Rethinking Time Encoding via Learnable Transformation Functions

Xi Chen [1] **Yateng Tang** [2] **Jiarong Xu** [3] **Jiawei Zhang** [4] **Siwei Zhang** [1] **Sijia Peng** [1] **Xuehao Zheng** [2] **Yun Xiong** [1]

Abstract

Effectively modeling time information and incorporating it into applications or models involving chronologically occurring events is crucial. Real-world scenarios often involve diverse and complex time patterns, which pose significant challenges for time encoding methods. While previous methods focus on capturing time patterns, many rely on specific inductive biases, such as using trigonometric functions to model periodicity. This narrow focus on single-pattern modeling makes them less effective in handling the diversity and complexities of real-world time patterns. In this paper, we investigate to improve the existing commonly used time encoding methods and introduce **Learnable Transformation-based Generalized Time Encoding (LeTE)**. We propose using deep function learning techniques to parameterize non-linear transformations in time encoding, making them learnable and capable of modeling generalized time patterns, including diverse and complex temporal dynamics. By enabling learnable transformations, LeTE encompasses previous methods as specific cases and allows seamless integration into a wide range of tasks. Through extensive experiments across diverse domains, we demonstrate the versatility and effectiveness of LeTE.

1. Introduction

Time-related data are commonly observed in real-world applications, such as user transaction data in financial institutions (Kazemi et al., 2020; Lezmi & Xu, 2023), purchase behavior sequences in e-commerce (Kang & McAuley, 2018; Rossi et al., 2020; Skarding et al., 2021), and climate observations in weather forecasting (Murat et al., 2018; Neumann et al., 2024). Adopting time series models and dynamic graph models to handle time-related data are two common approaches (Wu et al., 2023; Yu et al., 2023). In both cases, effectively incorporating time information is crucial for making accurate predictions. To achieve this, existing research works typically employ time encoding methods to capture and represent time information, with the resulting time embedding being treated as an independent feature in time series forecasting and dynamic graph representation learning models.

Early studies represent time by using hand-crafted temporal features designed specifically for downstream tasks (Choi et al., 2016; Baytas et al., 2017; Kwon et al., 2018). A prominent example is the time encoding method illustrated in Figure 1 (a), which is widely used in existing time series processing work (Wang et al., 2023; Wu et al., 2023). Such methods typically involve manually splitting timestamps into components (e.g., month, day, etc.), assigning a embedding to each component, and adding these embeddings to form the final time embedding. However, these methods are resource-intensive and often rely on domain expertise, which may limit their abilities to capture only specific pre-defined time patterns (Kazemi et al., 2019).

With the rapid development of attention mechanisms, which offer advantages such as better handling of long-range dependencies and adaptive weighting of time-related information, subsequent research on time series and dynamic graphs has increasingly leveraged these mechanisms (Xu et al., 2020; Yu et al., 2023; Liu et al., 2023). To better model time and ensure compatibility between time encoding methods and self-attention, Functional Time Encoding (FTE) methods were proposed, with two representative works: Functional Time Representation (Xu et al., 2019) and Time2Vec (Kazemi et al., 2019), as illustrated in Figure 1 (b). Nearly all subsequent dynamic graph representation learning research employs these methods to encode time (Yu et al., 2023). These techniques transform time input into multi-dimensional time embeddings by applying multiple linear transformations followed by pre-defined non-linear transformation functions. Due to their reliance on pre-defined non-linear transformations—such as trigonometric functions to capture periodic patterns—these methods are inherently limited to capturing fixed, specific time patterns.

[1]Shanghai Key Laboratory of Data Science, College of Computer Science and Artificial Intelligence, Fudan University, Shanghai, China [2]Tencent Weixin Group, Shenzhen, China [3]School of Management, Fudan University, Shanghai, China [4]IFM Lab, University of California, Davis, CA, USA. Correspondence to: Yun Xiong <yunx@fudan.edu.cn>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

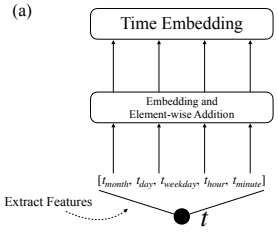
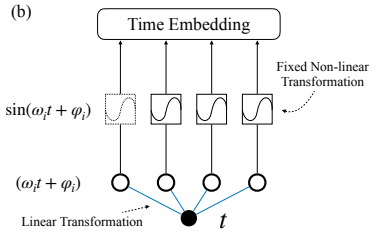
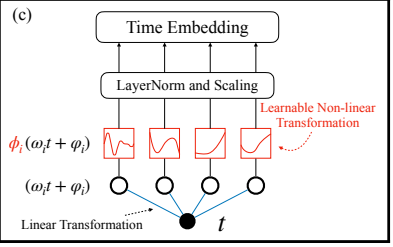
Time Encoding	Hand-Crafted Time Encoding (HCTE)	Functional Time Encoding (FTE)	**LeTE (Ours)**
Patterns Capturing	Pre-defined	Pre-defined	Diverse
Formula	$\mathbf{TE}(t)[i] = E_{\text{month}}(t_{\text{month}})[i] + E_{\text{day}}(t_{\text{day}})[i] + E_{\text{weekday}}(t_{\text{weekday}})[i] + E_{\text{hour}}(t_{\text{hour}})[i] + E_{\text{min}}(t_{\text{min}})[i]$	$\mathbf{TE}(t)[i] = \begin{cases} \sin(\omega_i t + \varphi_i) \text{ or } \omega_i t + \varphi_i, & \text{if } i = 1, \\ \sin(\omega_i t + \varphi_i), & \text{if } 2 \le i \le d \end{cases}$	$\mathbf{TE}(t)[i] = s_i \cdot \text{LayerNorm}\left(\phi_i(\omega_i t + \varphi_i)\right)$
Architecture			

Figure 1. A comparison of previous time encoding methods and proposed LeTE.

As a result, they often struggle to represent more complex, non-linear temporal dynamics (Kazemi et al., 2019; Wu et al., 2023) and require additional dimensions to account for diverse periodic components (Xu et al., 2020; Rossi et al., 2020; Zeng et al., 2024). Furthermore, these encoding methods frequently lack the capacity to effectively model non-periodic patterns, such as trends, irregularities.

We observe that previous time encoding methods—whether hand-crafted or functional—primarily introduce a strong inductive bias rooted in the periodic nature of human behavior and natural phenomena (Li et al., 2017; Xu et al., 2019; Kazemi et al., 2019). They mainly focus on capturing predefined periodic patterns, often struggling to capture more complex ones, such as non-periodic and mixed patterns.

However, in real-world scenarios, data often exhibits a complex interplay of mixed patterns, making accurate modeling more challenging. For instance, in financial risk control, periodic patterns—such as daily transaction peaks, weekly spending habits, and seasonal trends around holidays or salary payments—offer valuable insights into predictable behaviors. On the other hand, non-periodic events, such as sudden spikes in transactions caused by market fluctuations, regulatory changes, or potential fraudulent activities, necessitate flexible and adaptive modeling techniques. Moreover, different patterns often coexist. For example, fraudsters may blend regular periodic transactions with abnormal non-periodic activities to evade the detection by regulators. To illustrate the presence of complex mixed time patterns in real-world data, we conduct extensive investigations in this paper, with partial results in Appendix G.1.

This motivates us to rethink the design of time encodings. We argue that an effective time encoding method should adhere to a key principle to enable comprehensive and accurate analysis: Capacity for modeling diverse and complex time patterns, i.e., the method should be capable of capturing a wide range of time patterns, including periodic, non-periodic, and mixed patterns.

To better encode time information and simultaneously capture diverse time patterns, we propose **Learnable Transformation-based Generalized Time Encoding**, abbreviated as **LeTE**—a simple yet effective time encoding method. Instead of hand-crafting time encoding or relying on pre-defined non-linear transformations, we draw inspiration from deep function learning, which is known for its generalizability, interpretability, and reusability (Zhang, 2024; Liu et al., 2025), and propose to use learnable non-linear transformations for time encoding. Specifically, we parameterize non-linear transformation functions using techniques derived from deep function learning. This parameterization makes the transformations learnable and jointly optimizable with the model's parameters under supervision from downstream tasks, allowing them to flexibly adapt to both linear and arbitrary non-linear forms. With learnable transformations, LeTE adaptively models time information, enabling different dimensions of time encoding to capture complex time patterns—such as irregular trends, abrupt changes, and overlapping periodicities—that are commonly encountered in real-world scenarios and beyond the capabilities of previous methods. This generalization also allows our method to encompass previous approaches as specific cases. An illustration of LeTE is in Figure 1 (c).

LeTE also offers following advantages (cf. Section 3.2). Since time can be measured on different scales, its representation should be invariant to time rescaling (Kazemi et al., 2019). We prove that LeTE satisfies this property (cf. Appendix C.3). Furthermore, we prove that LeTE is a generalized version of previous methods and can integrate seamlessly with various models (cf. Section 3.2). By employing an interpretable deep function learning approach, LeTE achieves a high degree of interpretability (cf. Appendix G.3). Additionally, experimental results demonstrate that LeTE achieves superior results with fewer dimensions than previous time encodings, as the learnable transformations capture part of the complexity that would otherwise require higher-dimensional embeddings (cf. Section 4.5).

We highlight our contributions as follows:

- We reinvestigate the design of the existing time encoding methods, highlighting their limitations in handling real-world data and propose LeTE, a generalized time encoding method that allows the entire encoding process, including both linear and non-linear transformations, fully parameterized and learnable.
- LeTE has the capacity to model diverse and complex time patterns, and it offers additional benefits, including invariance to time rescaling, plug-and-play functionality, enhanced interpretability and improved dimensional efficiency.
- Through extensive experiments across diverse domains—including event-based image classification, time series forecasting, dynamic graph representation learning and real-world applications-we demonstrate the effectiveness and versatility of LeTE.

2. Preliminaries

2.1. Functional Time Encodings

Functional Time Encoding (FTE) methods can be viewed as feature mappings from 1-dimensional input time to a high-dimensional time embedding: $\Phi : t \in \mathbb{R}^1 \rightarrow \mathbf{TE} \in \mathbb{R}^d$, where $t \in [0, t_{\text{max}}]$ is from the value range bounded by t_{max}. Two representative works of FTE are Functional Time Representation (FTR) (Xu et al., 2019) and Time2Vec (T2V) (Kazemi et al., 2019). Although these methods construct time encodings from different perspectives, they are mathematically nearly identical (the only difference is that a separate dimension that undergoes only a linear transformation is used by T2V to capture non-periodic patterns). We state the following proposition, with details of the two methods and the proof provided in Appendix C.1.

Proposition 2.1. *Mathematically, with selected values for ω_i and φ_i, the aforementioned FTR and T2V can be unified into the following forms:*

Including the first dimension:

$$\boldsymbol{TE}(t)[i] = \begin{cases} \sin(\omega_i t + \varphi_i) \text{ or } \omega_i t + \varphi_i, & \text{if } i = 1, \\ \sin(\omega_i t + \varphi_i), & \text{if } 2 \leq i \leq d. \end{cases} \tag{1}$$

Or excluding the first dimension:

$$\boldsymbol{TE}(t)[i] = \sin(\omega_i t + \varphi_i), \tag{2}$$

or

$$\boldsymbol{TE}(t) = [\sin(\omega_1 t + \varphi_1), \cdots, \sin(\omega_d t + \varphi_d)] \tag{3}$$

For simplicity, we use Functional Time Encoding (FTE) to refer to both the FTR and T2V throughout the paper.

2.2. Deep Function Learning

Deep Function Learning refers to the approach of learning target functions by optimizing parameterized functions, such as polynomials, sinusoidal functions, or splines. This method leverages the flexibility of parameterized functions and optimizes their parameters using deep learning frameworks to approximate complex functions (Zhang, 2024).

Fourier Series Expansion: The Fourier series expresses a target function $f(x)$ as a combination of sine and cosine functions:

$$f(x) = a_0 + \sum_{n=1}^{N} (a_n \cos(n\omega x) + b_n \sin(n\omega x)) \tag{4}$$

Here, a_n and b_n are learnable coefficients, and ω is the fundamental frequency. By optimizing a_0, a_n and b_n through a learning process, the function $f(x)$ can approximate complex patterns (cf. Appendix B.1 for details). Unlike fixed sine functions, parameterized functions adjust their amplitude, frequency, and phase through downstream supervisory signals, enabling them to effectively model a wider range of patterns.

Spline Functions: Spline functions approximate a target function $f(x)$ using a sum of piecewise polynomial basis functions:

$$f(x) = \sum_{i=1}^{n} c_i B_i(x) \tag{5}$$

Here, c_i are control points, and $B_i(t)$ are the basis functions (e.g., B-splines). By learning and optimizing the control points, knot positions, and weights (cf. Appendix B.2 for details), splines provide a smooth and accurate representation of diverse functions. Their piecewise and localized structure makes them highly adaptable for complex function modeling.

3. Methods

3.1. LeTE

To address the limitations of previous time encodings—specifically, their restricted capacity to model fixed or pre-defined time patterns—we propose **Learnable Transformation-based Generalized Time Encoding** (referred to as **LeTE**). To capture diverse and complex patterns in time-related data, we propose techniques that make non-linear transformations learnable. This approach allows the model to dynamically adapt its transformations, enabling more precise representations of time patterns. To achieve this, we employ two distinct approaches for constructing learnable transformation functions: Fourier series expansion and Spline functions. Both methods share the ability to effectively capture and model complex, non-linear temporal patterns while maintaining flexibility in handling various

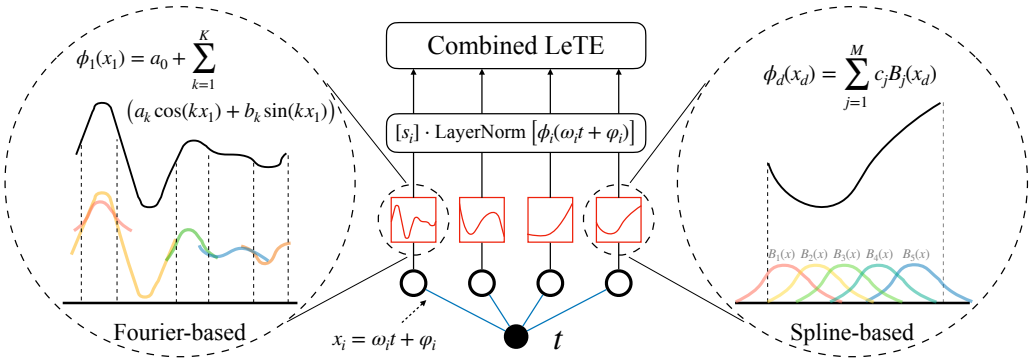


Figure 2. An illustration of Combined LeTE: the first dimension is parameterized by Fourier series expansion and the last dimension is parameterized by B-Splines.

time dynamics. Based on these methods, we propose three variations of LeTE. First, we construct the learnable transformation functions using these two approaches, categorizing them as Fourier-based LeTE and Spline-based LeTE according to their respective construction methods. We then integrate these two variations to develop a more generalized version, referred to as Combined LeTE, which leverages the strengths of both approaches.

For a scalar timestamp input t, LeTE for t, denoted as $\textbf{LeTE}(t)$, is a d-dimensional time embedding vector:

$$\textbf{LeTE}(t)[i] = \phi_i(\omega_i t + \varphi_i), \tag{6}$$

where ω_i and φ_i are learnable parameters, and ϕ_i are learnable functions that can be parameterized by either Fourier series expansion or B-splines.

Fourier-based LeTE: This method assumes that ϕ_i are parameterized by Fourier series expansion:

$$\phi_i(x) = a_0 + \sum_{k=1}^{K} \left(a_k \cos(kx) + b_k \sin(kx)\right), \tag{7}$$

where a_0, a_k, and b_k are the parameters to be learned, and K represents the number of terms in the expansion. LeTE can then be expressed as:

$$\begin{aligned}\textbf{LeTE}(t)[i] = a_{i,0} + \sum_{k=1}^{K} \Big(&a_{i,k} \cos(k(\omega_i t + \varphi_i)) \\ &+ b_{i,k} \sin(k(\omega_i t + \varphi_i))\Big)\end{aligned} \tag{8}$$

Spline-based LeTE: This method assumes that ϕ_i are parameterized by B-splines:

$$\phi_i(x) = \sum_{j=1}^{M} c_j B_j(x), \tag{9}$$

where M is the number of B-spline basis functions, and $B_j(x)$ is the j-th B-spline basis function. LeTE can then be expressed as:

$$\textbf{LeTE}(t)[i] = \sum_{j=1}^{M} c_{i,j} B_j(\omega_i t + \varphi_i). \tag{10}$$

Comparing these two approaches for constructing ϕ_i, the Fourier series expansion in LeTE enforces periodicity in ϕ_i, while the B-spline approach provides flexibility to model more complex ϕ_i functions, including both periodic and non-periodic patterns.

Recall that we neglect the first dimension of the linear transformation of time in Equations (2) and (3), which models the non-periodic patterns for Time2Vec. However, in LeTE , by allowing ϕ_i to be learnable, different ϕ_i at different dimensions can capture more complex non-periodic patterns based on the supervision signals from downstream tasks.

Combined LeTE: To enhance the capability of the time encoding to capture diverse and complex time patterns and to build a more generalized version of LeTE, we further propose a straightforward extension: applying Fourier-based LeTE to a portion of the time encoding dimensions and Spline-based LeTE to the remaining dimensions. The proportion of Fourier-based LeTE and Spline-based LeTE used can be controlled by a hyperparameter p. To address potential differences in the output scales of Fourier-based and Spline-based LeTE, we introduce a Layer Normalization layer followed by a learnable scaling weight for the time encoding. For a d-dimensional LeTE, the time embedding is formulated as:

$$\textbf{LeTE}(t)[i] = s_i \cdot \text{LayerNorm}\left(\phi_i(\omega_i t + \varphi_i)\right), \tag{11}$$

where $[s_i]$ is a d-length learnable scaling weight vector, and

$$\phi_i(x) = \begin{cases} \text{Equation(7)}, & \text{if } i \leq \lfloor p \cdot d \rfloor, \\ \text{Equation(9)}, & \text{if } i > \lfloor p \cdot d \rfloor. \end{cases} \tag{12}$$

When $p = 1$, the method corresponds to Fourier-based LeTE, and when $p = 0$, it corresponds to Spline-based LeTE. For the remainder of this paper, unless otherwise specified, **LeTE** will refer to the Combined LeTE, where p is set to 0.5. An illustration of LeTE is shown in Figure 2, and the implementation details are provided in Appendix D.

3.2. Properties of LeTE

In this subsection, we present the properties of our method from the perspective of theoretical analysis. Specifically, we discuss the strengths of Fourier-based LeTE and Spline-based LeTE individually. Since Combined LeTE integrates these two variations, it naturally inherits their respective properties.

