Effectively Designing 2-Dimensional Sequence Models for Multivariate Time Series

Anonymous Author(s)

Abstract

Although Transformers dominate fields like language modeling and computer vision, they often underperform simple linear baselines in time series tasks. Conversely, linear sequence models provide an efficient, causally biased alternative that excels at autoregressive processes. However, they are fundamentally limited to single-sequence modeling and cannot capture inter-variate dependencies in multivariate time series. Here, we introduce *Typhon*, a flexible framework that applies two sequence models to the time and variable dimensions, merging them with a *Dimension Mixer* module, allowing the inter-variate information flow in the learning process. Building on Typhon, we introduce T4 (Test Time Training with a cross-variate Transformer), which employs a a meta-model for on-the-fly forecasting across time, and a Transformer across variates to capture their dependencies. The Typhon framework's flexibility lets us benchmark T4 alongside various modern recurrent models, revealing that constant-memory recurrence struggles with long-term dependencies and error propagation. To address this, we introduce *Gated Multiresolution Convolution* (GMC)—a simple, attention-free Typhon variant. With a carefully designed constant-size multiresolution memory, GMC can capture long-term dependencies while mitigating error propagation. Our experiments validate Typhon's 2D inductive bias design and demonstrate GMC and T4's superior performance across diverse benchmarks.

Keywords

Multivariate Time Series, Transformers, Multiresolution convolution, Test Time Training

ACM Reference Format:
Anonymous Author(s). 2018. Effectively Designing 2-Dimensional Sequence Models for Multivariate Time Series. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 15 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Multivariate time series analysis plays a crucial role in understanding and predicting complex systems across a wide range of domains such as healthcare, finance, energy, transportation and weather [8, 28, 40, 43]. The complex nature of such multivariate data raises fundamental challenges to design effective and generalizable models: An effective model requires to (1) learn complex patterns, including multi-resolution, trend, and seasonal patterns in the time series data; (2) capture the complex dynamics of the dependencies between variate axes; and (3) be able to efficiently and effectively scale to long-context.

The emergence of deep learning has shifted the focus of time series prediction away from traditional statistical methods toward deep architectures, including Transformer-based [76, 88], recurrence-based [11, 41], and temporal convolutional-based [3, 64]. Despite the promising performance of Transformers [69] in various domains [24, 60, 76], several studies have highlighted that the inherent permutation equivariance of attentions in Transformers contradicts the causal nature of time series and often results in suboptimal performance compared to simple linear methods [81]. Also, their quadratic complexity can cause significant obstacles in large-scale time series applications.

Recently, sub-quadratic sequence models demonstrated significant potential as efficient alternatives to Transformers, mainly due to their efficiency and also ability to learn long-range dependencies based on their inductive temporal bias [67]. They, however, lack a two-dimensional inductive bias of multivariate time series (missing the complex dependencies across both time and variates), use fixed resolutions (missing the dense information in complex time series data), struggle with seasonal patterns, and/or rely on static update parameters. Furthermore, natural attempts to simply employ modern recurrent sequence models for long-term time series forecasting tasks results in (1) error propagation, and (2) poor performance on out-of-distribution test data. While existing studies often uses additional modules to mitigate the above challenges [11, 84], these additional modules result in almost doubling the number of parameters, limiting the number of effective parameters and so the expressive power of the model.

To address, explore, and validate the abovementioned challenges, we present Typhon, a simple yet effective framework that allows extending any sequence model to 2-dimensional data, and adapting them for multivariate time series tasks. Typhon uses two sequence models (not necessarily from the same architecture), each of which responsible to learn the dependencies across one of the dimensions (i.e., one across time and one across variate dimension). Then, it uses a dimension mixer module to inject 2D inductive bias into the model and combine the dimension-specific information along both time and variates.

The flexibility and effectiveness of Typhon, allows us to explore the different combinations of sequence models across time and variate dimensions. Performing extensive experimental evaluations on the combinations of recurrent models, SSMs, Transformers, and linear models, we found that while these hybrid models show outstanding performance in short-term forecasting tasks, they indeed suffer from error propagation in long-term forecasting and show poor performance when the test data is out of distribution.

To address this, we present two variants of Typhon–Test Time Training + Transformer (T4), and Gated Multiresolution Convolution

(GMC)–that shows outstanding performance in all downstream tasks: i.e., long-term and short-term forecasting, classification, imputation, and anomaly detection. T4 utilizes Test Time Training (TTT) layer across time, a meta-in-context model that learn how to learn at test time. Therefore, due to its meta-learning and causal nature, T4 is capable of generalization to out-of-distribution data at test time as it is based on test time training and can update its weights even at test time, adapting itself to new data. T4 further uses a Transformer across variate dimension. Variate dimension in multivariate time series data is naturally permutation equivariant and so Transformers are capable of capturing direct correlation of variates.
Evaluating the performance of T4 and comparing it with more than 100 combinations of sequence models, we find that the recurrent nature of T4 over time still results in error propagation in long-term forecasting tasks. To overcome this challenge, we present a new variant of Typons, Gated Multiresolution Convolution (GMC), that is attention and recurrence free. Our experimental results indicate that GMC show outstanding performance, outperforming T4 and other baselines in most cases over a diverse set of datasets and downstream tasks.

2 Related Work

Multiple mathematical models have been developed across various fields, including healthcare, meteorology, and finance, to address the challenges of time series forecasting. The research on time series forecasting has evolved from traditional statistical methods—such as those utilizing inherent patterns and properties of the data for prediction [5, 14, 15, 72]—to modern deep learning solutions that can capture more expressive temporal correlations. Additionally, techniques like state-space models (SSM), including the Kalman filter, have been widely used to model dynamic system behavior [2, 22, 35]. In these cutting-edge approaches, various neural architectures have driven remarkable advances in predictive accuracy and efficiency. Early work in time series forecasting adopted recurrent neural networks (RNNs)[26] and their variants, such as Long Short-Term Memory (LSTM) networks [36] and Gated Recurrent Units (GRUs) [18] due to their sequential nature, followed by the introduction of temporal convolutional networks (TCNs) [3, 70, 74], which excel at capturing local patterns due to their receptive field design. Meanwhile, Transformer-based models [69], have further revolutionized time series modeling by leveraging self-attention mechanisms to capture both short- and long-term dependencies, improving scalability and predictive performance across various time series tasks [71]. Although, their quadratic complexity poses optimization challenges [52, 76, 88, 89]. Recently, patch-based methods have been introduced to enhance efficiency in Transformer variants [61, 87]. Meanwhile, multilayer perceptrons (MLPs) remain a popular option for time series forecasting, owing to their simplicity and direct mapping capabilities [25]. In parallel, graph neural networks (GNNs) [77, 80] have been employed to capture relationships among multiple variables.

