, \ldots \,\|\, \delta_t^{(\ell_{\mathcal{I}_k})}] \in \mathbb{R}^{d+4+k} \tag{15}$$

where $\{\mathcal{I}_1, \ldots, \mathcal{I}_k\} \subseteq \{1, \ldots, p\}$ and $\ell_{\mathcal{I}_i} \in L$ meets that $\delta_t^{(\ell_{\mathcal{I}_i})} \neq 0$. Thus, the sequence is

$$\tilde{X} \in \mathbb{R}^{L\times(d+4+k)}. \tag{16}$$

This formulation captures both local statistics and short- to medium-term variations, providing richer semantic context.

3.3. WinStatFlex

WinStatFlex generalizes the approach by combining WinStatLag with several additive positional encodings using trainable weights. Specifically, given encodings X^{PE} (sinusoidal), X^{LPE} (learnable), X^{tAPE} (time-aware), and X^{TPE} (temporal hybrid), the final representation is

$$X^{final} = \tilde{w}_0 \cdot \tilde{X} + \tilde{w}_1 \cdot X^{PE} + \tilde{w}_2 \cdot X^{LPE} + \tilde{w}_3 \cdot X^{tAPE}, \tag{17}$$

where $\tilde{w} = \text{Softmax}(w)$ ensures normalized, sum-one trainable weights.

This design allows the model to learn the optimal contribution of statistical, lag-based, and additive encodings, adapting to dataset-specific requirements. A comprehensive visualization of this process is presented in Figure 2, which illustrates the complete data flow and specifically details how the final positional embedding is synthesized by aggregating multiple distinct encodings via learnable weights normalized to sum to unity. Notably, this normalized weighting scheme is a core architectural component, consistently applied across the TPE and SPE variants as well.

3.4. WinStatTPE

WinStatTPE follows the same weighted scheme as WinStatFlex but replaces $tAPE$ with TPE. This substitution exploits the hybrid nature of TPE, combining absolute sinusoidal structure with local semantic similarity:

$$X^{final} = \tilde{w}_0 \cdot \tilde{X} + \tilde{w}_1 \cdot X^{PE} + \tilde{w}_2 \cdot X^{LPE} + \tilde{w}_3 \cdot X^{TPE}. \tag{18}$$

The learnable weights control the balance between global periodicity, local statistics, and semantic similarity, making this variant particularly robust.

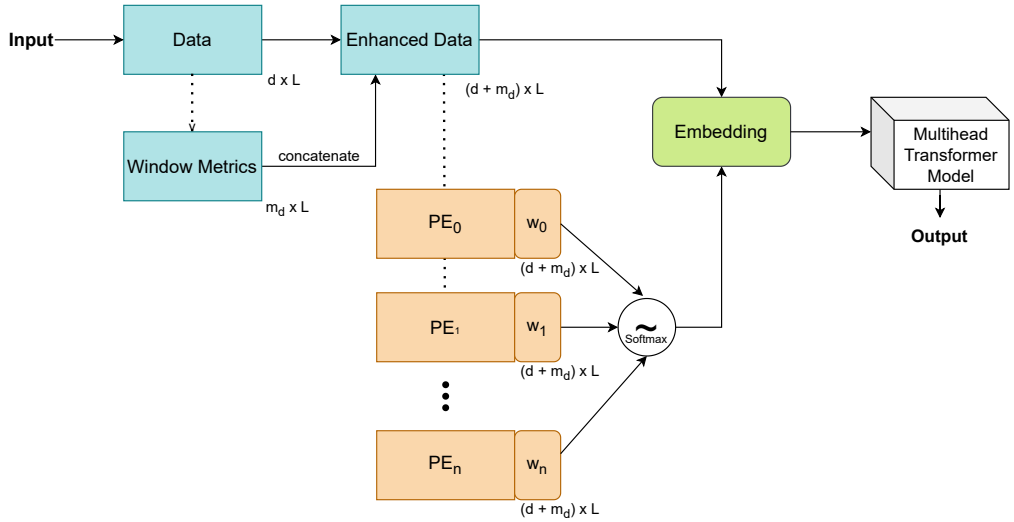

Figure 2. Schematic representation of the proposed WinStat data flow. The diagram highlights the embedding generation stage, demonstrating how the final vector is derived by combining multiple positional encodings via a set of learnable, normalized weights ($\sum w_i = 1$). This weighted aggregation strategy is fundamental to the proposed framework and is identically employed in the weighted variants.

3.5. WinStatSPE

Finally, *WinStatSPE* integrates the WinStatFlex scheme with *Stochastic Positional Encoding* (SPE), a convolution-based modification of the attention matrix. The enriched embedding is

$$X^{final} = \sum_i \tilde{w}_i \cdot X^{(i)} + \tilde{w}_{SPE} \cdot X^{SPE}, \tag{19}$$

where $X^{(i)}$ denotes the previous additive components, the weights are a sum-one normalized vector, and X^{SPE} introduces a convolutional structure in the attention mechanism. Although SPE is primarily designed for classification, its integration here allows an exploratory assessment of its potential for forecasting.

3.6. Theoretical Motivation of WinStat Positional Encoding Family

Transformers require positional information to break permutation invariance, yet standard absolute or Relative Positional Encodings provide purely geometric indices that do not reflect the temporal semantics of the signal. Time series data, however, exhibit local distributional structure and short-term dynamics that often determine predictive behavior. The WinStat encoding enriches positional representations by incorporating simple statistics of the local window and short-range difference-based features.

A common modeling assumption in time series analysis is local stationary, whereby over a short window $W_t = \{x_{t-w+1}, \ldots, x_t\}$ the process behaves approximately as if it were stationary with parameters θ_t that vary slowly in time. Under this assumption, the short-term future $x_{t+1:t+H}$ depends primarily on the local regime encoded in W_t, rather than on the remote past. In many families, especially exponential families such as the locally Gaussian case, low-dimensional statistics of W_t (e.g., empirical mean, variance, minimum, and maximum) act as approximate sufficient statistics for θ_t. Letting $s_t = [\mu_t, \sigma_t, m_t^{min}, m_t^{max}]$, WinStat exposes these regime descriptors directly to the Transformer. Because conditioning on additional variables cannot reduce mutual information, the representation (x_t, s_t) always satisfies

$$I(x_{t+1:t+H}; x_t, s_t) \geq I(x_{t+1:t+H}; x_t),$$

and under local stationary s_t captures most of the local distributional information relevant for short-horizon prediction.

Window statistics describe local state but not local dynamics, such as the direction and speed of change or the short-range autocorrelation structure. For this reason, WinStat includes difference-based features

$$d_t = [\delta_t^{\ell_1}, \delta_t^{\ell_2}, \dots],$$

which serve as empirical descriptors of short-term dynamics. Augmenting the representation further yields $z_t = [x_t || s_t || d_t]$. Again, information-theoretically, $I(x_{t+1:t+H}; x_t; s_t; d_t) \geq I(x_{t+1:t+H}; x_t; s_t)$, so the enriched embedding cannot be less informative about the short-term future.

Crucially, the additional features introduced by WinStat are not arbitrary. Window statistics and difference-based features necessarily carry predictive information under any non-trivial time series model. For any process exhibiting short-range temporal dependence, $I(x_{t+1:t+H}; x_{t-w+1:t}) > 0$. Since s_t and d_t are deterministic functions of the window, they inherit strictly positive mutual information with the future. Therefore, the added features are informative by construction as they capture distributional and dynamical components of the local context.

This enrichment also shapes the geometry of attention. Writing $Q_t = W_Q z_t$ and $K_t = W_K z_t$, and using the fact that linear map are Lipschitz, we obtain

$$||Q_t - Q_u|| \leq L_Q ||z_t - z_u||, \; ||K_t - K_u|| \leq L_K ||z_t - z_u||.$$

Since

$$||z_t - z_u||^2 = ||x_t - x_u||^2 + ||s_t - s_u||^2 + ||d_t - d_u||^2,$$

the embedding introduces structured geometric separation. When two positions belong to the same local regime, concentration of empirical statistics implies $s_t \approx s_u$ and $d_t \approx d_u$, so their keys and queries remain close. When they belong to different regimes, at least one of the components s_t or d_t differs systematically, introducing a positive lower bound on $||z_t - z_u||$ and thus separating their queries and keys. This induces a data-dependent semantic margin in the attention space, allowing the model to differentiate local regimes even when raw values x_t and x_u might be misleadingly similar. Unlike fixed geometric encodings, this margin depends on the actual temporal behavior of the series, reflecting both local distributional structure and local dynamics.

4. Experimental Setup

In this section, we describe the experimental framework designed to evaluate the effectiveness of the proposed WinStat family of positional encodings against established baselines. To isolate the impact of positional information, we maintain identical architectural configurations across four diverse domains: residential energy consumption (HPC), electricity infrastructure monitoring (ETT), urban mobility patterns (NYC), and industrial machinery monitoring (TINA). The experimental design follows rigorous protocols for temporal data, including chronological splitting to preserve sequential dependencies and systematic hyperparameter selection guided by preliminary sensitivity analyses.

4.1. Datasets

To comprehensively evaluate the proposed WinStat family of positional encodings, we selected four heterogeneous time series datasets that span different domains, temporal granularities, and forecasting challenges. These benchmarks were chosen to assess

robustness across varied data characteristics: from high-frequency household consumption data (HPC) to electrical grid monitoring with strong seasonal patterns (ETT), urban mobility aggregates (NYC), and large-scale industrial sensor streams (TINA). Each dataset presents distinct temporal dynamics—ranging from clear periodicity to mixed stationary-nonstationary behavior—ensuring that our positional encoding methods are tested under diverse forecasting scenarios representative of real-world applications.

4.1.1. Household Consumption Data(HPC)

The *Individual Household Electric Power Consumption* dataset (UCI) records minute-level consumption for a household near Paris from December 2006 to November 2010 (47 months; 2,075,259 timestamps) [34]. Although UCI lists nine attributes, the date and time fields are stored separately; when unified, each timestamp comprises eight variables (time plus seven electrical measurements). HPC blends clearly seasonal components (e.g., sub-metering channels) with variables closer to stationarity (e.g., reactive power), and shows strong associations between *global active power* and *intensity*. This combination of high frequency, long duration, and mixed temporal behavior makes HPC a demanding benchmark for positional encodings intended to capture both local and global temporal structure.