Generalizability: Compared with previous methods, which can only capture pre-defined time patterns—usually periodic ones—our method offers greater generalizability, enabling it to capture a wider range of diverse and complex patterns, including periodic, non-periodic and mixed ones. Naturally, Fourier-based LeTE (as formulated in Equation (8), the ϕ_i functions are parameterized by Fourier series expansion) can model periodicity, as it resembles a Fourier series expansion with weighted sums of sine and cosine terms at different frequencies and phases. By learning appropriate values for ω_i and incorporating different harmonics k, the function can approximate complex periodic patterns and capture repeating structures in time. The learnable parameters ω_i and φ_i enable the model to adapt to various periodic characteristics in the data. Although the Fourier series is inherently periodic, non-periodic patterns can also be modeled: the learnable parameters $a_{i,0}$, $a_{i,k}$, $b_{i,k}$, ω_i, and φ_i provide flexibility to approximate non-periodic behaviors. By using very small or large values for ω_i, the model can fit signals with long or slow-varying cycles, effectively creating non-repeating patterns over a finite interval. Additionally, the combination of learned frequencies, phases, and amplitudes can produce complex patterns that do not repeat over the observed range, thereby approximating non-periodic signals. For similar reasons, and given the generality of functions formed by splines, Spline-based LeTE (Equation (10), the ϕ_i are parameterized by B-splines) can also model both periodic and non-periodic patterns. Naturally, through multi-dimensional encoding, both Fourier-based and Spline-based LeTE are capable of capturing mixed time patterns. Although Fourier-based LeTE can capture non-periodic patterns, its inherent periodicity makes it particularly effective at modeling the periodicity of time. Conversely, while Spline-based LeTE is also capable of capturing periodic patterns, it exhibits a stronger ability to model non-periodic patterns. Therefore, by combining Fourier-based LeTE and Spline-based LeTE, the resulting Combined LeTE achieves enhanced capability to capture diverse patterns. Intuitively, the previous FTEs are special cases of LeTE; we present the following proposition with its corresponding proof to demonstrate this.

Proposition 3.1. *For an arbitrary input t, the network can learn a set of parameters such that LeTE can replicate the effects of previous time encodings, making previous time encodings specific cases of LeTE.*

Proof. For Equations (8) and (10), we only need to find a set of coefficients for Equations (7) and (9) to approximate the sine function, respectively. By selecting K =1, $a_{i,0} = 0$, $a_{i,1} = 0$, and $b_{i,1} = 1$, Equation (8) becomes:

$$\begin{aligned}\textbf{LeTE}(t)[i] &= 0 + \Big(0 \cdot \cos\big(1 \cdot (\omega_i t + \varphi_i)\big) \\ &\quad + 1 \cdot \sin\big(1 \cdot (\omega_i t + \varphi_i)\big)\Big) \\ &= \sin(\omega_i t + \varphi_i).\end{aligned} \tag{13}$$

Thus, $\sin(\omega_i t + \varphi_i)$ is indeed a special case of the more general formula in Equation (8). The proof continues in Appendix C.2, which demonstrates that $\sin(\omega_i t + \varphi_i)$ is also a special case of Equation (10). □

Moreover, since Xu et al. claim that absolute position encoding is a special case of functional time representation (Xu et al., 2019; 2020), it is straightforward to see that absolute position encoding is also a special case of our LeTE.

Invariance to Time Rescaling: Since time can be represented on various scales (such as days, hours, or seconds), a key characteristic of a time representation is its invariance to rescaling (Kazemi et al., 2019; Tallec & Ollivier, 2018). Similar to FTE, our proposed time encoding is also invariant to time rescaling, as shown in the following proposition with proof provided in Appendix C.3.

Proposition 3.2. *LeTE is invariant to time rescaling.*

Plug-and-Play: LeTE is designed in a plug-and-play manner, ensuring seamless compatibility with various models and architectures. By producing a d-dimensional time embedding vector similar to previous time encodings, it can be easily integrated without requiring significant modifications to existing frameworks. Unlike prior methods that rely on fixed non-linear transformation functions, LeTE employs parameterized and learnable transformations, enabling it to capture additional information and complexity. This design allows LeTE to achieve superior performance even with lower-dimensional time encodings compared to traditional methods (see Section 4.5 for experimental results).

Interpretability: Previous time encodings exhibit natural interpretability because they use a fixed non-linear activation function, i.e., the sine function, which has obvious periodicity. Our proposed time encoding uses a learnable non-linear transformation function. However, by examining

the learned parameters, we can reconstruct these transformation functions, enabling our method to also achieve strong interpretability. A visualization of our proposed time encoding is provided in Appendix G.3.

3.3. Use of Time Encoding

In time series forecasting research, time embeddings calculated by time encoding modules are usually directly added to feature embeddings and fed into the attention mechanism or Transformer (Vaswani et al., 2017). As a result, they typically share the same dimensions as the feature embeddings:

$$\mathbf{x} = \textbf{TokenEncode}(\mathbf{x}) + \textbf{TE}(t) \in \mathbb{R}^d. \tag{14}$$

Here, $\mathbf{x}$ represents the input, **TokenEncode** denotes a token encoding function, **TE** denotes the time encoding, and d represents the dimension of both feature embeddings and time embeddings.

In dynamic graph representation learning research, time embeddings are usually concatenated with node features and edge features as part of the input. This allows for more flexibility in the choice of time embedding dimensions:

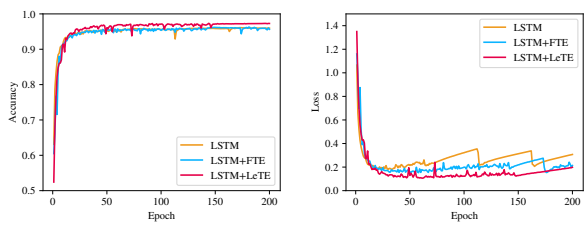
$$\mathbf{x} = \textbf{Node Features}\|\textbf{Edge Features}\|\textbf{TE}(t) \in \mathbb{R}^{d_n+d_e+d}. \tag{15}$$

Here, $\|$ denotes the concatenation operation, while d_n, d_e, and d represent the dimensions of node features, edge features, and time embeddings, respectively.

4. Experiments

4.1. Time as the Only Input

To evaluate the performance of the time encoding method in scenarios where the only input is time, and to compare different time representations while minimizing the influence of extraneous variables, we follow (Kazemi et al., 2019) and create a sequential (event-based) MNIST dataset (Fatahi et al., 2016; Campos et al., 2018; Bellec et al., 2018) and conduct image classification task (more details are shown in Appendix E.1).



(a) Testing Accuracy (b) Testing Loss

Figure 3. Time as the only input: Comparison of time encodings on sequential MNIST.

We then apply an LSTM with a 32-dimensional learnable embedding for the time input and compare it to models where the FTE or our LeTE is used for encoding time. The results, shown in Figure 3, indicate that the FTE achieves testing accuracy comparable to that of the LSTM without any time encoding method applied. However, our LeTE achieves significantly higher image classification accuracy. This simple experiment demonstrates that LeTE can efficiently encode time information for models. Next, we present experiments applying LeTE to time series tasks, dynamic graph tasks, and real-world applications.

4.2. Experiments on Time Series Tasks

For time series forecasting tasks, we select 5 baseline models where we can directly replace the time encoding methods with LeTE: vanilla Transformer (Vaswani et al., 2017), Pyraformer (Liu et al., 2021), Non-stationary Transformer (Liu et al., 2022), MICN (Wang et al., 2023), and TimesNet (Wu et al., 2023). We conduct long-term forecasting tasks on these baseline models using 4 datasets: ETT, Weather, Exchange (Lai et al., 2018), and Electricity, covering various real-world scenarios. Implementation details and introductions to baselines and datasets are provided in Appendix E.2. We apply LeTE and adjust the hyperparameter p in all experiments to capture more comprehensive time information. We report the results in the multivariate setting, as shown in Table 1 and 5. Because the time embeddings need to be added to the feature embeddings, they must have the same dimensions as the feature embeddings. Baseline models commonly apply hand-crafted time encoding (HCTE) with Date-Time Format inputs (e.g., ISO 8601 format, YYYY-MM-DD HH:mm:ss). Since our method, like FTE, takes UNIX timestamps as input, we include FTE in our experiments for comparison. In our approach, we transform the Date-Time Format timestamps into UNIX timestamps, encode them with our proposed LeTE , and feed the resulting time embeddings into the models in the same manner as the baselines. In this context, the input consists of absolute timestamps, and the time encoding can therefore be regarded as an *absolute time encoding*.

From the experimental results, we observe that: (1) When applying different time encoding methods to baseline models for time series forecasting, LeTE outperforms the benchmark in most cases, achieving an average win rate of 98% on MAE (Mean Absolute Error) and 95% on MSE (Mean Squared Error) across all baseline, dataset, and prediction length combinations, highlighting the effectiveness of the proposed time encoding. This demonstrates that our method can be seamlessly transferred to time series models, reliably achieving strong performance. (2) The improvements on baselines are considerable. For instance, applying LeTE to the Transformer model reduces the average MAE and MSE across all datasets by 25.1% and 46.5%, respectively. This

Table 1. Time series prediction: multivariate long-term forecasting task. The past sequence length is set to 96, while the prediction lengths are {96, 192, 336, 720}. The results are reported in terms of MAE, where lower values indicate better performance. HCTE (Hand-Crafted Time Encoding) is a method widely adopted in time series research. FTE stands for Functional Time Encoding. The win rate represents the percentage of cases where LeTE outperforms the HCTE. The best results for each baseline, dataset and prediction length combinations are in **bold**. ETT consists of 4 subsets. Here, we present the average results across these subsets, with the full results provided in Table 7.

MAE		Transformer			Pyraformer			NS Trans.			MINC			TimesNet			Win
TE		HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE	Rate
ETT	96	0.797	0.803	**0.550**	0.642	0.720	**0.583**	0.405	0.435	**0.377**	0.405	0.367	**0.350**	0.355	0.362	**0.352**	95%
	192	1.139	0.916	**0.712**	0.899	0.924	**0.738**	0.445	0.478	**0.414**	0.445	0.423	**0.395**	**0.385**	0.400	0.388	
	336	1.119	0.938	**0.821**	1.043	1.038	**0.863**	0.478	0.539	**0.449**	0.478	0.486	**0.448**	0.421	0.424	**0.413**	
	720	1.070	1.146	**0.878**	1.196	1.189	**0.959**	0.526	0.557	**0.490**	0.526	0.561	**0.505**	0.455	0.455	**0.429**	
Electricity	96	0.357	0.375	**0.347**	0.376	0.375	**0.365**	0.273	0.275	**0.265**	0.269	0.263	**0.254**	0.272	0.272	**0.267**	100%
	192	0.367	0.402	**0.353**	0.391	0.385	**0.372**	0.286	0.292	**0.278**	0.285	0.278	**0.271**	0.289	0.281	**0.277**	
	336	0.370	0.425	**0.357**	0.399	0.401	**0.369**	0.304	0.300	**0.293**	0.304	0.298	**0.294**	0.300	0.308	**0.291**	
	720	0.374	0.453	**0.363**	0.390	0.394	**0.380**	0.321	0.330	**0.317**	0.321	0.330	**0.317**	0.320	0.363	**0.316**	
Exchange	96	0.575	0.705	**0.547**	**0.570**	0.641	0.624	0.237	0.261	**0.237**	0.235	0.233	**0.203**	0.234	0.237	**0.230**	95%
	192	0.747	0.791	**0.744**	0.803	0.815	**0.786**	0.335	0.369	**0.319**	0.316	0.332	**0.289**	0.344	0.339	**0.332**	
	336	0.945	1.123	**0.879**	0.903	0.991	**0.859**	0.476	0.501	**0.439**	0.407	0.472	**0.402**	0.448	0.472	**0.446**	
	720	1.329	1.147	**1.066**	1.075	1.046	**0.938**	0.769	0.901	**0.612**	0.658	0.710	**0.622**	0.746	0.756	**0.751**	
Weather	96	0.422	0.257	**0.245**	0.303	0.296	**0.267**	0.223	0.222	**0.221**	0.229	0.258	**0.225**	0.220	0.221	**0.215**	100%
	192	0.523	0.308	**0.295**	0.336	0.317	**0.311**	0.285	0.271	**0.260**	0.281	0.306	**0.261**	0.261	0.263	**0.253**	
	336	0.607	**0.355**	0.365	0.403	0.377	**0.349**	0.338	0.321	**0.308**	0.331	0.335	**0.295**	0.306	0.302	**0.299**	
	720	0.690	0.459	**0.429**	0.434	0.417	**0.415**	0.410	0.357	**0.349**	0.356	0.387	**0.339**	0.359	0.350	**0.348**	
Win Rate		100%			94%			100%			100%			94%			98%

Table 2. Dynamic graph link prediction task: The results are reported in AP, where higher values indicate better performance. The better results are in **bold**. Here, we present the top-performing results across variations of LeTE, with the full results provided in Table 9. FTE represents Functional Time Encoding which is commonly used in dynamic graph research.

	AP	Wikipedia		Reddit		MOOC		LastFM	
	TE	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive
TGAT	FTE	96.95 ± 0.24	96.33 ± 0.26	98.53 ± 0.04	97.01 ± 0.05	85.34 ± 0.19	84.94 ± 0.04	72.73 ± 0.11	77.78 ± 0.13
	LeTE	**97.82 ± 0.09**	**97.34 ± 0.08**	**98.56 ± 0.01**	**97.05 ± 0.06**	**88.31 ± 0.10**	**88.37 ± 0.12**	**76.22 ± 0.25**	**81.32 ± 0.14**
TGN	FTE	98.45 ± 0.06	97.83 ± 0.04	98.63 ± 0.06	97.50 ± 0.07	89.15 ± 1.60	89.04 ± 1.17	77.07 ± 3.97	81.45 ± 4.29
	LeTE	**98.78 ± 0.07**	**98.19 ± 0.09**	**98.74 ± 0.00**	**97.65 ± 0.04**	**91.41 ± 0.55**	**90.87 ± 0.83**	**83.64 ± 2.00**	**87.55 ± 1.88**
TCL	FTE	96.47 ± 0.16	96.22 ± 0.17	97.53 ± 0.02	94.09 ± 0.07	82.38 ± 0.24	80.60 ± 0.22	67.27 ± 2.16	73.53 ± 1.66
	LeTE	**98.19 ± 0.04**	**97.89 ± 0.03**	**97.78 ± 0.03**	**94.99 ± 0.07**	**84.24 ± 0.10**	**82.72 ± 0.12**	**76.08 ± 0.79**	**80.68 ± 0.70**
DyG-Former	FTE	99.03 ± 0.02	98.59 ± 0.03	99.22 ± 0.01	98.84 ± 0.02	87.52 ± 0.49	86.96 ± 0.43	93.00 ± 0.12	94.23 ± 0.09
	LeTE	**99.13 ± 0.02**	**98.73 ± 0.00**	**99.24 ± 0.01**	**98.86 ± 0.01**	**88.70 ± 0.21**	**88.39 ± 0.15**	**93.64 ± 0.10**	**94.69 ± 0.12**

illustrates that our method can be applied to various time series forecasting models, consistently achieving strong performance. (3) FTE can occasionally outperform benchmarks; however, it also fails in many cases, whereas LeTE steadily outperforms benchmarks in such situations. This demonstrates our method's capability to model diverse time patterns, including periodic, non-periodic, and mixed, highlighting its generalizability across different models and data.

4.3. Experiments on Dynamic Graph Tasks

FTEs are widely used in dynamic graph representation learning models. Representative works include TGAT (Xu et al., 2020), TGN (Rossi et al., 2020), TCL (Wang et al., 2021), and DyGFormer (Yu et al., 2023). Thus, we apply these models as baselines and replace their time encodings with our LeTE. We conduct link prediction experiments on 4 real-world datasets: Wikipedia, Reddit, MOOC, and LastFM (Kumar et al., 2019). The details of the implementation, baseline methods and datasets are in Appendix E.3. The results are reported in both transductive and inductive settings, as shown in Tables 2 and 6. In this context, the time encoding module takes the relative time difference between the current edge and the most recent previous edge, and can therefore be regarded as a *relative time encoding*.

As shown in the experimental results, our proposed LeTE surpasses the benchmark results on all combinations of baselines and datasets, regardless of transductive or inductive settings, achieving state-of-the-art (SOTA) performance. This strongly demonstrates the effectiveness of our proposed time encoding and highlights its potential for improving the representation learning of dynamic graphs. The dimensions for the main experiments are set to 100, following the original settings in previous work (Rossi et al., 2020; Yu et al., 2023). However, since time embeddings are concatenated with node and edge features in dynamic graph models, this provides significant flexibility in setting their dimensions.

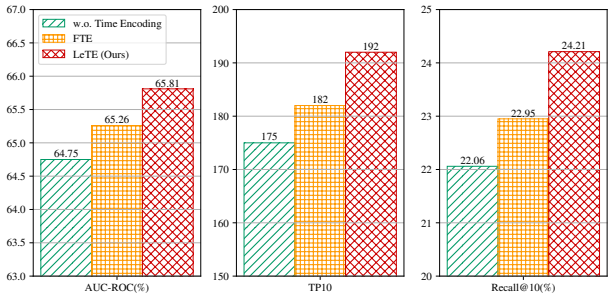
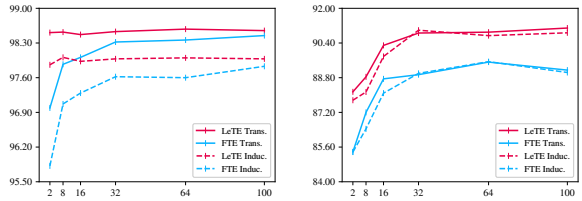


Figure 4. Results evaluated by AUC-ROC, TP10 and Recall@10 on real business datasets.



(a) Wikipedia/TGN (b) MOOC/TGN

Figure 5. Average Precision results comparing different dimensions of the FTE and Spline-based LeTE on Wikipedia/TGN and MOOC/TGN.

We compare the effects of time embeddings with different dimensions in Section 4.5. A comparison and analysis of the variations of LeTE are also provided in Appendix G.2.

4.4. Experiments on Real-World Application

Time information plays a crucial role in many real-world fields. We apply our proposed LeTE in a real-world financial risk control scenario to demonstrate its effectiveness in practical applications. In financial risk control, a user's historical transaction data is typically used to predict their credit risk, which can be framed as a classification problem based on historical transaction information. However, in this scenario, users' transaction behaviors often exhibit a combination of complex periodic and non-periodic patterns. For instance, users may regularly receive salary deposits and purchase daily necessities, whereas peer-to-peer transfers may lack strong periodicity. Using financial risk control data from Tencent Mobile Payment [1], we conduct comparative experiments without time information, with the FTE, and with LeTE to encode time information. The backbone model treats the time embedding as a feature, concatenates it with the user's raw features, and takes the concatenated features as input. The objective is to use users' historical transaction data to predict whether they have default risk. Details of the dataset are provided in Appendix E.4. The results are presented in Figure 4. The results indicate that the model without time encoding performs the worst, as it completely ignores time information. With FTE, the periodicity of user behavior at different frequencies is captured, resulting in improved performance compared to the case without time information. Using LeTE yields the best performance, as our time encoding effectively models periodic, non-periodic and mixed patterns in a more general manner.

[1]The data used in these experiments are properly sampled only for testing purposes and does not imply any commercial information. All users' private information is removed from the dataset. Moreover, the experiments were conducted locally on Tencent's server by formal employees who strictly followed data protection regulations.