Recently, deep state-space models have gained significant attention as efficient alternatives to Transformers, which suffer from quadratic computational complexity and demonstrated significant potential in addressing the long-range dependencies problem. Deep SSMs offer scalable training and inference, particularly efficient in long-context tasks [31]. These methods combine traditional SSMs with deep neural networks by parameterizing the sequence mixing layers of a neural network using multiple linear SSMs, addressing common training drawbacks of RNNs through the convolutional reformulation of SSMs [31–34, 66]. A recent advancement in expressive sequence modeling has emerged by specifying model parameters as functions of inputs, resulting in more expressive deep SSMs and RNNs [19, 21, 30], as well as long convolution models [42]. These architectures has been expanded beyond sequential tasks to diverse data modalities—including images [12, 42, 54, 58], point clouds [49], tabular data [1], graphs [9, 10, 39], and DNA modeling [30, 60, 63]—thereby enhancing its capacity for modeling long-range dependencies. To address the method's sensitivity to scan order, researchers have proposed bidirectional scanning [90], multi-directional scanning [47, 54], and even automatic direction determination [38]. However, there remains a paucity of work examining variable scan orders specifically within temporal contexts. Most relevant to this work is directly extending the 1-dimensional deep SSMs to their multi-dimensional analogs. Previous works have studied 2D State Space Models. Nguyen et al. [59] present S4ND, a multidimensional SSM layer that extends the continuous-signal modeling ability of SSMs to model videos and images. It not only considers M separate SSM for the M axes, but it also directly treat the system as a continuous system without discretization step. It has data-independent parameters and shows discritizing each 1D SSM results in resolution invariance and can be computed as a convolution as well. Baron et al. [4] present the 2D-SSM layer which is new spatial layer based on Roesser's model for multidimensional state space Kung et al. [45], the most general model for M-axial state space models. It has data-dependent weights and models images as discrete signals where initial SSM model is discrete and there is a lack of discretization step but can be computed as a convolution. The main difference between S4ND and 2-D SSM is that S4ND runs a standard 1-D SSM over each axis independently, and those functions are combined to form a global kernel. In contrast, 2D SSM learns multi-dimensional functions over multi-axes data directly, and 2D-SSM is a generalization of S4ND in 2 dimensions when setting $A_2 = A_3 = 0$ and A_1, A_4 to be the system matrices. Behrouz et al. [11] discuss this in their extension to 2D-Mamba and 2D-Mamba2.

3 Preliminaries

3.1 Notations

We focus on multivariate time series forecasting and classification tasks. Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\} \in \mathbb{R}^{T\times N}$ denote the input data, where T is the number of time steps and N is the number of features (variates). The value of feature v at time t is denoted by $x_{t,v}$. For forecasting tasks, given an input sequence $\mathbf{x}_i$, the goal is to predict the next H time steps, $\tilde{\mathbf{x}}_i \in \mathbb{R}^{H\times N}$, where H is the prediction horizon. For classification tasks, the goal is to assign a class label to each sequence. Anomaly detection can be seen as a binary classification task where 0 denotes "normal" and 1 denotes "anomalous."

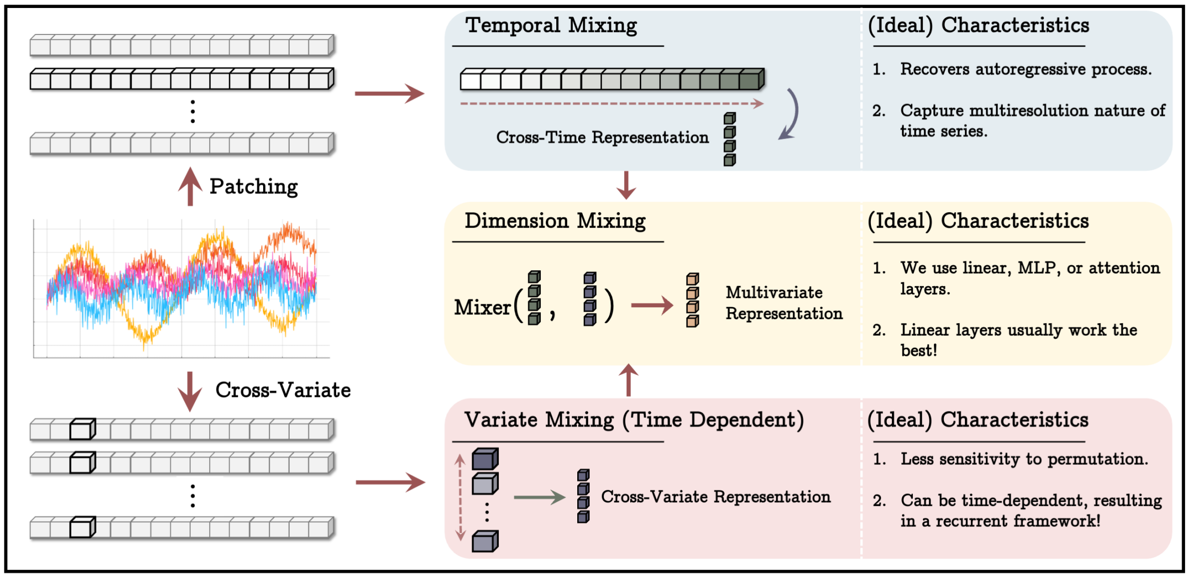
Patching
Cross-Variate
Temporal Mixing
Cross-Time Representation
(Ideal) Characteristics
1. Recovers autoregressive process.
2. Capture multiresolution nature of time series.
Dimension Mixing
Mixer
Multivariate Representation
1. We use linear, MLP, or attention layers.
2. Linear layers usually work the best!
Variate Mixing (Time Dependent)
Cross-Variate Representation
1. Less sensitivity to permutation.
2. Can be time-dependent, resulting in a recurrent framework!

Figure 1: Typhon integrates temporal mixing for cross-time dependencies, variate mixing for cross-variate interactions, and dimension mixing to unify temporal and feature representations. Typhon efficiently models complex multivariate time series dynamics while maintaining scalability.

3.2 (Seasonal) Autoregressive Processes

The autoregressive (AR) process is a foundational building block for time series modeling, capturing causal relationships in sequential data. For an order-p AR process, AR(p), the relationship between a value $\mathbf{x}_t \in \mathbb{R}^d$ and its past p values is given by:

$$\mathbf{x}_t = \sum_{i=1}^{p} \phi_i \mathbf{x}_{t-i}, \tag{1}$$

where $\phi_i \in \mathbb{R}^{d\times d}$ are the autoregressive coefficients. This formulation can be extended to account for seasonal patterns, resulting in a Seasonal Autoregressive (SAR) process, SAR(p, q, s):

$$\mathbf{x}_t = \sum_{i=1}^{p} \phi_i \mathbf{x}_{t-i} + \sum_{j=1}^{q} \eta_j \mathbf{x}_{t-js}, \tag{2}$$

where s is the seasonal period, and $\eta_j \in \mathbb{R}^{d\times d}$ are the seasonal coefficients. Here, the seasonal component captures periodic dependencies at lag s and its multiples.

