Leto: Modeling Multivariate Time Series with Memorizing at Test Time

Anonymous Author(s)

Abstract

Modeling multivariate time series data has been at the core of machine learning research efforts across diverse domains. However, effectively capturing dependencies across both time and variate dimensions, as well as temporal dynamics, have made this problem extremely challenging in realistic settings. The recent success of sequence models, such as Transformers, Convolutions, and Recurrent Neural Networks, in language modeling and computer vision tasks, has motivated various studies to adopt them for time series data. These models, however, are either: (1) natively designed for a univariate setup, missing the the rich information that comes from the inter-dependencies of time and variate dimensions; (2) inefficient for long-range time series; and/or (3) propagating the prediction error over time. In this work, we present Leto, a native 2-dimensional memory module that takes the advantage of temporal inductive bias across time while maintaining the permutation equivariance of variates. Leto uses meta in-context memory modules to learn and memorize patterns across the time dimension, and simultaneously, incorporates information from other correlated variates, if needed. Our experimental evaluation shows the effectiveness of Leto on extensive and diverse benchmarks, including time series forecasting (short, long, and ultra-long), classification, and anomaly detection.

Keywords

Multivariate Time series, Time Series Forecasting, Time Series Classification, Transformers, Recurrent neural networks

ACM Reference Format:
Anonymous Author(s). 2018. Leto: Modeling Multivariate Time Series with Memorizing at Test Time. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 18 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Modeling multivariate time series data is a well-established problem in the literature with a diverse set of applications ranging from healthcare [56, 96] and neuroscience [13] to finance [40, 86], energy [124], transportation management [34], and weather forecasting [4, 87]. Classical shallow models—such as State Space Models [6, 49], ARIMA [11], SARIMA [20], Exponential Smoothing (ETS) [101]—have long been the de-facto mathematical models for time series prediction, modeling diverse complex patterns (such as seasonal and trend patterns). Deploying these models at scale in real-world settings remains challenging due to their reliance on manual data preprocessing, sensitive model selection, and inherently sequential, non-parallelizable computations. Additionally, these models often fail to capture (1) the inter-dependencies of different variates, and (2) the complex *non-linear* dynamics inherent to multivariate time series data.

The emergence of deep learning has shifted the focus of recent time series research away from traditional statistical methods toward deep neural network architectures such as Transformer-based [106, 124], recurrence-based [17, 19, 58, 83], and temporal convolutional-based [9, 76, 90] models. Despite the outstanding performance of Transformers [97] across various diverse domains [33, 80, 106], recent studies have highlighted their frequent suboptimal performance compared to even linear methods, mainly due to their inherent permutation equivariance that contradicts the causal nature of time series [115]. Additionally, their quadratic time and memory complexity is a notable bottleneck for their use in large-scale long real-world settings with long-range prediction horizon.

In recent years, modern linear Recurrent Neural Networks (RNNs) have attracted much attention as the linear alternative to Transformers, improving Transformers' training and inference efficiency while maintaining their effectiveness [59, 60, 84, 92]. While these models have shown promising performance on clean and tokenized data modalities such as languages, applying them to multivariate time series modeling is more challenging as: (1) Contrary to text, time series data can be non-stationary and highly noisy, as demonstrated by complex temporal patterns. Accordingly, the additive nature of such recurrent models can cause error propagation in their predictions over time, requiring additional careful parametrization or design to achieve good performance [17, 58]; (2) These models are inherently designed for a single sequence and so their use for time series data overlooks the importance of variate dependencies in modeling multivariate time series data [81, 116, 119]. Moreover, simply mixing the variates to take advantage of cross-variate information can hinder the performance in the general case as variate dependencies are not always useful in practice; e.g., when the target variate is not correlated with other covariates [26]. Therefore, a major goal of effective modeling of multivariate time series is to develop a model which can *adaptively* mix cross variate information over time when appropriate; (3) To capture both cross-time and cross-variate information, recently several studies aim to perform selective 2-dimensional recurrence across both variates [17, 58]. These models, however, are sensitive to the order of variates, missing the permutation equivariance of information across variates.

Contributions. In this paper, to mitigate the above-mentioned limitations in existing time series models, we present Leto, a novel 2-dimensional architecture based on two meta in-context memory modules—called time and variate memory modules—that learns how to memorize cross-time and cross-variate patterns at test time,

Conference acronym 'XX, Woodstock, NY
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

respectively. While LETO updates the time memory module using a recurrent rule to take advantage of its temporal inductive bias, it uses an attention-like (with `Softmax`) non-parametric memory module across variates to accurately consider their permutation equivariance property. To capture the dynamics of dependencies across variates, LETO needs to mix the states of both time and variate memories at each time stamps. However, the non-parametric nature of variate memory module makes it state-less, empowering the memory to learn the dynamics of variate dependencies across time. To overcome this challenge, LETO uses a parametric approximation of the non-parametric memory and expresses the `Softmax` attention using its Taylor series. To the best of our knowledge, LETO is the first native 2-dimensional hybrid model. In our experiments, we perform various evaluations and compare LETO with state-of-the-art time series models on diverse downstream tasks, including: (1) short-, long-, and ultra-long-term forecasting, (2) classification, and (3) anomaly detection tasks. We further demonstrate the effectiveness of LETO for longer horizons and support the significance of LETO's design by performing ablation studies.

2 Preliminaries, Background, and Related Work

In this section, we first discuss the notation that we use through the paper and then provide an overview of the background concepts and related studies. A more detailed discussions of the related work is in Appendix B. Also, we build our model based on concepts like: (1) meta learning, (2) learning to memorize, and (3) Titans [19]. We provide a detailed explanation of these topics in Appendix A.

Notation. We let matrix $= \{{}_1, \dots, {}_V\} \in \mathbb{R}^{V \times T \times d_{\text{in}}}$ denote a multivariate time series, where T and V are the number of time stamps and variates, respectively, and d_{in} is the feature dimension of the input (often $d_{\text{in}} = 1$). We use $x_{v,t} \in \mathbb{R}^{d_{\text{in}}}$ to refer to the value of the time series in v-th variate at time t. In this paper, we mainly focus on forecasting, classification, and anomaly detection. In forecasting tasks, given the historical series $= \{{}_1, \dots, {}_V\}$, the model aims to predict the next H time steps. For classification and anomaly detection, the task is to assign a label to the sequence, where anomaly detection is treated as a binary classification problem, labeling variate as "normal" or "anomaly".

Autoregressive Process. Autoregressive (AR) process is a basic but fundamental concept for time series modeling. An AR process models the causal nature of time series by writing each element as the linear combination of its past samples. Given $p \in \mathbb{N}$, ${}_k \in \mathbb{R}^d$, the linear autoregressive relationships between ${}_k$ and its past samples ${}_{k-1}, {}_{k-2}, \dots, {}_{k-p}$ is modeled as

$$ {}_k = \zeta_1 {}_{k-1} + \zeta_2 {}_{k-2} + \dots, \zeta_p {}_{k-p} \qquad \text{(AR}(p)\text{ Process)} $$

where $\zeta_1, \dots, \zeta_p$ are coefficients. Note that we can simply extend the above autoregressive formulation to the multivariate setting by letting coefficients be vectors, replacing the product with element-wise product.

Time Series Models. The complexity of time series data—characterized by higher-order structures, multivariate dependencies, and domain variability—presents key challenges for model development. Models must capture both local and long-range dependencies, selectively leverage relevant covariates, and scale efficiently to long sequences without relying heavily on domain-specific pre-processing. Additionally, scalability to long sequences remains a critical requirement for practical deployment. Classical statistical models, such as ARIMA [5] and STL [30], effectively address periodic and trend components but are fundamentally limited when it comes to modeling non-linear and complex dependencies. Early efforts to enhance time series forecasting with deep learning methods adopted recurrent neural networks (RNNs) [37] and their variants, such as Long Short-Term Memory (LSTM) networks [50] and Gated Recurrent Units (GRUs) [27], owing to their natural suitability for sequential data. Subsequently, temporal convolutional networks (TCNs) [9, 98, 103] were introduced, excelling at capturing local patterns through carefully designed receptive fields. The introduction of Transformer-based models [97] marked a significant advancement, enabling more effective modeling of both short- and long-term dependencies with enhanced scalability and predictive performance across a wide range of time series tasks [100]. transformer-based architectures such as [72, 91, 126] exemplify the power of attention mechanisms to capture both local and global temporal patterns, surpassing earlier convolutional and recurrent approaches. These models further integrate frequency-domain representations and downsampling strategies to enhance computational efficiency without compromising accuracy. However, the quadratic complexity of standard Transformers has led to optimization challenges [71, 106, 124, 126]. In response, patch-based methods have been proposed to improve efficiency in Transformer variants [81, 123]. Meanwhile, multilayer perceptrons (MLPs) have maintained popularity for time series forecasting due to their simplicity and direct mapping capabilities [36].

Test Time Memorization and Time Series Modeling. In recent years, there have been growing interest in understanding the underlying mechanisms of sequence models and unifying (a subset of) them through a single perspective [16, 69, 89, 95]. In this work, we discuss a connection between test time memorization models, time series modeling, and autoregressive processes. In the associative memory perspective of sequence models, given the incoming input data ${}_t$, a sequence model is defined as an associative memory, $(\cdot)$ that aims to learn a mapping between a set of keys (i.e., $\{{}_i\}_{i=1}^N$) and values (i.e., $\{\vec{}_i\}_{i=1}^N$) based on an objective function $\ell((\cdot); {}_t, \vec{}_t)$. For example, in recurrent neural networks, this memory module is their hidden state. Since this memory module is updated for each incoming data (at test time), it is often called a test time learner or test time memorizer. It is notable that the process of training such memory is a meta learning process [52], where in the inner-loop the corresponding parameters to memory are optimized, while the outer-loop optimizes other parameters in the neural network. For additional discussions on the meta learning process and how architectures like Transformers and recurrent models can be formulated as associative memory module, we refer the reader to Behrouz et al. [16] and our background discussion in Appendix A.

In practice, given input data ${}_t$, keys and values are defined as the linear projections of the input, i.e.,

$$ {}_t = W_k {}_t \qquad \text{and} \qquad \vec{}_t = W_v {}_t, \tag{1} $$

where $W_k \in \mathbb{R}^d$ Another interpretation of this framework for associative memory is to see ${}_t$ as the corrupted version of the input, and define (.) as a model that can reconstruct a projection of the input from the corrupted version. In this interpretation, objective $\ell((\cdot);_t, \vec{}_t)$ measures the ability of in reconstructing the input projection. Despite the equivalence of these two interpretations, the later provides an interesting connection between modeling time series data with sequence models. That is, modeling time series data, in which given a lookback window of p time stamps, the model aims to predict the next $h \geq 1$ steps, is equivalent to reconstructing a time series of $h + p$ time stamps from its corrupted version that masks its last h steps. This reconstruction perspective and its connection to sequence models allows us to design sequence models that are theoretically expressive and capable of modeling time series data. Despite this advantage, it is important to note that this formulation is limited to a single sequence. Hence, there still remains an unanswered question: *"How can we design a native 2-dimensional model that learns to map underlying patterns of 2D data?"*

3 Leto: Learning to Memorize at Test Time with 2-Dimensional Memory

To address this question, we present our model: Leto, a native 2-dimensional architecture that takes advantage of two separate memory modules, each of which learns how to memorize patterns across either time or variate dimensions.

3.1 How to Memorize 2-Dimensional Data?

As discussed earlier, while sequence modeling and its test time memorization perspective can be an effective paradigm for modeling time series data, its design is limited to single sequences. Thus, for 2-dimensional data like multivariate time series, two memory modules are needed, each of which *learns* how to memorize patterns across each dimension (either time or variate) at test time. However, having memory modules that simply memorize the training data can significantly hinder the performance of the model, due to overfitting and the fact that time series data at test time can be out-of-distribution (OOD). To this end, we use a meta in-context memory, where the model learns *how to memorize patterns at test time.* This memory does not directly memorize training data, but instead uses the underlying patterns in the training data to learn *what patterns* need to be memorized and what patterns need to be forgotten.

Cross Time Dynamic. For the sake of simplicity and to demonstrate the process of modeling cross-time patterns, we fix the variate to v and remove it from subscript whenever the context is clear. Accordingly, for the input sequence this is a meta learning problem on the memory parameters, in which memory aims to reconstruct the projection of the time series (i.e., $\vec{}_i = W_{vi}$) from its corrupted version (i.e., $_i = W_{ki}$). That is, given an internal objective $\ell(\cdot)$ that measures the quality of reconstruction, the process of training the model performs two loops:

1. *Inner Loop*: In this loop the memory is optimized to reconstruct the sequence from its corrupted version using an optimization algorithm such as gradient descent. Therefore, the memory update is defined as:
$$_t = \alpha_{t\,t-1} - \eta_t \nabla \ell(_{t-1}; _{v,t}), \tag{2}$$
Note that in the inner loop we only optimize the memory parameters; all other parameters are considered fixed in this loop.
2. *Outer Loop*: The outer loop is responsible for the training of the entire model for a specific downstream task such as forecasting, classification, or anomaly detection. In this process, while all parameters in the model are optimized, memory parameters are fixed.

