HiPPO-KAN: Efficient KAN model for Time Series Analysis

SangJong Lee
XaaH Corp
Seoul, Korea
sangjong@xaah.xyz

Jin-Kwang Kim
XaaH Corp
Seoul, Korea
jinkwang@xaah.xyz

JunHo Kim
XaaH Corp
Seoul, Korea
demyank@xaah.xyz

TaeHan Kim
XaaH Corp
Seoul, Korea
taehankim@xaah.xyz

James Lee
XaaH Corp
Seoul, Korea
jaminyx@xaah.xyz

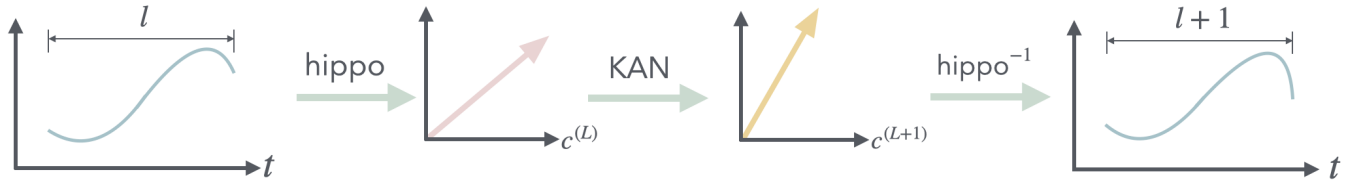

Figure 1: Overview of the HiPPO-KAN model architecture. The time series data is encoded using the HiPPO framework, transformed by the Kolmogorov-Arnold Network (KAN), and decoded back to the time domain, effectively serving as an auto-encoder.

Abstract

In this study, we introduce a parameter-efficient model that outperforms traditional models in time series forecasting, by integrating High-order Polynomial Projection (HiPPO) theory into the Kolmogorov-Arnold network (KAN) framework. This HiPPO-KAN model achieves superior performance on long sequence data without increasing parameter count. Experimental results demonstrate that HiPPO-KAN maintains a constant parameter count while varying window sizes, in contrast to KAN, whose parameter count increases linearly with window size. Surprisingly, although the HiPPO-KAN model keeps a constant parameter count as increasing window size, it significantly outperforms KAN model at larger window sizes. These results indicate that HiPPO-KAN offers significant parameter efficiency and scalability advantages for time series forecasting. Additionally, we address the lagging problem commonly encountered in time series forecasting models, where predictions fail to promptly capture sudden changes in the data. By modifying the loss function to compute the Mean Squared Error (MSE) directly on the coefficient vectors in the HiPPO domain, we effectively resolve the lagging problem, resulting in predictions that closely follow the actual time series data. By incorporating HiPPO theory into KAN, this study showcases an efficient approach for handling long sequences with improved predictive accuracy, offering practical contributions for applications in large-scale time series data.

ACM ICAIF P2P Workshop 2024, November 14, 2024, New York, USA

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

CCS Concepts

• **Information systems** → **Temporal data**; • **Computing methodologies** → **Neural networks**; • **Mathematics of computing** → *Nonlinear equations.*

Keywords

Time Series Forecasting, Kolmogorov-Arnold Network (KAN), Parameter Efficiency, Long-Term Dependencies, Coefficient-Based Loss Function, Lagging Problem, Financial Time Series Analysis, Nonlinear Dynamics

ACM Reference Format:
SangJong Lee, Jin-Kwang Kim, JunHo Kim, TaeHan Kim, and James Lee. 2018. HiPPO-KAN: Efficient KAN model for Time Series Analysis. In *Proceedings of From Prototype to Production: Deploying Real-World AI / ML Models in the Financial Industry (ACM ICAIF P2P Workshop 2024).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Deep learning aims to approximate complex functions, particularly those involving non-linearity or high dimensionality. Multilayer Perceptrons (MLPs) have been foundational in this area, with their ability to represent non-linear functions guaranteed by the universal approximation theorem [2, 6]. Recently, the Kolmogorov-Arnold Network (KAN) has emerged as a promising alternative to MLPs [17, 18]. Unlike MLPs, KAN learns activation functions rather than edge weights, drawing upon the Kolmogorov-Arnold Theorem (KAT) by assuming smooth activation functions. This approach allows KAN to outperform MLPs with better scaling laws, offering new avenues for modeling complex functions.