4 Typhon: a Double-Headed Model with 2D Inductive Bias

In this section, we present the general framework of Typhon and discuss its main constituents and properties. In Typhon framework, we break the architecture into three components (Figure 1 shows the three main steps of Typhon):

4.1 Time Mixer

Learning complex patterns and dependencies across time is a key component for understanding multivariate time series. Our intuition is to treat the time series data as a sequence of tokens (or patches) and then employ a sequence model (i.e., Transformers, linear RNNs, linear models, etc.) to encode the information across time. Notably, this encoding is done for each variate separately and it is mainly responsible to capture temporal dependencies. There are, however, three critical challenges to adapt existing sequence models:
(1) Transformer-based Models: The attention mechanism in Transformers is permutation equivariant and so is unable to recover autoregressive process by its nature, missing temporal patterns [11]. This lack of expressivity causes Transformers to even underperform simple linear models in several scenarios [68]; (2) Linear Models: Similar to Transformers, linear models also suffer from the lack of ability to recover autoregressive process. They further assumes a linear pattern in the temporal dependencies in data, resulting poor performance in real-world downstream tasks; (3) Recurrent Models: Contrary to Transformers and linear models, recurrence-based approaches are not naturally limited. That is, with careful parametrization and architectural design, recurrent models can recover autoregressive process [11, 85]. Their recurrence, however, can cause error propagation in inference time as the test data can be out-of-distribution with respect to the training data [11]. Accordingly, as we discuss later in § 4.5, we employ a test-time training layer to encode information across time, mitigating error propagation by dynamically adapt weights at test-time.
Given $\mathbf{X} \in \mathbb{R}^{T\times N}$ as the input data, the time mixer module is responsible to capture and learn temporal patterns in each variate $\mathbf{X}$ separately. Given a sequence model $\mathcal{T}(.)$, a look-back window length h, and prediction horizon H we use $\mathcal{T}$ across time dimension:

$$y_{T:T+H} = \mathcal{T}(\mathbf{X}_{h:}), \tag{3}$$

where $y_{T:T+H} \in \mathbb{R}^{H\times N}$ is the prediction output for next H time steps, and $\mathbf{X}_{h:}$ is the data for the last h time steps.

Later, we introduce two variants of Typhon, in which we specify and explain our time mixer module. In our experiments, we use Mamba [30], Transformers [69], xLSTM [6], and linear layers as the baseline modules for time mixing.

4.2 Variate Mixer

In understanding multivariate time series data, the dependencies across variates can be pivotal and play an important role in several real-world scenarios ranging from neuro-signals [8] and other bio-signals to stock prediction [76] and traffic forecasting [88]. For example, in neuro-signals (e.g., EEG, MEG, fMRI, etc.) the temporal dependencies is only important up to a binary label (i.e., active, deactive), while the dependencies of variates (i.e., co-activation of different brain regions) is a key to classify or forecast brain activity [7, 8].
Our approach to capture such dependencies is to treat the variate as un-ordered sequences, where each variate is described by its time stamps: i.e., each variate v is represented by $x_v \in \mathbb{R}^T$, where the i-th element is the value of variate v at time i. Accordingly, in Typhon, we use a bidierectional sequence model as the Variate Mixer module, which is responsible to learn pairwise dependencies of channels. Given a sequence model $\mathcal{V}(.)$, we define $\mathcal{V}^*(.)$ as the bidirectional variant of $\mathcal{V}(.)$. That is, if $\mathcal{V}(.)$ is causal by its nature, we define:

$$\mathcal{V}^*(x) = \mathcal{V}(x) + \mathcal{V}(\texttt{flip}(x)), \tag{4}$$

and if $\mathcal{V}(x)$ is bidirectional by its nature we define $\mathcal{V}^*(x) = \mathcal{V}(x)$. As an example, let $\mathcal{V}(.)$ be a SSM, then $\mathcal{V}^*(.)$ is defined by Equation 4 as SSMs are naturally causal. On the other hand, Transformers are permutation equivariant and so $\mathcal{V} = \mathcal{V}^*$. Therefore, given a sequence model $\mathcal{V}(.)$, the variate mixer module performs as follows:

$$y_V = \mathcal{V}^*(\mathbf{X}^\top). \tag{5}$$

Notably, as mentioned earlier, the main reason to define the bidirectional variants of a sequence model $\mathcal{V}(.)$ is the non-causal nature of variates. That is, variates are not naturally ordered and so a causal sequence model can make model sensitive to the initial order of variates.

4.3 Dimension Mixer

In the previous modules, we encode both time and variate dependencies. The resulting model by the combination of these two modules, however, still lacks 2D inductive bias as modules are working separately. In complex real-world scenarios, time and variate dimensions in a multivariate time series system are inter-connected, meaning that the dependencies of variates can affect the temporal patterns and vice versa. Accordingly, a powerful model needs to fuse information and learning process across both directions. To address this, in Typhon, we use a Dimension Mixer module. The main role of dimension mixer is to fuse information between these two dimension encoders. Given a neural network $\mathcal{D}(.)$, we obtain the final output of Typhon as:

$$o = \mathcal{D}\left(y_T || y_V\right). \tag{6}$$

There are different choices for $\mathcal{D}(.)$ in practice; however, in this paper, we focus on three variants of linear-model, MLP, and attention.

It is notable that our framework of Typhon is significantly different from linear mixer models such as TSMixer [17]. That is, Typhon, utilizes time and variate mixer modules in a parallel manner, while models like TSMixer consider a stack of time and variate mixers in a sequential manner. Accordingly, while the input of both time and variate mixer in Typhon is the data (and its transposed), the input of variate/time mixer in such models is the output of the previous layer.

4.4 Improving Typhon with Normalization and Time Series Decomposition

In this section, we first discuss a pre-processing step to improve the performance of Typhon with normalization of input data. Next, we present two natural ways to let model adaptively learn to decompose the time series data into seasonal and trend patterns.

Input Pre-processing and Embedding. In our framework, to stabilize training and capture time-dependent features, the input $\mathbf{X}$ is normalized along the temporal dimension:

$$\mathbf{e}_t = \frac{\mathbf{x}_t - \mu_t}{\sigma_t}, \quad \mu_t = \frac{1}{N}\sum_{v=1}^{N} x_{t,v}, \tag{7}$$

$$\sigma_t = \sqrt{\frac{1}{N}\sum_{v=1}^{N}(x_{t,v} - \mu_t)^2}. \tag{8}$$

The normalized sequence $\{\mathbf{e}_t\}_{t=1}^T$ is then embedded using a data embedding module:

$$\mathbf{z}_t = \texttt{Embedding}(\mathbf{e}_t, \mathbf{m}_t),$$

where $\mathbf{m}_t$ represents associated time features (e.g., timestamps or positional encodings).

Long-Term and Seasonal Decomposition. Real-world time series data is multi-resolution by its nature [81]. That is, temporal dependencies and its dynamic is happening in different scales. For example, seasonal patterns are patterns in a time series data that repeats every (almost) fixed period of time (e.g., each day, month, season, etc.), while trend patterns are long-term dynamic of the data. In this paper, we introduce two different methods to capture these multi-resolution patterns in time series data.
In the first approach, following previous studies on seasonal patterns in time series data [11], we split the sequence into long-term and seasonal components for specialized processing. Given the combined temporal and feature representations $\{\mathbf{h}_t^{(x)}, \mathbf{h}_t^{(y)}\}$, the decomposition is:

$$\mathbf{h}_t = \mathbf{h}_t^{\text{trend}} + \mathbf{h}_t^{\text{seasonal}}, \tag{9}$$

$$\mathbf{z}_t^{(1)} = \sigma\left(\mathbf{W}_1[\mathbf{h}_t^{\text{trend}}; \mathbf{h}_t^{\text{seasonal}}] + \mathbf{b}_1\right), \tag{10}$$

$$\mathbf{h}_t = \sigma\left(\mathbf{W}_2\mathbf{z}_t^{(1)} + \mathbf{b}_2\right), \tag{11}$$

where $\mathbf{W}_1, \mathbf{W}_2$ are learnable parameters and σ is an activation function such as Swish. Note that the dimension mixer does not need to be linear, though we obeserved that more complicated dimension mixers seem to lead to overfitting. Later, in our T4 model, we use this decomposition method.