4.1.2. Electricity Transformer Dataset (ETT)

The ETT benchmark, popularized by *Informer* and subsequent TSF studies, records load components and *oil temperature* (OT) from two power transformers installed in different substations [24]. The public release comprises two minute-level series (ETT-small-m1, ETT-small-m2; 2 years, 70,080 timestamps each) and their hourly counterparts (ETTh1, ETTh2) that downsample the same process to reduce computational burden [24]. Each timestamp contains eight features (time plus seven variables: HUFL, HULL, MUFL, MULL, LUFL, LULL, and OT). Empirically, ETTh1 exhibits marked seasonality and high one-step autocorrelation across variables, whereas ETTh2 retains seasonality with smaller amplitudes—differences that make ETTh2 comparatively harder to model. The OT variable is operationally salient because extreme temperatures impact transformer health and efficiency, which turns the dataset into a meaningful testbed for long-horizon forecasting under strong periodic structure.

4.1.3. Yellow Trip Data (NYC)

The *Yellow Trip Data Records* from NYC Open Data provide hourly aggregates of operational and monetary variables (e.g., passenger count, trip distance, fare amount, tip amount, total amount) [35]. In our setup, the 2015 series were preprocessed and then artificially extended to exceed two million timestamps to broaden temporal coverage while keeping moderate dimensionality (time + six variables). Autocorrelation analyses reveal pronounced daily periodicities (e.g., peaks around workday transitions in *passenger_count* and *fare_amount*), cross-variable synchrony (e.g., *tip_amount* with *passenger_count*/*fare_amount*), and near-stationary components (e.g., *trip_distance*).

4.1.4. Time Series Industrial Anomaly (TINA)

TINA is an industrial multivariate time series released by DaSCI for anomaly analysis in heavy machinery [36]. Data were collected from a mining machine at ArcelorMittal every two seconds between 6 September 2017 and 25 December 2018, yielding roughly 38 million timestamps. After anonymization and normalization, the dataset contains 108 columns (numeric and categorical), including categorical indicators such as `FEATURE76`, `FEATURE87`, maintenance flags (`m_id`, `m_subid`), and `alarms`. Its scale and heterogeneity make TINA an exacting test for both computational efficiency and robustness of positional encodings for TSF, particularly under the presence of rare, high-impact events.

4.2. Model Used and Rationale

As backbone, we adopt an *Informer-like* Transformer—chosen for its prevalence in the TSF literature and practical tooling—while *disabling ProbSparse* to recover *vanilla* full self-attention. This choice ensures a faithful assessment of positional encodings without confounding effects from sparsification or distillation that may erase locality, which is particularly harmful when encodings inject local statistics or semantic context. Using $O(L^2)$ attention provides a clean, comparable environment to isolate the effect of positional information and test procedures, such as completely removing PE.

Recent advancements in transformer-based models have led to a shift in how positional information is integrated. Positional encoding has increasingly become implicit, often incorporated directly within the architecture rather than explicitly tied to the input data. Contemporary architectures such as PatchTST [37] and TimeNET [38] exemplify this trend: PatchTST employs cross-patch attention and does not rely on explicit positional encodings, while TimeNET omits attention mechanisms entirely, effectively removing traditional positional signals. Consequently, existing approaches do not employ positional encodings in the traditional sense, preventing a straightforward comparison with our proposed methods.

We avoid architectures with implicit positional mechanisms (e.g., auto-correlation-based designs) so that improvements can be attributed to the encoding itself rather than architectural priors. Implementation follows a minimal, modifiable Transformer encoder–decoder stack suitable for multivariate TSF.

4.3. Validation Schema and Evaluation Metrics

To preserve temporal semantics, we use a chronological *last-block* split for testing and validation. Specifically, we adopt a 70–30 train–test partition and derive validation using a second-to-last block extracted from the training segment. We explicitly avoid blocked cross-validation due to its stationarity requirement and the prohibitive retraining costs associated with long sequences. This protocol yields an unbiased out-of-sample estimate while respecting order constraints.

Finally, dataset-specific horizons and windows follow a lightweight sensitivity study on ETT (e.g., typical setting seq $= 96$, label $= 48$, pred $= 24$) and are adapted when memory constraints arise in large and wide datasets (e.g., TINA with seq $= 288$, window $= 60$).

Training minimizes the mean squared error (MSE), the de facto regression loss in TSF, ensuring comparability to previous work. We report MSE and mean absolute error (MAE) as primary metrics.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{20}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{21}$$

4.4. Positional Window Size Selection

All WinStat variants share a single hyperparameter, the *window size* w, which determines how much local context (statistics and lagged differences) each position receives. Very large windows dilute locality, producing generic statistics that contribute little to attention; very small windows are noise–prone and may hinder pattern detection. Hence, w must balance local sensitivity and robustness.

Exhaustive search over w is prohibitive on large corpora, so we established a practical heuristic with a controlled study on the hourly ETT variants. Using the canonical Informer setting (seq $= 96$, label $= 48$, pred $= 24$), we swept $w \in \{3, \ldots, 96\}$ on ETTh1 and ETTh2, repeating each configuration 10 times to stabilize the MSE estimates. The best ranges fell

between $w = 24$ and $w = 36$, i.e., approximately one quarter to one third of the sequence length. We therefore adopt the rule-of-thumb $w \in [\lfloor L/4 \rfloor, \lfloor L/3 \rfloor]$ when feasible.

On very large or high-frequency datasets, we apply the heuristic subject to memory constraints. For instance, on TINA (sequence length $L = 288$) the 30-second sampling interval motivates broader context, but GPU memory constraints required us to cap the window at $w = 60$ (roughly the last half hour), instead of using $L/4$–$L/3$. This preserves informative locality while keeping training feasible.

4.5. Hyperparameters by Dataset

In Table 2, we report the dataset-specific configuration used across all positional-encoding variants so that architectural effects are held constant and differences can be attributed to the encoding itself (Transformer with *vanilla* self-attention, batch size, dropout, and number of runs as indicated).

Table 2. Common hyperparameters per dataset used across all positional-encoding variants. All runs use full (vanilla) self-attention to isolate the effect of positional encoding.

Dataset	Attention	Batch	Dropout	Runs	w	L	Label	Pred
HPC (UCI)	Vanilla	32	0.2	5	60	180	60	60
ETT (ETTh1/ETTh2)	Vanilla	32	0.2	5	24	96	48	24
Yellow Trip (NYC)	Vanilla	32	0.2	5	60	180	60	60
TINA (Industrial)	Vanilla	32	0.2	5	60 †	288	96	48

† On TINA, the window w was capped by GPU memory constraints, even though the ETT pilot study suggested $w \in [L/4, L/3]$.

4.6. Hardware and Software Framework

All experiments are implemented in `Python` (version 3.12.7) with `PyTorch` (version 2.5.1) as the deep learning framework; data handling and analysis rely on `NumPy` (version 1.26.4), `Pandas` (version 2.2.3), `Matplotlib` (version 3.9.2), and `statsmodels` (version 0.14.2). Training is carried out in a DGX cluster with 1× NVIDIA Tesla V100 (32 GB), 32 CPU cores (Intel® Xeon® E5-2698) and 96GB RAM. All implementation and experimental setup details are included in GitHub https://github.com/ari-dasci/S-WinStat (accessed on 8 October 2025).

5. Results and Analysis

In this section, we present a comprehensive empirical evaluation of the proposed WinStat family of positional encodings across all available heterogeneous benchmarks. Our experimental design aims to identify overall trends that may indicate the most effective encoding variant across diverse datasets. To this end, we evaluate all five proposed variants on each dataset—HPC, the ETT hourly variants, NYC, and TINA—alongside established baselines, using identical architectural configurations and training protocols to ensure fair comparison. Performance is assessed through MSE and MAE metrics across multiple independent runs. To validate that improvements stem from meaningful positional structure rather than incidental artifacts, we include decoder-input shuffling tests that systematically disrupt temporal order. This ablation approach allows us to establish a causal link between the proposed local semantic encodings and forecasting accuracy, while the learned mixture weights provide interpretable insights into which positional components contribute most effectively to each domain.

5.1. Comparative Results

We use the five previously introduced datasets as a comprehensive testing ground to evaluate the proposed encodings (WinStat, WinStatLag, WinStatFlex, WinStatTPE) against established baselines under identical architectural configurations and data splits. Collectively, these datasets constitute a heterogeneous benchmark environment, capturing a wide spectrum of temporal properties such as seasonality, stationarity, locality, and the coexistence of high- and low-frequency information. This diversity provides a rigorous basis for assessing the robustness and generality of the proposed methods across problems with distinct structural and semantic characteristics.

Table 3 consolidates all results across datasets and methods, including Informer (with `timeF`), sinusoidal APE only, a no-PE ablation, and the WinStat family. Informer serves as the reference baseline, with removal of PE leading to substantial performance degradation (higher MSE/MAE), whereas sinusoidal APE shows moderate losses compared with Informer. Figure 3 shows that, among the WinStat variants, WinStatFlex and WinStatTPE consistently achieve the lowest errors and most stable variances in HPC. Additionally, results on ETTh, NYC, and TINA datasets—representative of state-of-the-art benchmarks—confirm that the observed patterns generalize beyond the HPC dataset.

Table 3. Comparison of the WinStat family across five datasets (vs. Informer, sinusoidal, no PE, ROPE, and TUPE). Values represent the mean ± standard deviation over three runs. Bold values indicate the best results for each dataset and metric.