Using a reconstruction loss, i.e., $\ell(;_t) = \|_t - \vec{}_t\|_2^2$, where $_t$ and $\vec{}_t$ are defined as Equation 1, gives us a memory module with delta update rule (recurrence) [89] as:

$$\begin{aligned} _t &= _{t-1} - \eta_t \nabla \ell(_{t-1}; _t) = _{t-1} - (_{t-1\,t} - \vec{}_t)_t^{\top} \\ \Rightarrow \; _t &= (\mathbf{I} - \eta_{t\,t\,t}^{\top})_{t-1} + \eta_t \, \vec{}_t \, _t^{\top}, \end{aligned} \tag{3}$$

where $(\mathbf{I} - _t\,_t^{\top})$ is the transition matrix from state $_{t-1}$ to $_t$ and $\vec{}_t{}_t^{\top}$ is the transformation of the input data. This linear recurrent process is equivalent to a linear dynamical system with non-diagonal transition matrix, which is more expressive than its counterpart dynamical systems with diagonal transition [17, 64, 83]. In our later design of Leto in Equation Variant 2, we further enhance the above formulation by incorporating a gating mechanism from the Titans architecture [19]. Therefore, the update rule can be written as:

$$_t = (\alpha_t \mathbf{I} - \eta_{t\,t\,t}^{\top})_{t-1} + \eta_t \, \vec{}_t \, _t^{\top}, \tag{4}$$

where α controls the retention from the previous state of the memory. When $\alpha \longrightarrow 1$, it fully retains the past state (equivalent to Equation 3) and when $\alpha \longrightarrow 0$ it erases the past state of the memory.

Cross Variate Dynamic. In the previous section, we discuss a neural memory module that learns how to memorize cross-time patterns. However, in multivariate time series data, the dependencies of variates can be a rich source of information, sometimes even more important than cross-time patterns [12, 72, 96]. To this end, we aim to design a memory module that can learn from and memorize cross-variate patterns. One simple approach is to transpose the input data (re-ordering time and variate dimension) and apply our memory module introduced in Equation 4 across variates. However, the main drawback of such a method is its sensitivity to the order of variates. That is, while the temporal inductive bias of recurrent models is effective for capturing temporal patterns, it is indeed a caveat that when sampling data, the order of elements are arbitrary. In multivariate time series data, the order of variates is often arbitrary and so we expect the model to produce the same output (or its corresponding permutation) when we change the order of variates. This property is called "permutation equivariance" (resp. "permutation invariant"), where the output of the model permutes the same (resp. remains the same) with the permutation of the input.

Transformers are one of the most powerful architectures with the permutation equivariance property [110, 114]. Although this property makes their direct applicability to time series data limited, it makes them a great choice of architectural backbone for use in learning the cross-variate information [72]. To this end, given the input data $= \{_1, \ldots, _V\} \in \mathbb{R}^{V \times T \times d_{\text{in}}}$, one can define $\tilde{} = ^{\top} = \{_1, \ldots \tilde{}_T\} \in \mathbb{R}^{T \times V \times d_{\text{in}}}$ and then pass it to a Transformer block to capture the cross-variate dependencies:

$$\mathrm{Y} = \texttt{Transformer}\left(\tilde{\ }\right). \tag{5}$$

While the above method can satisfy both (1) fusing information across variates, and (2) preserving the robustness to the permutation of variates, it only models cross-variate patterns and misses the dynamics of variates dependencies [17, 58].

3.2 Leto: A Native 2-Dimensional Memory System

Previously we discussed how one can design an effective memory module that learns how to map underlying patterns across time *or* variate dimensions in the data. A simple and commonly used method in the literature is to use two different modules, each for one of the dimensions, and then mix their outputs for the final prediction [3, 28]. That is, given input $\in \mathbb{R}^{V\times T\times d_{\text{in}}}$, one can use $\texttt{Module}_1(\cdot)$ and $\texttt{Module}_2(\cdot)$ to fuse information across time and variates, respectively, and then combine them for the final output:

$$\begin{aligned} Y_{\text{time}} &= \texttt{Module}_1(),\\ Y_{\text{variate}} &= \texttt{Module}_2(\tilde{\ }),\\ Y_{\text{output}} &= \texttt{Combine}\left(Y_{\text{time}}, Y_{\text{variate}}\right). \end{aligned} \tag{Variant 1}$$

Another commonly used method is to employ $\texttt{Module}_1(\cdot)$ and $\texttt{Module}_2(\cdot)$ in a sequential manner (instead of the above parallel manner). However, all these models treat each dimension separately and thus miss the inter-dependencies of time and variate dimensions at each state of the system, resulting in less expressive power in modeling time series data (see Theorem 3.1 for the details). To this end, we present a native 2-dimensional memory system that not only has the temporal inductive bias across time, but also has the permutation equivariance property across variates.

We use two memory modules $^{(1)}(\cdot)$ and $^{(2)}(\cdot)$ to learn the underlying mappings/patterns across time and variate dimensions, respectively. As discussed in section 2 and section 3, to design such memory modules it is appropriate to use a reconstruction objective $\ell(\cdot)$ for the memory and then optimize this objective with an optimization algorithm (such as gradient descent). However, to capture the inter-dependencies of dimensions at each step of optimization, it is necessary to fuse the information between the memory modules as well. Therefore, the state of each memory module not only depends on its time stamp, but it also depends on its variate. Given $= \{_1, \dots, _V\}$ as the input, and arbitrary $v \in \{1, \dots, V\}$ we define the update of cross-time memory, as:

$$^{(1)}_{t,v} = \underbrace{\alpha^{(1)}_{t,v}\ _{t-1,v} - \eta_{t,v}\nabla\ell(^{(1)}_{t-1,v}; _{t,v})}_{\text{cross-time dynamic}} + \underbrace{\beta^{(2)}_{t,v}\ _{t-1,v} - \gamma_{t,v}\nabla\ell(^{(2)}_{t-1,v}; _{t,v})}_{\text{cross-variate dynamic}} \tag{6}$$

where $\ell(^{(j)}_{t-1,v}; _{t,v}) = \|^{(j)}_{t-1,v}\ _{t,v} - \vec{\ }_{t,v}\|_2^2$ for $j \in \{1,2\}$ and $v \in \{1,\dots,V\}$ and:

$$_{t,v} = W_k\ _{t,v}, \qquad \text{and} \qquad \vec{\ }_{t,v} = W_v\ _{t,v}. \tag{7}$$

Expanding the gradient for the above formulation results in the recurrent update rule for the cross-time memory module as follows:

$$^{(1)}_{t,v} = (\alpha_{t,v}\mathbf{I} - \eta_{t,v}\ _{t,v}\ _t^{\top})\ _{t-1,v} + \eta_{t,v}\ \vec{\ }_{t,v}\ ^{\top}_{t,v} + (\beta_{t,v}\mathbf{I} - \gamma_{t,v}\ _{t,v}\ ^{\top}_{t,v})\ _{t-1,v} + \gamma_{t,v}\ \vec{\ }_{t,v}\ ^{\top}_{t,v}.$$

The above formulation demonstrates how to update the cross-time memory. To get the final output from this memory, we need to multiply it by the input data $_{t,v}$ to achieve the $_{t,v}$'s corresponding information in the memory: i.e., $\mathrm{Y}^{(1)}_{t,v} = ^{(1)}_{t,v}\ _{t,v}$. One can similarly define the recurrence for the cross-variate memory module $^{(2)}_{t,v}$ as:

$$^{(2)}_{t,v} = \underbrace{\theta^{(1)}_{t,v}\ _{t,v-1} - \lambda_{t,v}\nabla\ell(^{(1)}_{t,v-1}; _{t,v})}_{\text{cross-time dynamic}} + \underbrace{\mu^{(2)}_{t,v}\ _{t,v-1} - \omega_{t,v}\nabla\ell(^{(2)}_{t,v-1}; _{t,v})}_{\text{cross-variate dynamic}}. \tag{8}$$

However, it is still sensitive to the order of variates. This sensitivity to variate ordering comes from the parametric nature of gradient descent algorithm as its iterations requires a series of ordered steps. Therefore, the use of any other parametric optimizer can cause such sensitivity to the order. To overcome this issue, we use the non-parametric estimate of our objective. Interestingly, with a small modification and using Nadaraya-Watson estimators [38, 122], the non-parametric estimate of the objective is equivalent to softmax attention mechanism in Transformers [97], as also discussed in previous studies [16, 95]. Therefore, due to this theoretical connection, we use an attention module for the cross-variate information mixing. The final output of this block can simply be defined as:

$$\mathbf{Y}^{(2)}_{t,v} = \theta_{t,v}\underbrace{\texttt{Attention}\left(\{^{(1)}_{t,i}\ _{t,i}\}_{i=1}^{V}\right)}_{\text{cross-time dynamic}} + \mu_{t,v}\underbrace{\texttt{Attention}\left(\{_{t,i}\}_{i=1}^{V}\right)}_{\text{cross-variate dynamic}}. \tag{9}$$

Note that $^{(1)}_{t,i}\ _{t,i}$ provides the $_{t,i}$'s corresponding information in cross-time memory module and so the first term combines the cross-time dynamic of all variates at the same time. While computation of the final output for the cross-variate memory is clear, we need to access its memory (i.e., $^{(2)}_{t,v}$) to use in the update of cross-time memory (i.e., Equation 6). The memory of Transformers are known to be the pair of key and value matrices $(\mathbf{K}, \mathbf{V})$ in the attention mechanism [19, 21, 107, 121]. However, incorporating a pair of matrices into the recurrence update rule of Equation 6 is unclear and challenging. Therefore, we utilize a kernelized variant of attention, in which we replace $\texttt{Softmax}$ with a separable kernel $\phi(\cdot)$ [7, 59, 60] (see Appendix A for the corresponding background and detailed formulation). This allows us to concretely define the memory of the Transformer with keys and values of $\{\hat{}_i\}$ and $\{_i\}$ as [60]:

$$^{(2)}_{t,v} = \sum_{i=1}^{V}\ _{t,i}\,\phi(^{\top}_{t,i}). \tag{10}$$

The question about what would be the optimal kernel $\phi(\cdot)$ to use in the above formulation remains. To answer this, we recall the formulation of $\texttt{Softmax}$ attention that is proportional to $\texttt{softmax}(^{\top}_{t}\ _t)\ \vec{\ }_t$. To replace softmax $\texttt{softmax}(\cdot)$ with a separable kernel $\phi(\cdot)$, we can choose the kernel to approximate the exponential term in softmax with its Taylor series. Accordingly, we use the first four terms of the Taylor series of $\exp(\cdot)$ defined as:

$$\exp(x) \approx \phi(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!}. \tag{11}$$

Combining the prior expressions, we can define our native 2-dimensional update rule as:

$$^{(1)}_{t,v} = \underbrace{\alpha^{(1)}_{t,v}\ _{t-1,v} - \eta_{t,v}\nabla\ell(^{(1)}_{t-1,v}, _{t,v})}_{\text{cross-time dynamic}} + \underbrace{\beta^{(2)}_{t,v}\ _{t-1,v} - \gamma_{t,v}\nabla\ell(^{(2)}_{t-1,v}, _{t,v})}_{\text{cross-variate dynamic}},$$

$$^{(2)}_{t,v} = \sum_{i=1}^{V} {}_{t,i}\phi(\hat{}^{\top}_{t,i}), \qquad \text{where } \phi(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!}. \tag{Variant 2}$$

Note that in the above formulation $_i$ and $\hat{}_i$ are keys and values of the Transformer block, coming from the keys and values of the cross-variate dynamic attention mentioned in Equation 9. In the next theorem, we show that this inter-connectivity of these two memories can enhance the expressive power of model, compared to two separate memory modules:

Theorem 3.1. *Let* $\texttt{Module}_i(\cdot)$ *be linear recurrent models, then inter-connected memory modules (i.e., Equation Variant 2) can express full-rank kernels with* $O(1)$ *parameters, while independent memory systems (i.e., Equation Variant 1) require at least* $O(N)$ *parameters to express matrix with rank* N.

3.3 Leto Model Design

While our recurrence formulation is theoretically motivated to capture both cross-time and variate dependencies, in practice, its training can be difficult due its recurrent nature, potentially limiting parallelizable training. In this section, we discuss the architectural details in Leto and present a fast parallelizable training approach. Figure 1 illustrates the architectural design of Leto.