Multi-resolution Decomposition. While the above approach in most cases achieves outstanding performance to model multivariate time series data, in some complex cases, the granularity of patterns in the time series data is more than 2 levels. Accordingly, an expressive an generalizable model needs to extract and learn all different multi-resolution patterns in different levels of granularity. Accordingly, given granularity levels of $\{\ell_1, \dots, \ell_k\}$, we decompose the time series into:

$$\mathbf{h}_t = \mathbf{h}_t^{(\ell_1)} + \mathbf{h}_t^{(\ell_2)} + \cdots + \mathbf{h}_t^{(\ell_k)}, \tag{12}$$

$$\mathbf{z}_t^{(1)} = \sigma\left(\mathbf{W}_1 [\mathbf{h}_t^{(\ell_1)}; \mathbf{h}_t^{(\ell_2)}; \dots; \mathbf{h}_t^{(\ell_k)}] + \mathbf{b}_1\right), \tag{13}$$

$$\mathbf{h}_t = \sigma\left(\mathbf{W}_2 \mathbf{z}_t^{(1)} + \mathbf{b}_2\right). \tag{14}$$

Later, we use a gated multi-resolution convolution to extract and learn different $\mathbf{h}_t^{(\ell_i)}$.

4.5 T4: A Double-headed Test-Time Training and Transformer Model

Earlier, we discuss the design of Typhon framework that allows us to employ any sequence model for 2D time series data (i.e., multivariate time series). However, given diverse choices for Time Mixer (e.g., Transformers [69], linear recurrent models [19], xLSTM [6], TTT [67], Mamba [30], etc.), Variate Mixer (e.g., the bidirectional variants of the same set of choices for time mixing), and Dimension Mixer (e.g., attention [69], linear, and/or MLPs, etc.), it is still an open question that what constitute a good time, variate, and dimension mixer. Accordingly, in this section, we present a powerful variant of Typhon, called Test Time Training with Transformer (T4) model.

Time Mixer. As discussed earlier, a good time mixer module should recover the autoregressive process and also mitigate the error propagation at test time. Accordingly, we use Test Time Training layer (TTT) [67] as our time mixer. More specifically, TTT is a meta-learning layer that aims to reconstruct different views of data in its inner-loop. Let X be the input, we corrupt the data using a linear layer W_θ and reconstruct it using another linear layer W_ϕ. Therefore, one can define the loss function as:

$$\mathcal{L}_{\text{inner}} = ||W_\theta X - \mathcal{M} \times (W_\phi X)||_2^2, \tag{15}$$

where $\mathcal{M} \in \mathcal{R}^{N \times N}$ is the hidden state of the layer. Note that the above loss function is the loss for the inner-loop of the meta-learning framework and so learnable parameters of W_θ and W_ϕ are considered hyperparameters in it. Given this loss function and time stamp t, we optimize it using mini-batch gradient descent with adaptive learning rate of η_t (input-dependent), resulting in the following recurrence:

$$M_{t+1} = M_t + \eta_t \nabla \mathcal{L}_{\text{inner}} \tag{16}$$

This meta model will learn how to learn at test time. Notably, the recurrence in the above equation is still valid at test time and so the model is always learning from the data. This adaptive nature and its continual learning results in more generalization and less sensitivity to out-of-distribution data as discussed in previous studies. Next theorem shows the power of the above layer (proof is simply derive from its definition):

Theorem 4.1. *The above TTT layer can recover autoregressive process.*

Variate Mixer. Using the above design across time (i.e., as the time mixer module) to learn the temporal patterns in data, we need to specify the variate mixer module. While the permutation equivariance property of Transformers make them less expressive to recover autoregressive process, that is indeed an advantage for learning patterns across variates. That is, a Transformer architecture with full attention is permutation equivariance and so is not sensitive to the order of variates. On the other hand, recurrent models are causal by nature and while their bidirectional versions can considerably avoid sensitivity to the order of variates, they cannot be full permutation equivariance. Therefore, in T4 design, we use an attention mechanism across variates to capture their pairwise dependencies. More specifically, let X be the input data, we use:

$$\texttt{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \texttt{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} \tag{17}$$

as the variate mixer, where $Q = W_Q X^\top, K = W_K X^\top$, and $V = W_V X^\top$.

Dimension Mixer. For our dimension mixer, we simply use a simple linear layer. The main reason for this choice was mainly motivated by our experimental observations, in which we did not see a notable improvement when using non-linear multilayer MLPs and/or attention.

Given the above choices for the Time, Variate, and Dimension Mixers, we also use input normalization and long-term and seasonal decomposition of time series, which we discussed both in the previous subsection.

5 Gated Multi-resolution Convolution

In the above, we discussed a variant of Typhon, in which we decompose the time series into two types of patterns. However, as discussed earlier, real-world complex time series data can have multiple scales of granularity and so requires a more general model to capture such temporal multi-scale patterns. In this section, we present another variant of Typhon, in which we use simple multiresolution convolutions across both time and variates. The multiresolution convolutions allow the model to capture dependencies in multiple levels and so automatically can extract such patterns, without any manual decomposition as T4.

We take a similar approach as Luo and Wang [57] to design a modern convolutional time series model and use pointwise convolutions. However, to capture both across time and variate dependencies, we use pointwise convolutuions across both of these dimensions. Figure 2 represents the Gated Multiresolution Convolution (GMC) block. More specifically, let $\mathbf{X} \in \mathbb{R}^{T \times N}$ represent the input time series, where T is the number of time steps and N is the number of variates. For a convolutional filter of size k, the operation is defined as:

$$\mathbf{H}_t^{(k)} = \mathbf{W}^{(k)} * \mathbf{X}_t + \mathbf{b}^{(k)},$$

where: $\mathbf{H}_t^{(k)} \in \mathbb{R}^{T \times d}$ is the output of the convolution at scale k, $\mathbf{W}^{(k)} \in \mathbb{R}^{k \times N \times d}$ are the learnable weights of the convolutional kernel, $\mathbf{b}^{(k)} \in \mathbb{R}^d$ is the bias term, and $*$ denotes the convolution operator.