Encoding	Metric	HPC	ETTh1	ETTh2	NYC	TINA
Informer	MAE	0.4139 ± 0.0122	0.5323 ± 0.0170	0.6799 ± 0.0830	$3.247 \times 10^{-3} \pm 1.311 \times 10^{-4}$	0.3569 ± 0.0143
	MSE	0.5309 ± 0.0104	0.5458 ± 0.0204	0.8866 ± 0.1000	$1.158 \times 10^{-5} \pm 1.159 \times 10^{-5}$	1.0556 ± 0.0946
	s/epoch	1622.5673	9.3117	7.0332	2586.0825	3381.5719
PE (sin/cos)	MAE	0.4140 ± 0.0058	0.5552 ± 0.0530	0.9354 ± 0.0513	$4.04 \times 10^{-3} \pm 8.892 \times 10^{-4}$	0.3926 ± 0.0156
	MSE	0.5377 ± 0.0098	0.5775 ± 0.0755	1.3881 ± 0.1581	$2.069 \times 10^{-5} \pm 8.5520 \times 10^{-6}$	1.1691 ± 0.1441
	s/epoch	1935.1905	9.4156	6.8421	2065.1050	1751.7543
WinStat base	MAE	0.3976 ± 0.0056	0.5738 ± 0.0285	0.5773 ± 0.0565	$\mathbf{1.110 \times 10^{-3} \pm 4.428 \times 10^{-3}}$	0.3172 ± 0.0047
	MSE	0.5010 ± 0.0035	0.6165 ± 0.0433	0.5808 ± 0.1028	$\mathbf{9.464 \times 10^{-6} \pm 8.821 \times 10^{-5}}$	$\mathbf{1.007 \pm 0.2307}$
	s/epoch	3094.5832	11.6796	12.1549	7691.8144	4975.6098
WinStatLag	MAE	0.3961 ± 0.0049	0.5800 ± 0.0401	0.6375 ± 0.0442	$2.696 \times 10^{-3} \pm 3.014 \times 10^{-3}$	$\mathbf{0.2986 \pm 0.0054}$
	MSE	0.4915 ± 0.0056	0.6166 ± 0.0604	0.7072 ± 0.0970	$2.078 \times 10^{-5} \pm 3.888 \times 10^{-5}$	1.1202 ± 0.1047
	s/epoch	3058.8799	12.7845	12.5865	7937.1738	4726.3677
WinStatFlex	MAE	0.3780 ± 0.0090	$\mathbf{0.4946 \pm 0.0195}$	$\mathbf{0.5128 \pm 0.0651}$	$2.729 \times 10^{-3} \pm 4.380 \times 10^{-3}$	0.3121 ± 0.0098
	MSE	0.4646 ± 0.0052	$\mathbf{0.4659 \pm 0.0277}$	$\mathbf{0.4676 \pm 0.1017}$	$3.5337 \times 10^{-5} \pm 6.6427 \times 10^{-6}$	1.1104 ± 0.0996
	s/epoch	3757.0456	12.3785	13.2354	8505.9394	6232.9690
WinStatTPE	MAE	$\mathbf{0.3737 \pm 0.0029}$	0.5071 ± 0.0312	0.5889 ± 0.0496	$3.859 \times 10^{-3} \pm 2.605 \times 10^{-3}$	0.3077 ± 0.0166
	MSE	$\mathbf{0.4639 \pm 0.0023}$	0.4896 ± 0.0481	0.5963 ± 0.0760	$2.607 \times 10^{-5} \pm 3.126 \times 10^{-5}$	1.0794 ± 0.1842
	s/epoch	4829.0695	15.4916	15.2376	10,965.4951	7894.6299
TUPE	MAE	0.4069 ± 0.0110	0.5112 ± 0.0261	0.5896 ± 0.0686	$4.556 \times 10^{-3} \pm 3.062 \times 10^{-3}$	0.3998 ± 0.0200
	MSE	0.5155 ± 0.0119	0.4871 ± 0.0396	0.5836 ± 0.1271	$3.5164 \times 10^{-5} \pm 4.556 \times 10^{-5}$	1.0702 ± 0.2117
	s/epoch	1595.0493	7.7966	9.6014	1585.5295	1649.8372
ROPE	MAE	0.4023 ± 0.0038	0.5838 ± 0.0383	0.9130 ± 0.1543	$2.664 \times 10^{-3} \pm 8.985 \times 10^{-4}$	0.3648 ± 0.0219
	MSE	0.5088 ± 0.0057	0.6329 ± 0.0842	1.5023 ± 0.5544	$9.592 \times 10^{-6} \pm 5.512 \times 10^{-6}$	1.0084 ± 0.0925
	s/epoch	3064.2061	9.7351	9.0590	2419.4463	4425.7903
No PE	MAE	0.4220 ± 0.0046	0.7419 ± 0.0509	0.7777 ± 0.1199	$4.592 \times 10^{-3} \pm 6.371 \times 10^{-3}$	0.3717 ± 0.0147
	MSE	0.5454 ± 0.0039	0.9234 ± 0.0841	1.0132 ± 0.2417	$2.337 \times 10^{-5} \pm 6.174 \times 10^{-6}$	1.1392 ± 0.1916
	s/epoch	1607.7277	8.9651	7.7267	2387.9838	3025.7112

On the ETT hourly benchmarks (ETTh1/ETTh2), both WinStatFlex and WinStatTPE markedly outperform Informer, while the *no-PE* ablation consistently ranks last, underscoring the necessity of explicit positional information. Figure 4 shows the performance in ETTh1: TUPE situates between WinStatFlex and WinStatTPE, highlighting its intermediate effectiveness. Sinusoidal APE behaves poorly, in some cases even worse than the shuffled

WinStat variants, suggesting that purely geometric absolute signals may misalign with the true temporal dependencies of this subset unless complemented by local or semantic components. As shown in Table 3, the ranking for ETTh2, from best to worst, is: WinStatFlex > WinStat base > TUPE > WinStatTPE > WinStatLag, with ROPE performing particularly poorly and occupying the last position. As illustrated in Figure 5, the visual comparison clearly emphasizes this gap and reveals the high variance in ROPE's results, underscoring its instability across runs. This confirms that semantic and local positional information, as captured by the WinStat variants, remains critical for accurately modeling long-range dependencies in these datasets.

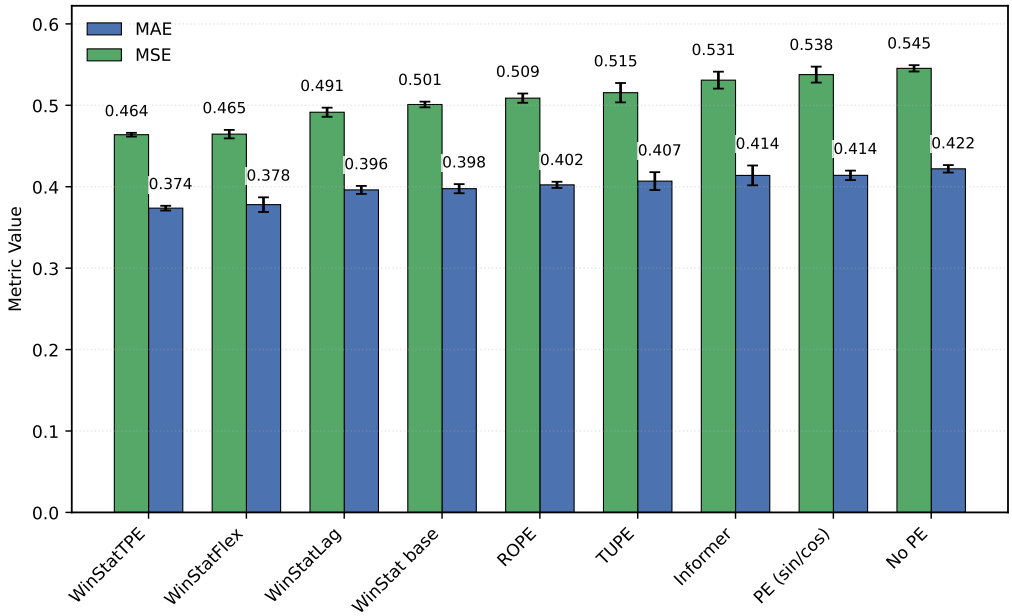

Figure 3. Mean squared error (MSE) and mean absolute error (MAE) calculated on the HPC dataset for all encodings (sorted by MAE).

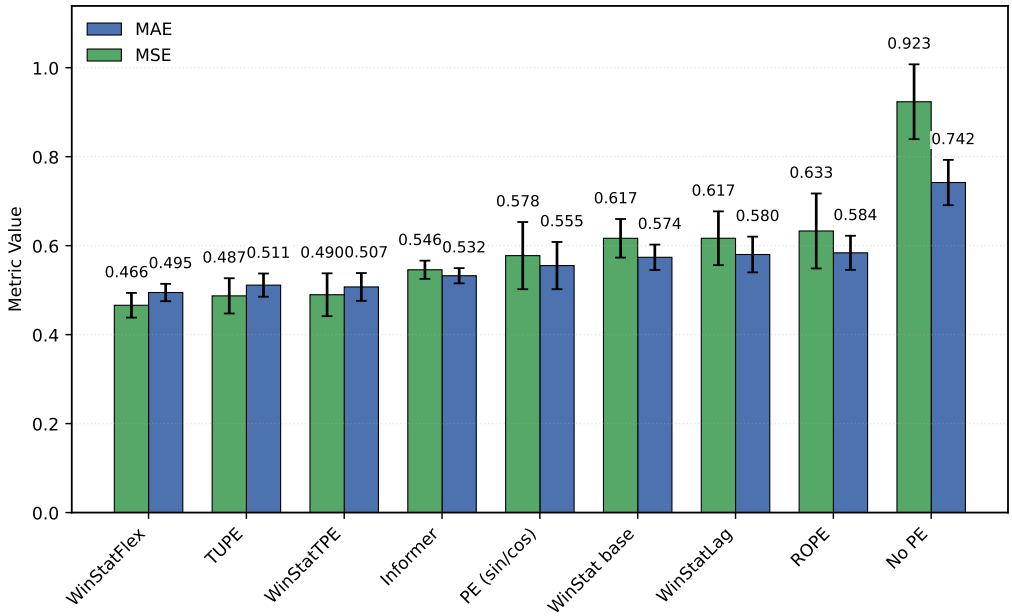

Figure 4. Mean squared error (MSE) and mean absolute error (MAE) calculated on the ETTh1 dataset for all encodings (sorted by MAE).

Regarding the hourly NYC aggregates, we caution that the series was artificially extended, which may bias the metrics toward optimistic values due to the smoothing effects of generative augmentation. With that caveat, WinStat base achieves the best performance overall. WinStatLag, WinStatFlex, and ROPE show very similar results, while TUPE performs poorly, approaching the performance level of the *no-PE* ablation. WinStatTPE still surpasses sinusoidal APE, but the ranking highlights that, in this dataset, the effectiveness of positional encoding varies substantially across methods. As shown in Table 3 and Figures 6 and 7, this variability is visually evident, revealing marked contrasts in performance stability between the WinStat family and methods like TUPE. For this reason, the figure has been divided to better read the results.