Parallelizable Training. Despite the recurrent nature of Leto, in this section, we build upon the training algorithms of Sun et al. [95] and Behrouz et al. [19] and present a parallel training process for our model. To begin, given a variate v, we divide its corresponding time series $\{_{1,v}, \dots, _{T,v}\}$ with length T into C subsequences of length $b = \frac{T}{C}$, each of which is represented by $\mathcal{S}_i = \{_{(i-1)b+1,v}, \dots, _{ib,v}\}$. Recall that the cross-variate dynamic term in Equation 9 is independent of time and variate states in our formulation and thus can be computed in advance. Note that the training procedure for the attention module is highly parallelizable. Given the output of the attention module, we can also calculate all the states of $^{(2)}$ memory using Equation 10. Therefore, we can calculate the gradient term with respect to $^{(2)}$ in (Variant 2), all in advance. Having the states of $^{(2)}$ and its corresponding gradient terms, we have calculated the cross-variate dynamic term in (Variant 2) in advance and so we only need to compute the cross-time dynamic term in a parallelizable manner. To this end, following the algorithms of Sun et al. [95] and Behrouz et al. [19], we approximate the gradient term $\nabla\ell(^{(1)}_{t-1,v}, _{t,v})$ with $\nabla\ell(^{(1)}_{t',v}, _{t,v})$, in which t' is the last state of the memory in the previous chunk, i.e., $t' = \lfloor \frac{t}{b} \rfloor \times b$. Therefore, we can calculate the gradients of each chunk in advance, making the recurrence linear, which is highly parallelizable. For a detailed discussion of parallelizable training see Appendix C.

Thus, we can parallelize the training process for each variate and by scanning the variates from top to bottom, we can encode all the states in the multivariate time series. We note that the training complexity is linear across time and is dominated by the attention module's complexity across variates.

4 Experiments

Goals and Baselines. In this section, we evaluate Leto on a wide range of time series tasks, comparing with the state-of-the-art multivariate time series models [31, 32, 68, 70, 72, 76, 83, 99, 102, 104, 106, 111, 117, 123, 126] on forecasting: long, ultra-long, and short term, classification, and anomaly detection tasks. In Section 4.2, we evaluate the significance of the Leto's components by performing ablation studies. Detailed dataset descriptions, complete experimental results, error bars, visualization of predictions, hyperparameters, metric descriptions, additional experimental results on the effect of lookback and other design choices are provided in Section E of the Appendix. Please note that we control the effect of parameters and all models use the same number of parameters.

4.1 Main Results: Classification and Forecasting

Long-Term Forecasting. We conduct experiments on the long-term forecasting tasks using commonly used benchmark datasets used by Zhou et al. [124]. The average performance across different horizons is summarized in Table 1. Leto consistently delivers strong results across different datasets, highlighting its robustness compared to recurrent, convolutional, SSM, and Transformer-based models.

Ultra Long-term Forecasting. We further extend the evaluation to ultra-long-range forecasting on the same benchmark datasets [124] to observe the effectiveness of Leto in longer horizons. The tasks on the left side of the Table 2 retain the same interpretation as in the standard long-term forecasting setting. The results in Table 2 demonstrate Leto's ability to capture long-term dependencies from extremely long historical inputs, maintaining its strong performance across various extended prediction horizons.

Classification and Anomaly Detection. We evaluate the performance of Leto on 10 multivariate datasets from the UEA Time Series Classification Archive [8] (see Figure 2 and Table 12). For anomaly detection, which is typically treated as a binary classification task, we conduct experiments on five widely-used benchmarks: SMD [93], SWaT [78], PSM [1], and SMAP [54] (see Figure 2 and Table 11). For each benchmark, we compare Leto against state-of-the-art methods for each respective task.

Short-Term Forecasting. Our evaluation on short-term forecasting tasks using the M4 benchmark datasets [43] is reported in Table 3 (with the full results provided in Table 8). We fix the input length to twice the prediction length and calculate Symmetric Mean Absolute Percentage Error (SMAPE), Mean Absolute Scaled Error (MASE), and Overall Weighted Average (OWA) as the evaluation metrics. The results demonstrate the strong performance of Leto compared to current baselines.

4.2 Ablation Study

To validate the effectiveness of our model design, we perform an ablation study on long-term forecasting tasks with averaged across 5 runs over the ETT, Weather, and Exchange datasets by removing key architectural components - see Table 4. The first row reports

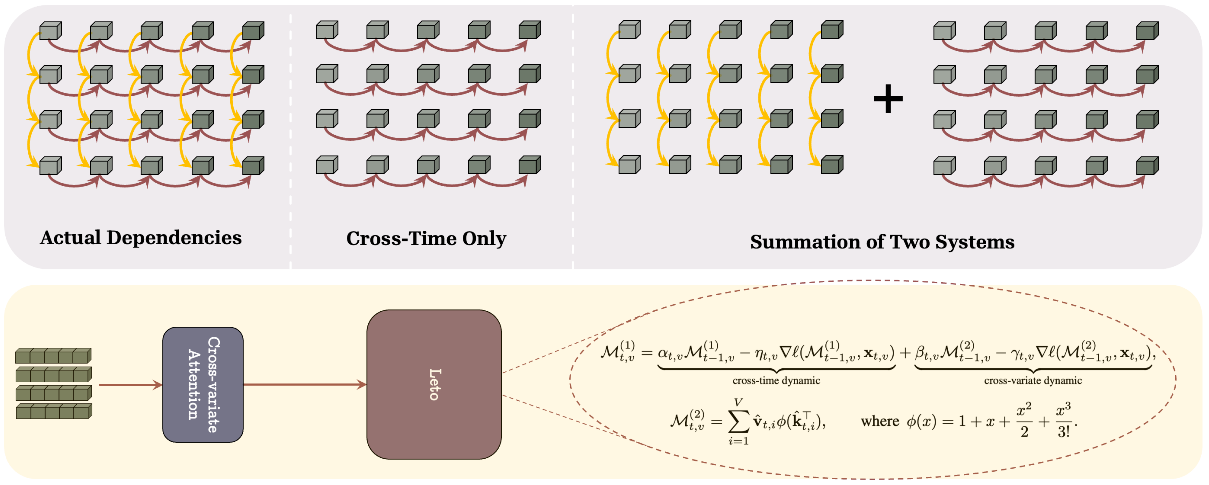
Actual Dependencies
Cross-Time Only
Summation of Two Systems
+
Cross-variate Attention
Leto
$\mathcal{M}^{(1)}_{t,v} = \underbrace{\alpha_{t,v}\mathcal{M}^{(1)}_{t-1,v} - \eta_{t,v}\nabla\ell(\mathcal{M}^{(1)}_{t-1,v}, \mathbf{x}_{t,v})}_{\text{cross-time dynamic}} + \underbrace{\beta_{t,v}\mathcal{M}^{(2)}_{t-1,v} - \gamma_{t,v}\nabla\ell(\mathcal{M}^{(2)}_{t-1,v}, \mathbf{x}_{t,v})}_{\text{cross-variate dynamic}},$
$\mathcal{M}^{(2)}_{t,v} = \sum_{i=1}^{V} \hat{\mathbf{v}}_{t,i}\phi(\hat{\mathbf{k}}^{\top}_{t,i}),$ where $\phi(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!}.$

Figure 1: An Overview of Leto's Architecture: We define two inter-connected memory blocks M^1, M^2 corresponding to time and variate axes, where the recurrence is updated by fusing together both cross-time and cross-variate information, using an approximation of softmax attention for M^2.

Table 1: Average performance on long term forecasting tasks over four prediction lengths: {96, 192, 336, 720}. A lower MAE and MSE indicates a better prediction. As a convention for all experimental results, best performance is highlighted in red, and the second-best is underlined.

Models	Leto (Ours)		TimeMixer		Simba		ModernTCN		iTransformer		RLinear		PatchTST		Crossformer		TiDE		TimesNet		DLinear	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.347**	**0.375**	0.381	0.385	0.383	0.396	0.351	0.381	0.407	0.410	0.414	0.407	0.387	0.400	0.513	0.496	0.419	0.419	0.400	0.406	0.403	0.407
ETTm2	**0.249**	**0.302**	0.275	0.323	0.271	0.327	0.253	0.314	0.288	0.332	0.286	0.327	0.281	0.326	0.757	0.610	0.358	0.404	0.291	0.333	0.350	0.401
ETTh1	**0.393**	**0.401**	0.447	0.440	0.441	0.432	0.404	0.420	0.454	0.447	0.446	0.434	0.469	0.454	0.529	0.522	0.541	0.507	0.458	0.450	0.456	0.452
ETTh2	**0.318**	0.381	0.364	0.395	0.361	0.391	0.322	0.379	0.383	0.407	0.374	0.398	0.387	0.407	0.942	0.684	0.611	0.550	0.414	0.427	0.559	0.515
Exchange	**0.297**	0.364	0.391	0.453	0.298	**0.363**	0.302	0.366	0.360	0.403	0.378	0.417	0.367	0.404	0.940	0.707	0.370	0.413	0.416	0.443	0.354	0.414
Traffic	0.408	**0.267**	0.484	0.297	0.493	0.291	**0.398**	0.270	0.428	0.282	0.626	0.378	0.481	0.304	0.550	0.304	0.760	0.473	0.620	0.336	0.625	0.383
Weather	**0.216**	**0.253**	0.240	0.271	0.255	0.280	0.224	0.264	0.258	0.278	0.272	0.291	0.259	0.281	0.259	0.315	0.271	0.320	0.259	0.287	0.265	0.317
ECL	**0.149**	**0.247**	0.182	0.272	0.185	0.274	0.156	0.253	0.178	0.270	0.219	0.298	0.205	0.290	0.244	0.334	0.251	0.344	0.192	0.295	0.212	0.300

Table 2: Average performance on Ultra long-term forecasting tasks (MSE / MAE)

Dataset	Metric	Leto		MICN		TimesNet		PatchTST		DLinear		FiLM		FEDformer		Autoformer		Informer	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	720–1440	0.4782	0.5614	1.0460	0.7765	0.6119	0.5962	0.8243	0.6704	0.4923	0.5473	**0.4730**	0.5336	0.4833	0.5393	1.4957	0.9533	0.5064	**0.5317**
ECL	1440–1440	**0.4639**	**0.5387**	0.8262	1.2207	0.5720	0.5712	0.9053	0.7328	0.5146	0.5615	0.4849	0.5429	0.5142	0.5571	1.7873	1.0283	0.7247	0.6920
	1440–2880	**0.6047**	**0.5868**	2.8936	1.3717	0.7683	0.6846	1.1282	0.8087	0.8355	0.7193	0.6847	0.6493	3.9018	1.5276	1.2867	0.8878	0.6152	0.5953
	720–1440	0.1672	0.2431	0.2876	0.3916	0.1882	0.2656	0.1904	0.2685	0.1639	**0.2412**	**0.1638**	0.2448	0.2753	0.3650	0.3104	0.4095	0.7614	0.6496
Traffic	1440–1440	**0.1521**	0.2497	0.2905	0.3923	0.2081	0.2712	0.1917	0.2764	0.1590	**0.2411**	0.1602	0.2437	0.2848	0.3681	0.2970	0.3999	0.7375	0.6414
	1440–2880	**0.1425**	0.2433	0.2823	0.3874	0.1560	**0.2409**	0.1819	0.2761	0.1550	0.2421	0.1744	0.2693	0.2952	0.3844	0.3035	0.3982	0.9408	0.7618
	720–1440	**0.1331**	**0.2943**	0.4640	0.5836	0.1391	0.3049	0.3708	0.4906	0.2952	0.4370	0.2949	0.4388	0.1768	0.3409	0.3298	0.4741	0.1378	0.3051
ETTh1	1440–1440	**0.1359**	0.3120	0.5188	0.6075	0.1404	**0.3093**	0.4475	0.5392	0.2200	0.3714	0.3226	0.4678	0.1928	0.3576	0.3618	0.5507	0.1402	0.3192
	1440–2880	**0.2591**	0.3949	0.7591	0.7215	0.2732	0.4094	0.9617	0.8072	0.3773	0.4794	0.3624	0.4705	0.2627	**0.3754**	0.3177	0.4733	0.3495	0.4111

the Leto's performance, while the second row removes the Cross Attention block, the third row removes the the linear attention mechanisms, and the fourth row removes the the weights for the final gating between each block. The results demonstrate that Leto with all components yields the strongest performance. Notably, the results without the linear attention component and Transformer Block perform the worst, highlighting the importance of maintaining separate time and variate memories, and including both in the recurrence.

5 Conclusion

In this paper, we present Leto, a native 2-dimensional memory module that takes the advantage of temporal inductive bias across time while maintaining the permutation equivariance of variates. Leto uses a meta in-context memory module to learn and memorize patterns across time dimension, and simultaneously, incorporates information from other correlated variates, if it is needed. Our experimental and theoretical results support the effectiveness of Leto

Table 3: Average performance on short-term forecasting tasks on the M4 dataset. A lower SMAPE, MASE, and OWA indicate better prediction. * is an abbreviation of the "former" term.