4.5. Dimensions of Time Encoding

Compared to the previous FTE methods, our LeTE takes a step forward by making the non-linear transformation learnable, thereby generalizing the time encoding. Since part of the information from the data is captured by the learnable non-linear transformation, we hypothesize that using lower-dimensional LeTE may still outperform the FTE (which relies on a fixed non-linear transformation). Therefore, we conduct experiments with lower-dimensional time encodings. Since the dimensionality of time encoding in dynamic graph tasks is more flexible, we conduct experiments on dynamic graph link prediction tasks, with the results presented in Figures 5, 6, and 7. As illustrated in the results, models using the FTE suffer from severe performance degradation as the dimension decreases, whereas models with LeTE demonstrate more stable performance and consistently outperform those using the FTE, even at lower dimensions. Notably, models using LeTE with significantly lower dimensions (e.g., 2, 8 or 16) outperform models with the 100-dimensional FTE. This demonstrates the effectiveness and generalizability of our method.

4.6. Additional Experiments

We provide the complete experimental results for the experiments mentioned in the main text in Appendix F. Additionally, we conduct further experiments to analyze the complex time patterns in real-world data (cf. Appendix G.1); compare different variants of LeTE (cf. Appendix G.2); illustrate the interpretability of LeTE (cf. Appendix G.3); demonstrate LeTE's ability to simultaneously capture diverse time patterns, including periodic, non-periodic, and mixed ones (cf. Appendix G.4); and assess LeTE's capability to fit various functions (cf. Appendix G.5).

5. Conclusion

In this paper, we propose a effective time encoding method—Learnable Transformation-based Generalized Time Encoding (LeTE)—designed to accept both absolute

timestamps and relative time differences as inputs, depending on the specific requirements of different models, enabling it to function as either an absolute or a relative time encoding method. Through comprehensive analysis, we demonstrate that our proposed LeTE is capable of modeling diverse and complex time patterns, including periodic, non-periodic, and mixed patterns. It is invariant to time rescaling, sufficiently simple for integration with various backbone models, and exhibits good interpretability and dimensional efficiency. Extensive experiments on event-based image classification, time-series forecasting tasks, dynamic graph link prediction tasks, and real-world financial risk control applications demonstrate the superior performance and generalizability of our method across various application scenarios.

Acknowledgements

This work is partially supported by the Noncommunicable Chronic Diseases-National Science and Technology Major Project (NO. 2024ZD0532400 and NO. 2024ZD0532403), the National Key Research and Development Plan Project 2022YFC3600901. This work is sponsored by the Tencent Rhino-Bird Focused Research Program. This work is partially supported by NSF through grant IIS-2106972.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Baytas, I. M., Xiao, C., Zhang, X., Wang, F., Jain, A. K., and Zhou, J. Patient subtyping via time-aware lstm networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 65–74, 2017.

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., and Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in Neural Information Processing Systems*, 31, 2018.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, 2013.

Braun, J. and Griebel, M. On a constructive proof of kolmogorov's superposition theorem. *Constructive approximation*, 30:653–675, 2009.

Campos, V., Jou, B., Giró-i Nieto, X., Torres, J., and Chang, S.-F. Skip rnn: Learning to skip state updates in recurrent neural networks. In *The Sixth International Conference on Learning Representations*, 2018.

Chen, X., Liao, Y., Xiong, Y., Zhang, Y., Zhang, S., Zhang, J., and Sun, Y. Speed: Streaming partition and parallel acceleration for temporal interaction graph embedding. *arXiv preprint arXiv:2308.14129*, 2023.

Chen, X., Xiong, Y., Zhang, S., Zhang, J., Zhang, Y., Zhou, S., Wu, X., Zhang, M., Liu, T., and Wang, W. Dtformer: A transformer-based method for discrete-time dynamic graph representation learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 301–311, 2024a.

Chen, X., Zhang, S., Xiong, Y., Wu, X., Zhang, J., Sun, X., Zhang, Y., Zhao, F., and Kang, Y. Prompt learning on temporal interaction graphs. *arXiv preprint arXiv:2402.06326*, 2024b.

Cho, K. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014.

Choi, E., Bahadori, M. T., Schuetz, A., Stewart, W. F., and Sun, J. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine learning for Healthcare Conference*, pp. 301–318. PMLR, 2016.

Fatahi, M., Ahmadi, M., Shahsavari, M., Ahmadi, A., and Devienne, P. evt_mnist: A spike based version of traditional mnist. In *1st International Conference on New Research Achievements in Electrical and Computer Engineering*, 2016.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pp. 1243–1252. PMLR, 2017.

Graves, A. and Graves, A. Long short-term memory. *Supervised Sequence Labelling with Recurrent Neural Networks*, pp. 37–45, 2012.

Kang, W.-C. and McAuley, J. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining*, pp. 197–206. IEEE, 2018.

Kazemi, S. M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., and Brubaker, M. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.

Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., and Poupart, P. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

Kolmogorov, A. N. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*, volume 114, pp. 953–956. Russian Academy of Sciences, 1957.

Kolmogorov, A. N. *On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables*. American Mathematical Society, 1961.

Kumar, S., Zhang, X., and Leskovec, J. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1269–1278, 2019.

Kwon, B. C., Choi, M.-J., Kim, J. T., Choi, E., Kim, Y. B., Kwon, S., Sun, J., and Choo, J. Retainvis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records. *IEEE Transactions on Visualization and Computer Graphics*, 25(1): 299–309, 2018.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.

Lezmi, E. and Xu, J. Time series forecasting with transformer models and application to asset management. *Available at SSRN 4375798*, 2023.

Li, Y., Du, N., and Bengio, S. Time-dependent representation for neural event sequence prediction. *arXiv preprint arXiv:1708.00065*, 2017.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *The Ninth International Conference on Learning Representations*, 2021.

Liu, Y., Wu, H., Wang, J., and Long, M. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, 2022.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2023.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., and Tegmark, M. Kan: Kolmogorov-arnold networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

Misra, D. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

Murat, M., Malinowska, I., Gos, M., and Krzyszczak, J. Forecasting daily meteorological time series using arima and regression models. *International Agrophysics*, 32(2), 2018.

Neumann, O., Beichter, M., Heidrich, B., Friederich, N., Hagenmeyer, V., and Mikut, R. Intrinsic explainable artificial intelligence using trainable spatial weights on numerical weather predictions. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*, pp. 551–559, 2024.

Pennebaker, J. W. Linguistic inquiry and word count: Liwc 2001, 2001.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., and Bronstein, M. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Skarding, J., Gabrys, B., and Musial, K. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, 9:79143–79168, 2021.

Tallec, C. and Ollivier, Y. Can recurrent neural networks warp time? In *The Sixth International Conference on Learning Representations*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., and Xiao, Y. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

Wang, L., Chang, X., Li, S., Chu, Y., Li, H., Zhang, W., He, X., Song, L., Zhou, J., and Yang, H. Tcl: Transformer-based dynamic graph modelling via contrastive learning. *arXiv preprint arXiv:2105.07944*, 2021.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., and Achan, K. Self-attention with functional time representation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., and Achan, K. Inductive representation learning on temporal graphs. In *The Eighth International Conference on Learning Representations*, 2020.

Yu, L., Sun, L., Du, B., and Lv, W. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36: 67686–67700, 2023.

Zeng, C., Tian, Y., Zheng, G., and Gao, Y. How much can time-related features enhance time series forecasting? *arXiv preprint arXiv:2412.01557*, 2024.

Zhang, J. Rpn: Reconciled polynomial network towards unifying pgms, kernel svms, mlp and kan. *arXiv preprint arXiv:2407.04819*, 2024.

Zhang, S., Xiong, Y., Zhang, Y., Sun, Y., Chen, X., Jiao, Y., and Zhu, Y. Rdgsl: Dynamic graph representation learning with structure learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 3174–3183, 2023a.

Zhang, S., Chen, X., Xiong, Y., Wu, X., Zhang, Y., Fu, Y., Zhao, Y., and Zhang, J. Towards adaptive neighborhood for advancing temporal interaction graph modeling. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4290–4301, 2024.

Zhang, S., Xiong, Y., Tang, Y., Chen, X., Jia, Z., Gu, Z., Xu, J., and Zhang, J. Unifying text semantics and graph structures for temporal text-attributed graphs with large language models. *arXiv preprint arXiv:2503.14411*, 2025.

Zhang, Y., Xiong, Y., Liao, Y., Sun, Y., Jin, Y., Zheng, X., and Zhu, Y. Tiger: Temporal interaction graph embedding with restarts. In *Proceedings of the ACM Web Conference 2023*, pp. 478–488, 2023b.

A. Related Work

Currently, commonly used time encoding methods or strategies for modeling temporal information can be broadly categorized into two types: Hand-Crafted Time Encodings (HCTE) and Functional Time Encodings (FTE).

HCTE involves manually designed temporal encodings tailored to specific downstream tasks. These methods rely on specific design choices and incorporate various inductive biases to capture fixed periodic patterns, constructing hand-crafted temporal features. These features are typically fed into models such as RNNs (Cho, 2014) or sequential architectures (Liu et al., 2021; 2022; Wang et al., 2023; Wu et al., 2023) to meet specific modeling requirements (Choi et al., 2016; Baytas et al., 2017; Kwon et al., 2018), as illustrated in Figure 1(a). Such approaches are often employed to address particular challenges in time series tasks.

Additionally, some methods in this category integrate time encoding directly with attention mechanisms (Vaswani et al., 2017), simplifying temporal modeling by adopting position encoding strategies (Gehring et al., 2017). Others embed discrete events into a continuous vector space to better capture event contexts in attention-based models (Bengio et al., 2013; Li et al., 2017). These methods, while effective, are often limited to representing fixed or narrowly defined temporal patterns.

FTE represents an advanced and generalized version of time encoding, designed to overcome part of the limitations of Hand-Crafted Time Encodings. Two representative works in this category are functional time representation, proposed by (Xu et al., 2019), and Time2Vec, proposed by (Kazemi et al., 2019), as shown in Figure 1(b). Importantly, the previously mentioned position encoding methods integrated with attention mechanisms can be considered a simplified version of FTE.

Both functional time representation and Time2Vec adopt similar implementation methods, which resemble a one-dimensional-to-d-dimensional MLP with a specially designed trigonometric non-linear activation function. In Time2Vec, experiments comparing different non-linear activation functions demonstrate that the sine function performs best across various downstream tasks. Despite limitations in modeling restricted aspects of time, FTE is widely adopted in dynamic graph representation learning due to its ease of application and effectiveness (Rossi et al., 2020; Zhang et al., 2023b;a; Chen et al., 2023; Yu et al., 2023; Chen et al., 2024b; Zhang et al., 2024; Chen et al., 2024a; Zhang et al., 2025).

Time encodings can be directly applied to sequential models such as RNNs and LSTMs (Graves & Graves, 2012), or easily integrated into attention-based architectures. In time series forecasting, for instance, many models now employ transformer-based structures (Vaswani et al., 2017), where time encoding is often treated similarly to position encoding. This is usually achieved by adding it to the input of the attention mechanism (Liu et al., 2021; 2022; Wu et al., 2023).

Dynamic graph representation learning models also require precise temporal modeling. For example, TGAT (Xu et al., 2020) directly replaces position encoding with functional time representation within its attention mechanism. Subsequent methods, such as TGN and TIGER (Rossi et al., 2020; Zhang et al., 2023b), have adopted similar approaches. DyGFormer (Yu et al., 2023), which applies a Transformer to dynamic graph representation learning, uses the same encoding method by concatenating it with node and edge features before processing them with a Transformer-based model.

B. Methods of Parameterize Continuous Functions

B.1. Fourier Series Expansion

A function $f(x)$ that is periodic with period T and satisfies certain conditions (Dirichlet conditions) can be represented as a Fourier Series. This series represents $f(x)$ as an infinite sum of sines and cosines (or, equivalently, complex exponentials) with specific coefficients. The series takes the form:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos \frac{2\pi n x}{T} + b_n \sin \frac{2\pi n x}{T} \right). \tag{16}$$

Here, a_0 is the average value of the function over one period, and a_n and b_n are Fourier coefficients that can be calculated by integrating $f(x)$ over the interval $[0, T]$.

These coefficients are given by:

$$a_n = \frac{2}{T} \int_0^T f(x) \cos \frac{2\pi n x}{T} \, dx, \tag{17}$$

$$b_n = \frac{2}{T} \int_0^T f(x) \sin \frac{2\pi n x}{T} \, dx. \tag{18}$$

Under these conditions, the Fourier Series converges to $f(x)$ at all points where f is continuous and converges to the average of the left-hand and right-hand limits at points of discontinuity.

B.2. KAN and Spline Functions

The Kolmogorov–Arnold Theorem (Kolmogorov, 1961; 1957; Braun & Griebel, 2009) states that for any continuous multivariate function $f(x_1, x_2, \ldots, x_n)$ on the unit cube $[0, 1]^n$, there exist continuous functions ϕ_i and ψ_{ij} such that:

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{2n+1} \phi_i \left(\sum_{j=1}^{n} \psi_{ij}(x_j) \right), \tag{19}$$

where ϕ_i are continuous functions of a single variable, enabling dimensionality reduction, and ψ_{ij} are continuous functions mapping each input variable x_j to a single output, contributing to the superposition structure. This theorem implies that every continuous function of multiple variables can be represented as a sum of compositions of univariate functions.

Building on the Kolmogorov–Arnold Theorem and the advantages of splines for function fitting, Liu et al. propose using splines to construct learnable non-linear activation functions for neural networks (Liu et al., 2025). We briefly introduce B-splines here. Given a knot vector $T = t_0, t_1, \ldots, t_m$ with non-decreasing values, the basis functions $N_{i,p}(x)$ for a B-spline of degree p are defined recursively as follows: For degree $p = 0$:

$$N_{i,0}(x) = \begin{cases} 1 & \text{if } t_i \leq x < t_{i+1} \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

For higher degrees $p > 0$:

$$N_{i,p}(x) = \frac{x - t_i}{t_{i+p} - t_i} N_{i,p-1}(x) + \frac{t_{i+p+1} - x}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(x). \tag{21}$$

The B-spline curve $C(x)$ of degree p with control points $\{P_0, P_1, \ldots, P_n\}$ is given by:

$$C(x) = \sum_{i=0}^{n} N_{i,p}(x) P_i. \tag{22}$$

Here, $N_{i,p}(x)$ are the B-spline basis functions of degree p , and P_i are the control points that influence the shape of the curve.

C. Proofs

C.1. Proof of Proposition 2.1

C.1.1. Details of Functional Time Representation and Time2Vec

FTR is designed to use the time difference $t = t_i - t_j$, where $0 \leq t_j \leq t_i \leq t_{\text{max}}$, as input. For the input time difference, a learnable frequency parameter is first applied. Next, a non-linear transformation is applied, using the cosine function on the odd dimensions and the sine function on the even dimensions. This method is mathematically represented as follows:

$$\mathbf{TE}(t)[i] = \begin{cases} \cos(\omega_i t), & \text{if } i \text{ is odd,} \\ \sin(\omega_i t), & \text{if } i \text{ is even,} \end{cases} \tag{23}$$

where d is the dimension of the time encoding, $1 \leq i \leq d$, and ω_i are learnable parameters representing the frequency of the trigonometric functions. Since this time encoding uses time differences as input, it can be considered as a *relative time encoding*.

T2V is designed to use timestamps t as input. A linear transformation is applied to the first dimension to capture non-periodic time patterns. For the remaining dimensions, a linear transformation is followed by a sine-based non-linear transformation to model periodic time patterns: Mathematically, this method is represented as follows:

$$\mathbf{TE}(t)[i] = \begin{cases} \omega_i t + \varphi_i, & \text{if } i = 1, \\ \sin(\omega_i t + \varphi_i), & \text{if } 2 \leq i \leq d, \end{cases} \tag{24}$$

where $\mathbf{TE}(t)[i]$ is the i^{th} element of the time encoding, and ω_i and φ_i are learnable parameters representing frequency and phase-shift of the sine function, respectively. Since this time encoding takes timestamps as input, it can be considered as an *absolute time encoding*.

C.1.2. Proof of Proposition 2.1

Proof. Since Equation (23) can be written as $\mathbf{TE}(t) = [\cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_d t), \sin(\omega_d t)]$, we show that the vector $[\cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_d t), \sin(\omega_d t)]$ can be expressed in the form $\sin(\omega_i t + \varphi_i)$ with suitable phase shifts φ_i. Recall the trigonometric identity:

$$\cos(\theta) = \sin\left(\theta + \frac{\pi}{2}\right). \tag{25}$$

Applying this identity, each cosine term in the vector can be rewritten as:

$$\cos(\omega_i t) = \sin\left(\omega_i t + \frac{\pi}{2}\right), \qquad \text{for} \quad i = 1, 2, \ldots, d. \tag{26}$$

The sine terms are already in the desired form with a zero phase shift:

$$\sin(\omega_i t) = \sin\left(\omega_i t + 0\right), \qquad \text{for} \quad i = 1, 2, \ldots, d. \tag{27}$$

With these transformations, the original vector becomes:

$$\mathbf{TE}(t) = \left[\sin\left(\omega_1 t + \frac{\pi}{2}\right), \sin\left(\omega_1 t + 0\right), \ldots, \sin\left(\omega_d t + \frac{\pi}{2}\right), \sin\left(\omega_d t + 0\right)\right], \tag{28}$$

or equivalently,

$$\mathbf{TE}(t) = \left[\sin\left(\omega_i t + \varphi_i\right)\right]_{i=1}^{2d}, \tag{29}$$

where the phase shifts φ_i are defined as follows:

$$\varphi_i = \begin{cases} \frac{\pi}{2}, & \text{if } i \text{ is odd} \\ 0, & \text{if } i \text{ is even.} \end{cases}$$

□

C.2. Continued Proof of Proposition 3.1

Proof. The function $\sin(\theta)$ is continuous and infinitely differentiable (i.e., C^∞) on $\mathbb{R}$. Thus, it is continuous on any closed interval $[a, b]$.

B-spline basis functions of degree k form a basis for the space of piecewise polynomial functions of degree k with continuity C^{k-1} at the knots. By the Weierstrass Approximation Theorem, any continuous function on a closed interval can be uniformly approximated by polynomials to any desired degree of accuracy.

Since B-splines are piecewise polynomials, they can uniformly approximate any continuous function on $[a, b]$. Specifically, for any $\epsilon > 0$, there exists a linear combination of B-spline basis functions that approximates $\sin(\theta)$ within ϵ over $[a, b]$.

To build the approximation of $\sin(\omega_i t + \varphi_i)$ using B-spline basis functions, we first select a knot vector $\mathbf{T} = \{t_0, t_1, \ldots, t_{n+k+1}\}$ that partitions the interval $[a, b]$ appropriately. The choice of the knot vector determines the placement and spacing of the knots, which in turn affect the flexibility and local support of the B-spline basis functions.

Next, we choose the degree k of the B-spline basis functions based on the desired smoothness and approximation quality. A higher degree allows for smoother basis functions, potentially improving the approximation at the cost of increased computational complexity.

With the knot vector and degree specified, we generate the B-spline basis functions $\{B_j(\theta)\}_{j=1}^{M}$ of degree k using standard recursive definitions. These basis functions possess local support and satisfy the partition of unity property, making them suitable for approximating functions over $[a, b]$.