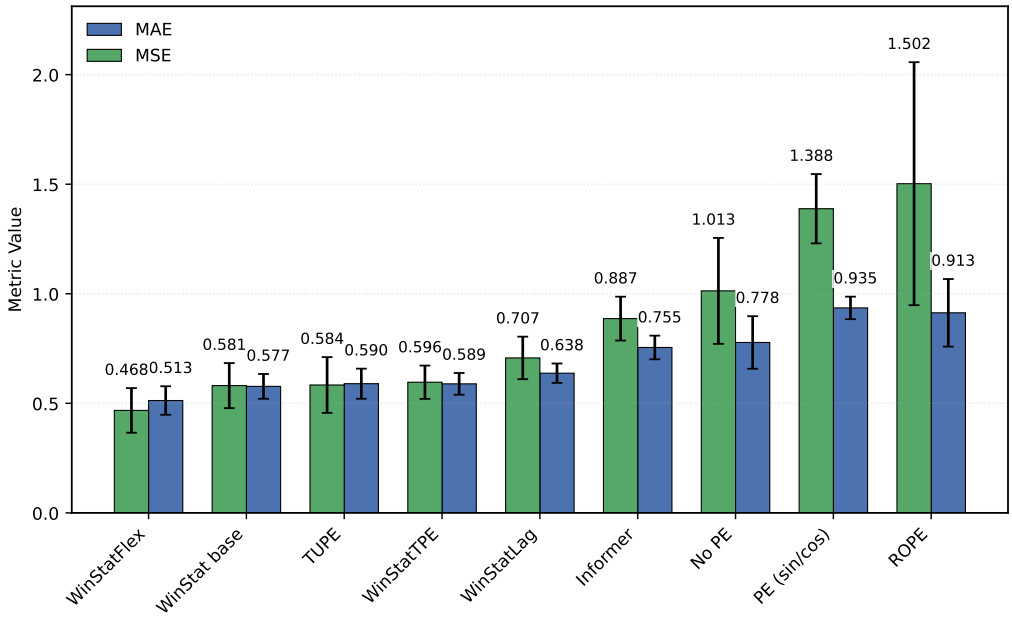

Figure 5. Mean squared error (MSE) and mean absolute error (MAE) calculated on the ETTh2 dataset for all encodings (sorted by MAE).

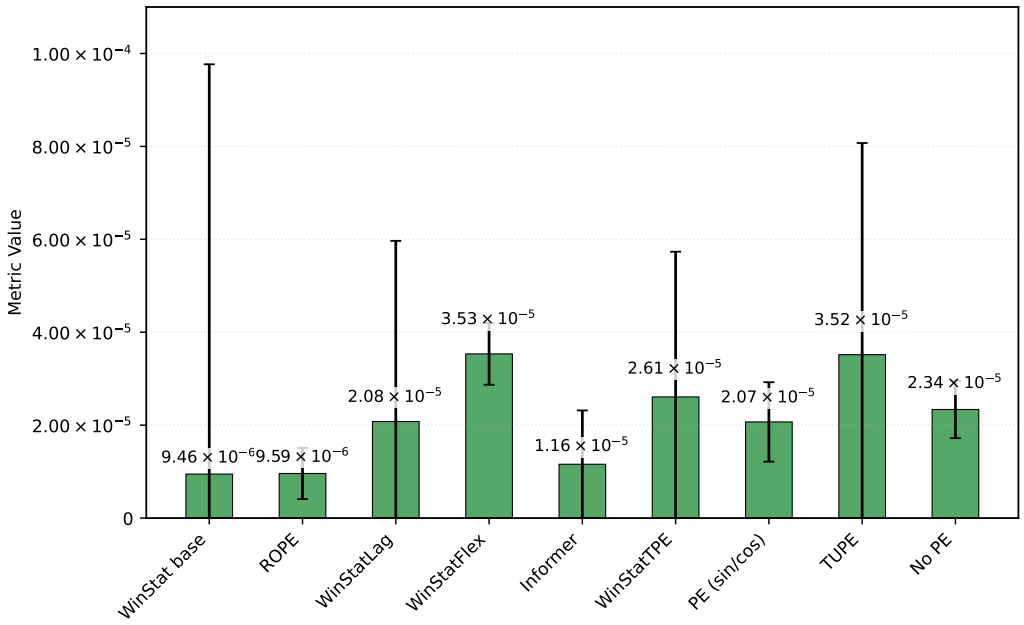

Figure 6. Mean squared error (MSE) calculated on the NYC dataset for all encodings (sorted by MSE).

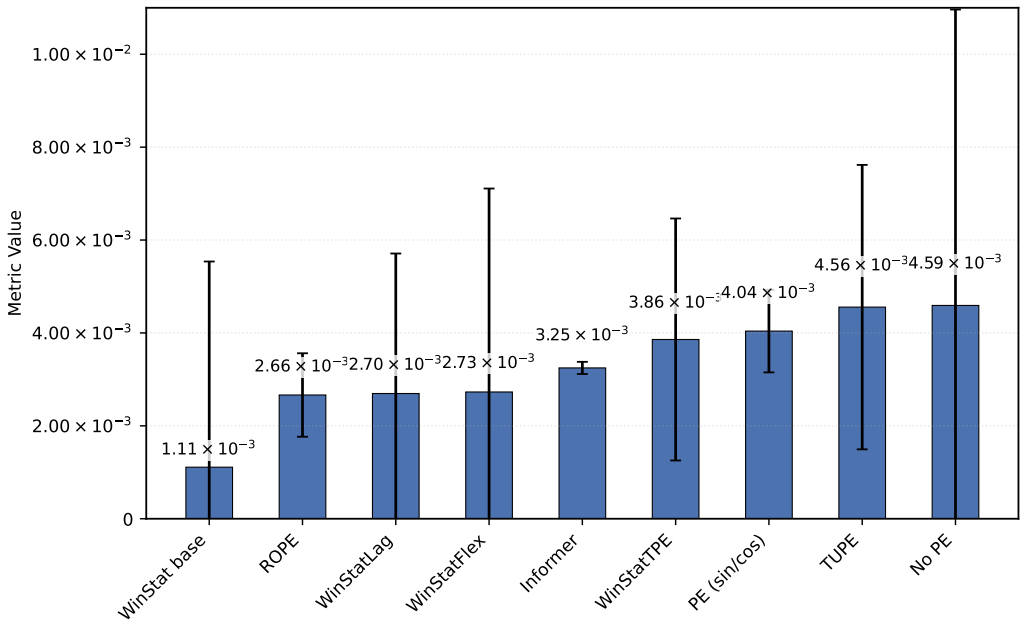

Figure 7. Mean absolute error (MAE) calculated on the NYC dataset for all encodings (sorted by MAE).

The fourth benchmark, TINA—an industrial, high-dimensional dataset originally curated for anomaly analysis—poses a stringent test of computational efficiency and robustness. Even after careful optimization, its scale requires substantial hardware and long runs; however, its breadth provides a valuable stress test for the positional mechanisms. As summarized in Table 3, the results indicate that, in terms of MAE, WinStatLag achieves the best performance, followed closely by the rest of the WinStat family, with Informer slightly behind; ROPE, Fixed PE, and TUPE all perform worse than Informer. TUPE ranks last, even below the *no-PE* ablation. These fluctuations highlight that the relative effectiveness of positional encodings can vary substantially due to the particular characteristics of this dataset. Graphically, the behavior of the encodings can be observed in Figure 8, where WinStatLag and WinStatTPE stand out, achieving substantially better results than the other methods. However, it becomes difficult to establish a clear hierarchy, as the relative performance varies depending on the metric considered.

Computational complexity, encompassing both execution time and memory resources, represents an additional dimension of this analysis. As detailed in the *Time/Epoch* rows of Table 3, the Informer and *no-PE* encoding methods demonstrate the highest temporal efficiency, resulting from their reduced computational load. Conversely, WinStatTPE exhibits the highest computational cost, attributable to the overhead of the TPE component calculation and the aggregation of weighted encodings. For the remaining WinStat variants, training duration scales by a factor of two to three—comparable to algorithms like ROPE or Informer on the largest datasets—reflecting the processing volume required for the additional calculated values.

To further evaluate computational cost in terms of execution time, we conducted a GPU profiling study using the manageable ETTh1 and ETTh2 datasets, as shown in Figure 9. This analysis corroborates the temporal impact across the four WinStat models. Notably, WinStatFlex and WinStatTPE models present a 33% increase in training duration when compared with the Informer architecture. This behavior is inherent to the model's design, attributable to the additional processing required to derive statistics within the sliding window.

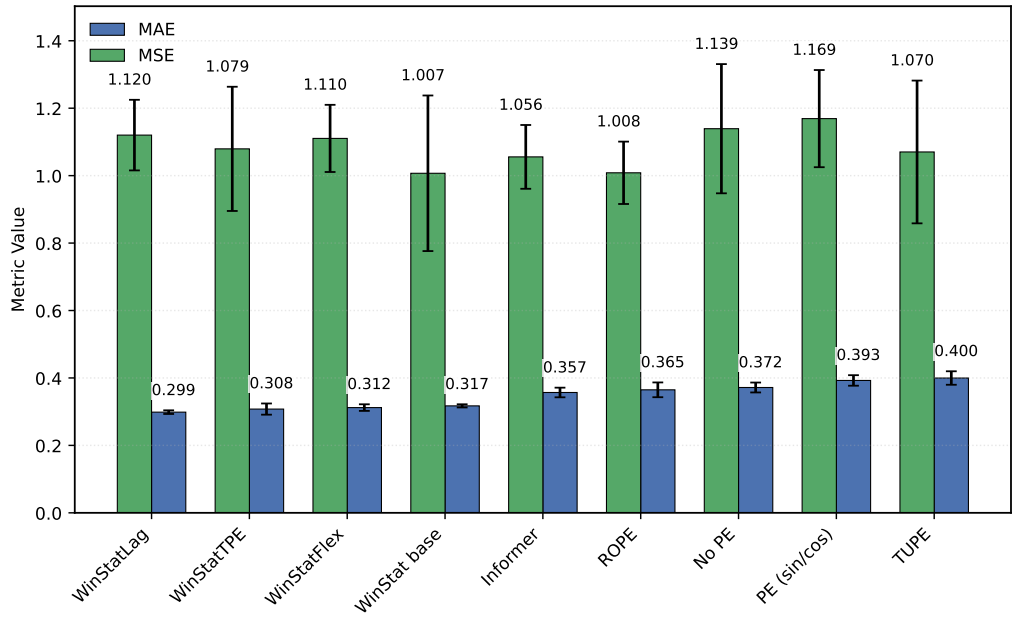

Figure 8. Mean squared error (MSE) and mean absolute error (MAE) calculated on the TINA dataset for all encodings (sorted by MAE).