Models		LETO (Ours)	ModernTCN 2024	TimeMixer 2024	PatchTST 2023	TimesNet 2023	N-HiTS 2022	N-BEATS* 2019	ETS* 2022	LightTS 2022	DLinear 2023	FED* 2022	Stationary 2022	Auto* 2021	Pyra* 2021
Weighted Average	SMAPE	**11.658**	<u>11.698</u>	11.723	11.807	11.829	11.927	11.851	14.718	13.525	13.639	12.840	12.780	12.909	16.987
	MASE	**1.541**	<u>1.556</u>	1.559	1.590	1.585	1.613	1.599	2.408	2.111	2.095	1.701	1.756	1.771	3.265
	OWA	**0.832**	<u>0.838</u>	0.840	0.851	0.851	0.861	0.855	1.172	1.051	1.051	0.918	0.930	0.939	1.480

Table 4: Ablation Study of LETO on ETT, Weather, and Exchange datasets

Model	ETTh1 MSE / MAE	ETTh2 MSE / MAE	ETTm1 MSE / MAE	ETTm2 MSE / MAE	Weather MSE / MAE	Exchange MSE / MAE
Full LETO	**0.393** / **0.401**	**0.318** / **0.381**	**0.347** / **0.375**	**0.243** / **0.302**	**0.216** / **0.253**	**0.297** / **0.364**
w/o Cross Variate Attention	0.458 / 0.447	0.400 / 0.427	0.394 / 0.419	0.320 / 0.362	0.244 / 0.274	0.311 / 0.398
w/o Linear Attention	0.454 / 0.454	0.392 / 0.421	0.407 / 0.410	0.341 / 0.370	0.258 / 0.278	0.360 / 0.403
w/o Weighted Gating	0.405 / 0.412	0.368 / 0.392	0.389 / 0.397	0.312 / 0.354	0.237 / 0.269	0.301 / 0.384

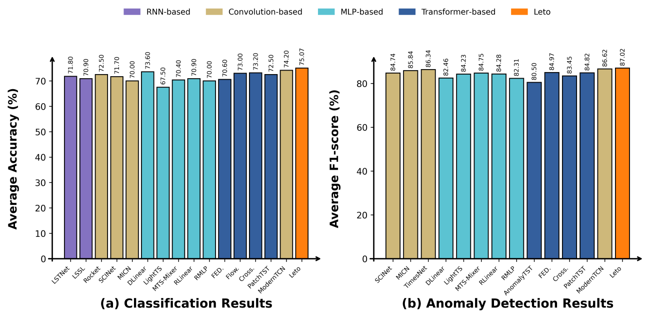

Figure 2: Anomaly detection and classification results of LETO and baselines. Higher accuracy/F1-score indicate better performance.

across a diverse set of tasks, including time series forecasting, classification, and anomaly detection tasks. Limitations, future research directions, and societal impacts are discussed in Section F of the Appendix.

References

[1] Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. 2021. Practical approach to asynchronous multivariate time series anomaly detection and localization. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2485–2494.

[2] Md Atik Ahamed and Qiang Cheng. 2024. MambaTab: A Simple Yet Effective Approach for Handling Tabular Data. *arXiv preprint arXiv:2401.08867* (2024).

[3] Md Atik Ahamed and Qiang Cheng. 2024. Timemachine: A time series is worth 4 mambas for long-term forecasting. In *ECAI 2024: 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain-Including 13th Conference on Prestigious Applications of Intelligent Systems. European Conference on Artificial Intelli*, Vol. 392. 1688.

[4] Anna Allen, Stratis Markou, Will Tebbutt, James Requeima, Wessel P Bruinsma, Tom R Andersson, Michael Herzog, Nicholas D Lane, Matthew Chantry, J Scott Hosking, et al. 2025. End-to-end data-driven weather prediction. *Nature* (2025), 1–3.

[5] Oliver D. Anderson and M. G. Kendall. 1976. Time-Series. 2nd edn. *The Statistician* (1976).

[6] Masanao Aoki. 2013. *State space modeling of time series*. Springer Science & Business Media.

[7] Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. 2024. Simple linear attention language models balance the recall-throughput tradeoff. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 1763–1840. https://proceedings.mlr.press/v235/arora24a.html

[8] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. 2018. The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075* (2018).

[9] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[10] Ethan Baron, Itamar Zimerman, and Lior Wolf. 2024. A 2-Dimensional State Space Layer for Spatial Inductive Bias. In *The Twelfth International Conference on Learning Representations*.

[11] David J Bartholomew. 1971. Time Series Analysis Forecasting and Control.

[12] Ali Behrouz, Parsa Delavari, and Farnoosh Hashemi. 2024. Unsupervised Representation Learning of Brain Activity via Bridging Voxel Activity and Functional Connectivity. In *International conference on machine learning (ICML)*.

[13] Ali Behrouz and Farnoosh Hashemi. 2024. Brain-Mamba: Encoding Brain Activity via Selective State Space Models. In *Proceedings of the fifth Conference on Health, Inference, and Learning (Proceedings of Machine Learning Research, Vol. 248)*, Tom Pollard, Edward Choi, Pankhuri Singhal, Michael Hughes, Elena Sizikova, Bobak Mortazavi, Irene Chen, Fei Wang, Tasmie Sarker, Matthew McDermott, and Marzyeh Ghassemi (Eds.). PMLR, 233–250.

[14] Ali Behrouz and Farnoosh Hashemi. 2024. Graph Mamba: Towards Learning on Graphs with State Space Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 119–130.

[15] Ali Behrouz, Ali Parviz, Mahdi Karami, Clayton Sanford, Bryan Perozzi, and Vahab S. Mirrokni. 2024. Best of Both Worlds: Advantages of Hybrid Graph Sequence Models. *arXiv preprint arXiv:2411.15671* (2024).

[16] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. 2025. It's All Connected: A Journey Through Test-Time Memorization, Attentional Bias, Retention, and Online Optimization. *arXiv preprint arXiv:2504.13173* (2025).

[17] Ali Behrouz, Michele Santacatterina, and Ramin Zabih. 2024. Chimera: Effectively Modeling Multivariate Time Series with 2-Dimensional State Space Models. In *Thirty-eighth Conference on Advances in Neural Information Processing Systems*.

[18] Ali Behrouz, Michele Santacatterina, and Ramin Zabih. 2024. MambaMixer: Efficient selective state space models with dual token and channel selection. *arXiv preprint arXiv:2403.19888* (2024).

[19] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. 2024. Titans: Learning to Memorize at Test Time. *arXiv preprint arXiv:2501.00663* (2024).

[20] Michael Bender and Slobodan Simonovic. 1994. Time-series modeling for long-range stream-flow forecasting. *Journal of Water Resources Planning and Management* 120, 6 (1994), 857–870.

[21] Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. 2023. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems* 36 (2023), 1560–1588.

[22] Leon Bottou and Vladimir Vapnik. 1992. Local learning algorithms. *Neural computation* 4, 6 (1992), 888–900.

[23] George EP Box and Gwilym M Jenkins. 1968. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 17, 2 (1968), 91–109.

[24] Daniel Yiming Cao, Ali Behrouz, Ali Parviz, Mahdi Karami, Michele Santacatterina, and Ramin Zabih. 2025. Effectively Designing 2-Dimensional Sequence Models for Multivariate Time Series. In *ICLR 2025 Workshop on World Models: Understanding, Modelling and Scaling*. https://openreview.net/forum?id=xkFRduCUNz

[25] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. 2022. N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting. *arXiv preprint arXiv:2201.12886* (2022).

[26] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. 2023. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053* (2023).

[27] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[28] Panayiotis Christou, Shichu Chen, Xupeng Chen, and Parijat Dube. 2024. Test time learning for time series forecasting. *arXiv preprint arXiv:2409.14012* (2024).

[29] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[30] R. B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma J. Terpenning. 1990. STL: A seasonal-trend decomposition procedure based on loess (with discussion).

[31] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan K Mathur, Rajat Sen, and Rose Yu. 2023. Long-term Forecasting with TiDE: Time-series Dense Encoder. *Transactions on Machine Learning Research* (2023).

[32] Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Peter Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim Alabdulmohsin, Avital Oliver, Piotr Padlewski, Alexey A. Gritsenko, Mario Lucic, and Neil Houlsby. 2023. Patch n' Pack: NaViT, a Vision Transformer for any Aspect Ratio and Resolution. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[33] Dazhao Du, Bing Su, and Zhewei Wei. 2023. Preformer: Predictive Transformer with Multi-Scale Segment-Wise Correlations for Long-Term Time Series Forecasting. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1–5. doi:10.1109/ICASSP49357.2023.10096881

[34] Pablo L Durango-Cohen. 2007. A time series analysis framework for transportation infrastructure management. *Transportation Research Part B: Methodological* 41, 5 (2007), 493–505.

[35] Vijay Prakash Dwivedi, Ladislav Rampaek, Mikhail Galkin, Alipanah Parviz, Guy Wolf, Anh Tuan Luu, and D. Beaini. 2022. Long Range Graph Benchmark. *ArXiv* abs/2206.08164 (2022). https://api.semanticscholar.org/CorpusID:249712241

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

[37] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14, 2 (1990), 179–211.

[38] Jianqing Fan. 2018. *Local polynomial modelling and its applications: monographs on statistics and applied probability 66*. Routledge.

[39] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. 2019. Unsupervised Scalable Representation Learning for Multivariate Time Series. In *NeurIPS*.

[40] Kelum Gajamannage, Yonggi Park, and Dilhani I Jayathilake. 2023. Real-time forecasting of time series in financial markets using sequentially trained dual-LSTMs. *Expert Systems with Applications* 223 (2023), 119879.

[41] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. 2022. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems* 35 (2022), 29374–29385.

[42] Julia Gastinger, Shenyang Huang, Michael Galkin, Erfan Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, et al. 2024. Tgb 2.0: A benchmark for learning on temporal knowledge graphs and heterogeneous graphs. *Advances in neural information processing systems* 37 (2024), 140199–140229.

[43] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I Webb, Rob J Hyndman, and Pablo Montero-Manso. 2021. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643* (2021).

[44] Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752* (2023).

[45] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems* 33 (2020), 1474–1487.

[46] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. 2022. On the Parameterization and Initialization of Diagonal State Space Models. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=yJE7iQSAep

[47] Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*. https://openreview.net/forum?id=uYLFoz1vlAC

[48] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. 2021. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems* 34 (2021), 572–585.

[49] Andrew C Harvey. 1990. Forecasting, structural time series models and the Kalman filter. *Cambridge university press* (1990).

[50] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[51] S. Hochreiter and J. Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* (1997).

[52] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence* 44, 9 (2021), 5149–5169.

[53] Yinan Huang, Siqi Miao, and Pan Li. 2024. What Can We Learn from State Space Models for Machine Learning on Graphs? *ArXiv* abs/2406.05815 (2024).

[54] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 387–395.

[55] Romain Ilbert, Ambroise Odonnat, Vasilii Feofanov, Aladin Virmaux, Giuseppe Paolo, Themis Palpanas, and Ievgen Redko. 2024. Unlocking the Potential of Transformers in Time Series Forecasting with Sharpness-Aware Minimization and Channel-Wise Attention. *arXiv preprint arXiv:2402.10198* (2024).

[56] Plamen Ch Ivanov, Luis A Nunes Amaral, Ary L Goldberger, Shlomo Havlin, Michael G Rosenblum, Zbigniew R Struzik, and H Eugene Stanley. 1999. Multifractality in human heartbeat dynamics. *Nature* 399, 6735 (1999), 461–465.

[57] Vidit Jain and Erik Learned-Miller. 2011. Online domain adaptation of a pre-trained cascade of classifiers. In *CVPR 2011*. IEEE, 577–584.

[58] Yuxin Jia, Youfang Lin, Xinyan Hao, Yan Lin, Shengnan Guo, and Huaiyu Wan. 2023. WITRAN: Water-wave Information Transmission and Recurrent Acceleration Network for Long-range Time Series Forecasting. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[59] Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. 2023. Polysketchformer: Fast transformers via sketches for polynomial kernels. *arXiv preprint arXiv:2310.01655* (2023).

[60] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*. PMLR, 5156–5165.

[61] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*.

[62] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *SIGIR*.

[63] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In *NeurIPS*.

[64] Shufan Li, Harkanwar Singh, and Aditya Grover. 2024. Mamba-ND: Selective State Space Modeling for Multi-Dimensional Data. *arXiv preprint arXiv:2402.05892* (2024).

[65] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. *arXiv: Learning* (2017).

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

[67] Dingkang Liang, Xin Zhou, Xinyu Wang, Xingkui Zhu, Wei Xu, Zhikang Zou, Xiaoqing Ye, and Xiang Bai. 2024. PointMamba: A Simple State Space Model for Point Cloud Analysis. *arXiv preprint arXiv:2402.10739* (2024).

[68] Bryan Lim and Stefan Zohren. 2021. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A* 379, 2194 (2021), 20200209.

[69] Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. 2024. Longhorn: State space models are amortized online learners. *arXiv preprint arXiv:2407.14207* (2024).

[70] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. 2022. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems* 35 (2022), 5816–5828.

[71] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations (ICLR)*.

[72] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

[73] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. 2024. Vmamba: Visual state space model. *arXiv*

preprint arXiv:2401.10166 (2024).

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

[75] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Non-stationary Transformers: Rethinking the Stationarity in Time Series Forecasting. In *NeurIPS*.

[76] Donghao Luo and Xue Wang. 2024. ModernTCN: A Modern Pure Convolution Structure for General Time Series Analysis. In *The Twelfth International Conference on Learning Representations*.