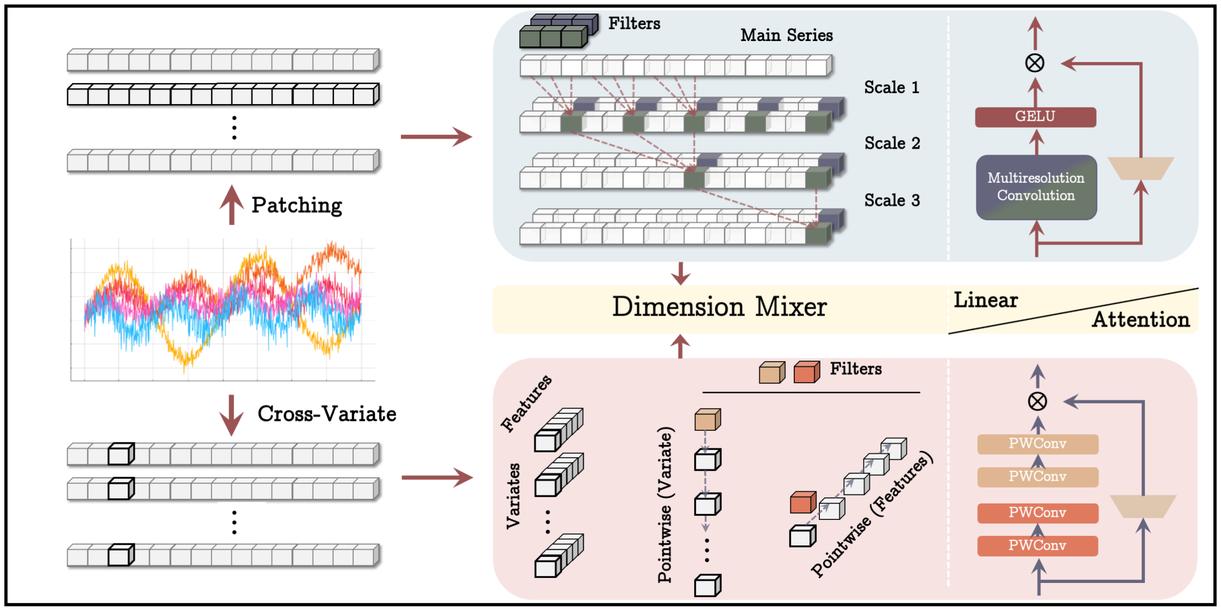

Figure 2: The GMC block processes the input through stacked filters across different resolutions, using GELU activations and gating mechanisms to enhance expressiveness. The processed representations are combined and passed to the dimension mixer for integrating temporal and feature interactions, leveraging either linear or attention-based layers for downstream tasks.

Next, to accommodate multi-resolution processing [65], we recursively apply convolution filters with size K to the time dimension. Therefore, after s-th iteration, the output of the s-th scale for filters $\mathbf{h}_0$ and $\mathbf{h}_1$ is as:

$$h_t^{(s)} = \sum_{i=0}^{K-1} h_{t-2^{s-1}i}^{(s-1)} \mathbf{h}_0 \tag{18}$$

$$q_t^{(s)} = \sum_{i=0}^{K-1} q_{t-2^{s-1}i}^{(s-1)} \mathbf{h}_1. \tag{19}$$

We simply mix scales by a linear layer in this setup and use $\mathbf{H}_t$ to denote the output of this multiresolution convolution. Therefore, the output, $\mathbf{H}_t$, is the mix of all scales and can learn to weight different scales in a data-driven manner.

Gated Convolution. To enhance expressiveness, we follow the backbone architecture of modern sequence models [13, 19, 30, 67] and add a gating mechanism that modulates the multi-resolution convolution outputs. Let X be the input data, $\mathbf{H}_t$ be the output in the above process, the gate branch is defined as:

$$\mathbf{G}_t = \sigma(\mathbf{W}_g X_t + \mathbf{b}_g),$$

where $\mathbf{G}_t \in \mathbb{R}^{T\times d}$ is the gating signal, $\mathbf{W}_g \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_g \in \mathbb{R}^d$ are learnable parameters, σ is a non-linear activation function (e.g., GELU or sigmoid). Given this gated branch, we define the output of the gated convolution as:

$$\mathbf{H}_t^{\text{gated}} = \mathbf{G}_t \odot \mathbf{H}_t, \tag{20}$$

where $\odot$ denotes element-wise multiplication. This gating mechanism allows the model to selectively amplify or suppress specific patterns, enabling a more dynamic representation of the input data.

6 Experiments

We evaluate Typhon's performance on the standard baselines for multivariate time series tasks, comparing Typhon with the state of the art multivariate time series models, including recent models like: TimesNet [75], ModernTCN [57], iTransformer [53], Autoformer [76], ETSFormer [73], CrossFormer [87], FedFormer [89], etc. [20, 50, 51]. Specifically for time series tasks, we test Typhon's variants on short term forecasting 6.1, long term forecasting 6.2, imputation 6.3, and anomaly detection and classification 6.4. We further evaluate the significance of the Typhon's components by performing an ablation study in 6.5. We also provide results evaluating whether the strong performance of Typhon coincides with its efficiency and also test its generalizability on unseen variates and its ability to filter irrelevant context. Additional model combinations, experimental details for reproducibility, and the complete experiment results are provided in the appendix. Note that the order in which the 2 models is stated throughout our results is always in the order of time variate dimension and feature variate dimension. All the experiments were run on 4 NVIDIA RTX A6000 GPUs.

6.1 Short-term Forecasting

We perform experiments in short-term forecasting task on the M4 benchmark dataset datasets [29] and report the results in Table 1. Interestingly, the performance of both Typhon's variants (i.e., T4 and GMC) are close and both outperform state-of-the-art approaches like ModernTCN, PatchTST, etc. These results highlight the expressivity of Typhon's design to capture cross time dependencies. However, since the data is one dimensional, it is questionable whether or not the use of the more advanced dimension mixer or the use of the number of dimensions of Typhon may help. Our results show that in those cases the model tends to overfit. In our case we find that the best results are obtained when using only 1 layer of Typhon and a single linear layer dimension mixer. The complete results are in Table 8.

6.2 Long-term Forecasting

Despite the outstanding performance of Typhon's variants, still it is not clear if our designs perform well when we have long-term time series data. Accordingly, we perform experiments in long-term forecasting task on commonly-used benchmark datasets [88]. The summary of results is reported in Table 2 and the full results can be found in Table 9. Typhon outperforms extensively studied MLP-based, convolution-based, and Transformer-based models providing a better balance of performance and efficiency, as well as recurrent models. Comparing with other baselines that also use time series decomposition (i.e., seasonal and trend pattern), the superior performance of Typhon's variants show their expressivity in capturing both time and variate dependencies.

6.3 Imputation

Real-world systems always work continuously and are monitored by automatic observation equipment. However, due to malfunctions, the collected time series can be partially missing, making the downstream analysis difficult which begs the need for imputation. For imputation task we select the datasets from the electricity and weather scenarios as our benchmarks, including ETT [88], Electricity [78] and Weather, where the data-missing problem happens commonly. To compare the model capacity under different proportions of missing data, we randomly mask the time points in the following ratios: 12.5%, 25%, 37.5%, 50%. The main results are summarized in Table 3.

6.4 Classification and Anomaly Detection

Anomaly detection is generally viewed as a binary classification task, where 0 denotes "normal" and 1 denotes "anomaly". We let $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \in \mathbb{R}^{N\times T}$ be the input sequences, where N is the number of variates and T is the time steps. We use $x_{v,t}$ to refer to the value of the series v at time t. In classification (anomaly detection) tasks, we aim to classify input sequences and for forecasting tasks, given an input sequence $\mathbf{x}_i$, we aim to predict $\tilde{\mathbf{x}}_i \in \mathbb{R}^{1\times H}$, i.e., the next H time steps for variate $\mathbf{x}_i$, where H is called horizon. We evaluate the performance of Gated Multiresolution Convolution and T4 in anomaly detection task by aggregating over their respective datasets for their specific tasks, and report the results in Figure 3. Typhon's variants achieve outstanding performance and outperform all baselines from different group of models (i.e., transformer-based, linear-based, and convolutional).

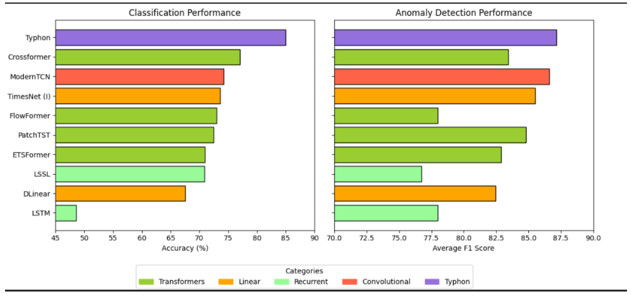

Figure 3: Anomaly detection and classification results of Typhon and baselines.