Various models based on MLPs, Recurrent Neural Networks (RNNs), and Long Short-Term Memory networks (LSTMs) have been developed for time-series analysis, aiming to capture complex patterns and non-linearities [5, 11, 13, 16, 23]. Additionally, deep state space models have gained prominence in time-series forecasting [19]. These models combine the strengths of traditional state space models with deep learning to capture complex temporal dynamics more effectively [12, 15, 19]. However, these methods often struggle with capturing complex patterns and, especially, learning long-term dependencies [1].

Long-term dependencies are particularly important in time-series analysis, as real-world datasets in finance, weather forecasting, and energy consumption involve patterns that evolve over extended periods. Capturing these dependencies enables models to make more accurate predictions by considering broader trends and delayed effects. To address these challenges, Gu et al. introduced the High-order Polynomial Projection Operator (HiPPO) theory and the Structured State Space (S4) model [7–10], effectively capturing long-range dependencies by performing online function approximation with special initial conditions in the state space transition equation.

Building upon the HiPPO theory, we enhance the capabilities of the Kolmogorov-Arnold Network for time-series analysis. The HiPPO framework uses a special combination of matrices A and B in the state space model's transition equation to map sequential data into a finite-dimensional space expanded by well-defined polynomial bases. This representation allows time-series data to be encapsulated as a coefficient vector whose dimension is independent of the sequence length.

Leveraging this property, we propose the HiPPO-KAN model, which effectively forecasts future time series with fewer parameters. The model encodes time-series data into a fixed-dimensional coefficient vector using the HiPPO framework, transforms this vector within the same dimensional space using KAN as a function approximator, and decodes the transformed coefficient vector back into the time domain using the inverse HiPPO function. This process is analogous to an autoencoder, where the encoder and decoder are defined by the HiPPO transformations, and the latent space manipulation is handled by KAN.

Our contributions demonstrate that HiPPO-KAN achieves superior parameter efficiency in univariate time-series prediction tasks, with the coefficient vector's dimension remaining fixed regardless of input sequence length. This scalability is crucial for practical applications involving large datasets. We show that HiPPO-KAN outperforms traditional KAN and other models specialized in handling sequential data, such as RNNs and LSTMs, particularly in long-range forecasting scenarios. By effectively capturing long-term dependencies through the HiPPO framework, our model provides more accurate predictions compared to KAN alone. Furthermore, the integration of HiPPO coefficients offers a concise and interpretable state representation of the time-series system. Combined with KAN's transparent architecture, this allows for better understanding and interpretability of the model's internal workings.

By incorporating HiPPO theory into the KAN framework, we introduce an efficient approach for handling long sequences with improved predictive accuracy, offering practical contributions for applications in large-scale time-series data.

2 Backgrounds

2.1 State Space Model

State space model can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \tag{1}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \tag{2}$$

where $\mathbf{u}(t) \in \mathbb{R}^l$ is an input vector, $\mathbf{x}(t) \in \mathbb{R}^N$ is a hidden state vector, and $\mathbf{y}(t) \in \mathbb{R}^k$ is an output vector. Eq.(1) describes the state dynamics, showing how the state $\mathbf{x}(t)$ evolves over time based on its current value and the input $\mathbf{u}(t)$. The matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ defines the influence of the current state on its rate of change, while $\mathbf{B} \in \mathbb{R}^{N\times l}$ defines how the input affects the state dynamics. Eq.(2) represents the output equation, illustrating how the current state and input produce the output $\mathbf{y}(t)$. The matrix $\mathbf{C} \in \mathbb{R}^{k\times N}$ maps the state to the output, and $\mathbf{D} \in \mathbb{R}^{k\times l}$ maps the input directly to the output.