In terms of memory consumption, while the computation of window-based statistics introduces an additional processing load, the specific impact on memory allocation remains minimal. The observed constraints are inherent to the underlying Transformer architecture rather than the positional encoding scheme. Since the WinStat method computes encodings via a sliding window without retaining auxiliary data, it maintains a memory footprint comparable to other standard methods, exhibiting no additional overhead.

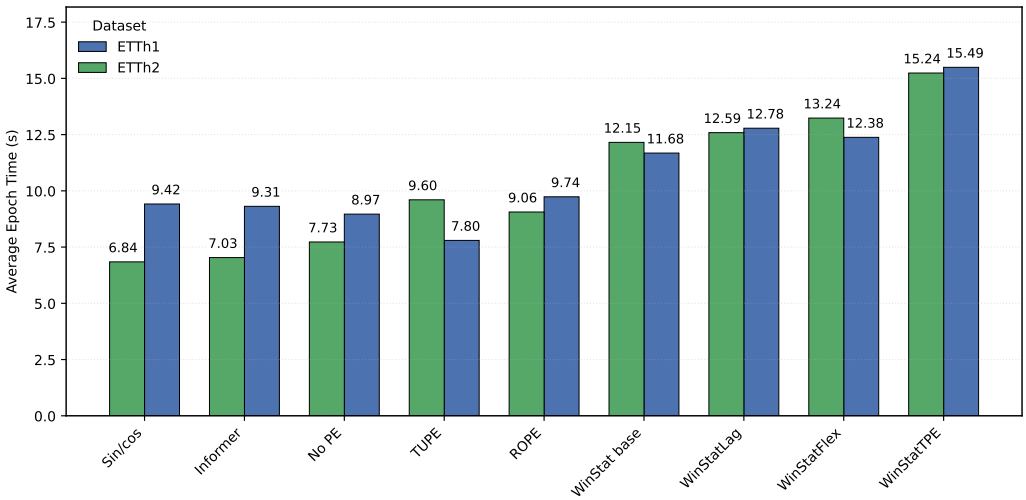

Figure 9. Comparison of the mean execution time per epoch (in seconds) across all encoding methods for the ETTh1 and ETTh2 datasets.

This observation is further supported by Figure 10, which visualizes memory usage for the ETTh1 and ETTh2 datasets. The results indicate negligible variance across most models, with only a minor increment observed in WinStatFlex. While WinStatTPE presents a distinct outlier with increased consumption—likely due to implementation specifics or GPU memory allocation dynamics—the primary proposed method, WinStatFlex, confirms that the memory overhead remains efficient and within reasonable limits.

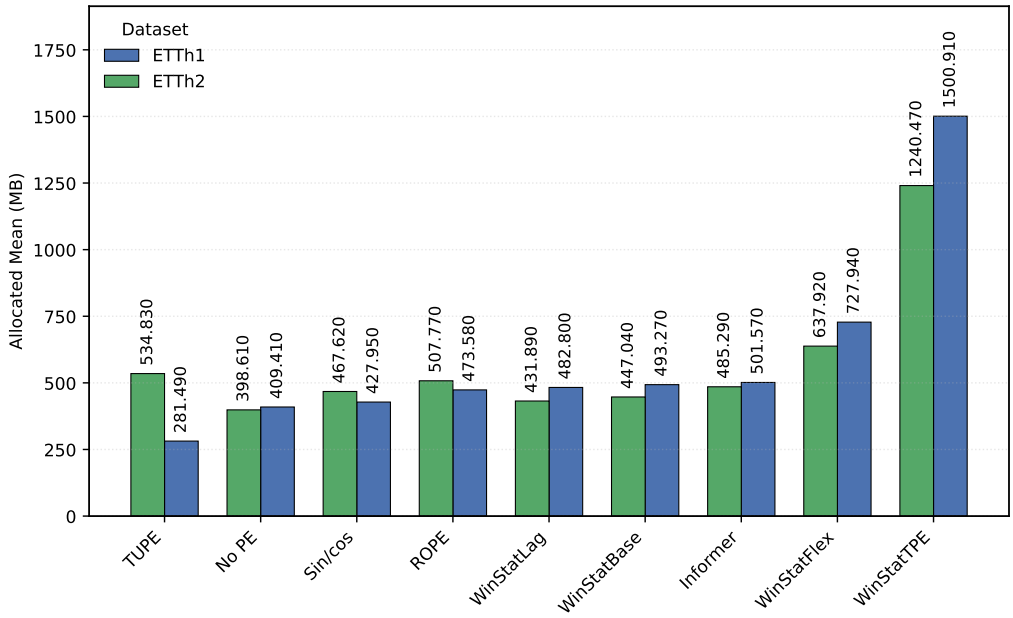

Figure 10. Comparison of mean memory usage across all encoding methods on the ETTh1 and ETTh2 datasets.

Finally, to validate the performance of WinStatFlex, we conducted a Bayesian Signed-Rank Test, as visualized in Figure 11. By establishing a Region of Practical Equivalence (ROPE) interval of 0.05, the analysis confirms that WinStatFlex significantly outperforms the *no-PE*, *PE (sin/cos)*, and Informer baselines. In these cases, the probability mass is located almost entirely outside the interval of equivalence, demonstrating a clear statistically significant advantage. A degree of statistical equivalence is observed only when comparing against the TUPE and RoPE encodings; while WinStatFlex maintains favorable performance, these specific methods emerge as the closest competitors, falling partly within the equivalence boundaries. It should be noted that, due to the lack of independent samples for the test, the results may not be as strong as they could be. Nevertheless, these results provide further evidence of the performance of the WinStat family over the other positional encoding solutions.

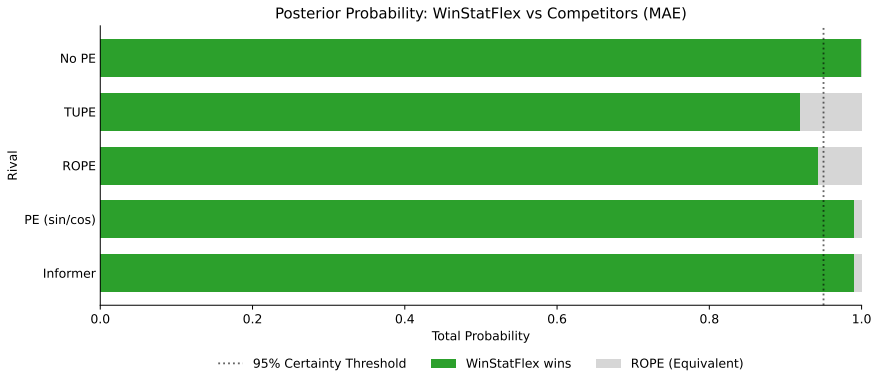

Figure 11. Posterior distributions resulting from the Bayesian Signed-Rank Test comparing WinStatFlex against baseline models. The green regions signify the probability of WinStatFlex achieving superior performance. Conversely, while red bars would denote instances where the competitor algorithm prevails, their absence indicates that such an outcome does not occur in the present case. Gray denotes the Region of Practical Equivalence (ROPE). The dashed vertical line marks the defined 0.05 significance threshold. A result is considered statistically significant only if the probability of winning (green bar) extends beyond this 0.05 boundary.

5.2. Analysis of the Learned Mixture Weights

To gain a clearer insight into the internal behavior of the proposed encodings, we analyze the relative contribution of each component within the normalized weighting scheme of WinStatFlex and WinStatTPE. For this purpose, we compile tables reporting the normalized weights associated with each component, enabling us to assess their relative importance in the overall computation. This analysis allows us to identify which parts dominate the representation and to evaluate whether the observed weighting patterns align with the intended design of the encodings.

5.2.1. HPC

The learned mixture weights, as reported in Table 4, indicate that the strongest variants assign non-trivial mass to the local statistics component together with absolute/learnable encodings (and, for TPE, a semantic term), rationalizing their superiority over fixed APE.

Table 4. HPC: learned mixture weights in WinStat variants (normalized). The symbol * identifies a component not considered in the weighting definition.

Component	WinStatFlex	WinStatTPE
Stats	0.3209	0.2520
PE	0.2577	0.2590
LPE	0.2569	0.2340
tAPE	0.1645	*
TPE	*	0.2550

5.2.2. ETT

In the case of the ETT benchmarks, both datasets exhibit a fairly similar and balanced behavior. For the first subset, ETTh1 (Table 5), and the WinStatFlex variant, the statistical component is marginally higher than the others, although the relevance distribution remains close to one-fourth across all components. For WinStatTPE, this slight increase shifts toward the TPE component itself, potentially due to its semantic contribution, which facilitates the modeling of outcomes. Overall, however, the weighting can be described as nearly uniform across components for both variants. This effect becomes even more pronounced in ETTh2 (Table 6), where both models achieve values that closely approximate an equal proportionality among components.

Table 5. ETTh1: learned mixture weights in WinStat variants (normalized). The symbol * identifies a component not considered in the weighting definition.

Component	WinStatFlex	WinStatTPE
Stats	0.2704	0.2580
PE	0.2506	0.2470
LPE	0.2420	0.2320
tAPE	0.2371	*
TPE	*	0.2630

Table 6. ETTh2: learned mixture weights in WinStat variants (normalized). The symbol * identifies a component not considered in the weighting definition.

Component	WinStatFlex	WinStatTPE
Stats	0.2569	0.2510
PE	0.2504	0.2490
LPE	0.2513	0.2490
tAPE	0.2413	*
TPE	*	0.2510

5.2.3. Yellow Trip Data (NYC): Aggregated Urban Mobility

Learned mixture weights, as reported in Table 7, show non-trivial mass on *Stats* and *TPE*, indicating that local statistics and semantic similarity are the dominant contributors in this domain. In this case, the differences among components are more pronounced than in the previous datasets. For the WinStatFlex variant, *Stats* and *tAPE* gain greater relevance, reducing the proportional weight of the fixed encoding, which appears to contribute little useful information. For WinStatTPE, the emphasis again falls on the semantic components, namely *Stats* and *TPE*. Interestingly, however, the learnable encoding becomes the least relevant component in this setting, which slightly increases the relative importance of the fixed PE.