6.5 Ablation Study

To evaluate the significance of the Typhon's design, we perform an ablation study for Gated Multiresolution Convolution and remove one of its components at each time, keeping other parts unchanged. We first report Gated Multiresolution Convolution's performance with each of its components, while the next row removes dimension mixer, third row removes multiresolution convolution and instead uses a simple convolution, and the last two rows removes gating from time and variate mixing, respectively. The results are reported in Table 5 and demonstrate that removing each component significantly degrades the performance of the model, supporting the importance of our design.

We then perform an ablation study on T4 architecture. The results for T4 where we remove one of the directions for the bidirectioning model encoding, the dimension mixer, and each of the long term and seasonal components, are reported in Table 4. The results demonstrate that these changes on long term time forecasting. The full ablation study is in the appendix.

Table 4: Ablation Study Results for Typhon: TTT-Linear and Transformer (T4)

Model Variations	ETTh1		ETTm1		ETTh2	
	MSE	MAE	MSE	MAE	MSE	MAE
Typhon	0.438	0.444	0.374	0.399	0.373	0.410
Uni.-directional	0.501	0.463	0.485	0.437	0.431	0.523
w/o Dim Mixer	0.522	0.476	0.391	0.414	0.389	0.413
w/o Long term	0.471	0.498	0.361	0.389	0.372	0.401
w/o Seasonal	0.456	0.471	0.357	0.403	0.395	0.425

Table 5: Ablation Study Results for Typhon: Gated Multiresolution (GMC)

Model Variations	ETTh1		ETTm1		ETTh2	
	MSE	MAE	MSE	MAE	MSE	MAE
Gated Multiresolution	0.398	0.409	0.344	0.373	0.316	0.377
w/o Dim Mixer	0.448	0.462	0.394	0.419	0.367	0.412
w/o multiresolution	0.405	0.431	0.354	0.383	0.328	0.380
w/o time gating	0.405	0.412	0.366	0.389	0.324	0.379
w/o variate gating	0.400	0.412	0.357	0.383	0.331	0.386

Table 1: Average performance on short-term forecasting tasks on the M4 dataset. Full results are reported in the appendix. For the Typhon architecture results we denote GMC as Gated Multiresolution Convolution variant of Typhon, and T4 as TTT layer and Transformer

Models		Typhon (T4)	Typhon (GMC)	ModernTCN	PatchTST	TimesNet	N-HiTS	N-BEATS*	ETS*	LightTS	DLinear	FED*	Stationary	Auto*	Pyra*
		(Ours)	(Ours)	2024	2023	2023	2022	2019	2022	2022	2023	2022	2022	2021	2021
Weighted Average	SMAPE	11.917	**11.614**	11.698	11.807	11.829	11.927	11.851	14.718	13.525	13.639	12.840	12.780	12.909	16.987
	MASE	1.744	**1.534**	1.556	1.590	1.585	1.613	1.599	2.408	2.111	2.095	1.701	1.756	1.771	3.265
	OWA	0.932	**0.825**	0.838	0.851	0.851	0.861	0.855	1.172	1.051	1.051	0.918	0.930	0.939	1.480

Table 2: Average performance on long term forecasting tasks.

Models	Typhon (T4)		Typhon (GMC)		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
ETTm1	0.374	0.399	0.344	0.373	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	0.275	0.325	0.251	0.307	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	0.438	0.444	0.398	0.409	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	0.373	0.410	0.316	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
Exchange	0.363	0.406	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	0.436	0.278	0.392	0.264	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	0.245	0.276	0.211	0.258	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

Table 3: Average performance on imputation tasks. We randomly mask 12.5%, 25%, 37.5%, 50% time points in length-96 time series.

Models	Typhon (T4)		Typhon (GMC)		FedFormer		ModernTCN		Reformer		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	0.027	0.104	0.033	0.124	0.062	0.177	0.020	0.093	0.407	0.410	0.070	0.166	0.045	0.133	0.041	0.143	0.419	0.419	0.027	0.107	0.093	0.206
ETTm2	0.026	0.099	0.029	0.103	0.101	0.215	0.019	0.082	0.288	0.332	0.032	0.108	0.028	0.098	0.046	0.149	0.358	0.404	0.022	0.088	0.096	0.208
ETTh1	0.076	0.186	0.082	0.195	0.117	0.246	0.050	0.150	0.288	0.332	0.141	0.242	0.133	0.236	0.132	0.251	0.358	0.404	0.078	0.187	0.201	0.306
ETTh2	0.058	0.159	0.053	0.148	0.163	0.279	0.042	0.131	0.288	0.332	0.066	0.165	0.066	0.164	0.122	0.240	0.358	0.404	0.049	0.146	0.142	0.306
Weather	0.033	0.051	0.039	0.060	0.099	0.203	0.027	0.044	0.288	0.332	0.034	0.058	0.033	0.057	0.036	0.090	0.358	0.404	0.030	0.054	0.052	0.110

7 Conclusion

We present Typhon, a general and flexible framework which adapts 1-dimensional sequence models to multivariate time series. We use two 1-dimensional sequence models across time variate and feature variate dimensions, using a dimension mixer and discretization and demonstrate that this better helps capture and tie together the information across the time and feature variate dimensions. We provide a special case of Typhon - a Gated Multiresolution Convolution architecture - which uses convolutions with iterative kernel dimensions to retain as much information as possible when moving autoregressively. We evaluate on a variety of time series tasks such as classification and long term forecasting, demonstrating the state of the art performance of Typhon. We also ascertain the importance of each component in contributing to the strong performance of Typhon through an ablation study.

We believe there is great potential for improvement of efficiency, particularly in the parallel scan, possibly through using more hardware-aware implementations and optimizations. We also leave possible methods from numerical linear algebra and control theory in developing a more optimal dimension mixer. We also note that a promising direction is to explore the potential of Typhon with its 2D inductive bias for other high dimensional data modalities and different tasks such as images, videos, multi-channel speech where prior 1 dimensional sequence models have been applied.

References

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

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

[3] 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).

[4] 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*.

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

[6] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. 2024. xLSTM: Extended Long Short-Term Memory. *arXiv preprint arXiv:2405.04517* (2024).

[7] Ali Behrouz and Farnoosh Hashemi. 2023. Learning Temporal Higher-order Patterns to Detect Anomalous Brain Activity. In *Proceedings of the 3rd Machine Learning for Health Symposium (Proceedings of Machine Learning Research, Vol. 225)*, Stefan Hegselmann, Antonio Parziale, Divya Shanmugam, Shengpu Tang, Mercy Nyamewaa Asiedu, Serina Chang, Tom Hartvigsen, and Harvineet Singh (Eds.). PMLR, 39–51.

[8] 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.

[9] 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.

[10] 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).

[11] 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.*

[12] 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).

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

[14] 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.

[15] 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.

[16] 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).

[17] 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).

[18] 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).

[19] Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In *International Conference on Machine Learning (ICML).*

[20] 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).

[21] Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. 2024. Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models. *arXiv preprint arXiv:2402.19427* (2024).

[22] Emmanuel de Bézenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. 2020. Normalizing kalman filters for multivariate time series analysis. *Advances in Neural Information Processing Systems* 33 (2020), 2995–3007.

[23] 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.*

[24] 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

[25] 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.

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

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

[28] 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.

[29] 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).

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

[31] 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.

[32] 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.).

[33] Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations.*

[34] 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.

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

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

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

[38] Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. 2024. LocalMamba: Visual State Space Model with Windowed Selective Scan. *arXiv preprint arXiv:2403.09338* (2024).