[77] Jun Ma, Feifei Li, and Bo Wang. 2024. U-Mamba: Enhancing Long-range Dependency for Biomedical Image Segmentation. *arXiv preprint arXiv:2401.04722* (2024).

[78] Aditya P Mathur and Nils Ole Tippenhauer. 2016. SWaT: A water treatment testbed for research and training on ICS security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*. IEEE, 31–36.

[79] Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. 2019. Online model distillation for efficient video inference. In *Proceedings of the IEEE/CVF International conference on computer vision*. 3573–3582.

[80] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. 2024. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems* 36 (2024).

[81] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations (ICLR)*.

[82] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *ICLR* (2019).

[83] Badri N. Patro and Vijay S. Agneeswaran. 2024. SiMBA: Simplified Mamba-Based Architecture for Vision and Multivariate Time series. arXiv:2403.15360 [cs.CV]

[84] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048* (2023).

[85] Bo Peng, Eric Alcaide, Quentin G. Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, G Kranthikiran, Xingjian Du, Xuming He, Haowen Hou, Przemyslaw Kazienko, Jan Kocoń, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Xiangru Tang, Bolun Wang, Johan Sokrates Wind, Stansilaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui Zhu. 2023. RWKV: Reinventing RNNs for the Transformer Era. In *Conference on Empirical Methods in Natural Language Processing*.

[86] Steve Pincus and Rudolf E Kalman. 2004. Irregularity, volatility, risk, and financial market time series. *Proceedings of the National Academy of Sciences* 101, 38 (2004), 13709–13714.

[87] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. 2025. Probabilistic weather forecasting with machine learning. *Nature* 637, 8044 (2025), 84–90.

[88] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting* 36, 3 (2020), 1181–1191.

[89] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. 2021. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*. PMLR, 9355–9366.

[90] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. 2019. Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[91] Xiao Long Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. 2024. Time-MoE: Billion-Scale Time Series Foundation Models with Mixture of Experts. *ArXiv* abs/2409.16040 (2024).

[92] Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. 2023. Simplified State Space Layers for Sequence Modeling. In *The Eleventh International Conference on Learning Representations*.

[93] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2828–2837.

[94] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive Network: A Successor to Transformer for Large Language Models. *ArXiv* abs/2307.08621 (2023).

[95] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. 2024. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620* (2024).

[96] Siyi Tang, Jared A Dunnmon, Qu Liangqiong, Khaled K Saab, Tina Baykaner, Christopher Lee-Messer, and Daniel L Rubin. 2023. Modeling multivariate biosignals with graph neural networks and structured state space models. In *Conference on health, inference, and learning*. PMLR, 50–71.

[97] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 30.

[98] Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. 2023. MICN: Multi-scale Local and Global Context Modeling for Long-term Series Forecasting. In *International Conference on Learning Representations (ICLR)*.

[99] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and JUN ZHOU. 2024. TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=7oLshfEIC2

[100] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2022. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125* (2022).

[101] Peter R Winters. 1960. Forecasting sales by exponentially weighted moving averages. *Management science* 6, 3 (1960), 324–342.

[102] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven C. H. Hoi. 2022. ETSformer: Exponential Smoothing Transformers for Time-series Forecasting. *arXiv preprint arXiv:2202.01381* (2022).

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

[104] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *The Eleventh International Conference on Learning Representations*.

[105] Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2022. Flowformer: Linearizing Transformers with Conservation Flows. In *ICML*.

[106] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[107] Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing Transformers. In *International Conference on Learning Representations*. https://openreview.net/forum?id=TrjbxzRcnf-

[108] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 753–763.

[109] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *International Joint Conference on Artificial Intelligence*.

[110] Hengyuan Xu, Liyao Xiang, Hangyu Ye, Dixi Yao, Pengzhi Chu, and Baochun Li. 2024. Permutation equivariance of transformers and its applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5987–5996.

[111] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2021. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy. In *ICLR*.

[112] Kun Yi, Qi Zhang, Wei Fan, Hui He, Liang Hu, Pengyang Wang, Ning An, Longbing Cao, and Zhendong Niu. 2024. FourierGNN: Rethinking multivariate time series forecasting from a pure graph perspective. *Advances in Neural Information Processing Systems* 36 (2024).

[113] Ting Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal Graph Convolutional Neural Network: A Deep Learning Framework for Traffic Forecasting. *ArXiv* abs/1709.04875 (2017).

[114] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020. Are Transformers universal approximators of sequence-to-sequence functions?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ByxRM0Ntvr

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

[116] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting?, In AAAI. *AAAI*.

[117] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 9 (Jun. 2023), 11121–11128.

[118] Hao Zhang, Alexander C Berg, Michael Maire, and Jitendra Malik. 2006. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. IEEE, 2126–2136.

[119] Michael Zhang, Khaled Kamal Saab, Michael Poli, Tri Dao, Karan Goel, and Christopher Re. 2023. Effectively Modeling Time Series with Simple Discrete

State Spaces. In *The Eleventh International Conference on Learning Representations*.

[120] T. Zhang, Yizhuo Zhang, Wei Cao, J. Bian, Xiaohan Yi, Shun Zheng, and Jian Li. 2022. Less Is More: Fast Multivariate Time Series Forecasting with Light Sampling-oriented MLP Structures. *arXiv preprint arXiv:2207.01186* (2022).

[121] Yizhe Zhang and Deng Cai. 2022. Linearizing Transformer with Key-Value Memory. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 346–359. doi:10.18653/v1/2022.emnlp-main.24

[122] Yufeng Zhang, Boyi Liu, Qi Cai, Lingxiao Wang, and Zhaoran Wang. 2022. An analysis of attention via the lens of exchangeability and latent variable models. *arXiv preprint arXiv:2212.14852* (2022).

[123] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*.

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

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

[126] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning (ICML)*.

A Preliminaries and Background

Transformers and their Permutation Equivariance Property. Transformers [97] have been the de facto backbone for many deep learning models and are based on attention module. Let $x \in \mathbb{R}^{N \times d_{\text{in}}}$ be the input, attention computes output $\mathbf{y} \in \mathbb{R}^{N \times d_{\text{in}}}$ based on softmax over input dependent key, value, and query matrices:

$$\mathbf{Q} = x\mathbf{W}_{\mathbf{Q}}, \qquad \mathbf{K} = x\mathbf{W}_{\mathbf{K}}, \qquad \mathbf{V} = x\mathbf{W}_{\mathbf{V}}, \tag{12}$$

$$\mathbf{y}_i = \sum_{j=1}^{N} \frac{\exp\left(\mathbf{Q}_i^{\top}\mathbf{K}_j/\sqrt{d_{\text{in}}}\right)\mathbf{V}_j}{\sum_{\ell=1}^{N}\exp\left(\mathbf{Q}_i^{\top}\mathbf{K}_\ell/\sqrt{d_{\text{in}}}\right)}, \tag{13}$$

where $\mathbf{W}_{\mathbf{Q}}, \mathbf{W}_{\mathbf{K}}$, and $\mathbf{W}_{\mathbf{V}} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ are learnable parameters. This formulation of attention makes it permutation equivariant, meaning that the permutation of the input cannot change the output but permute it. That is, let $\pi(.)$ be a permutation, and $\mathcal{A}(\cdot)$ be the above attention module, we have:

$$\mathcal{A}(\pi(x)) = \pi(\mathcal{A}(x)). \tag{14}$$

The property, which is called permutation equivariance, is a desirable property for the data that is permutation equivariant, such as variates in the multivariate time series. When encoding the multivariate time series, we do not want the output of the model to be sensitive to the order of the input (variates) and so transformers are great architectures as any change to the order, does not change the output, but just permute it.

Learning to Memorize at Test Time. The concept of learning to memorize at test time is derived from the learning at test time or learning to learn, which backs to very early studies on local learning [22]: i.e., training each test sample on its neighbors before making a prediction [41, 118]. Later, test time training shows promising results in vision tasks [57, 79], mainly because of the ability to properly address out-of-distribution cases. Using this perspective, recently this idea has been applied on sequence modeling [16, 19, 95]. These methods that aim to train a memory module that learns how to memorize the context at test time, have shown promising results in language and sequence modeling tasks. In this work, we also take this perspective and design a 2-dimensional test time memorizer that generalizes all these methods to 2-dimensional data modality.

B Additional Related Work

Classical Approach. Time series modeling has been a fundamental research topic, Classical approaches include a range of statistical models such as exponential smoothing [101], ARIMA [11], SARIMA [20], and the Box-Jenkins methodology [23], with later advancements introducing state-space models [6, 49]. While these models offer interpretability, they often fall short in capturing complex non-linear dynamics and typically rely on manual inspection of time series characteristics—such as trend and seasonality—limiting their adaptability across diverse datasets.

Transformer-based models. Transformer-based architectures have become increasingly prominent in multivariate time series forecasting, particularly when modeling complex inter-variable and temporal dependencies [55, 61, 71, 81, 106, 116, 123, 124, 126]. A line of research has focused on designing specialized attention mechanisms that leverage the unique structure of time series data [102], while others have explored strategies for capturing long-term temporal patterns to improve forecasting accuracy [81, 125].

In parallel, recent works have revisited linear recurrent neural networks (Linear RNNs) as efficient alternatives to Transformers, aiming to reduce the quadratic complexity while maintaining competitive performance on long-range dependency modeling [85, 94, 104]. For instance, Chen et al. [26] introduce TSMixer, a purely MLP-based model that demonstrates strong performance on time series forecasting tasks. Notably, the expressive capacity of certain linear models aligns with 2D state space models (SSMs), suggesting that these architectures can be interpreted as specific instances within the broader 2D SSM framework. Additionally, convolution-based models have shown renewed promise [76], where the use of global convolutional kernels facilitates an expanded receptive field for capturing long-range dynamics.

Recurrent-based models. Another line of research closely related to our work involves deep sequence modeling. Recurrent neural networks (RNNs), including variants such as GRUs [29], LSTMs [51], and DeepAR [88], have been widely used for sequential data. However, these models suffer from well-known limitations such as vanishing and exploding gradients, along with inherently sequential computation that slows down training and inference. To address these inefficiencies, recent efforts have explored linear attention mechanisms as faster alternatives [59, 60, 89]. For instance, Katharopoulos et al. [60] propose a linear attention model with a recurrent formulation, enabling efficient inference and reduced computational complexity.

In parallel, deep state space models (SSMs) have gained momentum as a compelling alternative to Transformer-based architectures [97], offering improved scalability and training efficiency [45]. These models blend classical state space formulations with deep learning by parameterizing neural network layers using multiple linear SSMs. This hybrid formulation leverages the convolutional interpretation of SSMs to mitigate the optimization challenges typically associated with RNNs [45–48, 92]. Recently, Gu and Dao [44] introduced Mamba, a novel deep SSM architecture where parameters dynamically depend on input features. This approach has been successfully extended to various modalities—including images [18, 73, 77], point clouds [67], tabular data [2], graphs [14, 15, 53], and time series [17, 24]—demonstrating strong capabilities in modeling long-range dependencies across domains.

Other Methods. Graph-based models have emerged as powerful tools for time series forecasting[108, 112], especially when the data exhibits spatial or relational structure across variables or entities. Approaches such as graph neural networks (GNNs) model dependencies through learned graph representations, enabling effective spatiotemporal forecasting in domains like traffic [65, 113] and sensor networks [109]. Recent work has extended these ideas by incorporating dynamic graphs [35, 42, 104], learning graph structures jointly with temporal dynamics to better capture evolving relationships over time. These methods offer strong performance in settings where explicit or latent graph structure underpins multivariate time series behavior.

C Parallelizable Training of Leto

While the recurrence-based formulation of Leto enables it to better capture joint temporal and variate dependencies, as well as their independent dynamics, it introduces sequential dependencies that can hinder training efficiency. To address this, we develop a parallelizable training strategy inspired by recent advances in test-time memorization frameworks [19, 95].

Specifically, for a given variate v, we divide its time series $\{x_{1,v}, \dots, x_{T,v}\}$ into C disjoint chunks of length $b = T/C$. Each chunk $S_i = \{x_{(i-1)b+1,v}, \dots, x_{ib,v}\}$ can be treated as an independent subsequence for computing the inner-loop updates of the memory module. This chunking allows us to approximate the gradient $\nabla \ell(M^{(1)}_{t-1,v}, x_{t,v})$ with $\nabla \ell(M^{(1)}_{t',v}, x_{t,v})$, where $t' = \lfloor t/b \rfloor \cdot b$ is the last time step of the previous chunk. Since t' is fixed for each chunk, this gradient can be computed in parallel for all time steps within a chunk.

Moreover, the cross-variate dynamic component—modeled via the attention mechanism—is independent of time and can be computed in advance. We precompute the attention-based memory $M^{(2)}_{t,v}$ for all variates using equation above with a Taylor-approximated softmax kernel. This enables us to also precompute $\nabla \ell(M^{(2)}_{t,v}, x_{t,v})$, further decoupling the cross-variate dynamics from the sequential recurrence.

With the cross-variate memory and its corresponding gradient terms available, the remaining computation in each chunk reduces to a linear update over the cross-time memory using the precomputed components. As a result, we obtain a recurrence that is linear within chunks and can be parallelized across both time and variates.