Table 7. NYC: learned mixture weights (normalized).The symbol * identifies a component not considered in the weighting definition.

Component	WinStatFlex	WinStatTPE
Stats	0.2666	0.2740
PE	0.2506	0.2290
LPE	0.2380	0.2230
tAPE	0.2448	*
TPE	*	0.2740

5.2.4. TINA (Industrial): High-Dimensional, Long Sequences

In the particular case of the TINA dataset, as detailed in Table 8, the differences among variants are even more pronounced. For WinStatFlex, the largest share of the weighting falls on the concatenated statistical values, as the high complexity of the time series and its specific characteristics make the semantic information provided by this component especially critical. The tAPE component, despite being specifically designed to adapt the traditional encoding to the properties of the series, does not appear to achieve the intended effect, possibly introducing desynchronization and thus ranking last in relevance. For WinStatTPE, however, the differences among components are minimal, potentially due to the strong semantic contribution of the involved components, but also as a result of the particular characteristics of this dataset.

Table 8. TINA: learned mixture weights in WinStat variants (normalized). The symbol * identifies a component not considered in the weighting definition.

Component	WinStatFlex	WinStatTPE
Stats	0.3877	0.2680
PE	0.2272	0.2410
LPE	0.1888	0.2310
tAPE	0.1963	*
TPE	*	0.2600

5.2.5. Lessons Learned from Weight Training

After estimating the weights associated with each of the methods integrated into WinStat, several noteworthy conclusions can be drawn. First, the results highlight the high adaptability of WinStat, which stems from the diversity of its constituent components. This adaptability is clearly illustrated by the heatmap in Figure 12, which highlights the substantial variation in weights across different datasets. The visualization makes it evident that the algorithm is capable of dynamically adjusting its weighting strategy according to the characteristics of each specific scenario. Such flexibility is a desirable property in practical applications, as it enhances the robustness and generalizability of the method across heterogeneous contexts.

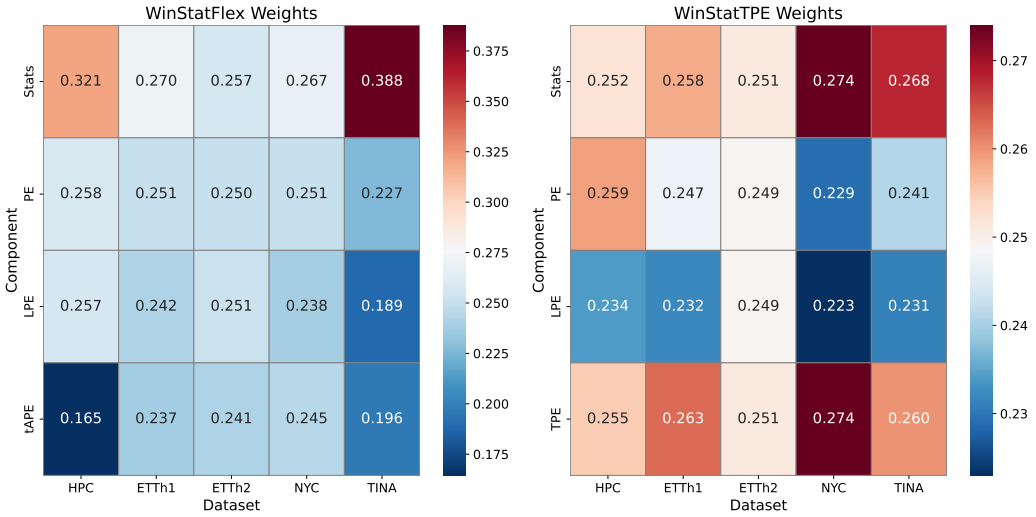

Figure 12. Component-wise weight values in WinStatFlex and WinStatTPE.

In addition, the statistical computation component emerges as particularly influential, consistently receiving a relatively high share of importance. For instance, in both the TINA and HPC datasets, this component accounts for more than 30% of the total weight, thereby reinforcing its central role in the overall performance of the framework. The prominence of this component indicates that the extraction and processing of statistical information remain a key driver of predictive capacity within the WinStat architecture.

At the same time, a certain degree of uniformity in the choice of weights is observed across several datasets. This tendency suggests that, while the algorithm adapts to scenario-specific requirements, it also identifies stable patterns of relevance among its components. Taken together with the strong performance metrics consistently achieved by the proposed method, these observations emphasize not only the complementary nature of the individual components but also their collective contribution to the effectiveness of WinStat. Consequently, the results provide empirical support for the design choices underlying the framework and point to its potential applicability in a wide range of real-world tasks.

5.3. Comparing the Semantic Quality of the Encoding

In this section, we highlight the semantic role of positional encodings and their importance for evaluating encoding quality. Although semantics lie outside the architecture, whose layers process only numerical tensors, such information may be implicitly captured through locality and patterns, which our proposed encodings aim to exploit.

A straightforward way to assess this is to shuffle the decoder inputs during inference and compare the performance with intact sequences, as suggested in [6]. This procedure provides an intuitive measure of whether an encoding preserves order and locality, as a significant drop in performance indicates that the model relies on sequential structure to make accurate predictions. In contrast, if shuffling has little effect, it suggests that the encoding does not effectively capture temporal dependencies.

To examine this property, we employed the five datasets evaluated previously, selected to study the impact of input shuffling across problems with diverse temporal and semantic characteristics, particularly in terms of seasonality and stationary patterns. The decoder shuffling procedure was applied to the WinStat family, as well as to TUPE, ROPE, and the Informer model itself, enabling a direct comparison of its effect across these different approaches.

As shown in Table 9, the performance of the original Informer and its shuffled variant is nearly identical, especially in HPC, which is consistent with the findings of the original study in [6]. The version without positional encoding performs at an intermediate level, leading to overall similar results across all three models and underscoring the limited semantics and weak-order preservation provided by the traditional sinusoidal scheme.

Table 9. Comparison of differences between shuffled and original encodings on 5 datasets. Values are the mean differences (Shuf − Orig) over 3 runs. If (Shuf − Orig) < 0, the shuffled version performs better; if (Shuf-Orig) > 0, the original version performs better. The higher the values, the better.

Encoding	Metric	HPC	ETTh1	ETTh2	NYC	TINA
Informer (Shuf.)	MAE	0.0291	0.1035	0.1358	2.509×10^{-3}	0.0014
	MSE	0.0520	0.3297	0.1924	3.1×10^{-5}	−0.0626
WinStat base (Shuf.)	MAE	0.0815	0.1423	0.0173	9.03×10^{-3}	0.1169
	MSE	0.1577	0.4235	0.0170	4.25×10^{-4}	0.1650
WinStatLag (Shuf.)	MAE	0.2102	0.2125	0.1059	3.323×10^{-3}	0.1707
	MSE	0.5049	0.5444	0.2477	5.1×10^{-5}	0.1580
WinStatFlex (Shuf.)	MAE	0.2156	0.1366	0.1800	5.5405×10^{-2}	0.1889
	MSE	0.3463	0.2237	0.3932	8.4287×10^{-2}	0.1130
WinStatTPE (Shuf.)	MAE	0.2062	0.1657	0.1003	-1.6×10^{-5}	0.1379
	MSE	0.3883	0.3139	0.2330	8×10^{-6}	0.3248
ROPE (Shuf.)	MAE	0.0217	−0.0195	−0.1334	-2.45×10^{-4}	−0.0075
	MSE	0.0361	−0.0218	−0.4728	-2×10^{-6}	0.0333
TUPE (Shuf.)	MAE	0.0133	0.0124	0.0512	-1.251×10^{-3}	−0.0283
	MSE	0.0146	0.0374	0.0613	-1.8×10^{-5}	−0.0351

In contrast, WinStatFlex exhibits a marked drop in performance when inputs are shuffled, a desirable behavior that reflects its ability to capture both contextual dependencies and the intrinsic sequential structure of the data.

A similar effect is observed for WinStatTPE, whose degradation under shuffling clearly exceeds that of the baseline Informer. This outcome indicates that the semantic information captured by T-PE is likewise sensitive to disorder, highlighting its superior ability to encode a meaningful temporal structure compared with the traditional approach.

Although these two methods achieved the best results in the previous experiments within our model family, the remaining variants exhibit a consistent pattern: all experience a substantial performance degradation when input shuffling is applied. This provides empirical evidence, in a general sense, that our proposed encodings are significantly sensitive to structural changes in the data. Consequently, it demonstrates that they successfully inject meaningful positional information with a strong semantic component into the datasets.

Across the evaluated benchmarks, WinStat base consistently demonstrates superior performance on the TINA and NYC, reflecting the particular temporal characteristics of these series. In TINA, the time series exhibits complex, low-frequency fluctuations over a wide temporal range, making the statistical component especially critical for capturing relevant patterns. In Taxi, the synthetic extension of the series similarly emphasizes the importance of statistical features, helping to stabilize learning despite the artificial augmentation. In contrast, TUPE consistently displays a counterintuitive behavior on these same datasets: applying input shuffling leads to better performance than the unshuffled model, contrary to the expected clear degradation. This can be attributed both to the distinct nature of these mechanisms—being more strongly driven by attention computations than by the explicit encoding itself—and to the specific characteristics of the datasets. The smoothing effects in Taxi and the wide-ranging fluctuations in TINA reduce the direct impact of

positional information on performance, highlighting a fundamental difference compared with the WinStat family, whose encodings are more robustly anchored in the statistical and semantic structure of the data.

We can analyze the differential behavior graphically by examining visualizations that compare the original and shuffled models across the evaluated datasets. In these representations, higher values indicate better performance, meaning that semantic and ordering information is being effectively injected into the model. These visual comparisons highlight how the encoding methods from the WinStat family impact the model's ability to capture meaningful temporal or structural dependencies across datasets.

First, the HPC dataset exhibits minimal differences between the original and shuffled Informer model, as illustrated in Figure 13, which is particularly noteworthy and indicates the limited ordinal information contributed by this approach. A similar behavior is observed for the other tested methods, TUPE and ROPE. In contrast, all four WinStat variants show a substantially larger differential, highlighting a strong ordinal contribution to the model that is effectively disrupted when input shuffling is applied.