[39] 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).

[40] 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.

[41] 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.*

[42] Mahdi Karami and Ali Ghodsi. 2024. Orchid: Flexible and Data-Dependent Convolution for Sequence Modeling. In *Thirty-eighth Conference on Advances in Neural Information Processing Systems.*

[43] Shruti Kaushik, Abhinav Choudhury, Pankaj Kumar Sheron, Nataraj Dasgupta, Sayee Natarajan, Larry A Pickett, and Varun Dutt. 2020. AI in healthcare: time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in big data* 3 (2020), 4.

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

[45] Sun-Yuan Kung, Bernard C Levy, Martin Morf, and Thomas Kailath. 1977. New results in 2-D systems theory, Part II: 2-D state-space models—realization and the notions of controllability, observability, and minimality. *Proc. IEEE* 65, 6 (1977), 945–961.

[46] 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.*

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

[48] 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).

[49] 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).

[50] 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.

[51] 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.

[52] 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).*

[53] 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.*

[54] 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).

[55] 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.

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

[57] 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.*

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

[59] Eric Nguyen, Karan Goel, Albert Gu, Gordon Downs, Preey Shah, Tri Dao, Stephen Baccus, and Christopher Ré. 2022. S4nd: Modeling images and videos

as multidimensional signals with state spaces. *Advances in neural information processing systems* 35 (2022), 2846–2861.

[60] 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).

[61] 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)*.

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

[63] Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. 2024. Caduceus: Bi-directional equivariant long-range dna sequence modeling. *arXiv preprint arXiv:2403.03234* (2024).

[64] 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)*.

[65] Jiaxin Shi, Ke Alexander Wang, and Emily Fox. 2023. Sequence modeling with multiresolution convolutional memory. In *International Conference on Machine Learning*. PMLR, 31312–31327.

[66] 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*.

[67] 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).

[68] William Toner and Luke Darlow. 2024. An Analysis of Linear Time Series Forecasting Models. *International conference on machine learning (ICML)* (2024).

[69] 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.

[70] 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)*.

[71] 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).

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

[73] 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).

[74] 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*.

[75] 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*.

[76] 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)*.

[77] 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.

[78] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. 2018. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*. 418–434.

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

[80] 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).

[81] 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.

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

[83] 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.

[84] Michael Zhang, Khaled K Saab, Michael Poli, Tri Dao, Karan Goel, and Christopher Ré. 2023. Effectively modeling time series with simple discrete state spaces. *arXiv preprint arXiv:2303.09489* (2023).

[85] 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*.

[86] 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).

[87] 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*.

[88] 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.

[89] 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)*.

[90] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. 2024. Vision Mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417* (2024).

A Appendix

A.1 Background Information: 1-D State Space Models

1D Space State Models (SSMs) are linear time-invariant systems that map input sequence $x(t) \in \mathbb{R}^L \mapsto y(t) \in \mathbb{R}^L$ [2]. SSMs use a latent state $h(t) \in \mathbb{R}^{N \times L}$, transition parameter $\mathbf{A} \in \mathbb{R}^{N \times N}$, and projection parameters $\mathbf{B} \in \mathbb{R}^{N \times 1}, \mathbf{C} \in \mathbb{R}^{1 \times N}$ to model the input and output as:

$$h'(t) = \mathbf{A}\, h(t) + \mathbf{B}\, x(t), \qquad y(t) = \mathbf{C}\, h(t). \tag{21}$$

Most existing SSMs [12, 30, 33], first discretize the signals $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. That is, using a parameter Δ and zero-order hold, the discretized formulation is defined as:

$$h_t = \bar{\mathbf{A}}\, h_{t-1} + \bar{\mathbf{B}}\, x_t, \qquad y_t = \mathbf{C}\, h_t, \tag{22}$$

where $\bar{\mathbf{A}} = \exp(\Delta\mathbf{A})$ and $\bar{\mathbf{B}} = (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A} - I))\ .\ \Delta\mathbf{B}$. [31] show that discrete SSMs can be interpreted as both convolutions and recurrent networks: i.e.,

$$\begin{aligned} \bar{\mathbf{K}} &= \left(\mathbf{C}\bar{\mathbf{B}}, \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}}, \dots, \mathbf{C}\bar{\mathbf{A}}^{L-1}\bar{\mathbf{B}}\right), \\ y &= x * \bar{\mathbf{K}}, \end{aligned} \tag{23}$$

which makes their training and inference very efficient as a convolution and recurrent model, respectively.

A.2 Experiment Details

The experimental details are reported in Table 6.

A.3 Full Experimental Results

A.3.1 Short Term Forecasting Full Results.

A.3.2 Long Term Forecasting Full Results.

A.3.3 Anomaly Detection.

A.3.4 Imputation.

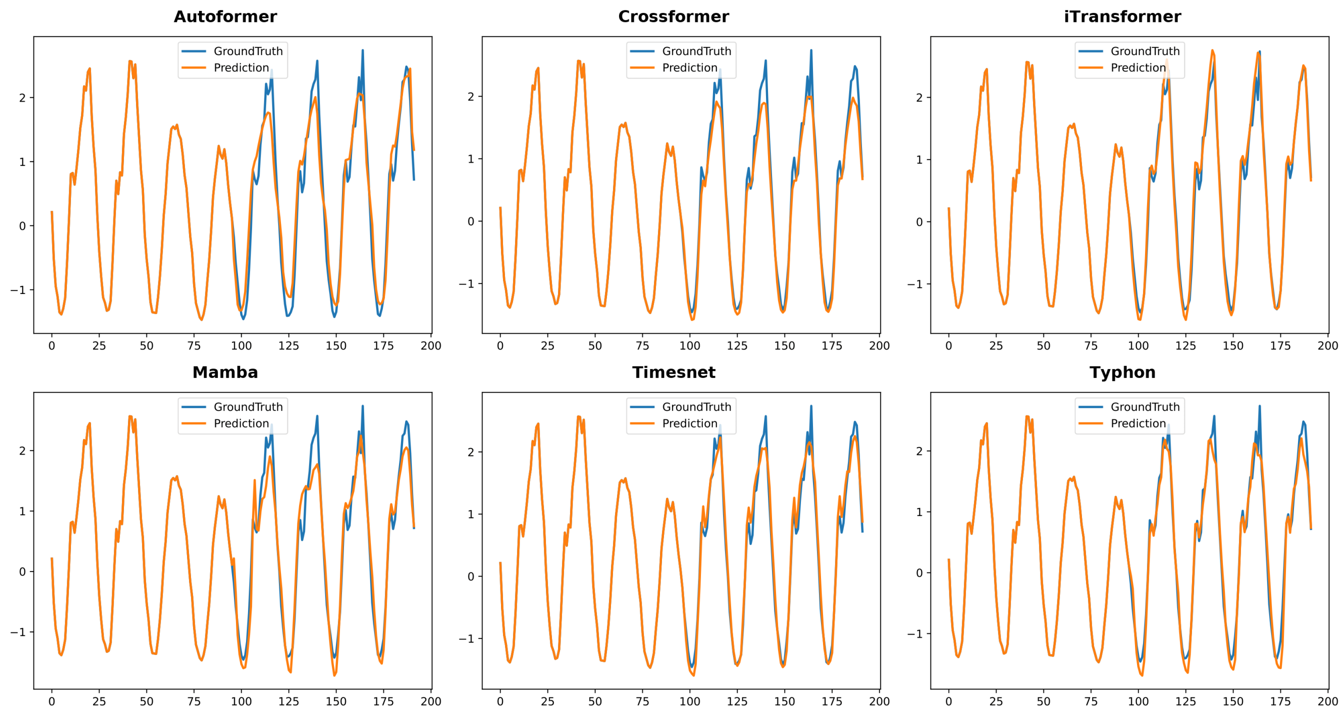

Figure 4: Visualization of Traffic Long Term Forecasting results given by models under the input-96-predict-336 setting. The blue lines stand for the ground truth and the orange lines stand for predicted values.