D Dataset and Experimental Details

The experimental details are reported in Table 5.

E Additional Experimental Results

E.1 Metrics

We utilize the mean square error (MSE) and mean absolute error (MAE) for long-term forecasting. For short-term forecasting on the M4 datasets, we follow the methodology of N-BEATS [82] and utilize the symmetric mean absolute percentage error (SMAPE), mean absolute scaled error (MASE), and overall weighted average (OWA) as metrics. It is worth noting that OWA is a specific metric utilized in the M4 competition. The calculations of these metrics are:

$$\text{RMSE} = \Big(\sum_{i=1}^{F} (\mathbf{X}_i - \widehat{\mathbf{X}}_i)^2\Big)^{\frac{1}{2}}, \qquad \text{MAE} = \sum_{i=1}^{F} |\mathbf{X}_i - \widehat{\mathbf{X}}_i|,$$

$$\text{SMAPE} = \frac{200}{F} \sum_{i=1}^{F} \frac{|\mathbf{X}_i - \widehat{\mathbf{X}}_i|}{|\mathbf{X}_i| + |\widehat{\mathbf{X}}_i|}, \qquad \text{MAPE} = \frac{100}{F} \sum_{i=1}^{F} \frac{|\mathbf{X}_i - \widehat{\mathbf{X}}_i|}{|\mathbf{X}_i|},$$

$$\text{MASE} = \frac{1}{F} \sum_{i=1}^{F} \frac{|\mathbf{X}_i - \widehat{\mathbf{X}}_i|}{\frac{1}{F-s}\sum_{j=s+1}^{F} |\mathbf{X}_j - \mathbf{X}_{j-s}|}, \qquad \text{OWA} = \frac{1}{2}\Big[\frac{\text{SMAPE}}{\text{SMAPE}_{\text{Naïve2}}} + \frac{\text{MA}}{\text{MASE}}$$

where s is the periodicity of the data. $\mathbf{X}, \widehat{\mathbf{X}} \in \mathbb{R}^{F \times C}$ are the ground truth and prediction results of the future with F time pints and C dimensions. $\mathbf{X}_i$ means the i-th future time point. For classification, we use accuracy as the metric. Lastly for anomaly detection, we use F1-Score as the metric.

E.2 Short Term Forecasting

The complete results of short term forecasting are reported in Table 8.

E.3 Long Term Forecasting

The complete results of long term forecasting are reported in 9.

E.4 Anomaly Detection

The complete results of Anomaly Detection are reported in Table 11.

E.5 Classification

The complete results of Classification are reported in 12.

F Limitations and Future Work

We note LETO has a few limitations worth acknowledging. First, the use of gradient-based meta in-context updates at test time, while powerful, introduces additional computational overhead compared to traditional non-adaptive sequence models. Although our dual-form implementation and parallel training strategies mitigate some of this cost, the memory and compute requirements may still be prohibitive in resource-constrained settings, particularly for long-horizon forecasting tasks.

Second, while LETO is designed to model both cross-time and cross-variate dependencies, its reliance on Taylor approximations for the variate attention mechanism may limit its capacity to fully capture complex, high-order variate interactions in some datasets. More expressive non-parametric approximators or learned kernel functions could offer improved generalization and efficiency.

Finally, our current formulation assumes access to reasonably stationary statistics at test time for the meta-memorization process

Table 5: Dataset descriptions. The dataset size is organized in (Train, Validation, Test).

Tasks	Dataset	Dim	Series Length	Dataset Size	Information (Frequency)
Forecasting (Long-term)	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)
	Traffic	862	{96, 192, 336, 720}	(12185, 1757, 3509)	Transportation (Hourly)
	Weather	21	{96, 192, 336, 720}	(36792, 5271, 10540)	Weather (10 mins)
	Exchange	8	{96, 192, 336, 720}	(5120, 665, 1422)	Exchange rate (Daily)
Forecasting (short-term)	M4-Yearly	1	6	(23000, 0, 23000)	Demographic
	M4-Quarterly	1	8	(24000, 0, 24000)	Finance
	M4-Monthly	1	18	(48000, 0, 48000)	Industry
	M4-Weakly	1	13	(359, 0, 359)	Macro
	M4-Daily	1	14	(4227, 0, 4227)	Micro
	M4-Hourly	1	48	(414, 0, 414)	Other
Imputation	ETTm1, ETTm2	7	96	(34465, 11521, 11521)	Electricity (15 mins)
	ETTh1, ETTh2	7	96	(8545, 2881, 2881)	Electricity (15 mins)
	Weather	21	96	(36792, 5271, 10540)	Weather (10 mins)
Classification (UEA)	EthanolConcentration	3	1751	(261, 0, 263)	Alcohol Industry
	FaceDetection	144	62	(5890, 0, 3524)	Face (250Hz)
	Handwriting	3	152	(150, 0, 850)	Handwriting
	Heartbeat	61	405	(204, 0, 205)	Heart Beat
	JapaneseVowels	12	29	(270, 0, 370)	Voice
	PEMS-SF	963	144	(267, 0, 173)	Transportation (Daily)
	SelfRegulationSCP1	6	896	(268, 0, 293)	Health (256Hz)
	SelfRegulationSCP2	7	1152	(200, 0, 180)	Health (256Hz)
	SpokenArabicDigits	13	93	(6599, 0, 2199)	Voice (11025Hz)
	UWaveGestureLibrary	3	315	(120, 0, 320)	Gesture
Anomaly Detection	SMD	38	100	(566724, 141681, 708420)	Server Machine
	MSL	55	100	(44653, 11664, 73729)	Spacecraft
	SMAP	25	100	(108146, 27037, 427617)	Spacecraft
	SWaT	51	100	(396000, 99000, 449919)	Infrastructure
	PSM	25	100	(105984, 26497, 87841)	Server Machine

Table 7: Standard deviation and statistical tests for our LETO method and the strongest baseline ModernTCN on the M4 dataset (short-term forecasting). Lower is better. Confidence is derived from a paired two-tailed t-test over five runs.

Frequency	LETO (Ours)			ModernTCN (2024)			Confidence
	SMAPE	MASE	OWA	SMAPE	MASE	OWA	
Yearly	13.183 ± 0.115	2.941 ± 0.028	0.754 ± 0.022	13.226 ± 0.118	2.957 ± 0.031	0.777 ± 0.025	99%
Quarterly	9.953 ± 0.101	1.150 ± 0.015	0.851 ± 0.015	9.971 ± 0.105	1.167 ± 0.017	0.878 ± 0.018	95%
Monthly	12.517 ± 0.115	0.935 ± 0.014	0.853 ± 0.014	12.556 ± 0.120	0.917 ± 0.015	0.866 ± 0.016	95%
Others	4.583 ± 0.084	2.797 ± 0.027	0.900 ± 0.021	4.715 ± 0.090	3.107 ± 0.028	0.986 ± 0.024	99%
Averaged	11.658 ± 0.112	1.541 ± 0.022	0.832 ± 0.018	11.698 ± 0.120	1.556 ± 0.024	0.838 ± 0.020	95%

Table 8: Full results for the short-term forecasting task in the M4 dataset. ∗. in the Transformers indicates the name of ∗former. *Stationary* means the Non-stationary Transformer. A lower SMAPE, MASE, and OWA indicate a better prediction. As a convention for all experimental results, best performance is highlighted in red, and the second-best is underlined. We take the average of 5 separate runs for each prediction frequency.

Models		LETO (Ours)	ModernTCN [2024]	PatchTST [2023]	TimesNet [2023]	N-HiTS [2023]	N-BEATS* [2022]	ETS∗ [2019]	LightTS [2022]	DLinear [2022b]	FED∗ [2023b]	Stationary [2022b]	Auto∗ [2022b]	Pyra∗ [2021]	In∗ [2021]	Re∗ [2021]
Yearly	SMAPE	**13.183**	<u>13.226</u>	13.258	13.387	13.418	13.436	18.009	14.247	16.965	13.728	13.717	13.974	15.530	14.727	16.169
	MASE	**2.941**	<u>2.957</u>	<u>2.985</u>	2.996	3.045	3.043	4.487	3.109	4.283	3.048	3.078	3.134	3.711	3.418	3.800
	OWA	**0.754**	<u>0.777</u>	0.781	0.786	0.793	0.794	1.115	0.827	1.058	0.803	0.807	0.822	0.942	0.881	0.973
Quarterly	SMAPE	**9.953**	9.971	10.179	10.100	10.202	10.124	13.376	11.364	12.145	10.792	10.958	11.338	15.449	11.360	13.313
	MASE	**1.150**	1.167	0.803	1.182	1.194	1.169	1.906	1.328	1.520	1.283	1.325	1.365	2.350	1.401	1.775
	OWA	<u>0.851</u>	0.878	**0.803**	0.890	0.899	0.886	1.302	1.000	1.106	0.958	0.981	1.012	1.558	1.027	1.252
Monthly	SMAPE	**12.517**	12.556	12.641	12.670	12.791	12.677	14.588	14.014	13.514	14.260	13.917	13.958	17.642	14.062	20.128
	MASE	0.935	**0.917**	<u>0.930</u>	0.933	0.969	0.937	1.368	1.053	1.037	1.102	1.097	1.103	1.913	1.141	2.614
	OWA	**0.853**	<u>0.866</u>	0.876	0.878	0.899	0.880	1.149	0.981	0.956	1.012	0.998	1.002	1.511	1.024	1.927
Others	SMAPE	**4.583**	<u>4.715</u>	4.946	4.891	5.061	4.925	7.267	15.880	6.709	4.954	6.302	5.485	24.786	24.460	32.491
	MASE	**2.797**	3.107	2.985	3.302	3.216	3.391	5.240	11.434	4.953	3.264	4.064	3.865	18.581	20.960	33.355
	OWA	**0.9001**	<u>0.986</u>	1.044	1.035	1.040	1.053	1.591	3.474	1.487	1.036	1.304	1.187	5.538	5.013	8.679
Weighted Average	SMAPE	**11.658**	<u>11.698</u>	11.807	11.829	11.927	11.851	14.718	13.525	13.639	12.840	12.780	12.909	16.987	14.086	18.200
	MASE	**1.541**	<u>1.556</u>	1.590	1.585	1.613	1.599	2.408	2.111	2.095	1.701	1.756	1.771	3.265	2.718	4.223
	OWA	**0.832**	<u>0.838</u>	0.851	0.851	0.861	0.855	1.172	1.051	1.051	0.918	0.930	0.939	1.480	1.230	1.775

to be effective. In highly non-stationary environments or under strong distribution shifts, the learned test-time updates may generalize poorly, leading to suboptimal performance.

G Broader Impact

LETO has demonstrated strong performance as a general-purpose model for time series pattern recognition, achieving competitive results across a wide range of tasks including forecasting, classification, and anomaly detection. Its versatility makes it well-suited for deployment in diverse real-world scenarios, such as energy and power demand forecasting with pronounced seasonal trends, weather prediction under complex and dynamic conditions, financial market modeling in rapidly shifting environments, and demand forecasting within supply chains. Furthermore, LETO has shown particular promise in industrial anomaly detection tasks, which often require robustness to noise and structural variability. These capabilities highlight LETO's potential as a foundational model for advancing time series analysis across multiple applied domains.

H Compute Resources

For experiments, we utilized up to 4 NVIDIA A6000 and A6000 ADA GPUs.

I Proof of Theorem 3.1

To prove this theorem, we show that our LETO can recover the 2D linear recurrent models that are proven to model full-rank matrices [10, 17]. To this end, we show that a special instance of our LETO is equivalent to these linear 2D recurrent models. We let the chunk size to be the size of the sequence length. Therefore, for every $1 \leq t \leq T$, we have:

$$\nabla \ell({}_0^{(1)};{}_t, \rightarrow t) = ({}_0^{(1)}\,{}_t - \rightarrow t){}_t^{\top}, \tag{15}$$

where ${}_0^{(1)}$ is the initial state of the memory, which we let ${}_0^{(1)} = \mathbf{I}$ for the simplicity. Replacing this gradient in Equation Variant 2, we have:

$${}_{t,v}^{(1)} = \alpha_{t,v}^{(1)}\,{}_{t-1,v} - \eta_{t,v}\left(\underbrace{({}_t - \rightarrow t)^{\top}}_{\mathbf{u}_t}\;{}_t\right) + \beta_{t,v}^{(2)}\,{}_{t-1,v} - \gamma_{t,v}\left({}_t^{(2)}\,{}_t{}_t^{\top} - \rightarrow t_t^{\top}\right), \tag{16}$$

where we let $\eta_{t,v} = \gamma_{t,v} = 1$. Also, for the attention module, we use polynomials with degree 1 to approximate the softmax attention (which is the special instance and the weaker version of our design, i.e., considering only the first two terms of the Taylor series). The

Table 9: Complete experiments on long term forecasting tasks over four prediction lengths: {96, 192, 336, 720}. A lower MAE and MSE indicates a better prediction. As a convention for all experimental results, best performance is highlighted in red, and the second-best is underlined. We take the average of 5 separate runs for each prediction length.