In many cases, especially when implementing skip connections akin to those in deep learning architectures, we can set $\mathbf{D} = 0$. This simplifies the output equation to

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t). \tag{3}$$

By doing so, the output depends solely on the internal state, allowing the model to focus on the learned representations within $\mathbf{x}(t)$ without direct influence from the immediate input $\mathbf{u}(t)$. Gu et al. showed that when the system is linear time-invariant (LTI), the SSM reduces to a sequence-to-sequence mapping by defining a convolution mapping

$$K(t) = \mathbf{C}e^{t\mathbf{A}}\mathbf{B}, \quad \mathbf{y}(t) = (\mathbf{K} * \mathbf{u})(t). \tag{4}$$

Gu et al.[7] also showed that, by selecting specific initial conditions for the parameters $(\mathbf{A}, \mathbf{B})$, $e^{t\mathbf{A}}\mathbf{B}$ becomes a vector of N basis functions. This result enables the state-space model to perform online function approximation using the HiPPO theory.

2.2 HiPPO Theory

In the context of continuous time series, the Legendre Memory Unit (LMU) [21] exemplifies an approach that employs continuous orthogonal functions—specifically, Legendre polynomials—to maintain a compressed representation of the entire history of input data. Building upon these principles, Gu et al. [7] established a strong theoretical foundation by connecting memorization to state-space models. Specifically, they demonstrated that a special initialization of the transition equation in the state-space model enables closed-form function approximation, effectively capturing long-term dependencies in sequential data. The HiPPO framework treats memorization as an online function approximation.

Suppose we have a univariate time series function:

$$f : \mathbb{R}_{\geq 0} \to \mathbb{R}, \quad t \mapsto f(t). \tag{5}$$

Since we are considering online function approximation, we define:

$$x_n(t) = \int_0^t \mathrm{d}s\, \omega(t,s) p_n(t,s) u(s), \tag{6}$$

$$\langle p_n,\, p_m \rangle_\omega \equiv \int_0^t \mathrm{d}s\, \omega(t,s) p_n(t,s) p_m(t,s) = \delta_{n,m}, \tag{7}$$

which states that for every fixed t, the function p_n belong to a Hilbert space $\mathcal{H}$ and form an orthonormal basis with respect to the measure $\omega(t, s)$. Rewriting Eq.(6), we have:

$$x_n(t) = \int_0^t \mathrm{d}s\omega(t,s)\, p_n(t,s)u(s) = \langle u,\ p_n(t)\rangle_\omega\,, \tag{8}$$

which indicates that the state vector $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_N(t)]^T$ represents the projection of $u(s)$ for $s \le t$ onto an orthonormal basis with respect to weighted inner product defined by $\omega(t, s)$.

If we assume completeness, we have:

$$u(s) = \lim_{N\to\infty} \sum_{n=1}^{N} x_n(t)p_n(t,\ s) \tag{9}$$

for all $s \le t$ due to the completeness of the basis function. Since we are dealing with a finite N, by choosing an appropriate cutoff, we obtain an approximate representation of the function $u(s)$. Gu et al. [10] defined this problem as online function approximation in the HiPPO theory.

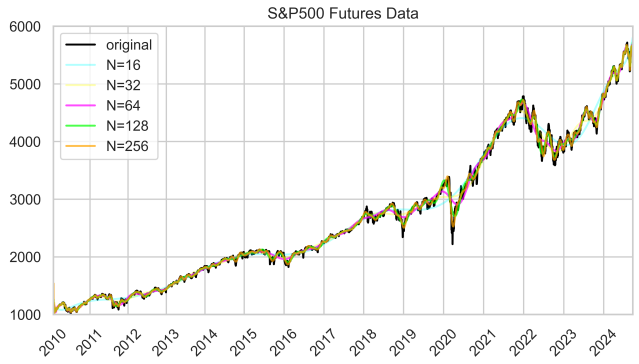

Figure 2: Comparison of S&P 500 data with approximated data using HiPPO for different state space dimensions.