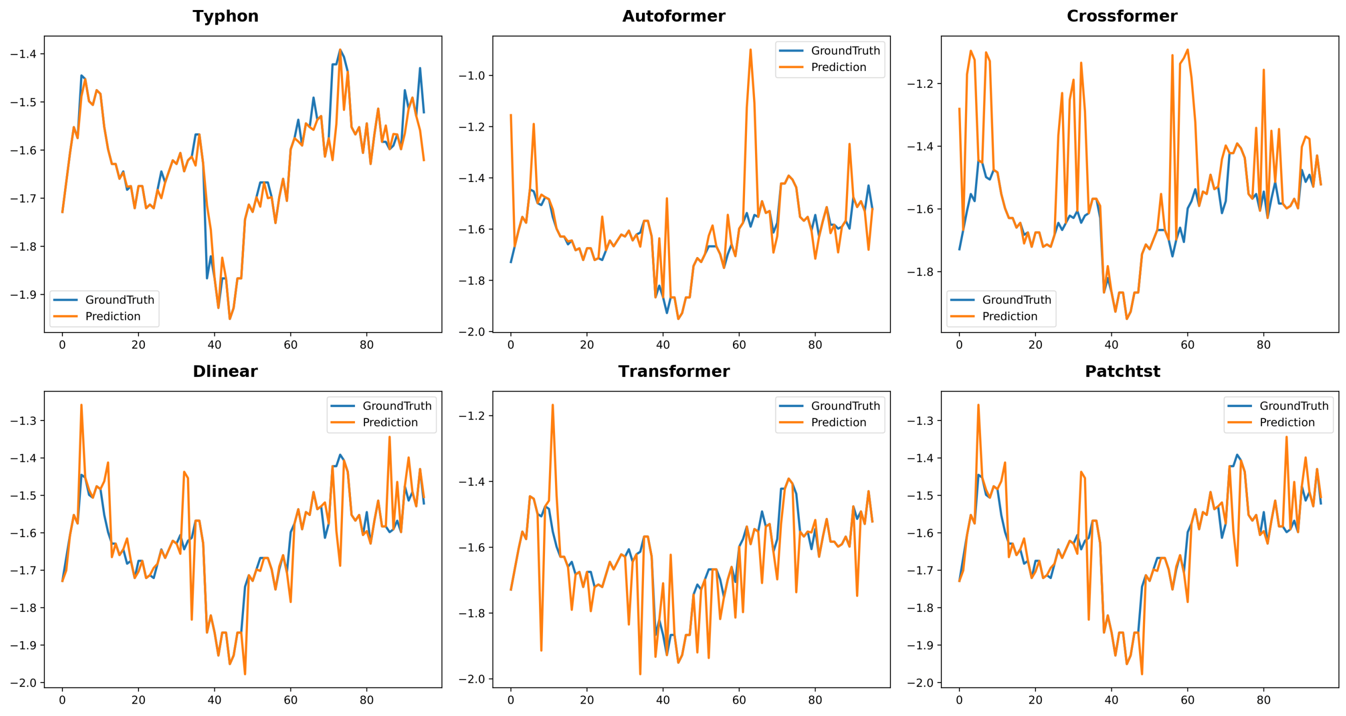

Figure 5: Visualization of ETTm1 imputation results given by models under the 25% mask ratio setting. The blue lines stand for the ground truth and the orange lines stand for predicted values.

Table 6: 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 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.

Models		Typhon (**Ours**)	ModernTCN [2024])	PatchTST [2023]	TimesNet [2023]	N-HiTS [2023]	N-BEATS* [2022]	ETS* [2019]	LightTS [2022]	DLinear [2022]	FED* [2023b]	Stationary [2022]	Auto* [2022b]	Pyra* [2021]	In* [2021]	Re* [2021]
Yearly	SMAPE	13.455	13.226	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	3.023	2.957	2.985	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	0.801	0.777	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	10.243	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	1.192	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	0.908	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	12.752	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	0.937	0.917	0.930	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	0.887	0.866	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	4.848	4.715	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	3.236	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	1.004	0.986	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	11.917	11.698	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	1.744	1.556	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	0.932	0.838	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

Table 9: Long-term forecasting task with different horizons H. The best results are bolded. We include the results for Patching as well. Note that & represents Typhon with Gated Multiresolution Convolution, $*$ represents Typhon with TTT-Linear [67] and Transformer [69], % denotes Typhon with TTT-Linear and Transformer with Patching with patching dimension 4, stride dimension 1, and pad dimension 3.

		Typhon* (ours)		Typhon$^\%$ (ours)		TCN [2024]		iTransformer [2024a]		RLinear [2023]		PatchTST [2023]		Crossformer [2023]		TiDE [2023]		TimesNet [2023]		DLinear [2023a]		SCINet [2022c]		FEDformer [2022]		Stationary [2022a]		Autoformer [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
ETTm1	96	0.335	0.371	0.324	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	0.365	0.391	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	0.398	0.406	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	0.407	0.431	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	0.374	0.399	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	0.174	0.263	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	0.231	0.302	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	0.306	0.344	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	0.389	0.401	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	0.275	0.325	0.281	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	0.376	0.399	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	0.431	0.440	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	0.473	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	0.486	0.476	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	0.438	0.444	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	0.301	0.370	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	0.392	0.403	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	0.396	0.423	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	0.406	0.447	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	0.373	0.410	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	0.954	0.723	0.437	0.449	0.526	0.516	0.450	0.459
Exchange	96	0.09	0.209	0.089	0.201	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	0.223	0.338	0.232	0.351	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	0.401	0.455	0.416	0.445	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	0.741	0.623	0.771	0.789	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	0.363	0.406	0.377	0.446	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	0.461	0.263	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	0.408	0.277	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	0.427	0.274	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	0.449	0.301	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	0.436	0.278	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	0.164	0.218	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	0.208	0.256	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	0.261	0.267	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	0.357	0.364	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	0.245	0.276	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

Table 10: 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. For Typhon models we let & represent Gated Multiresolution Convolution, $*$ denotes TTT-Linear [67] and Transformer [69], % denotes Typhon with xLSTM [6] and TTT-Linear, and # denotes Typhon with TTT-Linear [83] and TTT-Linear.

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	[2022a]	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	[2022]	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	[2022]	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
Typhon$^{\#}$	(ours)	88.10	82.72	85.33	89.29	76.54	81.23	89.91	55.35	69.80	91.22	95.55	93.33	98.33	88.87	96.96	85.33
Typhon$^{\%}$	(ours)	87.31	80.76	83.90	89.54	79.90	81.42	90.05	56.81	70.98	90.02	95.29	92.58	97.91	95.88	96.89	85.54
Typhon*	(ours)	88.16	82.74	85.05	89.16	88.37	81.79	90.35	58.12	71.55	92.18	95.19	92.30	98.23	96.19	96.97	86.53