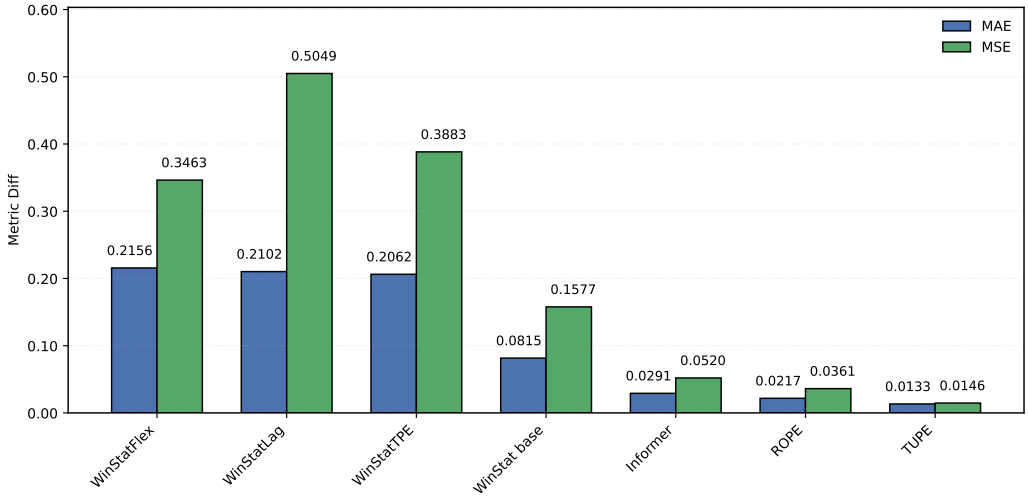

Figure 13. Comparison of original and shuffled models on the HPC dataset. Higher values indicate better incorporation of semantic and ordering information. A larger difference suggests stronger ordinal dependence of the encoding, which is disrupted when shuffling.

For ETTh1 and ETTh2, the observed results differ considerably. In ETTh1, as shown in Figure 14, input shuffling imposes a substantial penalty on Informer, although this effect is smaller in terms of MAE compared with the WinStat family. TUPE, in contrast, exhibits almost negligible degradation, while ROPE even shows a negative difference, indicating better performance under shuffling than in the original configuration, which suggests that this method is not well suited for this dataset. In ETTh2, illustrated in Figure 15, the behavior shifts: WinStatFlex now suffers the largest degradation (in contrast to WinStatLag in ETTh1). Most notably, ROPE exhibits a pronounced negative delta, reflecting a substantial improvement under shuffling. As these experiments were repeated multiple times, this is not an artifact, but rather a clear indication that this mechanism is ineffective for the ETTh2 dataset.

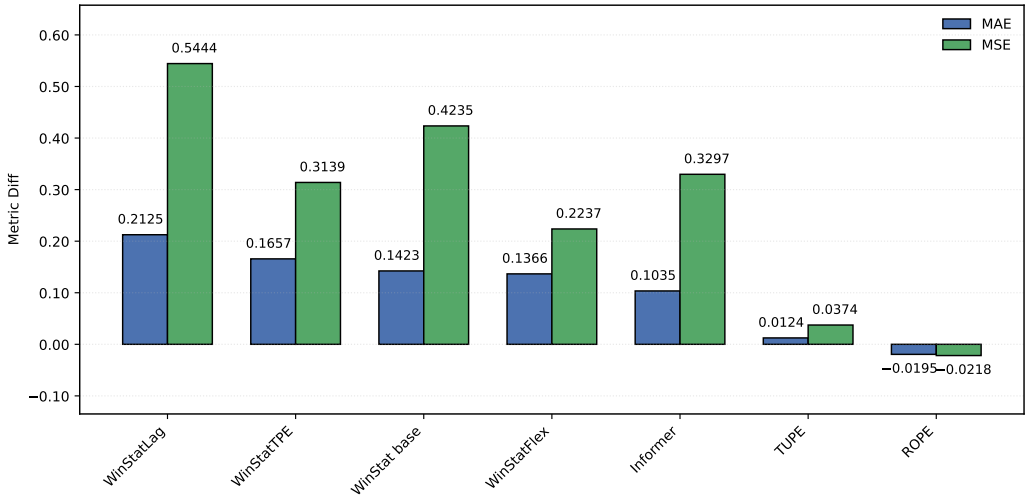

Figure 14. Comparison of original and shuffled models on the ETTh1 dataset. Higher values better reflect the use of semantic and ordering information. Greater differences reveal stronger ordinal dependence in the encoding, broken by shuffling.

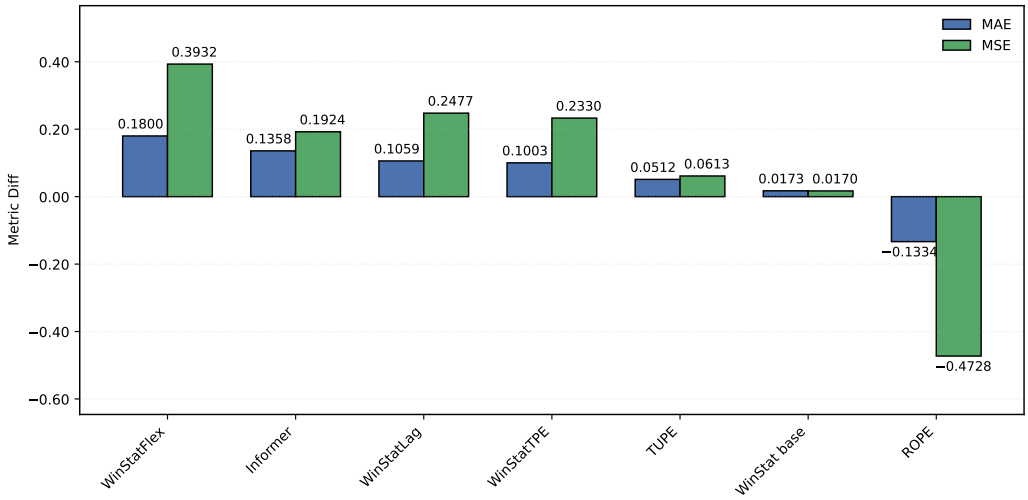

Figure 15. Comparison of original and shuffled models on the ETTh2 dataset. Higher values show more effective use of semantic and ordering information. A larger gap indicates higher ordinal dependence of the encoding, disrupted by shuffling.

The NYC dataset, shown in Figure 16, exhibits one of the largest discrepancies observed in this experiment, with the *WinStatFlex* encoding achieving a difference on the order of 10^2. This represents, without doubt, the most pronounced result across all benchmarks. Although similar trends can be identified in other models within the WinStat family (WinStat base and WinStatLag), the magnitude of the difference in this case is considerably greater, highlighting the distinct sensitivity of this dataset to the proposed encoding.

Finally, as illustrated in Figure 17, in the TINA dataset, several encodings—such as ROPE, TUPE, and the original Informer—exhibit negative differences, meaning that the shuffled models actually outperform their unshuffled counterparts. While this behavior is theoretically counterintuitive, it reinforces the notion that these positional encoding mechanisms are not well suited to the specific characteristics of this dataset. In contrast, all variants of the WinStat family display the expected degradation under shuffling—most notably WinStatTPE—thereby confirming their stronger capacity to encode meaningful positional and semantic information.

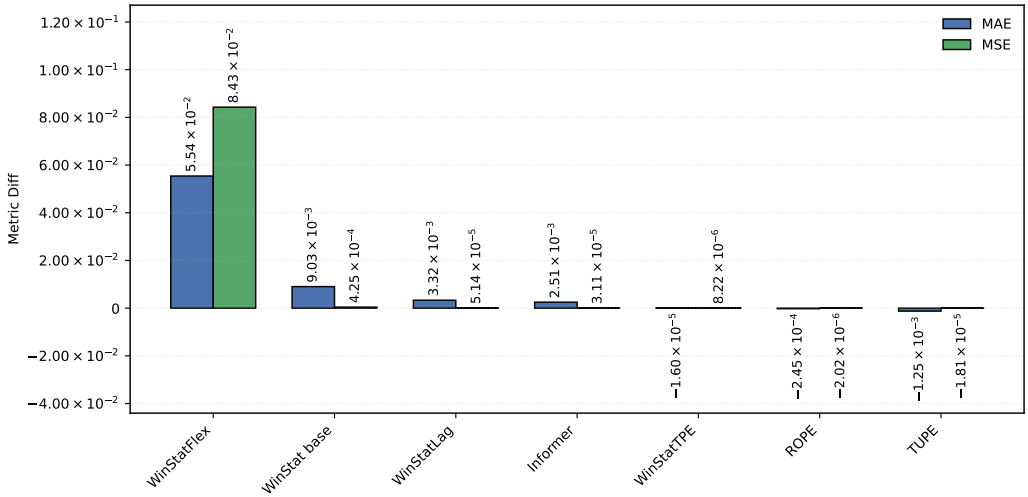

Figure 16. Comparison of original and shuffled models on the NYC dataset. Higher values denote stronger integration of semantic and ordering information. Greater differences imply stronger ordinal dependence in the encoding, which is lost when data are shuffled.

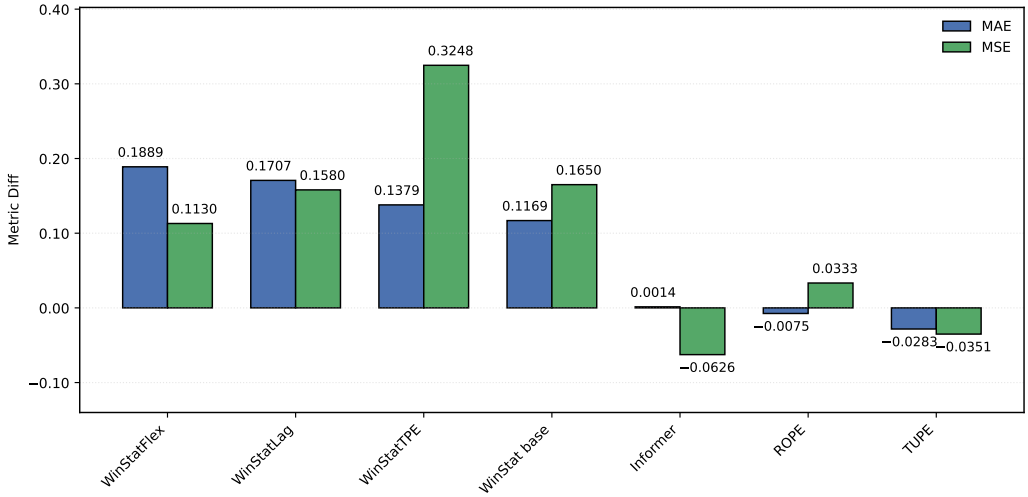

Figure 17. Comparison of original and shuffled models on the TINA dataset. Higher values indicate better incorporation of semantic and ordering information. A larger difference points to greater ordinal dependence of the encoding, which is broken by shuffling.