To determine the coefficients c_{ij} that yield the best approximation of $\sin(\omega_i t + \varphi_i)$, we formulate an optimization problem. Specifically, we set up a minimization problem that seeks to minimize the squared difference between the sine function and

the weighted sum of B-spline basis functions over the interval $[a, b]$:

$$\min_{c_{i,1},c_{i,2},\ldots,c_{i,M}} \int_a^b \left[\sin(\omega_i t + \varphi_i) - \sum_{j=1}^{M} c_{i,j}\, B_j(\omega_i t + \varphi_i)\right]^2 dt. \tag{30}$$

This minimization problem is a standard least squares problem, where the objective is to find the coefficients $c_{i,j}$ that minimize the integral of the squared error. Solving this problem can be accomplished using numerical methods such as the normal equations or singular value decomposition, leading to the optimal coefficients for the approximation.

By leveraging the properties of B-spline basis functions and the Weierstrass Approximation Theorem, we can assert that, for any $\epsilon > 0$, there exists a sufficiently large M and appropriate coefficients $c_{i,j}$ such that:

$$\sup_{t\in[a,b]} \left|\sin(\omega_i t + \varphi_i) - \sum_{j=1}^{M} c_{i,j}\, B_j(\omega_i t + \varphi_i)\right| < \epsilon. \tag{31}$$

This inequality indicates that the maximum deviation between the sine function and its B-spline approximation over $[a, b]$ is less than ϵ, satisfying the condition of uniform approximation.

Since $\epsilon > 0$ is arbitrary, we can make the approximation as accurate as desired by increasing M and choosing appropriate coefficients $c_{i,j}$. Therefore, the sine function $\sin(\omega_i t + \varphi_i)$ can be represented as a sum of B-spline basis functions, making it a special case of Equation (10) in the limit as $M \to \infty$. □

C.3. Proof of Proposition 3.2

A class $\mathcal{C}$ of models is considered invariant to time rescaling if, for any model $\mathcal{M}_1 \in \mathcal{C}$ and any scalar $\alpha > 0$, there exists a model $\mathcal{M}_2 \in \mathcal{C}$ that responds to αt (where t is scaled by α) in the same way that $\mathcal{M}_1$ responds to the original t values. We provide the following proof to show that LeTE is invariant to time rescaling.

Proof. Consider time encoding $\mathcal{M}_1$, mapped by LeTE:

$$\textbf{LeTE}(t)[i] = \phi_i(\omega_i t + \varphi_i). \tag{32}$$

If we replace t with $\alpha \cdot t$ (where $\alpha > 0$), the time encoding updates as follows:

$$\textbf{LeTE}(\alpha \cdot t)[i] = \phi_i(\omega_i(\alpha \cdot t) + \varphi_i). \tag{33}$$

To preserve the behavior of the original model $\mathcal{M}_1$ under time rescaling, consider a new time encoding $\mathcal{M}_2$ with adjusted frequencies $\omega_i' = \frac{\omega_i}{\alpha}$. With this frequency adjustment, $\mathcal{M}_2$ behaves identically to $\mathcal{M}_1$ on $\alpha \cdot t$, demonstrating that LeTE is invariant to time rescaling. □

D. Implementation Details of LeTE

Previous implementations of FTEs can be summarized as inputting a timestamp or time difference between events into a single-layer MLP with a fixed trigonometric function as the non-linear activation function. Inspired by this, our method can be viewed as making the fixed activation function learnable by parameterizing it with a Fourier series expansion or B-spline functions. Following KAN (Liu et al., 2025), which makes activation functions in deep learning models trainable, we use a similar implementation method.

D.1. Fourier-based LeTE

The implementation of Fourier-based LeTE is straightforward and is given by:

$$\phi_j(x) = \sum_{i=1}^{D}\sum_{m=1}^{K} \left(W_{j,i,m}^{(\cos)} \cos(mx_i) + W_{j,i,m}^{(\sin)} \sin(mx_i)\right) + b_j, \tag{34}$$

where $i = 1, 2, \ldots, D$ indexes the input dimension, $j = 1, 2, \ldots, M$ indexes the output dimension (with $D = M$ in our method), and $m = 1, 2, \ldots, K$ indexes the Fourier frequencies. Here, K is a hyper-parameter that determines the grid size. The parameters $\mathbf{W}^{(\cos)} \in \mathbb{R}^{M\times D\times K}$, $\mathbf{W}^{(\sin)} \in \mathbb{R}^{M\times D\times K}$, and $\mathbf{b} \in \mathbb{R}^{M}$ are learnable weights and biases.

D.2. Spline-based LeTE

By using B-spline functions, we make the ϕ_i functions in Equation (6) learnable as follows:

$$\phi_i(x) = b_i(x) + \text{spline}_i(x), \tag{35}$$

$$b(x) = \text{Tanh}(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{36}$$

$$\text{spline}_i(x) = \sum_j c_{ij} B_j(x), \tag{37}$$

where c_{ij} are learnable. Unlike the original KAN, we use Tanh as the basis function here, as we found it performs better in practice.

E. More Details for Experimental Setting

E.1. Details for "Time as the Only Input" Experiments

Following (Kazemi et al., 2019), We generate a sequential event-based version of MNIST by flattening the images and recording the positions of pixels with intensities greater than a threshold (0.9 in our experiment). After this transformation, each image is represented as an array of increasing numbers, such as $[t_1, t_2, t_3, \ldots, t_m]$. These values are treated as event times and can be used for image classification task. The backbone model we used is a 128-dimensional LSTM, with a batch size of 512, aligned with the settings in (Kazemi et al., 2019).

E.2. Time Series Baselines and Datasets

E.2.1. Experiment Implementation Details

Our experiments setting on time series tasks aligns with the definition of long-term forecasting. The baseline results for Transformer and Pyraformer are based on our implementation, while the results for the other baselines are taken from their original papers. Baseline models typically use a hand-crafted time encoding method that applies date and timestamps to represent various time features—including minutes, hours, weekdays, days, and months. The mapped vectors are then added together and added to the feature embeddings and fed into the models. The results are evaluated using MAE (Mean Absolute Error) (Table 1) and MSE (Mean Squared Error) (Table 5), both of which are widely used metrics in time series forecasting research.

E.2.2. Baselines

We select 5 commonly used time series prediction baselines—Transformer (Vaswani et al., 2017), Pyraformer (Liu et al., 2021), Non-stationary Transformer (Liu et al., 2022), MICN (Wang et al., 2023), and TimesNet (Wu et al., 2023)—and replace their original hand-crafted time encodings with our proposed LeTE to demonstrate that LeTE can be effectively applied to time series prediction models and improve the performance of downstream tasks. A brief introduction to these baseline models is provided below:

- **Transformer** (Vaswani et al., 2017) leverages the self-attention mechanism to model long-range dependencies in sequences, making it a powerful tool for time series forecasting, especially in cases with complex time patterns. Its global context modeling capability enables it to capture intricate relationships between time steps effectively.
- **Pyraformer** (Liu et al., 2021) introduces a pyramid attention mechanism that hierarchically reduces the computational burden while preserving the ability to model both local and global dependencies. This design makes it particularly well-suited for handling long time series with improved efficiency and scalability.
- **Non-stationary Transformer** (Liu et al., 2022): addresses challenges in forecasting non-stationary time series by incorporating dynamic feature adjustments and context-aware attention mechanisms. This allows the model to adapt to evolving data distributions, ensuring robust and accurate predictions in dynamic environments.
- **MICN** (Wang et al., 2023) integrates multi-scale architectures to capture both short-term patterns and long-term dependencies in time series data. By combining local convolutional operations and global attention, it provides a balanced approach to handling diverse temporal characteristics.

- **TimesNet** (Wu et al., 2023) innovatively models time series data in the frequency domain, leveraging discrete Fourier transformations to capture periodicity and trends. This approach enhances its ability to predict time series with prominent seasonal and cyclical behaviors efficiently.

E.2.3. DATASETS

We utilize 4 real-world datasets to evaluate the effectiveness of our method on time series prediction tasks, encompassing various real-world scenarios. The dataset statistics are presented in Table 3, with detailed descriptions provided below.

Table 3. Time Series Dataset Statistics: The dataset size is organized in (Train, Validation, Test). Please refer to (Wu et al., 2023) for the original table.

Dataset	Dim	Series Length	Dataset Size	Information (Frequency)
ETTm1, ETTm2	7	{96, 192, 336, 720}	(34465, 11521, 11521)	Electricity (15 mins)
ETTh1, ETTh2	7	{96, 192, 336, 720}	(8545, 2881, 2881)	Electricity (15 mins)
Electricity	321	{96, 192, 336, 720}	(18317, 2633, 5261)	Electricity (Hourly)
Exchange	8	{96, 192, 336, 720}	(5120, 665, 1422)	Exchange rate (Daily)
Weather	21	{96, 192, 336, 720}	(36792, 5271, 10540)	Weather (10 mins)

- **ETT**[2] dataset includes time series data for oil temperature and power load measurements from electricity transformers, collected between July 2016 and July 2018. Specifically, the subsets ETTm1 and ETTm2 are sampled at 15-minute intervals, while ETTh1 and ETTh2 are recorded hourly.
- **Electricity**[3] dataset provides hourly electricity consumption data for 321 clients, spanning the period from 2012 to 2014.
- **Exchange** (Lai et al., 2018) dataset offers daily panel data on exchange rates from eight countries, covering the years 1990 to 2016.
- **Weather**[4] dataset contains meteorological time series data, comprising 21 weather indicators recorded at 10-minute intervals in 2020 by the Weather Station of the Max Planck Biogeochemistry Institute.

E.3. Dynamic Graph Baselines and Datasets

E.3.1. EXPERIMENT IMPLEMENTATION DETAILS

The hyper-parameters are based on the best configurations reported in the papers, and we keep them unchanged across different experiments for each baseline model to ensure a fair comparison. We rerun the baseline models TGAT with batch size 100 and reuse the baseline results reported in the DyGFormer paper for other baselines. The results are evaluated using Average Precision, i.e., AP (Table 2) and Area Under the Receiver Operating Characteristic Curve, i.e., AUC-ROC (Table 6), both of which are widely used metrics in dynamic graph representation learning research.

E.3.2. BASELINES

We select 4 commonly used continuous dynamic graph representation learning baselines—TGAT (Xu et al., 2020), TGN (Rossi et al., 2020), TCL (Wang et al., 2021), and DyGFormer (Yu et al., 2023)—and replace the Functional Time Encoding methods (Kazemi et al., 2019; Xu et al., 2019) with LeTE to demonstrate its optimal performance on the dynamic graph link prediction task. A brief introduction to these baseline models is provided below:

- **TGAT** (Xu et al., 2020) introduces a temporal attention mechanism to aggregate information from temporal-topological neighbors, thereby generating temporal node representations in temporal graphs. Additionally, it proposes a trainable

[2] https://github.com/zhouhaoyi/ETDataset
[3] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[4] https://www.bgc-jena.mpg.de/wetter/

time encoding function to capture distinguishable temporal information, which has been widely adopted in subsequent dynamic graph network architectures.

- **TGN** (Rossi et al., 2020) integrates key ideas from previous models and introduces a memory module that maintains a state vector for each node. The memory is updated dynamically whenever nodes participate in interactions. Additionally, TGN incorporates a message-passing module, a memory update module, and a temporal embedding module to generate effective temporal representations for nodes within temporal graphs.
- **TCL** (Wang et al., 2021) utilizes a breadth-first search algorithm to construct a temporal dependency interaction subgraph, extracting interaction sequences. It employs a Transformer encoder that integrates both topological and temporal information to learn representations of central nodes. Additionally, TCL introduces a cross-attention mechanism within the Transformer to model the inter-dependencies between interacting node pairs.
- **DyGFormer** (Yu et al., 2023) leverages 1-hop neighbor information for learning temporal graph representations. It employs a Transformer encoder enhanced with a patching technique to effectively capture long-term dependencies among nodes in temporal graphs. Furthermore, DyGFormer incorporates a Neighbor Co-occurrence Feature to preserve the correlation information between source and target nodes.

E.3.3. DATASETS

We utilize 4 real-world datasets (Kumar et al., 2019) to evaluate the effectiveness of our method on dynamic graph representation learning tasks, encompassing various real-world scenarios. The dataset statistics are presented in Table 4, with detailed descriptions provided below.

Table 4. Dynamic Graph Dataset Statistics: Dim$n$ and Dim$e$ represent the dimensions of node features and edge features, respectively. For non-attributed graphs, we follow previous studies (Xu et al., 2020; Rossi et al., 2020) and use 172-dimensional zero vectors as padding.

Dataset	Dim$_n$	Dim$_e$	# Nodes	# Edges	Information	Duration	Time Granularity
Wikipedia	-	172	9,227	157,474	Social	1 month	Unix timestamps
Reddit	-	172	10,984	672,447	Social	1 month	Unix timestamps
MOOC	-	4	7,144	411,749	Interaction	17 months	Unix timestamps
LastFM	-	-	1,980	1,293,103	Interaction	1 month	Unix timestamps

- **Wikipedia** records editing activities on Wikipedia pages over a one-month timeframe. Nodes in this graph represent users or pages, and temporal links with timestamps capture the edits. Each link is associated with a 172-dimensional feature vector based on LIWC (Linguistic Inquiry and Word Count) (Pennebaker, 2001).
- **Reddit** captures user activity across subreddits over a one-month period. In this dataset, nodes represent users or subreddits, while timestamped links denote posting actions. Each link is further characterized by a 172-dimensional feature vector derived from LIWC.
- **MOOC** captures the interactions of users on a widely used MOOC platform, structured as a directed, temporal network. In this representation, nodes correspond to users and course activities (referred to as targets), while edges denote the actions performed by users on these targets
- **LastFM** records interaction data where users listen to songs over a month. In LastFM, nodes correspond to users and songs, and the links represent listening activities performed by users.

E.4. Real-World Application Dataset

The dataset used for real-world application experiments is a financial risk control dataset, containing records of 483,379 users' transaction behavior at various merchants over a 60-day period. It includes a total of 26,850,000 transactions. Each user is represented by a 585-dimensional feature, each merchant by a 128-dimensional feature, and each transaction by a 202-dimensional feature, with all transactions labeled with UNIX timestamps. The ratio of positive users (with default risk) to negative users (without default risk) is 1:9.92 in the training dataset and 1:20.25 in the test dataset. The backbone model

employs a specially designed Transformer-based architecture to aggregate users' historical transaction features, merchant features, user features, and an optional time embedding into user embeddings, which are then used to predict whether the user has default risk.

F. Additional Experimental Results

Here, we provide the complete results of the experiments discussed in the main text.

F.1. Results of multivariate time series long-term forecasting task evaluated using MSE

The results of the multivariate time series long-term forecasting task, evaluated using MSE, are presented in Table 5. These results are organized in the same manner as those in Table 1.

Table 5. Time series prediction: multivariate long-term forecasting task. The past sequence length is set to 96, while the prediction lengths are {96, 192, 336, 720}. The results are reported in terms of MSE, where lower values indicate better performance. HCTE (Hand-Crafted Time Encoding) is a method widely adopted in time series research. FTE stands for Functional Time Encoding. The win rate represents the percentage of cases where LeTE outperforms the HCTE. The best results for each baseline, dataset and prediction length combinations are in **bold**. ETT consists of 4 subsets. Here, we present the average results across these subsets, with the full results provided in Table 8.

MSE		Transformer			Pyraformer			NS Trans.			MICN			TimesNet			Win
TE		HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE	Rate
ETT	96	1.219	1.204	**0.568**	0.794	0.960	**0.650**	0.392	0.486	**0.347**	0.392	0.310	**0.296**	0.312	0.322	**0.305**	95%
	192	2.601	1.643	**0.959**	1.667	1.812	**1.058**	0.446	0.548	**0.410**	0.446	0.396	**0.357**	**0.365**	0.390	0.368	
	336	2.438	1.557	**1.192**	1.981	1.962	**1.297**	0.492	0.651	**0.467**	0.492	0.487	**0.430**	0.419	0.422	**0.404**	
	720	1.998	2.217	**1.313**	2.589	2.442	**1.612**	0.552	0.660	**0.531**	0.552	0.630	**0.517**	0.467	0.468	**0.441**	
Electricity	96	0.258	0.280	**0.252**	0.285	0.281	**0.267**	0.169	0.171	**0.163**	0.164	0.156	**0.150**	0.168	0.168	**0.164**	95%
	192	0.266	0.310	**0.259**	0.298	0.288	**0.274**	0.182	0.192	**0.178**	0.177	0.170	**0.166**	0.184	0.179	**0.179**	
	336	0.275	0.339	**0.265**	0.307	0.306	**0.274**	0.200	0.201	**0.190**	0.193	0.189	**0.184**	0.198	0.210	**0.193**	
	720	0.288	0.395	**0.276**	0.304	0.304	**0.293**	**0.222**	0.237	0.226	0.212	0.228	**0.208**	0.220	0.287	**0.220**	
Exchange	96	0.545	0.788	**0.464**	**0.505**	0.616	0.584	0.111	0.137	**0.102**	0.102	0.098	**0.080**	0.107	0.108	**0.103**	95%
	192	0.950	1.028	**0.903**	1.015	0.983	**0.904**	0.219	0.273	**0.193**	0.172	0.186	**0.152**	0.226	0.217	**0.211**	
	336	1.462	1.866	**1.114**	1.263	1.420	**1.056**	0.421	0.463	**0.342**	0.272	0.356	**0.265**	0.367	0.413	**0.367**	
	720	2.569	2.002	**1.736**	1.762	1.714	**1.316**	1.092	1.546	**0.682**	0.714	0.833	**0.636**	0.964	0.989	**0.958**	
Weather	96	0.393	0.184	**0.172**	0.225	0.207	**0.181**	0.173	0.172	**0.168**	**0.161**	0.198	0.166	0.172	0.172	**0.166**	95%
	192	0.547	0.249	**0.220**	0.252	0.238	**0.230**	0.245	0.224	**0.215**	0.220	0.243	**0.209**	0.219	0.221	**0.212**	
	336	0.678	0.328	**0.309**	0.362	0.324	**0.284**	0.321	0.295	**0.277**	0.278	0.285	**0.251**	0.280	0.278	**0.274**	
	720	0.844	0.480	**0.425**	0.411	0.394	**0.383**	0.414	0.350	**0.338**	0.311	0.350	**0.303**	0.365	0.354	**0.352**	
Win Rate		100%			94%			94%			94%			94%			95%

F.2. Results of dynamic graph link prediction task evaluated using AUC-ROC

The results of the dynamic graph link prediction task, evaluated using AUC-ROC, are presented in Table 6.

Table 6. Dynamic graph link prediction task: The results are reported in AUC-ROC, where higher values indicate better performance. The better results are in **bold**. Here, we present the top-performing results across variations of LeTE, with the full results provided in Table 10. FTE represents Functional Time Encoding which is commonly used in dynamic graph research.