		LETO		TimeMixer		Simba		TCN		iTransformer		RLinear		PatchTST		Crossformer		TiDE		TimesNet		DLinear		SCINet		FEDformer		Stationary		Autoformer	
		(ours)		[2024]		[2024]		[2024]		[2024a]		[2023]		[2023]		[2023]		[2023]		[2023]		[2023a]		[2022c]		[2022b]		[2022a]		[2021]	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	0.312	**0.343**	0.320	0.357	0.342	0.360	**0.292**	0.346	0.334	0.368	0.355	0.376	0.329	0.367	0.404	0.426	0.364	0.387	0.338	0.375	0.345	0.372	0.418	0.438	0.379	0.419	0.386	0.398	0.505	0.475
	192	**0.330**	**0.365**	0.361	0.381	0.363	0.382	0.332	0.368	0.377	0.391	0.391	0.392	0.367	0.385	0.450	0.451	0.398	0.404	0.374	0.387	0.380	0.389	0.439	0.450	0.426	0.441	0.459	0.444	0.553	0.496
	336	**0.355**	**0.384**	0.390	0.404	0.395	0.405	0.365	0.391	0.426	0.420	0.424	0.415	0.399	0.410	0.532	0.515	0.428	0.425	0.410	0.411	0.413	0.413	0.490	0.485	0.445	0.459	0.495	0.464	0.621	0.537
	720	**0.391**	**0.408**	0.454	0.441	0.451	0.437	0.416	0.417	0.491	0.459	0.487	0.450	0.454	0.439	0.666	0.589	0.487	0.461	0.478	0.450	0.474	0.453	0.595	0.550	0.543	0.490	0.585	0.516	0.671	0.561
	Avg	**0.347**	**0.375**	0.381	0.395	0.383	0.396	0.351	0.381	0.407	0.410	0.414	0.407	0.387	0.400	0.513	0.496	0.419	0.419	0.400	0.406	0.403	0.407	0.485	0.481	0.448	0.452	0.481	0.456	0.588	0.517
ETTm2	96	**0.164**	0.248	0.175	0.258	0.177	0.263	0.166	0.256	0.180	0.264	0.182	0.265	0.175	0.259	0.287	0.366	0.207	0.305	0.187	0.267	0.193	0.292	0.286	0.377	0.203	0.287	0.192	0.274	0.255	0.339
	192	0.217	0.284	0.237	0.299	0.245	0.306	0.222	0.293	0.250	0.309	0.246	0.304	0.241	0.302	0.414	0.492	0.290	0.364	0.249	0.309	0.284	0.362	0.399	0.445	0.269	0.328	0.280	0.339	0.281	0.340
	336	0.266	0.312	0.298	0.340	0.304	0.343	0.272	0.324	0.311	0.348	0.307	0.342	0.305	0.343	0.597	0.542	0.377	0.422	0.321	0.351	0.369	0.427	0.637	0.591	0.325	0.366	0.334	0.361	0.339	0.372
	720	0.349	0.363	0.391	0.396	0.400	0.399	0.351	0.381	0.412	0.407	0.407	0.398	0.402	0.400	1.730	1.042	0.558	0.524	0.408	0.403	0.554	0.522	0.960	0.735	0.421	0.415	0.417	0.413	0.433	0.432
	Avg	**0.249**	**0.302**	0.275	0.323	0.271	0.327	0.253	0.314	0.288	0.332	0.286	0.327	0.281	0.326	0.757	0.610	0.358	0.404	0.291	0.333	0.350	0.401	0.571	0.537	0.305	0.349	0.306	0.347	0.327	0.371
ETTh1	96	0.365	0.383	0.375	0.400	0.379	0.395	0.368	0.394	0.386	0.405	0.386	0.395	0.414	0.419	0.423	0.448	0.479	0.464	0.384	0.402	0.386	0.400	0.654	0.599	0.376	0.419	0.513	0.491	0.449	0.459
	192	0.396	0.400	0.429	0.421	0.432	0.424	0.405	0.413	0.441	0.436	0.437	0.424	0.460	0.445	0.471	0.474	0.525	0.492	0.436	0.429	0.437	0.432	0.719	0.631	0.420	0.448	0.534	0.504	0.500	0.482
	336	0.461	0.462	0.484	0.458	0.473	0.443	0.391	0.412	0.487	0.458	0.479	0.446	0.501	0.466	0.570	0.546	0.565	0.515	0.491	0.469	0.481	0.459	0.778	0.659	0.459	0.465	0.588	0.535	0.521	0.496
	720	0.427	0.428	0.498	0.482	0.483	0.469	0.450	0.461	0.503	0.491	0.481	0.470	0.500	0.488	0.653	0.621	0.594	0.558	0.521	0.500	0.519	0.516	0.836	0.699	0.506	0.507	0.643	0.616	0.514	0.512
	Avg	**0.393**	**0.401**	0.447	0.440	0.441	0.432	0.404	0.420	0.454	0.447	0.446	0.434	0.469	0.454	0.529	0.522	0.541	0.507	0.458	0.450	0.456	0.452	0.747	0.647	0.440	0.460	0.570	0.537	0.496	0.487
ETTh2	96	0.258	0.337	0.289	0.341	0.290	0.339	0.263	0.332	0.297	0.349	0.288	0.338	0.302	0.348	0.745	0.584	0.400	0.440	0.340	0.374	0.333	0.387	0.707	0.621	0.358	0.397	0.476	0.458	0.346	0.388
	192	0.316	0.379	0.372	0.392	0.373	0.390	0.320	0.374	0.380	0.400	0.374	0.390	0.388	0.400	0.877	0.656	0.528	0.509	0.402	0.414	0.477	0.476	0.860	0.689	0.429	0.439	0.512	0.493	0.456	0.452
	336	0.309	0.379	0.386	0.414	0.376	0.406	0.313	0.376	0.428	0.432	0.415	0.426	0.426	0.433	1.043	0.731	0.643	0.571	0.452	0.452	0.594	0.541	1.000	0.744	0.496	0.487	0.552	0.551	0.482	0.486
	720	0.389	0.430	0.412	0.434	0.407	0.431	0.392	0.433	0.427	0.445	0.420	0.440	0.431	0.446	1.104	0.763	0.874	0.679	0.462	0.468	0.831	0.657	1.249	0.838	0.463	0.474	0.562	0.560	0.515	0.511
	Avg	**0.318**	0.381	0.364	0.395	0.361	**0.377**	0.322	0.379	0.383	0.407	0.374	0.398	0.387	0.407	0.942	0.684	0.611	0.550	0.414	0.427	0.559	0.515	0.954	0.723	0.437	0.449	0.526	0.516	0.450	0.459
Exchange	96	0.079	0.208	0.090	0.235	-	-	0.080	0.196	0.086	0.206	0.093	0.217	0.088	0.205	0.256	0.367	0.094	0.218	0.107	0.234	0.088	0.218	0.267	0.396	0.148	0.278	0.111	0.237	0.197	0.323
	192	0.164	0.298	0.187	0.343	-	-	0.166	0.288	0.177	0.299	0.184	0.307	0.176	0.299	0.470	0.509	0.184	0.307	0.226	0.344	0.176	0.315	0.351	0.459	0.271	0.315	0.219	0.335	0.300	0.369
	336	0.308	0.329	0.353	0.473	-	-	0.307	0.398	0.331	0.417	0.351	0.432	0.301	0.397	1.268	0.883	0.349	0.431	0.367	0.448	0.313	0.427	1.324	0.853	0.460	0.427	0.421	0.476	0.509	0.524
	720	0.637	0.621	0.934	0.761	-	-	0.656	0.582	0.847	0.691	0.886	0.714	0.901	0.714	1.767	1.068	0.852	0.698	0.964	0.746	0.839	0.695	1.058	0.797	1.195	0.695	1.092	0.769	1.447	0.941
	Avg	**0.297**	0.364	0.391	0.453	-	-	0.302	0.366	0.360	0.403	0.378	0.417	0.367	0.404	0.940	0.707	0.370	0.413	0.416	0.443	0.354	0.414	0.750	0.626	0.519	0.429	0.461	0.454	0.613	0.539
Traffic	96	0.380	0.247	0.462	0.285	0.468	0.268	0.368	0.253	0.395	0.268	0.649	0.389	0.462	0.295	0.522	0.290	0.805	0.493	0.593	0.321	0.650	0.396	0.788	0.499	0.587	0.366	0.612	0.338	0.613	0.388
	192	0.391	0.258	0.473	0.296	0.413	0.317	0.379	0.261	0.417	0.276	0.601	0.366	0.466	0.296	0.530	0.293	0.756	0.474	0.617	0.336	0.598	0.370	0.789	0.505	0.604	0.373	0.613	0.340	0.616	0.382
	336	0.409	0.266	0.498	0.296	0.529	0.284	0.397	0.270	0.433	0.283	0.609	0.369	0.482	0.304	0.558	0.305	0.762	0.477	0.629	0.336	0.605	0.373	0.797	0.508	0.621	0.383	0.618	0.328	0.622	0.337
	720	0.452	0.297	0.506	0.313	0.564	0.297	0.440	0.296	0.467	0.302	0.647	0.387	0.514	0.322	0.589	0.328	0.719	0.449	0.640	0.350	0.645	0.394	0.841	0.523	0.626	0.382	0.653	0.355	0.660	0.408
	Avg	0.408	**0.267**	0.484	0.297	0.493	0.291	**0.398**	0.270	0.428	0.282	0.626	0.378	0.481	0.304	0.550	0.304	0.760	0.473	0.620	0.336	0.625	0.383	0.804	0.509	0.610	0.376	0.624	0.340	0.628	0.379
Weather	96	0.155	0.203	0.163	0.209	0.176	0.219	0.149	0.200	0.174	0.214	0.192	0.232	0.177	0.218	0.158	0.230	0.202	0.261	0.172	0.220	0.196	0.255	0.221	0.306	0.217	0.296	0.173	0.223	0.266	0.336
	192	0.173	0.240	0.222	0.260	0.222	0.260	0.196	0.245	0.221	0.254	0.240	0.271	0.225	0.259	0.206	0.277	0.242	0.298	0.219	0.261	0.237	0.296	0.261	0.340	0.276	0.336	0.245	0.285	0.307	0.367
	336	0.232	0.260	0.251	0.287	0.275	0.297	0.238	0.277	0.278	0.296	0.292	0.307	0.278	0.297	0.272	0.335	0.287	0.335	0.280	0.306	0.283	0.335	0.309	0.378	0.339	0.380	0.321	0.338	0.359	0.395
	720	0.307	0.309	0.350	0.349	0.350	0.349	0.314	0.334	0.358	0.347	0.364	0.353	0.354	0.348	0.398	0.418	0.351	0.366	0.365	0.359	0.345	0.381	0.377	0.427	0.403	0.428	0.414	0.410	0.419	0.428
	Avg	**0.216**	**0.253**	0.240	0.271	0.255	0.280	0.224	0.264	0.258	0.278	0.272	0.291	0.259	0.281	0.259	0.315	0.271	0.320	0.259	0.287	0.265	0.317	0.292	0.363	0.309	0.360	0.288	0.314	0.338	0.382
ECL	96	0.136	0.233	0.153	0.247	0.165	0.253	0.129	0.226	0.148	0.240	0.201	0.281	0.181	0.270	0.219	0.314	0.237	0.329	0.168	0.272	0.197	0.282	0.247	0.345	0.193	0.308	0.169	0.273	0.201	0.317
	192	0.144	0.221	0.166	0.256	0.173	0.262	0.143	0.239	0.162	0.253	0.201	0.283	0.188	0.274	0.231	0.322	0.236	0.330	0.184	0.289	0.196	0.285	0.257	0.355	0.201	0.315	0.182	0.286	0.222	0.334
	336	0.154	0.253	0.185	0.277	0.188	0.277	0.161	0.259	0.178	0.269	0.215	0.298	0.204	0.293	0.246	0.337	0.249	0.344	0.198	0.300	0.209	0.301	0.269	0.369	0.214	0.329	0.200	0.304	0.231	0.338
	720	0.162	0.261	0.225	0.310	0.214	0.305	0.191	0.286	0.225	0.317	0.257	0.331	0.246	0.324	0.280	0.363	0.284	0.373	0.220	0.320	0.245	0.333	0.299	0.390	0.246	0.355	0.222	0.321	0.254	0.361
	Avg	**0.149**	**0.247**	0.182	0.272	0.185	0.274	0.156	0.253	0.178	0.270	0.219	0.298	0.205	0.290	0.244	0.334	0.251	0.344	0.192	0.295	0.212	0.300	0.268	0.365	0.214	0.327	0.193	0.296	0.227	0.338

resulting formula can be written as:

$${}^{(1)}_{t,v} = \alpha^{(1)}_{t,v}\,{}_{t-1,v} - \eta_{t,v}\mathbf{u}_t{}_t^{\top} + \beta^{(2)}_{t,v}\,{}_{t-1,v} - \gamma_{t,v}{}_{v_t}^{(2)} + \gamma_{t,v}\mathbf{u}_t{}_t^{\top}, \qquad (17)$$

which is equivalent to the 2-dimensional linear recurrence with diagonal transition matrix. Therefore, as proven by Baron et al. [10], the recurrence can model full-rank matrix.