6. Conclusions and Future Work

This work examined positional information in Transformer-based time series forecasting and identified a central limitation of classical absolute encodings: they inject only global, index-based geometry and omit local temporal semantics. To address this gap, we introduced window-statistics positional signals and additive, learnable mixtures that combine heterogeneous encodings within a single embedding. These designs enrich each timestamp with local descriptive context (e.g., mean, dispersion, extremes, short-term differences) and allow the model to learn how much to trust absolute, learnable, and semantic components, thereby aligning positions with similar behavior across the series.

Across the evaluated benchmarks, WinStatFlex consistently improved predictive accuracy over the strong Transformer baseline (Informer). Observed gains typically ranged from 10% to 30%, with improvements reaching up to approximately 90% in specific high-precision regimes. Furthermore, the overall ranking identifies WinStatFlex as the top performer. The Bayesian significance tests confirm a clear statistical advantage over standard baselines, while establishing practical equivalence with RoPE and TUPE.

Shuffle-based stress tests further confirmed that the benefits come from the learned order-sensitive local/semantic structure. When decoder inputs were randomly permuted, the advantage of the proposed encodings collapsed, while conventional baselines like Informer, TUPE, and ROPE were less sensitive to permutation. From a practical standpoint, the only extra hyperparameter shared by the window variants is the window length. A dedicated pre-experiment on ETTh1/ETTh2 suggested a simple, reproducible heuristic to bound this choice without exhaustive search, striking a trade-off between overly generic statistics (large windows) and noisy, uninformative context (small windows). Furthermore, the additive mixture with trainable weights improved performance without requiring the practitioner to hand-tune component balances; softmax normalization provided stable training dynamics.

Several directions emerge naturally. First, extend the proposed positional mechanisms beyond forecasting to time series classification and anomaly detection, where localized patterns and semantics are equally critical (e.g., medical signals or industrial monitoring). Second, broaden empirical coverage with domains such as medicine, physics, and meteorology to distill cross-domain regularities and simplify the mixture. Third, explore hybridization with frequency-domain architectures (e.g., Fourier/Wavelet-based modules) to learn positional structure across temporal and spectral representations. Finally, investigate distributed and federated training to mitigate the higher training cost and enable privacy-preserving deployment across sites with sensitive data.

Author Contributions: Conceptualization, I.A.-M., D.G.-G. and J.L.; data curation, I.A.-M.; formal analysis, C.M.-M.; funding acquisition, J.L.; investigation, C.M.-M.; methodology, I.A.-M., D.G.-G. and J.L.; project administration, J.L.; software, C.M.-M.; supervision, D.G.-G.; validation, C.M.-M.; writing—original draft, C.M.-M., I.A.-M., D.G.-G. and J.L.; writing—review and editing, I.A.-M., D.G.-G. and J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This publication is part of the IAFER TSI-100927-2023-1 project, funded by the Recovery, Transformation, and Resilience Plan of the Next Generation of the EU through the Spanish Ministry of Digital Transformation and the Civil Service. This work was also supported by the Spanish Ministry of Science and Technology under grants PID2023-150070NB-I00 and TED2021-132702B-C21 funded by MICIU/AEI/10.13039/501100011033 and by "ERDF/EU" and the "European Union NextGenerationEU/PRTR", respectively. Ignacio Aguilera-Martos was supported by the Ministry of Science of Spain under the FPI programme PRE2021-100169.

Data Availability Statement: All data will be available upon request through the corresponding author. Most datasets used are open and appropriately cited in the manuscript. All implementation and experimental setup details are included in GitHub https://github.com/ari-dasci/S-WinStat (accessed on 8 October 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Palma, G.; Chengalipunath, E.S.J.; Rizzo, A. Time series forecasting for energy management: Neural circuit policies (ncps) vs. long short-term memory (lstm) networks. *Electronics* **2024**, *13*, 3641. [CrossRef]
2. Rezaei, Z.; Samghabadi, S.S.; Amini, M.A.; Shi, D.; Banad, Y.M. Predicting Climate Change: A Comparative Analysis of Time Series Models for CO_2 Concentrations and Temperature Anomalies. *Environ. Model. Softw.* **2025**, *192*, 106533. [CrossRef]
3. Buczyński, M.; Chlebus, M.; Kopczewska, K.; Zajenkowski, M. Financial time series models—comprehensive review of deep learning approaches and practical recommendations. *Eng. Proc.* **2023**, *39*, 79.
4. He, R.; Chiang, J.N. Simultaneous forecasting of vital sign trajectories in the ICU. *Sci. Rep.* **2025**, *15*, 14996. [CrossRef]
5. Cheng, C.H.; Tsai, M.C.; Cheng, Y.C. An intelligent time-series model for forecasting bus passengers based on smartcard data. *Appl. Sci.* **2022**, *12*, 4763. [CrossRef]
6. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are Transformers Effective for Time Series Forecasting? In Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 11121–11128.

7. Foumani, N.M.; Tan, C.W.; Webb, G.I.; Salehi, M. Improving position encoding of transformers for multivariate time series classification. *Data Min. Knowl. Discov.* **2024**, *38*, 22–48. [CrossRef]
8. Irani, H.; Metsis, V. Positional Encoding in Transformer-Based Time Series Models: A Survey. *arXiv* **2025**, arXiv:2502.12370. [CrossRef]
9. Ariyo, A.A.; Adewumi, A.; Ayo, C. Stock price prediction using the ARIMA model. In Proceedings of the 2014 IEEE UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, Cambridge, UK, 26–28 March 2014; pp. 106–112.
10. Liu, X.; Wang, W. Deep time series forecasting models: A comprehensive survey. *Mathematics* **2024**, *12*, 1504. [CrossRef]
11. Su, L.; Zuo, X.; Li, R.; Wang, X.; Zhao, H.; Huang, B. A systematic review for transformer-based long-term series forecasting. *Artif. Intell. Rev.* **2025**, *58*, 80. [CrossRef]
12. Oliveira, J.M.; Ramos, P. Evaluating the effectiveness of time series transformers for demand forecasting in retail. *Mathematics* **2024**, *12*, 2728. [CrossRef]
13. Shaw, P.; Uszkoreit, J.; Vaswani, A. Self-Attention with Relative Position Representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, US, 1–6 June 2018; Association for Computational Linguistics: Stroudsburg, PA, USA, 2018; Volume 2 (Short Papers), pp. 464–468.
14. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the NAACL-HLT, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
15. Ma, D.; Yuan, D.; Huang, M.; Dong, L. VGC-GAN: A multi-graph convolution adversarial network for stock price prediction. *Expert Syst. Appl.* **2024**, *236*, 121204. [CrossRef]
16. Wang, J.; Liao, L.; Zhong, K.; Deveci, M.; du Jardin, P.; Tan, J.; Kadry, S. MRRFGNN: Multi-relation reconstruction and fusion graph neural network for stock crash prediction. *Inf. Sci.* **2025**, *689*, 121507. [CrossRef]
17. Taylor, S.J.; Letham, B. Forecasting at scale. *Am. Stat.* **2018**, *72*, 37–45. [CrossRef]
18. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
19. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
20. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**, arXiv:1803.01271. [CrossRef]
21. Lim, B.; Zohren, S. Time-series forecasting with deep learning: A survey. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **2021**, *379*, 20200209. [CrossRef] [PubMed]
22. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
23. Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.X.; Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 5243–5253.
24. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual Event, 19–21 May 2021; Volume 35, pp. 11106–11115.
25. Xu, J.; Wu, H.; Wang, J.; Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Proceedings of the Advances in Neural Information Processing Systems, Virtual Event, 6–14 December 2021; Volume 34, pp. 22419–22430.
26. Liu, M.; Zeng, A.; Chen, M.; Xu, Q. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In Proceedings of the International Conference on Learning Representations, Virtual Event, 25–29 April 2022.
27. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; pp. 27268–27286.
28. Cirstea, R.G.; Bica, I.; van der Schaar, M. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting. In Proceedings of the International Joint Conference on Artificial Intelligence, Vienna, Austria, 23–29 July 2022; pp. 2054–2060.
29. Cao, F.; Yang, S.; Chen, Z.; Liu, Y.; Cui, L. Ister: Inverted Seasonal-Trend Decomposition Transformer for Explainable Multivariate Time Series Forecasting. *arXiv* **2024**, arXiv:2412.18798. [CrossRef]
30. Lan, Y.H. Gateformer: Advancing Multivariate Time Series Forecasting via Temporal and Variate-Wise Attention with Gated Representations. In Proceedings of the 1st ICML Workshop on Foundation Models for Structured Data, Vancouver, BC, Canada, 18 July 2025.
31. Ke, G.; He, D.; Liu, T.Y. Rethinking Positional Encoding in Language Pre-training. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 3–7 May 2021.

32. Su, J.; Lu, Y.; Pan, S.; Wen, B.; Liu, Y. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing* **2024**, *568*, 127063. [CrossRef]
33. Alioghli, A.A.; Okay, F.Y. Enhancing multivariate time-series anomaly detection with positional encoding mechanisms in transformers. *J. Supercomput.* **2025**, *81*, 282–306. [CrossRef]
34. Hebrail, G.; Berard, A. *Individual Household Electric Power Consumption*; UCI Machine Learning Repository: Irvine, CA, USA, 2006. [CrossRef]
35. NYC Open Data—Yellow Taxi Trip Records. In *City of New York Open Data*; NYC OpenData: New York, USA, 2015.
36. Aguilera-Martos, I.; López, D.; García-Barzana, M.; Luengo, J.; Herrera, F. Time-Series Industrial Anomaly (TINA) Dataset. DaSCI Institute Dataset. 2022. Available online: https://github.com/ari-dasci/OD-TINA (accessed on 8 October 2025).
37. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *arXiv* **2022**, arXiv:2211.14730.
38. Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; Long, M. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In Proceedings of the International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.