	AUC	Wikipedia		Reddit		MOOC		LastFM	
	TE	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive
TGAT	FTE	96.69 ± 0.26	95.95 ± 0.33	98.48 ± 0.04	96.90 ± 0.07	86.44 ± 0.24	86.04 ± 0.19	70.89 ± 0.10	76.11 ± 0.11
	LeTE	**97.63 ± 0.11**	**97.07 ± 0.10**	**98.51 ± 0.01**	**96.96 ± 0.05**	**89.46 ± 0.08**	**89.50 ± 0.05**	**74.24 ± 0.28**	**79.63 ± 0.14**
TGN	FTE	98.37 ± 0.07	97.72 ± 0.03	98.60 ± 0.06	97.39 ± 0.07	91.21 ± 1.15	91.24 ± 0.99	78.47 ± 2.94	82.61 ± 3.15
	LeTE	**98.73 ± 0.07**	**98.11 ± 0.10**	**98.72 ± 0.00**	**97.55 ± 0.05**	**92.68 ± 0.37**	**92.41 ± 0.73**	**83.94 ± 1.85**	**87.74 ± 1.75**
TCL	FTE	95.84 ± 0.18	95.57 ± 0.20	97.42 ± 0.02	93.80 ± 0.07	83.12 ± 0.18	81.43 ± 0.19	64.06 ± 1.16	70.84 ± 0.85
	LeTE	**97.84 ± 0.06**	**97.56 ± 0.02**	**97.68 ± 0.03**	**94.63 ± 0.05**	**84.73 ± 0.13**	**83.25 ± 0.23**	**70.17 ± 0.47**	**75.86 ± 0.44**
DyG-Former	FTE	98.91 ± 0.02	98.48 ± 0.03	99.15 ± 0.01	98.71 ± 0.01	87.91 ± 0.58	87.62 ± 0.51	93.05 ± 0.10	94.08 ± 0.08
	LeTE	**99.04 ± 0.01**	**98.67 ± 0.02**	**99.17 ± 0.00**	**98.74 ± 0.01**	**89.18 ± 0.21**	**89.16 ± 0.30**	**93.65 ± 0.07**	**94.52 ± 0.08**

F.3. Full results of the multivariate long-term forecasting task on 4 ETT subsets

We present the full results of the multivariate long-term forecasting task on the 4 ETT subsets in Tables 7 and 8, as Tables 1 and 5 report the average results.

Table 7. Time series prediction: multivariate long-term forecasting task on 4 subsets of ETT. The past sequence length is set to 96, while the prediction lengths are {96, 192, 336, 720}. The results are reported in terms of MAE. The best results for each baseline, dataset and prediction length combinations are in **bold**.

MAE		Transformer			Pyraformer			NS Trans.			MINC			TimesNet		
TE		HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE
ETTm1	96	0.621	0.601	**0.507**	0.581	0.522	**0.521**	0.398	0.409	**0.389**	0.398	0.376	**0.372**	**0.375**	0.389	0.378
	192	0.703	0.550	**0.526**	0.577	0.547	**0.531**	0.444	0.427	**0.417**	0.444	**0.393**	0.399	**0.387**	0.418	0.404
	336	0.795	0.755	**0.596**	0.675	0.637	**0.610**	0.464	0.496	**0.439**	0.464	0.425	**0.426**	**0.411**	0.418	0.418
	720	0.798	0.782	**0.663**	0.760	0.700	**0.645**	0.516	0.498	**0.470**	0.516	0.467	**0.465**	**0.450**	0.456	0.454
ETTm2	96	0.506	0.451	**0.420**	0.458	0.756	**0.411**	**0.274**	0.304	0.278	0.274	0.287	**0.271**	0.267	0.268	**0.262**
	192	0.908	0.700	**0.558**	0.649	**0.611**	0.625	0.339	0.379	**0.325**	0.339	0.349	**0.321**	0.309	0.306	**0.302**
	336	0.796	0.806	**0.753**	0.811	0.902	**0.775**	0.361	0.400	**0.353**	**0.361**	0.439	0.364	0.351	0.351	**0.342**
	720	1.192	1.301	**0.851**	1.416	1.491	**1.137**	**0.413**	0.446	0.420	**0.413**	0.506	0.432	0.403	0.409	**0.399**
ETTh1	96	0.739	0.814	**0.524**	0.637	0.613	**0.593**	0.491	0.616	**0.452**	0.491	0.413	**0.409**	**0.402**	0.425	0.408
	192	0.762	0.815	**0.632**	0.738	0.778	**0.647**	0.504	0.631	**0.483**	0.504	0.465	**0.452**	0.429	0.457	**0.439**
	336	0.772	0.786	**0.679**	0.794	0.758	**0.736**	**0.535**	0.730	0.541	0.535	0.511	**0.502**	0.469	0.469	**0.453**
	720	0.800	0.878	**0.724**	0.776	0.804	**0.782**	0.616	0.792	**0.610**	0.616	0.598	**0.565**	0.500	0.490	**0.421**
ETTh2	96	1.323	1.349	**0.752**	0.892	0.989	**0.806**	0.458	0.413	**0.390**	0.458	0.392	**0.349**	0.374	0.367	**0.359**
	192	2.184	1.599	**1.133**	1.632	1.760	**1.150**	0.493	0.475	**0.434**	0.493	0.485	**0.408**	0.414	0.418	**0.406**
	336	2.113	1.405	**1.256**	1.893	1.856	**1.330**	0.551	0.528	**0.463**	0.551	0.569	**0.500**	0.452	0.458	**0.438**
	720	1.488	1.623	**1.276**	1.832	1.761	**1.274**	0.560	0.492	**0.460**	0.560	0.673	**0.558**	0.468	0.466	**0.443**

Table 8. Time series prediction: multivariate long-term forecasting task on 4 subsets of ETT. The past sequence length is set to 96, while the prediction lengths are {96, 192, 336, 720}. The results are reported in terms of MSE. The best results for each baseline, dataset and prediction length combinations are in **bold**.

MSE		Transformer			Pyraformer			NS Trans.			MINC			TimesNet		
TE		HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE	HCTE	FTE	LeTE
ETTm1	96	0.713	0.667	**0.511**	0.708	**0.581**	0.584	0.386	0.430	**0.370**	0.386	0.329	**0.325**	**0.338**	0.365	0.339
	192	0.866	0.592	**0.573**	0.693	0.621	**0.560**	0.459	0.464	**0.418**	0.459	**0.364**	0.373	**0.374**	0.430	0.392
	336	1.063	1.035	**0.692**	0.848	0.754	**0.689**	0.495	0.651	**0.451**	0.495	0.403	**0.396**	**0.410**	0.412	0.412
	720	1.075	1.062	**0.788**	1.009	0.855	**0.777**	0.585	0.584	**0.510**	0.585	0.469	**0.463**	0.478	0.482	**0.477**
ETTm2	96	0.486	0.354	**0.327**	0.384	1.047	**0.310**	**0.192**	0.246	0.195	0.192	0.190	**0.187**	0.187	0.191	**0.182**
	192	1.499	0.834	**0.522**	0.730	**0.653**	0.672	0.280	0.378	**0.266**	0.280	0.271	**0.241**	0.249	0.253	**0.248**
	336	1.107	1.113	**0.949**	1.144	1.377	**1.023**	0.334	0.387	**0.315**	0.334	0.398	**0.307**	0.321	0.327	**0.310**
	720	2.609	2.931	**1.104**	3.624	3.708	**2.195**	**0.417**	0.491	0.431	0.417	0.525	**0.414**	0.408	0.427	**0.408**
ETTh1	96	0.876	1.032	**0.535**	0.727	0.664	**0.632**	0.513	0.841	**0.470**	0.513	0.381	**0.379**	**0.384**	0.412	0.387
	192	0.919	1.083	**0.710**	0.903	0.949	**0.734**	0.534	0.786	**0.522**	0.534	0.452	**0.434**	0.436	0.461	**0.433**
	336	0.960	1.003	**0.805**	1.011	0.931	**0.887**	**0.588**	0.941	0.638	0.588	0.520	**0.501**	0.491	0.483	**0.470**
	720	1.030	1.183	**0.856**	0.992	1.012	**0.961**	**0.643**	1.052	0.735	0.643	0.646	**0.578**	0.521	0.509	**0.455**
ETTh2	96	2.802	2.762	**0.897**	1.357	1.548	**1.074**	0.476	0.427	**0.355**	0.476	0.339	**0.294**	0.340	0.319	**0.313**
	192	7.123	4.066	**2.030**	4.342	5.025	**2.268**	0.512	0.563	**0.436**	0.512	0.495	**0.380**	0.402	0.416	**0.398**
	336	6.621	3.078	**2.322**	4.922	4.786	**2.589**	0.552	0.625	**0.466**	0.552	0.625	**0.516**	0.452	0.464	**0.422**
	720	3.279	3.695	**2.502**	4.733	4.191	**2.516**	0.562	0.513	**0.448**	**0.562**	0.880	0.615	0.462	0.456	**0.426**

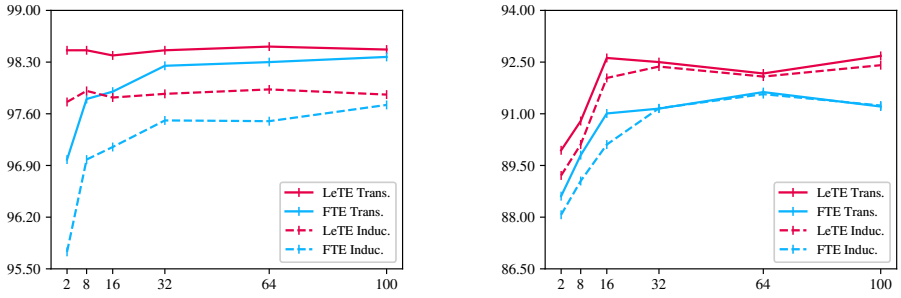
F.4. Dimensions of Time Embedding

We present the AUC-ROC results for Wikipedia/TGN and MOOC/TGN with different time embedding dimensions (for both FTE and LeTE) in Figure 6. To cover a broader range of scenarios, we also include the results for DyGFormer on the Wikipedia dataset in Figure 7.

G. More Experiments

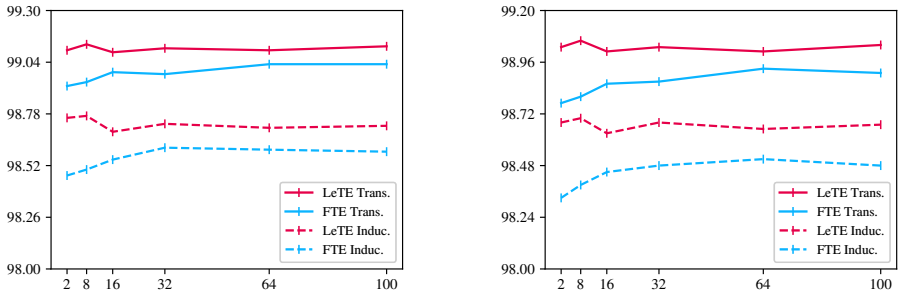
G.1. Statistic Analysis of the Complex Time Patterns in Data

Time-related data often contains mixed and complex patterns, which can primarily be categorized as periodic and non-periodic. To investigate the periodic and non-periodic patterns in the data, we analyze four dynamic graph datasets



(a) Wikipedia/TGN (b) MOOC/TGN

Figure 6. AUC-ROC results comparing different dimensions of the FTE and Spline-based LeTE on Wikipedia/TGN and MOOC/TGN.



(a) AP (b) AUC-ROC

Figure 7. AP and AUC-ROC results comparing different dimensions of the FTE and Spline-based LeTE on Wikipedia/DyGFormer.

using spectral entropy (Shannon, 1948). First, we normalize the time or time differences (since previous dynamic graph representation learning methods typically use time differences as inputs to the time encoding, we include this analysis here as well) for each node with more than five interactions, mapping the values to the range $[0, 1]$. We then treat each node's interaction times as a signal sequence. The spectral entropy for each node is computed as follows: We begin by applying the Fast Fourier Transform (FFT) to the normalized signal sequences: $X(f) = \text{FFT}(t_{\text{norm}})$, where $X(f)$ is the frequency-domain representation of the signal. Next, we calculate the magnitude of the frequency components $M(f) = |X(f)|$. The magnitudes are then normalized to form a probability distribution: $P(f) = \frac{M(f)}{\sum_f M(f)}$. Finally, the spectral entropy is computed as: $H(P) = -\sum_f P(f) \log P(f)$, which measures the uniformity of the frequency components. A lower entropy value indicates periodicity, while a higher entropy value suggests randomness.

We present the density plots of the spectral entropy in Figure 8. As shown in the figures, only a small portion of the nodes exhibit strong periodicity in their interaction times or time differences, while most nodes show high entropy, indicating non-periodic behavior. This suggests that capturing periodic patterns alone is insufficient; it is also important to model non-periodic patterns to enhance the efficiency and expressiveness of the time encoding.

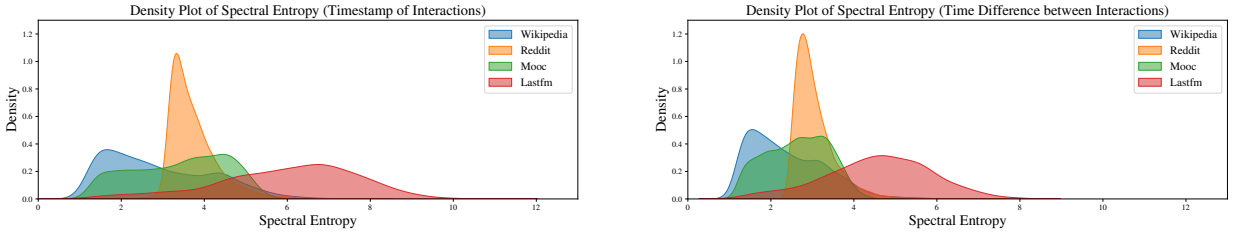


Figure 8. Density plots of spectral entropy for dynamic graph datasets.

G.2. Comparative Analysis of Different Variants of LeTE

We conducted a set of additional experiments on dynamic graph link prediction tasks, applying different variants of LeTE and comparing their downstream task performance, evaluated by AP and AUC-ROC. The results, presented in Tables 9 and 10, indicate that, in most cases, Combined LeTE achieves the best performance among the three variants of LeTE. This outcome is intuitive, as Combined LeTE leverages the strengths of both Fourier-based LeTE and Spline-based LeTE, enabling it to effectively model diverse time patterns.

Due to differences in the periodicity and non-periodicity of node interactions across datasets, the effectiveness of Fourier-based LeTE and Spline-based LeTE varies. Nonetheless, in most cases, both methods outperform the benchmark. This demonstrates that even when using only Fourier-based LeTE or Spline-based LeTE, they can effectively model different patterns, including periodic, non-periodic and mixed patterns, in the data.

Table 9. Comparing Functional Time Encoding (FTE), Fourier-based LeTE (F-LeTE), Spline-based LeTE (S-LeTE) and Combined LeTE (C-LeTE): Dynamic graph link prediction results in AP. The best results are in **bold**.

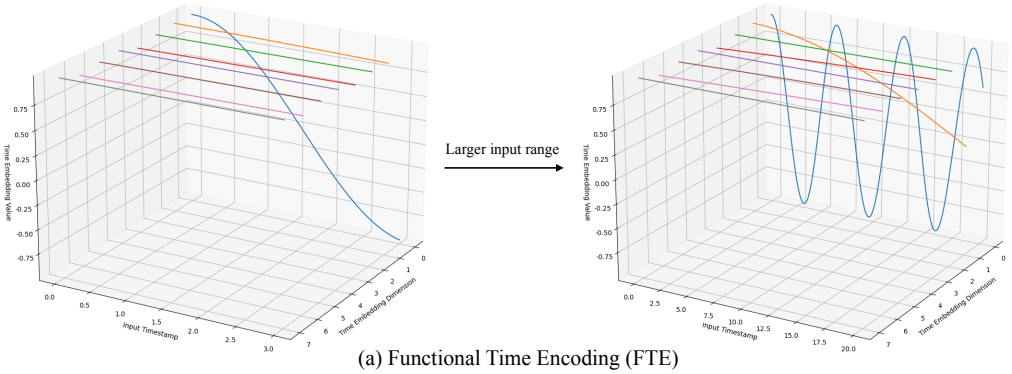
	AP	Wikipedia		Reddit		MOOC		LastFM	
	TE	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive
TGAT	FTE	96.95 ± 0.24	96.33 ± 0.26	98.53 ± 0.04	97.01 ± 0.05	85.34 ± 0.19	84.94 ± 0.04	72.73 ± 0.11	77.78 ± 0.13
	F-LeTE	96.82 ± 0.16	96.31 ± 0.13	98.54 ± 0.03	97.03 ± 0.02	85.25 ± 0.29	85.08 ± 0.29	72.31 ± 0.30	77.19 ± 0.38
	S-LeTE	97.54 ± 0.06	97.06 ± 0.05	98.56 ± 0.01	**97.05 ± 0.06**	**88.31 ± 0.10**	88.13 ± 0.28	75.68 ± 0.55	80.61 ± 0.42
	C-LeTE	**97.82 ± 0.09**	**97.34 ± 0.08**	**98.56 ± 0.01**	96.99 ± 0.06	88.30 ± 0.05	**88.37 ± 0.12**	**76.22 ± 0.25**	**81.32 ± 0.14**
TGN	FTE	98.45 ± 0.06	97.83 ± 0.04	98.63 ± 0.06	97.50 ± 0.07	89.15 ± 1.60	89.04 ± 1.17	77.07 ± 3.97	81.45 ± 4.29
	F-LeTE	98.57 ± 0.09	97.94 ± 0.08	98.66 ± 0.01	97.39 ± 0.07	90.04 ± 0.67	89.94 ± 0.49	77.58 ± 5.22	82.82 ± 6.53
	S-LeTE	98.55 ± 0.06	97.98 ± 0.08	**98.74 ± 0.00**	**97.65 ± 0.04**	91.09 ± 0.20	**90.87 ± 0.83**	82.26 ± 2.27	86.46 ± 0.77
	C-LeTE	**98.78 ± 0.07**	**98.19 ± 0.09**	98.74 ± 0.01	97.52 ± 0.12	**91.41 ± 0.55**	90.17 ± 0.69	**83.64 ± 2.00**	**87.55 ± 1.88**
TCL	FTE	96.47 ± 0.16	96.22 ± 0.17	97.53 ± 0.02	94.09 ± 0.07	82.38 ± 0.24	80.60 ± 0.22	67.27 ± 2.16	73.53 ± 1.66
	F-LeTE	97.83 ± 0.05	97.58 ± 0.10	97.74 ± 0.03	94.75 ± 0.20	83.40 ± 1.32	81.75 ± 1.43	**76.08 ± 0.79**	**80.68 ± 0.70**
	S-LeTE	97.33 ± 0.06	97.05 ± 0.13	**97.78 ± 0.03**	**94.99 ± 0.07**	83.87 ± 0.30	82.34 ± 0.31	69.92 ± 0.46	76.44 ± 0.42
	C-LeTE	**98.19 ± 0.04**	**97.89 ± 0.03**	97.75 ± 0.09	94.83 ± 0.20	**84.24 ± 0.10**	**82.72 ± 0.12**	72.76 ± 4.64	78.70 ± 3.67
DyG-Former	FTE	99.03 ± 0.02	98.59 ± 0.03	99.22 ± 0.01	98.84 ± 0.02	87.52 ± 0.49	86.96 ± 0.43	93.00 ± 0.12	94.23 ± 0.09
	F-LeTE	99.04 ± 0.01	98.66 ± 0.05	99.22 ± 0.01	98.85 ± 0.02	87.60 ± 0.26	87.15 ± 0.22	93.06 ± 0.05	94.11 ± 0.09
	S-LeTE	99.12 ± 0.01	98.72 ± 0.03	99.17 ± 0.10	98.78 ± 0.13	88.66 ± 0.20	88.37 ± 0.25	93.50 ± 0.12	94.57 ± 0.15
	C-LeTE	**99.13 ± 0.02**	**98.73 ± 0.00**	**99.24 ± 0.01**	**98.86 ± 0.01**	**88.70 ± 0.21**	**88.39 ± 0.15**	**93.64 ± 0.10**	**94.69 ± 0.12**

Table 10. Comparing Functional Time Encoding (FTE), Fourier-based LeTE (F-LeTE), Spline-based LeTE (S-LeTE) and Combined LeTE (C-LeTE): Dynamic graph link prediction results in AUC-ROC. The best results are in **bold**.