Table 10: Standard deviation and statistical tests for LETO vs. the strongest baseline ModernTCN on long-term forecasting (lower is better). Confidence levels derive from a paired two-tailed t-test over five seeds.

Dataset	LETO (Ours)		ModernTCN (2024)		Confidence
	MSE	MAE	MSE	MAE	
ETTm1	0.347 ± 0.010	0.375 ± 0.012	0.351 ± 0.011	0.381 ± 0.013	99%
ETTm2	0.249 ± 0.009	0.302 ± 0.011	0.253 ± 0.010	0.314 ± 0.013	95%
ETTh1	0.393 ± 0.012	0.401 ± 0.014	0.404 ± 0.013	0.420 ± 0.015	99%
ETTh2	0.318 ± 0.010	0.381 ± 0.012	0.322 ± 0.011	0.379 ± 0.013	95%
Exchange	0.297 ± 0.016	0.364 ± 0.018	0.302 ± 0.017	0.366 ± 0.019	95%
Traffic	0.408 ± 0.020	0.267 ± 0.012	0.398 ± 0.019	0.270 ± 0.013	90%
Weather	0.216 ± 0.009	0.253 ± 0.011	0.224 ± 0.010	0.264 ± 0.012	95%
ECL	0.149 ± 0.007	0.247 ± 0.009	0.156 ± 0.008	0.253 ± 0.010	99%

Table 11: Full results for the anomaly detection task. The P, R and F1 represent the precision, recall and F1-score in percentage respectively. A higher value of P, R and F1 indicates a better performance. Best performance is highlighted in red, and the second-best is underlined. We take the average of 5 separate runs for each dataset.

Datasets		SMD			MSL			SMAP			SWaT			PSM			Avg F1
Metrics		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	(%)
LSTM	[1997b]	78.52	65.47	71.41	78.04	86.22	81.93	91.06	57.49	70.48	78.06	91.72	84.34	69.24	99.53	81.67	77.97
Transformer	[2017]	83.58	76.13	79.56	71.57	87.37	78.68	89.37	57.12	69.70	68.84	96.53	80.37	62.75	96.56	76.07	76.88
LogTrans	[2019]	83.46	70.13	76.21	73.05	87.37	79.57	89.15	57.59	69.97	68.67	97.32	80.52	63.06	98.00	76.74	76.60
TCN	[2019]	84.06	79.07	81.49	75.11	82.44	78.60	86.90	**59.23**	70.45	76.59	95.71	85.09	54.59	**99.77**	70.57	77.24
Reformer	[2020]	82.58	69.24	75.32	**85.51**	83.31	84.40	90.91	57.44	70.40	72.50	96.53	82.80	59.93	95.38	73.61	77.31
Informer	[2021]	86.60	77.23	81.65	81.77	86.48	84.06	90.11	57.13	69.92	70.29	96.75	81.43	64.27	96.33	77.10	78.83
Anomaly*	[2021]	**88.91**	82.23	85.49	79.61	87.37	83.31	91.85	58.11	71.18	72.51	97.32	83.10	68.35	94.72	79.40	80.50
Pyraformer	[2021]	85.61	80.61	83.04	83.81	85.93	84.86	92.54	57.71	71.09	87.92	96.00	91.78	71.67	96.02	82.08	82.57
Autoformer	[2021]	88.06	82.35	85.11	77.27	80.92	79.05	90.40	58.62	71.12	89.85	95.81	92.74	99.08	88.15	93.29	84.26
LSSL	[2021]	78.51	65.32	71.31	77.55	88.18	82.53	89.43	53.43	66.90	79.05	93.72	85.76	66.02	92.93	77.20	76.74
Stationary	[2022b]	88.33	81.21	84.62	68.55	89.14	77.50	89.37	59.02	71.09	68.03	96.75	79.88	97.82	96.76	97.29	82.08
DLinear	[2023b]	83.62	**71.52**	77.10	84.34	85.42	84.88	92.32	55.41	69.26	80.91	95.30	87.52	98.28	89.26	93.55	82.46
ETSformer	[2022]	87.44	79.23	83.13	85.13	84.93	85.03	92.25	55.75	69.50	90.02	**80.36**	84.91	**99.31**	85.28	91.76	82.87
LightTS	[2022b]	87.10	78.42	82.53	82.40	75.78	78.95	92.58	55.27	69.21	91.98	94.72	93.33	98.37	95.97	97.15	84.23
FEDformer	[2022b]	87.95	82.39	85.08	77.14	80.07	78.57	90.47	58.10	70.76	90.17	96.42	93.19	97.31	97.16	97.23	84.97
TimesNet (I)	[2023]	87.76	82.63	85.12	82.97	85.42	84.18	91.50	57.80	70.85	88.31	96.24	92.10	98.22	92.21	95.21	85.49
TimesNet (R)	[2023]	88.66	83.14	85.81	83.92	86.42	85.15	92.52	58.29	71.52	86.76	97.32	91.74	98.19	96.76	**97.47**	86.34
CrossFormer	[2023]	83.6	76.61	79.70	84.68	83.71	84.19	92.04	55.37	69.14	88.49	93.48	90.92	97.16	89.73	93.30	83.45
PatchTST	[2023]	87.42	81.65	84.44	84.07	86.23	85.14	92.43	57.51	70.91	80.70	94.93	87.24	98.87	93.99	96.37	84.82
ModernTCN	[2024]	87.86	83.85	85.81	83.94	85.93	84.92	93.17	57.69	71.26	91.83	95.98	93.86	98.09	96.38	97.23	86.62
Leto	(ours)	**88.20**	**85.52**	**86.84**	83.50	**89.27**	**86.29**	**93.20**	57.10	70.81	**92.00**	96.73	**94.31**	99.20	94.61	96.85	**87.02**

Table 12: Full results for the classification task (accuracy %). We omit "former" from the names of Transformer-based methods. For all methods, the standard deviation is less than 0.1%. A higher average accuracy indicates a better prediction. Best performance is highlighted in red, and the second-best is underlined. We take the average of 5 separate runs for each dataset.

Datasets / Models	LSTM [1997b]	LSTNet [2018]	LSSL	Trans. [2017]	Re. [2020]	In. [2021]	Pyra. [2021]	Auto. [2021]	Station. [2022b]	FED. [2022b]	/ETS. [2022]	/Flow. [2022c]	/DLinear [2023b]	/LightTS. [2022b]	/TimesNet [2023]	/PatchTST [2023]	/MTCN [2024]	/Leto (ours)
EthanolConcentration	32.3	39.9	31.1	32.7	31.9	31.6	30.8	31.6	32.7	31.2	28.1	33.8	32.6	29.7	35.7	32.8	36.3	**38.8**
FaceDetection	57.7	65.7	66.7	67.3	68.6	67.0	65.7	68.4	68.0	66.0	66.3	67.6	68.0	67.5	68.6	68.3	70.8	**71.3**
Handwriting	15.2	25.8	24.6	32.0	27.4	32.8	29.4	36.7	31.6	28.0	32.5	33.8	27.0	26.1	32.1	29.6	30.6	**32.9**
Heartbeat	72.2	77.1	72.7	76.1	77.1	80.5	75.6	74.6	73.7	73.7	71.2	77.6	75.1	75.1	78.0	74.9	77.2	**78.3**
JapaneseVowels	79.7	98.1	98.4	98.7	97.8	98.9	98.4	96.2	99.2	98.4	95.9	98.9	96.2	96.2	98.4	97.5	**98.8**	98.5
PEMS-SF	39.9	86.7	86.1	82.1	82.7	81.5	83.2	82.7	87.3	80.9	86.0	83.8	75.1	88.4	89.6	89.3	89.1	**89.6**
SelfRegulationSCP1	68.9	84.0	90.8	92.2	90.4	90.1	88.1	84.0	89.4	88.7	89.6	92.5	87.3	89.8	91.8	90.7	93.4	**94.4**
SelfRegulationSCP2	46.6	52.8	52.2	53.9	56.7	53.3	53.3	50.6	57.2	54.4	55.0	56.1	50.5	51.1	57.2	57.8	60.3	**61.1**
SpokenArabicDigits	31.9	100.0	100.0	98.4	97.0	100.0	99.6	100.0	100.0	100.0	100.0	98.8	81.4	**100.0**	99.0	98.3	98.7	98.7
UWaveGestureLibrary	41.2	87.8	85.9	85.6	85.6	85.6	83.4	85.9	87.5	85.3	85.0	86.6	82.1	80.3	85.3	85.8	86.7	**87.1**
Average Accuracy	48.6	71.8	70.9	71.9	71.5	72.1	70.8	71.1	72.7	70.7	71.0	73.0	67.5	70.4	73.6	72.5	74.2	**75.07**

On the other hand, the univariate version of this recurrence (i.e., $\gamma_{t,v} = 0$) results in linear attention formulation, which is limited and cannot express full-rank matrices.

J Visualizations

J.1 Long Term Forecasting

J.2 Ultra Long Term Forecasting

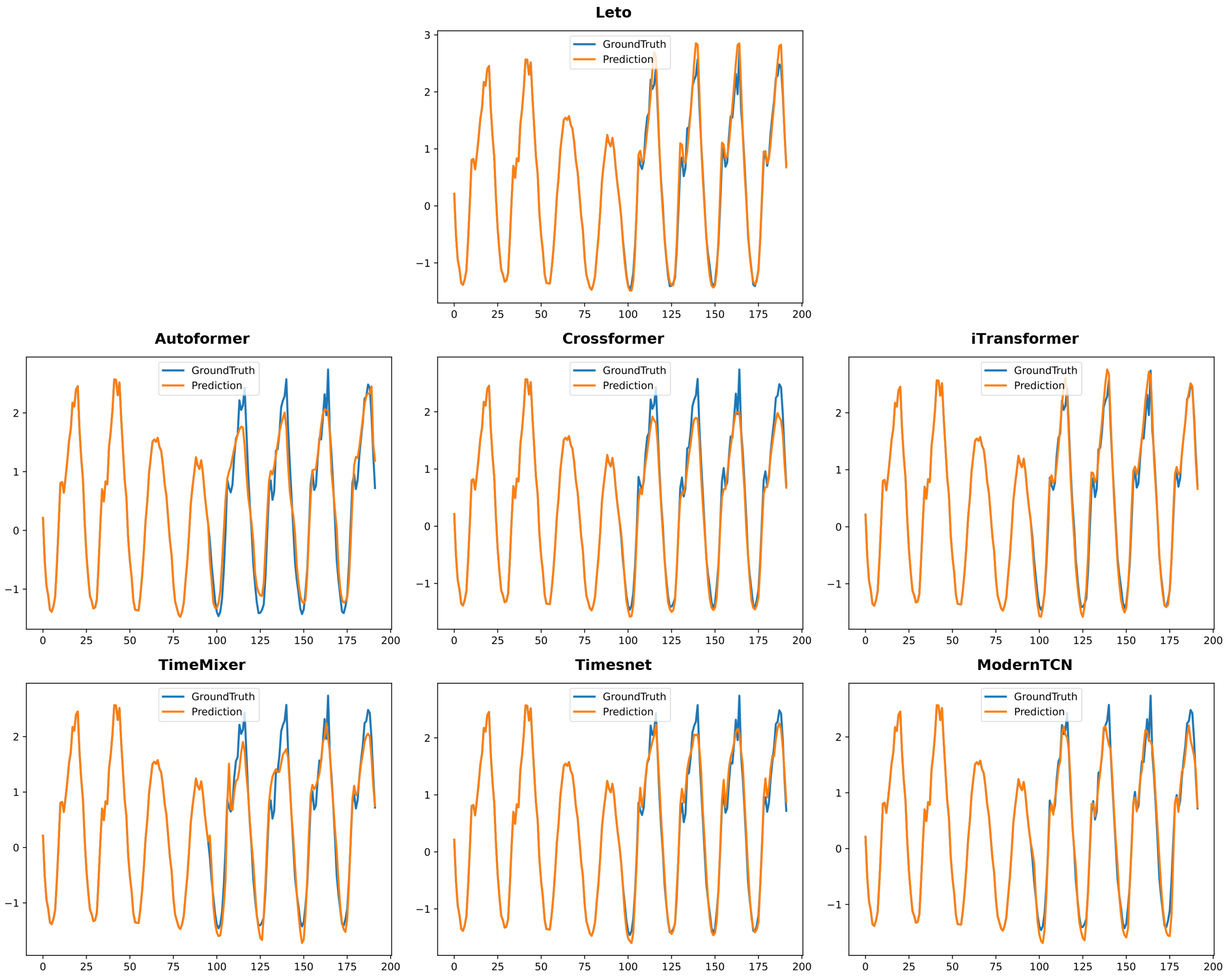

Figure 3: Visualization of Traffic Long Term Forecasting results given by models under the input-96-predict-96 setting. The blue lines stand for the ground truth and the orange lines stand for predicted values.