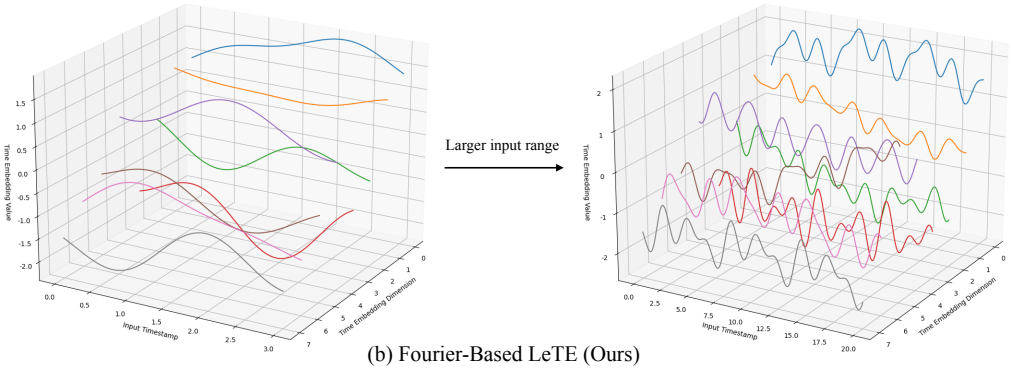
	AUC	Wikipedia		Reddit		MOOC		LastFM	
	TE	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive
TGAT	FTE	96.69 ± 0.26	95.95 ± 0.33	98.48 ± 0.04	96.90 ± 0.07	86.44 ± 0.24	86.04 ± 0.19	70.89 ± 0.10	76.11 ± 0.11
	F-LeTE	96.53 ± 0.17	95.94 ± 0.20	98.48 ± 0.03	96.91 ± 0.03	86.36 ± 0.29	86.18 ± 0.33	70.53 ± 0.18	75.61 ± 0.22
	S-LeTE	97.33 ± 0.07	96.76 ± 0.06	98.51 ± 0.02	**96.96 ± 0.05**	89.38 ± 0.14	89.14 ± 0.24	73.89 ± 0.51	79.17 ± 0.44
	C-LeTE	**97.63 ± 0.11**	**97.07 ± 0.10**	**98.51 ± 0.01**	96.88 ± 0.05	**89.46 ± 0.08**	**89.50 ± 0.05**	**74.24 ± 0.28**	**79.63 ± 0.14**
TGN	FTE	98.37 ± 0.07	97.72 ± 0.03	98.60 ± 0.06	97.39 ± 0.07	91.21 ± 1.15	91.24 ± 0.99	78.47 ± 2.94	82.61 ± 3.15
	F-LeTE	98.50 ± 0.10	97.86 ± 0.09	98.63 ± 0.02	97.27 ± 0.10	91.71 ± 0.65	91.53 ± 0.33	78.24 ± 4.86	83.16 ± 6.28
	S-LeTE	98.47 ± 0.06	97.86 ± 0.06	**98.72 ± 0.00**	**97.55 ± 0.05**	**92.68 ± 0.37**	**92.41 ± 0.73**	82.48 ± 2.17	86.49 ± 0.62
	C-LeTE	**98.73 ± 0.07**	**98.11 ± 0.10**	98.72 ± 0.02	97.43 ± 0.11	92.65 ± 0.48	91.27 ± 0.85	**83.94 ± 1.85**	**87.74 ± 1.75**
TCL	FTE	95.84 ± 0.18	95.57 ± 0.20	97.42 ± 0.02	93.80 ± 0.07	83.12 ± 0.18	81.43 ± 0.19	64.06 ± 1.16	70.84 ± 0.85
	F-LeTE	97.35 ± 0.07	97.14 ± 0.13	97.62 ± 0.03	94.42 ± 0.22	83.73 ± 0.92	82.11 ± 0.98	**70.17 ± 0.47**	**75.86 ± 0.44**
	S-LeTE	96.90 ± 0.08	96.62 ± 0.15	**97.68 ± 0.03**	**94.63 ± 0.05**	84.50 ± 0.20	83.02 ± 0.23	67.32 ± 0.50	74.37 ± 0.51
	C-LeTE	**97.84 ± 0.06**	**97.56 ± 0.02**	97.64 ± 0.08	94.49 ± 0.17	**84.73 ± 0.13**	**83.25 ± 0.23**	69.16 ± 3.21	75.85 ± 2.64
DyG-Former	FTE	98.91 ± 0.02	98.48 ± 0.03	99.15 ± 0.01	98.71 ± 0.01	87.91 ± 0.58	87.62 ± 0.51	93.05 ± 0.10	94.08 ± 0.08
	F-LeTE	98.94 ± 0.02	98.55 ± 0.03	99.16 ± 0.01	98.72 ± 0.03	88.09 ± 0.16	87.89 ± 0.14	93.09 ± 0.03	94.00 ± 0.04
	S-LeTE	**99.04 ± 0.01**	**98.67 ± 0.02**	99.08 ± 0.13	98.61 ± 0.19	**89.18 ± 0.21**	**89.16 ± 0.30**	93.56 ± 0.06	94.46 ± 0.06
	C-LeTE	99.04 ± 0.02	98.65 ± 0.01	**99.17 ± 0.00**	**98.74 ± 0.01**	89.17 ± 0.20	89.14 ± 0.09	**93.65 ± 0.07**	**94.52 ± 0.08**

G.3. Visualization and Interpretability

As mentioned in Section 3.2, the learned parameters of our proposed method can be used to reconstruct the time embedding feature map or the non-linear transformations. Since the previous time encoding method (FTE) uses fixed non-linear transformation functions, we present an example of its feature map and compare it with the Fourier-based LeTE in Figure 9.



(a) Functional Time Encoding (FTE)



(b) Fourier-Based LeTE (Ours)

Figure 9. Example of feature map for the FTE and Fourier-based LeTE at different dimensions (the total dimension is 8, the parameters weight are based on a learned TGN model on Wikipedia dataset).

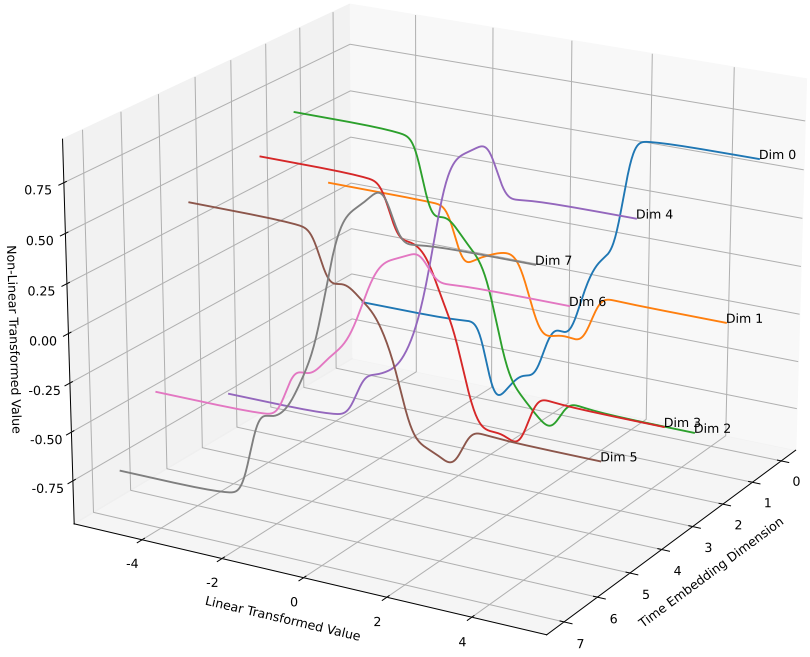


Figure 10. Example of non-linear transformation for Spline-based LeTE at different dimensions.

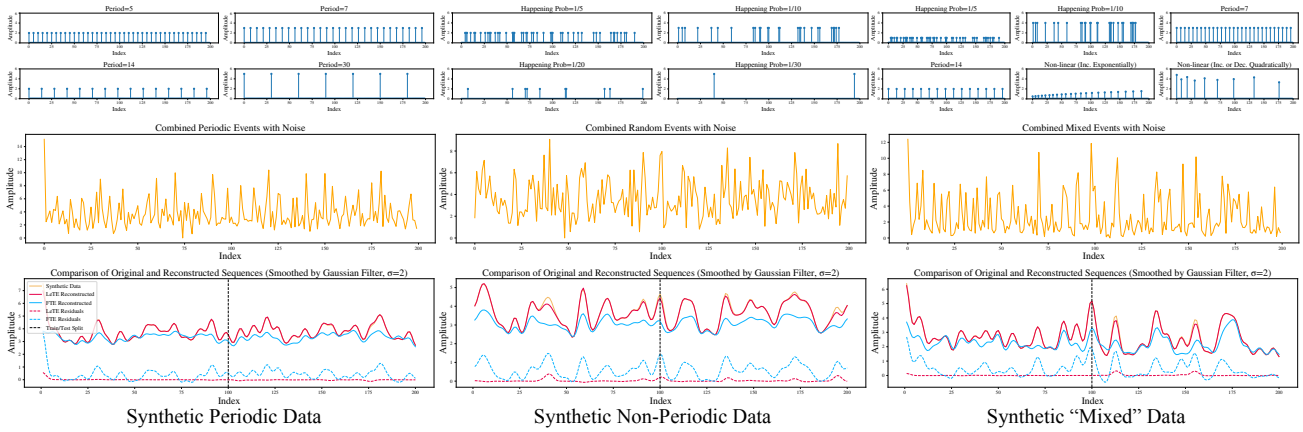


Figure 11. Capturing periodic, non-periodic and mixed patterns in synthetic data.

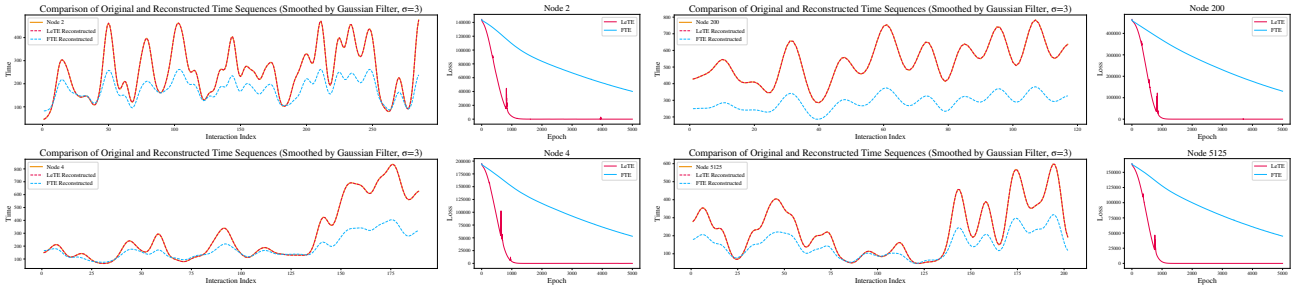


Figure 12. Capturing periodic, non-periodic and mixed patterns in real data.

We compare the local and global mappings of the two time encoding methods. As shown in the figure, our method captures richer and more detailed time patterns in different dimensions of the time encoding for local mappings. By contrast, the FTE method exhibits periodicity in only one dimension. This occurs because the learned frequency parameters in other dimensions are too small to capture sufficient periodicity locally. Similarly, for global mappings, strong periodicity is still observed in the feature map of our method, alongside varying degrees of non-periodicity. The FTE continues to exhibit periodicity in only one dimension. In other dimensions, the similar frequency parameters result in insufficient periodicity modeling and a lack of non-periodic pattern representation.

We also present the non-linear transformation sketches of Spline-based LeTE in Figure 10. The figure shows that the learned non-linear activation functions vary across dimensions, significantly enhancing the model's expressiveness. Additionally, it demonstrates the modeling of both periodic and non-periodic patterns at local and global scales. Since Combined LeTE is a combination of Fourier-based LeTE and Spline-based LeTE, it is intuitive that Combined LeTE inherits the same interpretability as its components. Furthermore, as our method supports the reconstruction of non-linear activation functions, it retains strong interpretability.

G.4. Capturing Periodic, Non-Periodic and Mixed Patterns in Data

Complex periodic and non-periodic patterns often coexist in real-world data, forming mixed time patterns. To demonstrate that our method surpasses previous methods in modeling such patterns, we design a mini reconstruction task using both synthetic data and real data from the Wikipedia dataset. Specifically, we construct an encoder-decoder model to reconstruct the data. The encoder is either (d-dimensional) our LeTE or the FTE, while the decoder is a simple linear layer mapping a d-dimensional vector to a 1-dimensional output. The reconstruction objective minimizes the MSE loss, which also quantifies the modeling capability of the time encodings. Additionally, the reconstructed time sequence plots visually indicate the models' ability to fit the data.

To isolate periodic and non-periodic patterns, we first generate synthetic data containing purely periodic signals, purely

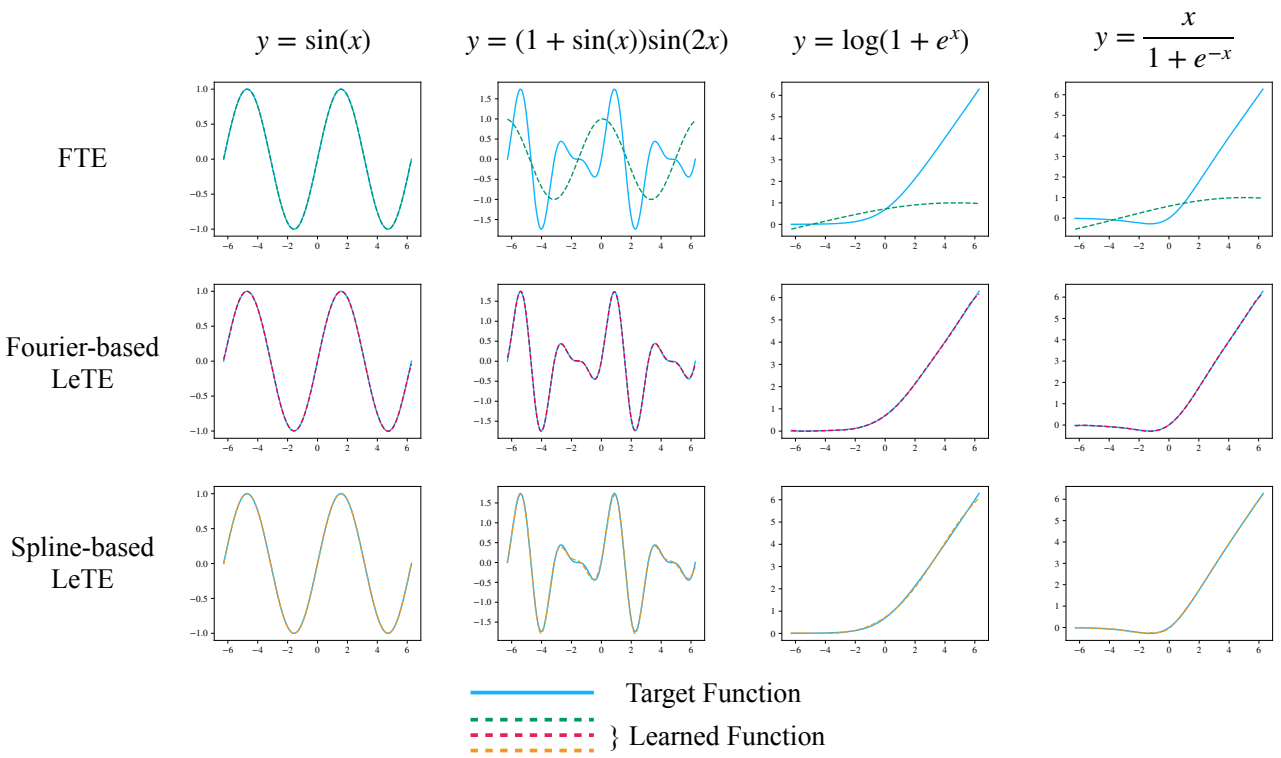


Figure 13. FTE, Fourier-based LeTE and Spline-based LeTE fitting different functions.

non-periodic signals, and mixed signals. These data is used to evaluate the performance of different encoders (LeTE or FTE). The ground truth and reconstructed sequences are shown in Figure 11. As illustrated, the FTE method performs reasonably well on periodic data but struggles with non-periodic and mixed data. In contrast, our method consistently outperforms FTE, demonstrating its capability to model both periodic and non-periodic patterns effectively.

Real-world data often exhibit complex combinations of periodic and non-periodic patterns, i.e., mixed patterns. To further evaluate our method, we randomly select 4 nodes' interaction sequences from the Wikipedia dataset and perform the same reconstruction experiments. The results are presented in Figure 12, where the time sequences are smoothed using a 1D Gaussian filter for clarity. As shown, the time sequences reconstructed using our LeTE align more closely with the original data compared to those reconstructed using FTE. Additionally, the loss of our LeTE is significantly lower than that of FTE, further validating our method's ability to capture complex periodic and non-periodic patterns in real-world data.

The experimental results show that, regardless of whether the sequence is periodic or non-periodic, our method consistently outperforms better. This is primarily due to the incorporation of learnable non-linear transformations into our time encoding approach.

G.5. Fitting Ability

We conduct a simple toy experiment to further demonstrate that both Fourier-based LeTE and Spline-based LeTE are capable of capturing different patterns. Consequently, Combined LeTE inherits this ability as well. To illustrate this, we generate a set of training data using 4 different non-linear transformation functions. Two of these functions are periodic: the sine function $y = \sin(x)$, and a more complex periodic function $y = (1 + \sin(x))\sin(2x)$. The other two functions are non-periodic: the Softplus activation function $y = \log(1 + e^x)$ (Misra, 2019), and the Swish activation function $y = \frac{x}{1+e^{-x}}$ (Radford et al., 2019).

We fit the data using simple 1-dimensional FTE, Fourier-based LeTE and Spline-based LeTE, evaluating their ability to capture complex patterns, including both periodic and non-periodic. The learned non-linear transformation functions are plotted in Figure 13. As shown in the figure, both Fourier-based LeTE and Spline-based LeTE successfully capture diverse

patterns. We also compare our method with FTE. Due to the fixed non-linear transformation functions used in FTE, it fails to capture the complex periodic and non-periodic patterns present in the data. These results demonstrate that our proposed LeTE has the capability to model complex patterns in data effectively and is more general than previous time encoding methods.

H. More Explanations and Examples about Interpretability

We choose to use a 4-dimensional Combined LeTE to present our analysis related to interpretability of LeTE. The experiments are conducted on the Wikipedia and MOOC datasets, with TGN and DyGFormer as backbone models. The training process and settings are consistent with those used in the main experiments. We will demonstrate the interpretability of our model from the following perspectives:

1. Reconstructing the learned non-linear transformation functions and plotting them to provide a clear and intuitive analysis.
2. Analyzing each dimension to understand what information it represents. Specifically, the first two dimensions of the time encoding are Fourier-based, while the last two dimensions are Spline-based.
3. Comparing different datasets under the same backbone model.
4. Comparing different backbone models' LeTE under the same dataset.
5. Comparing the plots of low- vs. high-dimensional LeTE to assess the impact of dimensionality on interpretability.

H.1. Reconstructing

As previously discussed, the previous time encoding methods exhibit a degree of interpretability by using fixed sinusoidal functions, which inherently reflect periodic patterns. However, this strong inductive bias also limits their expressiveness and generalization to complex or non-periodicity.

In contrast, our proposed LeTE is a fully learnable time encoding, and the learnable non-linear functions can still be reconstructed and visualized from learned parameters, allowing for interpretability analysis through function inspection.

We demonstrate this interpretability using a 4-dimensional Combined LeTE, trained on the Wikipedia/TGN. Figure 14 shows the learned transformation functions for each dimension. The first two dimensions are Fourier-based, and the last two are Spline-based.

H.2. Analyzing Each Dimension

Fourier-based: The Fourier coefficients explicitly encode frequency components, offering a clear, intuitive view of the captured periodicity. Compared to fixed sinusoidal functions, our learnable Fourier-based time encoding captures periodic patterns with finer granularity and greater flexibility, enabling the representation of both periodic signals and subtle non-periodicities within specific ranges.

For a single dimension, low-frequency components capture long-term trends, while high-frequency components focus on short-term fluctuations. This allows the model to encode both long-term dynamics and short-term variations simultaneously. As an example, we apply this to the Wikipedia dataset, which records editing activities, where nodes represent users or pages, and edges with timestamps capture editing events (frequency magnitude spectrum is shown in Figure 15, note that the inputs are time differences in this case).

Specifically, Dim 0 shows a strong high-frequency response. The learned coefficients include $\cos(3x\prime) : +0.29$, $\cos(4x\prime) : -0.16$, $\cos(5x\prime) : -0.42$. This suggests that Dim 0 is sensitive to short-term repetitive edits, i.e., high-frequency editing behavior.

Dim 1 captures low- to mid-frequency patterns, with large coefficients: $\sin(1x\prime) : +0.96$, $\sin(4x\prime) : +0.61$, $\cos(4x\prime) : +0.29$. These reflect longer-term periodic behaviors. For example, frequency-1 may correspond to daily or weekly editing cycles, while frequency-4 may capture sub-daily repeated interactions. This dimension may reflect user habits or regular community editing patterns. Thus, LeTE's Fourier-based dimensions not only retain the periodic interpretability of sine

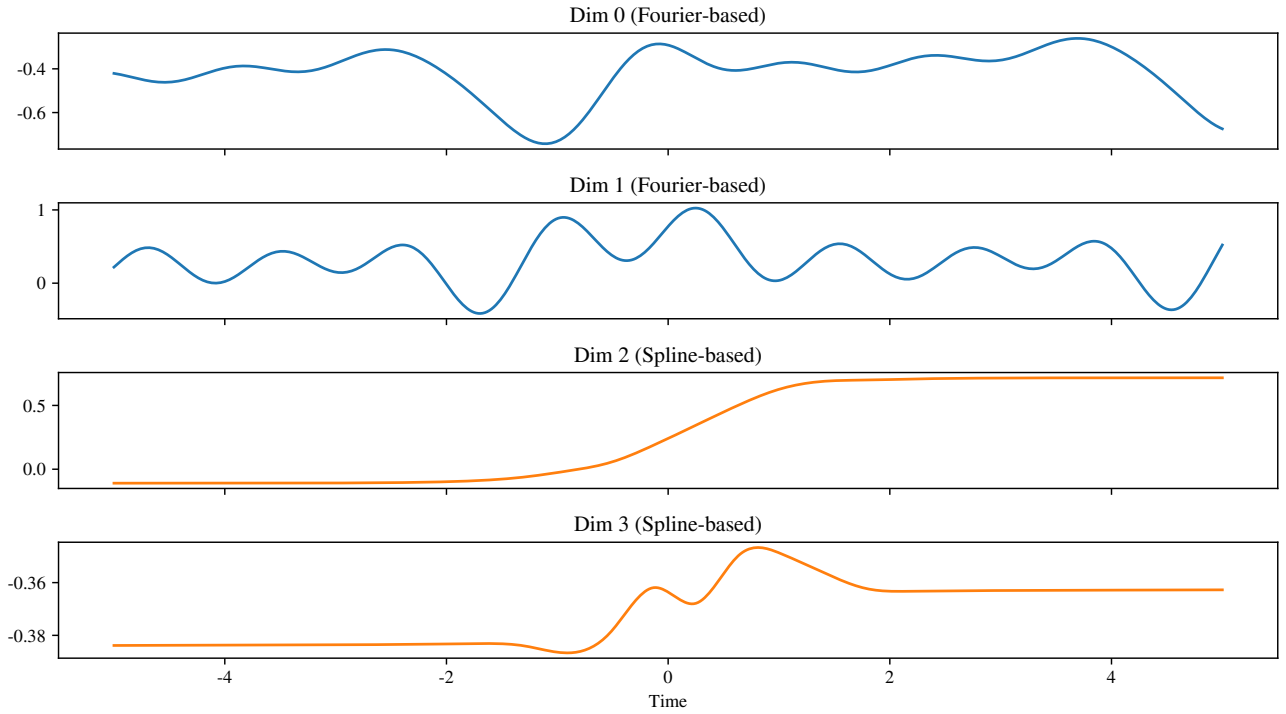


Figure 14. Plots of the four dimensions of the non-linear transformation functions of a 4-dimensional LeTE trained on Wikipedia/TGN. We further present the four functions here, the parameters are learned and read from the trained model:

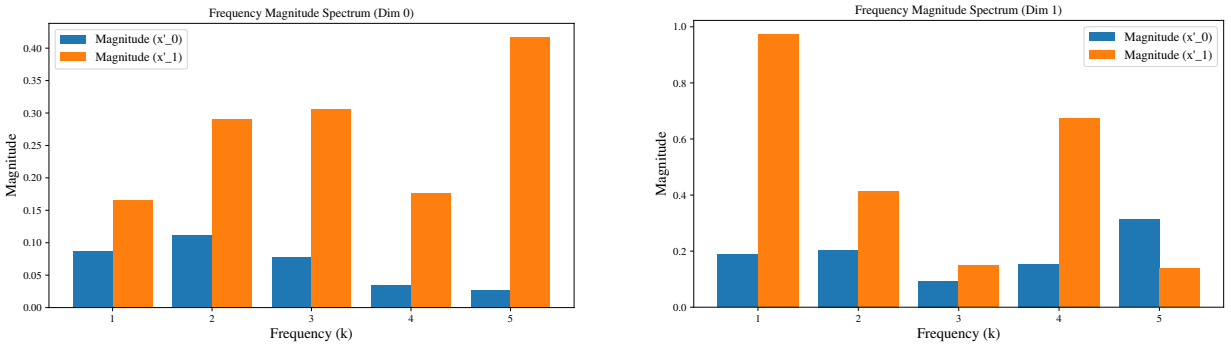
$f_0(x) = -0.0444 \cdot \cos(1 \cdot x'_0) + 0.0758 \cdot \sin(1 \cdot x'_0) + 0.0875 \cdot \cos(2 \cdot x'_0) + 0.0704 \cdot \sin(2 \cdot x'_0) + 0.0712 \cdot \cos(3 \cdot x'_0) - 0.0327 \cdot \sin(3 \cdot x'_0) + 0.0040 \cdot \cos(4 \cdot x'_0) - 0.0340 \cdot \sin(4 \cdot x'_0) + 0.0150 \cdot \cos(5 \cdot x'_0) - 0.0220 \cdot \sin(5 \cdot x'_0) + 0.0710 \cdot \cos(1 \cdot x'_1) + 0.1506 \cdot \sin(1 \cdot x'_1) - 0.1483 \cdot \cos(2 \cdot x'_1) + 0.2502 \cdot \sin(2 \cdot x'_1) + 0.2938 \cdot \cos(3 \cdot x'_1) + 0.0878 \cdot \sin(3 \cdot x'_1) - 0.1641 \cdot \cos(4 \cdot x'_1) + 0.0640 \cdot \sin(4 \cdot x'_1) - 0.4155 \cdot \cos(5 \cdot x'_1) + 0.0395 \cdot \sin(5 \cdot x'_1) - 0.0762,$

$f_1(x) = +0.1860 \cdot \cos(1 \cdot x'_0) + 0.0267 \cdot \sin(1 \cdot x'_0) + 0.1971 \cdot \cos(2 \cdot x'_0) - 0.0510 \cdot \sin(2 \cdot x'_0) - 0.0225 \cdot \cos(3 \cdot x'_0) - 0.0909 \cdot \sin(3 \cdot x'_0) - 0.0501 \cdot \cos(4 \cdot x'_0) + 0.1460 \cdot \sin(4 \cdot x'_0) + 0.0952 \cdot \cos(5 \cdot x'_0) + 0.2974 \cdot \sin(5 \cdot x'_0) - 0.1604 \cdot \cos(1 \cdot x'_1) + 0.9609 \cdot \sin(1 \cdot x'_1) + 0.2323 \cdot \cos(2 \cdot x'_1) - 0.3430 \cdot \sin(2 \cdot x'_1) - 0.0441 \cdot \cos(3 \cdot x'_1) - 0.1428 \cdot \sin(3 \cdot x'_1) + 0.2930 \cdot \cos(4 \cdot x'_1) + 0.6073 \cdot \sin(4 \cdot x'_1) + 0.0330 \cdot \cos(5 \cdot x'_1) - 0.1345 \cdot \sin(5 \cdot x'_1) - 0.0130,$

here, for both $f_0(x)$ and $f_1(x)$, $x'_0 = 1.0069 \cdot x + 0.0069$ and $x'_1 = 0.0054 \cdot x + 0.0108$.

$f_2(x) = +0.0013 \cdot B_0(x)$ (support: [-2.20,-1.80]) $+ 0.0047 \cdot B_1(x)$ (support: [-1.80,-1.40]) $- 0.0353 \cdot B_2(x)$ (support: [-1.40,-1.00]) $- 0.0321 \cdot B_3(x)$ (support: [-1.00,-0.60]) $- 0.0455 \cdot B_4(x)$ (support: [-0.60,-0.20]) $- 0.0273 \cdot B_5(x)$ (support: [-0.20, 0.20]) $+ 0.0211 \cdot B_6(x)$ (support: [0.20, 0.60]) $+ 0.0248 \cdot B_7(x)$ (support: [0.60, 1.00]) $+ 0.4133 \cdot \mathrm{Tanh}(x)$,

$f_3(x) = -0.0008 \cdot B_0(x)$ (support: [-2.20,-1.80]) $- 0.0072 \cdot B_1(x)$ (support: [-1.80,-1.40]) $- 0.0040 \cdot B_2(x)$ (support: [-1.40,-1.00]) $+ 0.0227 \cdot B_3(x)$ (support: [-1.00,-0.60]) $- 0.0067 \cdot B_4(x)$ (support: [-0.60,-0.20]) $+ 0.0248 \cdot B_5(x)$ (support: [-0.20, 0.20]) $+ 0.0165 \cdot B_6(x)$ (support: [0.20, 0.60]) $+ 0.0078 \cdot B_7(x)$ (support: [0.60, 1.00]) $+ 0.0100 \cdot \mathrm{Tanh}(x)$.



(a) Dim 0 (Fourier-based)

(b) Dim 1 (Fourier-based)

Figure 15. Frequency Magnitude Spectrum for the first two dimensions of the non-linear transformation functions of LeTE trained on Wikipedia/TGN. These two dimensions are Fourier-based.

functions but also exhibit richer frequency composition, allowing it to simultaneously capture both short-term bursts and long-term rhythms.

In addition, this approach could be extended to analyze more complex patterns. However, as our goal here is to present the underlying idea, we will not go deeper here.

Spline-based: The Spline-based functions offer complementary advantages, particularly for non-periodicity.

In our spline-based dimensions, where we applied a basis function (Tanh), if the weight of the basis function is higher, it may dominate a specific dimension—such as Dim 2 in Figure 14. However, there are other dimensions where splines dominate, such as Dim 3. To further clarify this, we combine the specific Wikipedia dataset and explain:

Dim 2: The output increases monotonically with time difference, indicating a time-decay-like effect — the longer the time since last edit, the stronger the encoding response. This may suggest the time encoding has learned that re-activation after long inactivity is a significant event in this specific case.

Dim 3: The function exhibits sharp peaks and local bumps, indicating that the model assigns particular importance to certain time intervals. These may correspond to known active editing windows or reaction delays. The sharpness of some coefficients suggests the model has captured rare but important temporal phenomena, such as one-off campaigns or anomaly spikes.

The Spline coefficients inherently capture local temporal features, indicating specific time intervals that the model considers critical or active. Sharp peaks coefficients within these curves suggest the occurrence of sudden events or anomalies. This local characteristic is advantageous for identifying rare phenomena.

H.3. Different Datasets

We reconstructed and plotted the four non-linear functions for a 4-dimensional LeTE trained on MOOC/TGN (shown in Figure 16). By comparing these results to those from the Wikipedia (Figure 14), it can be seen that the Dim 0 exhibit a lack of periodicity. From the reconstructed equations of Dim 0, the higher-frequency terms do have coefficients with some magnitude, but they are generally small. For instance, the coefficients of $\cos(5x\prime)$ and $\sin(5x\prime)$ are relatively small (e.g., -0.0134 and -0.0136), suggesting that their contribution is minimal and insufficient to generate significant fluctuations. As a result, the overall function primarily exhibits slow oscillations, making the plot appear to be predominantly non-periodic within a certain input window.

This observation aligns with the findings in Appendix G.1 and Figure 8, where the spectral entropy statistics also show that the Wikipedia exhibits stronger periodicity compared to the MOOC.

Thus, by comparing the non-linear functions of LeTE across different datasets, we can indirectly explore the periodic or non-periodic nature of the data present.

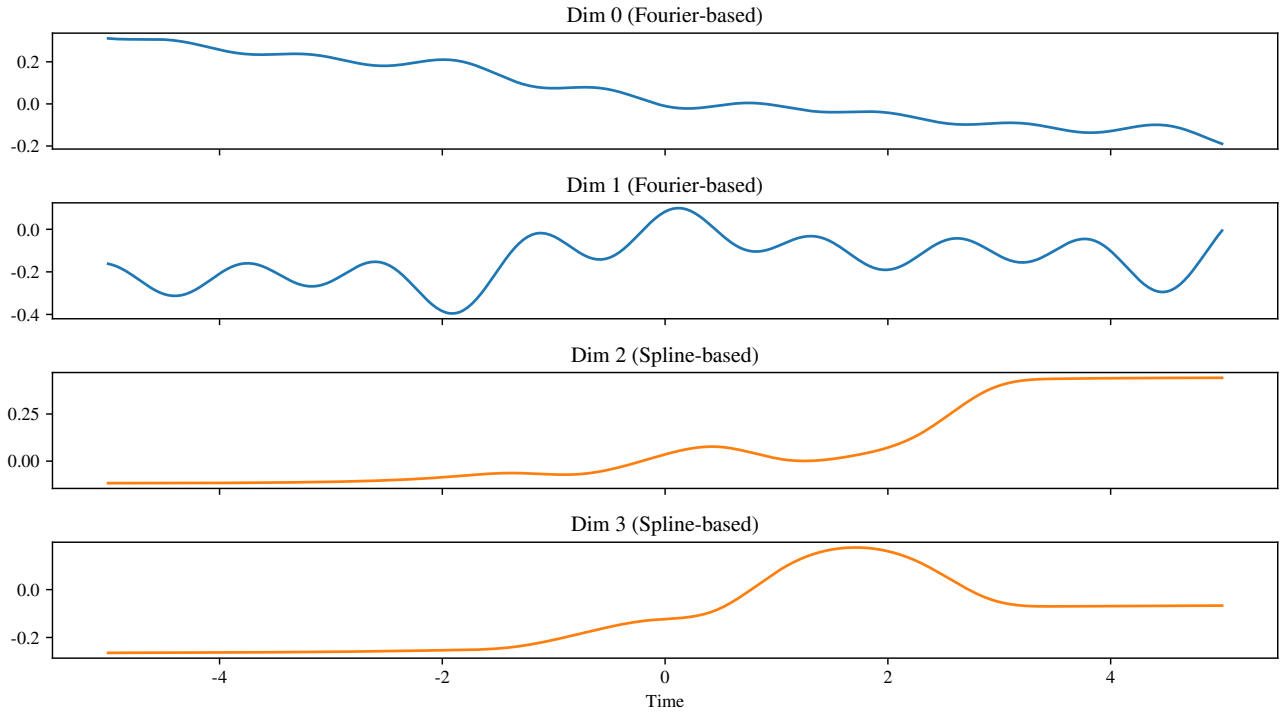


Figure 16. Plots of the four dimensions of the non-linear transformation functions of a 4-dimensional LeTE trained on MOOC/TGN. We further present the four functions here, the parameters are learned and read from the trained model:

$f_0(x) = -0.0186 \cdot \cos(1 \cdot x_0') - 0.0131 \cdot \sin(1 \cdot x_0') - 0.0173 \cdot \cos(2 \cdot x_0') + 0.0039 \cdot \sin(2 \cdot x_0') + 0.0019 \cdot \cos(3 \cdot x_0') + 0.0013 \cdot \sin(3 \cdot x_0') + 0.0032 \cdot \cos(4 \cdot x_0') - 0.0096 \cdot \sin(4 \cdot x_0') - 0.0134 \cdot \cos(5 \cdot x_0') - 0.0136 \cdot \sin(5 \cdot x_0') + 0.3140 \cdot \cos(1 \cdot x_1') + 0.5998 \cdot \sin(1 \cdot x_1') - 0.2781 \cdot \cos(2 \cdot x_1') + 0.4591 \cdot \sin(2 \cdot x_1') + 0.1376 \cdot \cos(3 \cdot x_1') + 0.2296 \cdot \sin(3 \cdot x_1') - 0.0880 \cdot \cos(4 \cdot x_1') - 0.0198 \cdot \sin(4 \cdot x_1') - 0.2621 \cdot \cos(5 \cdot x_1') + 0.0030 \cdot \sin(5 \cdot x_1') + 0.0016,$

$f_1(x) = +0.0816 \cdot \cos(1 \cdot x_0') + 0.0362 \cdot \sin(1 \cdot x_0') + 0.0588 \cdot \cos(2 \cdot x_0') - 0.0046 \cdot \sin(2 \cdot x_0') - 0.0199 \cdot \cos(3 \cdot x_0') - 0.0082 \cdot \sin(3 \cdot x_0') - 0.0121 \cdot \cos(4 \cdot x_0') + 0.0325 \cdot \sin(4 \cdot x_0') + 0.0591 \cdot \cos(5 \cdot x_0') + 0.0609 \cdot \sin(5 \cdot x_0') - 0.4787 \cdot \cos(1 \cdot x_1') + 1.1011 \cdot \sin(1 \cdot x_1') + 0.0467 \cdot \cos(2 \cdot x_1') - 0.4438 \cdot \sin(2 \cdot x_1') - 0.1300 \cdot \cos(3 \cdot x_1') - 0.2972 \cdot \sin(3 \cdot x_1') + 0.3344 \cdot \cos(4 \cdot x_1') + 0.1396 \cdot \sin(4 \cdot x_1') + 0.0593 \cdot \cos(5 \cdot x_1') - 0.0841 \cdot \sin(5 \cdot x_1') + 0.1166,$

here, for both $f_0(x)$ and $f_1(x)$, $x_0' = 0.9857 \cdot x + 0.0971$ and $x_1' = -0.0187 \cdot x + 0.0885$.

$f_2(x) = +0.0020 \cdot B_0(x)$ (support: [-2.20, -1.80])$+0.0071 \cdot B_1(x)$ (support: [-1.80, -1.40])$-0.0979 \cdot B_2(x)$ (support: [-1.40, -1.00])$-0.0916 \cdot B_3(x)$ (support: [-1.00, -0.60]) $- 0.1039 \cdot B_4(x)$ (support: [-0.60, -0.20]) $- 0.3442 \cdot B_5(x)$ (support: [-0.20, 0.20]) $- 0.3597 \cdot B_6(x)$ (support: [0.20, 0.60]) $- 0.3093 \cdot B_7(x)$ (support: [0.60, 1.00]) $+ 0.2827 \cdot \mathrm{Tanh}(x)$,

$f_3(x) = -0.0011 \cdot B_0(x)$ (support: [-2.20, -1.80])$-0.0099 \cdot B_1(x)$ (support: [-1.80, -1.40])$+0.0364 \cdot B_2(x)$ (support: [-1.40, -1.00])$+0.0712 \cdot B_3(x)$ (support: [-1.00, -0.60]) $+ 0.0169 \cdot B_4(x)$ (support: [-0.60, -0.20]) $+ 0.2372 \cdot B_5(x)$ (support: [-0.20, 0.20]) $+ 0.2886 \cdot B_6(x)$ (support: [0.20, 0.60]) $+ 0.2155 \cdot B_7(x)$ (support: [0.60, 1.00]) $+ 0.0947 \cdot \mathrm{Tanh}(x)$.

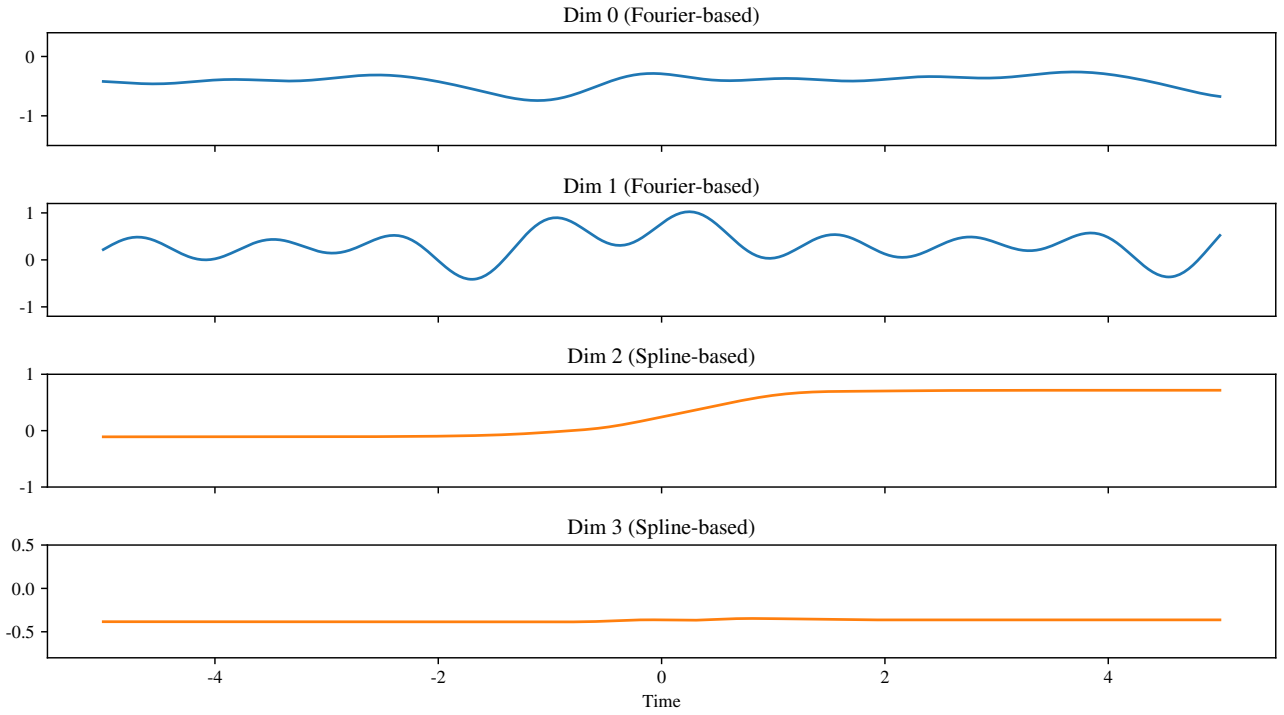


Figure 17. Plots of the four dimensions of the non-linear transformation functions of a 4-dimensional LeTE trained on Wikipedia/TGN. The y-axes have been scaled to the same level as 18 to facilitate a direct comparison. The reconstructed functions are the same as in the Figure 14.

H.4. Different Backbones

We provide plots of the same dataset trained with TGN and DyGFormer, shown in Figure 17 and Figure 18, with the y-axes set to the same level for each backbone to facilitate a direct comparison). As the figures demonstrate, despite using different backbone models, the learned functions exhibit similar trends and shapes for each dimension. This illustrates the stability of our method and makes the interpretability process more reliable.

Of course, there may be some detailed differences between LeTEs trained on different backbones. This is intuitively due to the presence of various influencing factors, such as the model architecture, the interaction of LeTE with other modules, the optimization process and etc. However, we can validate the idea by inspecting the plot in a simplified manner.

H.5. Comparing Lower- and Higher-dimensional LeTE

We further compare the lower- and higher-dimensional LeTE by reconstructing the non-linear functions and plotting them (please refer to and compare Figure 14 and 19). Intuitively, the higher-dimensional representation will provide more information. As seen from the plots, Dim 2 in Figure 19 is dominated by the basis function, partially losing the information captured by Dim 3 in Figure 14.

From the perspective of the reconstructed functions, for the Fourier-based dimensions, the LeTE with only one Fourier-based dimension has a single input transformation, $x\prime_0$, and all frequency components are computed based on this transformation. This means the LeTE encodes on a broader time scale (reminder: we used the Wikipedia dataset) and models the time difference variations of editing activities without distinguishing patterns at different scales. Since there is only one Fourier-based dimension, all frequency components are controlled by the same input transformation, making it harder for the model to interpret editing patterns at different time scales. In contrast, for the LeTE with two Fourier-based dimensions, each dimension has different input transformations ($x\prime_0$ and $x\prime_1$), enabling the model to capture more detailed editing behaviors at different scales. For example, Dim 0 might rely more on $x\prime_0$ (with a larger scaling factor), focusing on short-term fluctuations (high-frequency components), while Dim 1 might rely more on $x\prime_1$ (with a smaller scaling factor), focusing

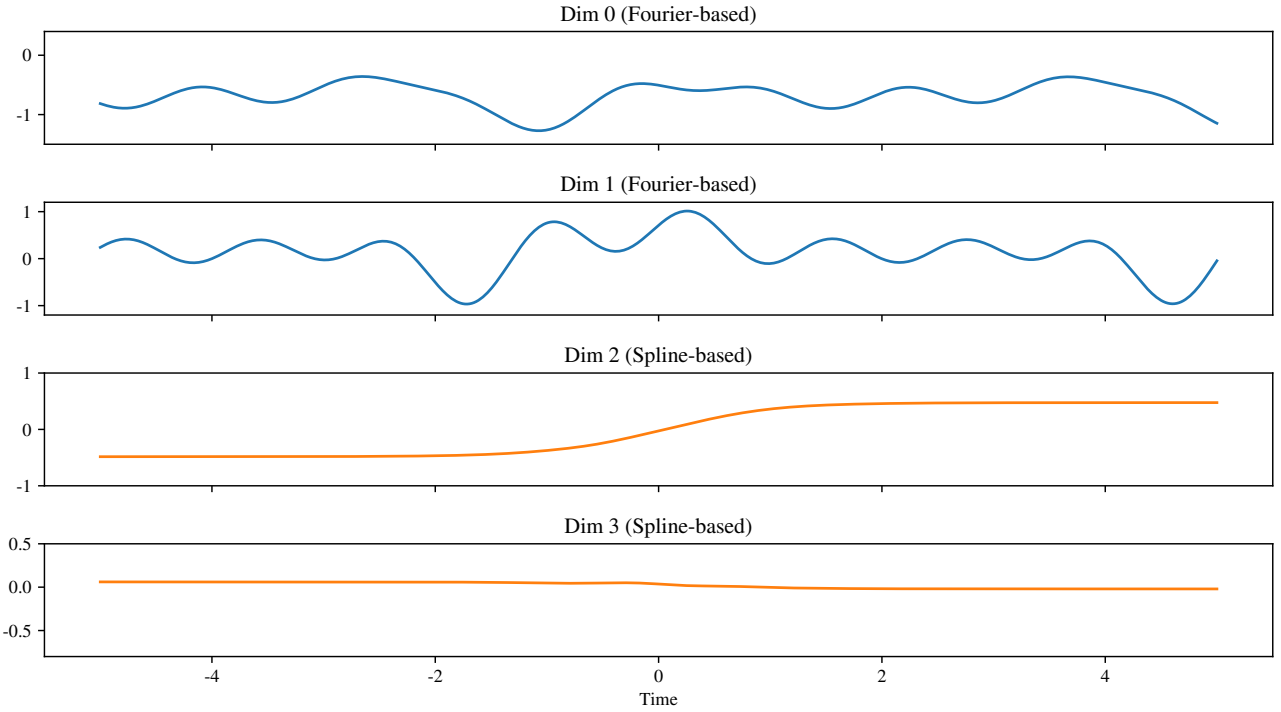


Figure 18. Plots of the four dimensions of the non-linear transformation functions of a 4-dimensional LeTE trained on Wikipedia/TGN. The y-axes have been scaled to the same level as 17 to facilitate a direct comparison. We further present the four functions here, the parameters are learned and read from the trained model:

$f_0(x) = -0.0727 \cdot \cos(1 \cdot x'_0) + 0.0704 \cdot \sin(1 \cdot x'_0) + 0.1529 \cdot \cos(2 \cdot x'_0) + 0.1720 \cdot \sin(2 \cdot x'_0) + 0.1554 \cdot \cos(3 \cdot x'_0) - 0.0320 \cdot \sin(3 \cdot x'_0) - 0.0385 \cdot \cos(4 \cdot x'_0) - 0.0220 \cdot \sin(4 \cdot x'_0) - 0.0086 \cdot \cos(5 \cdot x'_0) - 0.1107 \cdot \sin(5 \cdot x'_0) + 0.0758 \cdot \cos(1 \cdot x'_1) + 0.2909 \cdot \sin(1 \cdot x'_1) - 0.2430 \cdot \cos(2 \cdot x'_1) + 0.3166 \cdot \sin(2 \cdot x'_1) + 0.2097 \cdot \cos(3 \cdot x'_1) + 0.0334 \cdot \sin(3 \cdot x'_1) - 0.3308 \cdot \cos(4 \cdot x'_1) - 0.0664 \cdot \sin(4 \cdot x'_1) - 0.4329 \cdot \cos(5 \cdot x'_1) - 0.1351 \cdot \sin(5 \cdot x'_1) - 0.0009,$

$f_1(x) = +0.2232 \cdot \cos(1 \cdot x'_0) + 0.1531 \cdot \sin(1 \cdot x'_0) + 0.2963 \cdot \cos(2 \cdot x'_0) - 0.0852 \cdot \sin(2 \cdot x'_0) - 0.0409 \cdot \cos(3 \cdot x'_0) - 0.1263 \cdot \sin(3 \cdot x'_0) - 0.0938 \cdot \cos(4 \cdot x'_0) + 0.1772 \cdot \sin(4 \cdot x'_0) + 0.1431 \cdot \cos(5 \cdot x'_0) + 0.3435 \cdot \sin(5 \cdot x'_0) - 0.2795 \cdot \cos(1 \cdot x'_1) + 1.0935 \cdot \sin(1 \cdot x'_1) + 0.1297 \cdot \cos(2 \cdot x'_1) - 0.4577 \cdot \sin(2 \cdot x'_1) - 0.0711 \cdot \cos(3 \cdot x'_1) - 0.3519 \cdot \sin(3 \cdot x'_1) + 0.4151 \cdot \cos(4 \cdot x'_1) + 0.5552 \cdot \sin(4 \cdot x'_1) + 0.0696 \cdot \cos(5 \cdot x'_1) - 0.1653 \cdot \sin(5 \cdot x'_1) - 0.0245,$

here, for both $f_0(x)$ and $f_1(x)$, $x'_0 = 0.9936 \cdot x - 0.0016$ and $x'_1 = 0.0009 \cdot x - 0.0678$.

$f_2(x) = +0.0012 \cdot B_0(x)$ (support: [-2.20, -1.80]) $+ 0.0043 \cdot B_1(x)$ (support: [-1.80, -1.40]) $- 0.0096 \cdot B_2(x)$ (support: [-1.40, -1.00]) $- 0.0071 \cdot B_3(x)$ (support: [-1.00, -0.60]) $- 0.0197 \cdot B_4(x)$ (support: [-0.60, -0.20]) $- 0.0075 \cdot B_5(x)$ (support: [-0.20, 0.20]) $+ 0.0045 \cdot B_6(x)$ (support: [0.20, 0.60]) $+ 0.0038 \cdot B_7(x)$ (support: [0.60, 1.00]) $+ 0.4794 \cdot \text{Tanh}(x)$,

$f_3(x) = -0.0008 \cdot B_0(x)$ (support: [-2.20, -1.80]) $- 0.0074 \cdot B_1(x)$ (support: [-1.80, -1.40]) $+ 0.0026 \cdot B_2(x)$ (support: [-1.40, -1.00]) $+ 0.0294 \cdot B_3(x)$ (support: [-1.00, -0.60]) $- 0.0009 \cdot B_4(x)$ (support: [-0.60, -0.20]) $+ 0.0179 \cdot B_5(x)$ (support: [-0.20, 0.20]) $+ 0.0037 \cdot B_6(x)$ (support: [0.20, 0.60]) $+ 0.0031 \cdot B_7(x)$ (support: [0.60, 1.00]) $- 0.0395 \cdot \text{Tanh}(x)$.

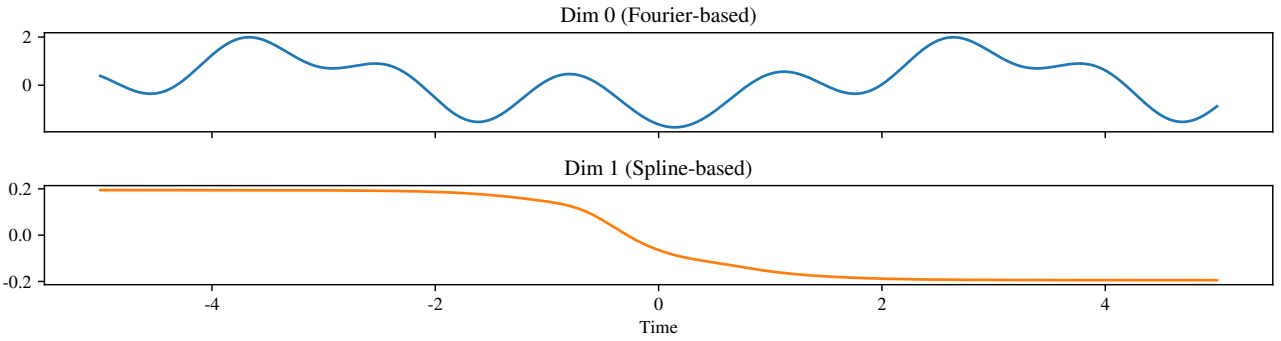


Figure 19. Plots of the two dimensions of the non-linear transformation functions of a 2-dimensional LeTE trained on Wikipedia/TGN. We further present the four functions here, the parameters are learned and read from the trained model:

$f_0(x) = -0.9560 \cdot \cos(1 \cdot (x')) + 0.3821 \cdot \sin(1 \cdot (x')) + 0.2430 \cdot \cos(2 \cdot (x')) - 0.3138 \cdot \sin(2 \cdot (x')) - 0.4807 \cdot \cos(3 \cdot (x')) - 0.1918 \cdot \sin(3 \cdot (x')) - 0.6372 \cdot \cos(4 \cdot (x')) - 0.2811 \cdot \sin(4 \cdot (x')) + 0.1555 \cdot \cos(5 \cdot (x')) + 0.0840 \cdot \sin(5 \cdot (x')) + 0.0334,$

here, $x' = 0.9963 \cdot x - 0.0092$.

$f_1(x) = -0.0020 \cdot B_0(x)$ (support: [-2.20,-1.80]) $- 0.0022 \cdot B_1(x)$ (support: [-1.80,-1.40]) $+ 0.0097 \cdot B_2(x)$ (support: [-1.40,-1.00]) $- 0.0636 \cdot B_3(x)$ (support: [-1.00,-0.60]) $- 0.0556 \cdot B_4(x)$ (support: [-0.60,-0.20]) $- 0.0123 \cdot B_5(x)$ (support: [-0.20, 0.20]) $- 0.0054 \cdot B_6(x)$ (support: [0.20, 0.60]) $- 0.0011 \cdot B_7(x)$ (support: [0.60, 1.00]) $- 0.1945 \cdot \mathrm{Tanh}(x)$.

more on long-term trends (low-frequency components). Thus, higher dimensions allow the model to handle editing behaviors at different time scales, providing higher interpretability.

Similarly, for the LeTE with only one Spline-based dimension, it primarily focuses on adjusting a single level, potentially describing how time affects editing behaviors. However, relying on just one Spline-based dimension may make it difficult to capture relatively complex time dynamics. For the LeTE with two Spline-based dimensions, the weights of the coefficients are more distributed, granting the overall LeTE stronger local adjustment capabilities. Moreover, since a dimension may be dominated by basis function or Spline functions, higher dimensions naturally have stronger expressive power.

Although higher-dimensional LeTEs offer stronger performance and better explain the information captured by the model, the interpretability analysis of such higher-dimensional LeTEs becomes more complex and may require a dimension-by-dimension analysis.

I. Limitation and Future Work

In this paper, we introduce LeTE, a general time encoding method. Generally, Combined LeTE offers better performance as it leverages the strengths of both Fourier-based LeTE and Spline-based LeTE, enabling it to effectively capture both the periodicity and non-periodicity of time. However, in practical scenarios, the choice of the hyperparameter p or among the three variants may depend on the characteristics of the data and the specific task requirements.

We also explore the impact of the time encoding dimension on downstream task performance. Similarly, selecting an appropriate time encoding dimension may vary depending on the data and tasks. Notably, we observe that even with a small dimension, LeTE can achieve acceptable results in downstream tasks.

Additionally, we mention that certain position encoding methods can be considered special cases of our approach. However, as position encoding is not the primary focus of this paper, we did not provide formal proofs. We believe that extending the ideas proposed in this paper to models that use position encoding could yield improved results, making this a promising direction for future research.

J. Code Implementation

The codes are available at a GitHub Repository.