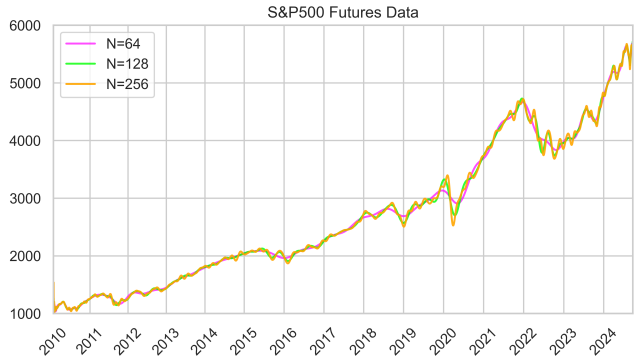

Figure 3: HiPPO approximation of S&P 500 data for state space dimensions $N = 64, 128, 256$.

From a physical standpoint, this is analogous to a multipole expansion, where each term has a specific physical interpretation. In the case of a nonlinear function that takes the coefficients of a multipole expansion as inputs, each coefficient corresponds to a node within the function. Ideally, during the process of learning this nonlinear function, deriving a closed-form solution or understanding how each node operates would greatly aid in physical interpretation. This understanding can provide significant insights into the underlying physics and how the model represents the system. To further enhance this interpretability, we utilized KAN to model the mapping from the coefficients of sequential data of length l to sequential data of length $l + 1$.

2.3 KAN

2.3.1 Kolmogorov-Arnold Theorem. The Kolmogorov-Arnold Representation Theorem states that any continuous multivariate function f defined on a bounded domain I^n, where n is the number of variables and $I = [0, 1]$, can be expressed as a finite sum of compositions of continuous univariate functions and addition. Specifically, for a smooth function f:

$$f : I^n \to \mathbb{R}, \quad \mathbf{x} \in I^n \mapsto \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right), \tag{10}$$

where each $\phi_{q,p} : I \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$ are continuous univariate functions. This theorem reveals that any multivariate continuous function can be constructed using only univariate continuous functions and addition, significantly simplifying their analysis. This decomposition reduces the complexity inherent in multivariate functions, making them more tractable for approximation methods.

2.3.2 Kolmogorov-Arnold Network. Building upon the Kolmogorov-Arnold representation theorem, the Kolmogorov-Arnold Network (KAN) is designed to explicitly parametrize this representation for practical function approximation in neural networks [17, 18]. Since we have decomposed the multivariate function into univariate functions, the problem reduces to parametrizing these univariate functions. To achieve this, we can use B-splines due to their flexibility and smoothness properties, which are advantageous for interpolations. From the perspective of generalizing the Kolmogorov–Arnold (KA) representation theorem and extending it to deeper networks, the network architecture can be expressed as follows:

$$[n_0, n_1, \cdots, n_L], \tag{11}$$

where n_l is the number of nodes in the l-th layer. The pre-activation values are given by:

$$x_{l+1,j} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i}), \quad l = 0, \ldots, L-1;\ j = 1, \ldots, n_{l+1} \tag{12}$$

where $\phi_{l,j,i}$ are the univariate functions with learnable parameters in the l-th layer.

In practice, the univariate functions $\phi_{l,j,i}$ in KAN are parametrized using B-splines to capture complex nonlinearities while maintaining smoothness and flexibility. To enhance the representational capacity of the network and facilitate efficient training, KAN employs residual activation functions that combine a basis function with a spline function. Specifically, the activation function at each node is defined as

$$\phi(x) = w_b\, b(x) + w_s\, \mathrm{spline}(x) \tag{13}$$

where $b(x)$ is a predefined basis function, w_b and w_s are learnable weights, and $\text{spline}(x)$ is a spline function constructed from B-spline basis functions. The basis function $b(x)$ is typically chosen as the SiLU (Sigmoid Linear Unit) activation function due to its smoothness and nonlinearity.

The overall network function is then:

$$\text{KAN}(\mathbf{x}) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_0)(\mathbf{x}). \tag{14}$$

In this expression, Φ_l represents the vector of univariate functions at layer l, and the composition of these functions across layers forms the basis of KAN's ability to approximate multivariate functions.

In the context our work, we extend KAN by integrating with the HiPPO framework to efficiently handle time series data. This integration allows us to leverage KAN's function approximation capabilities while benefiting from HiPPO's ability to represent sequential data in a fixed-dimensional space.

2.4 Time-series forecasting using KAN

Since its introduction, KAN have been proved to be a powerful tool for time-series forecasting due to their effective approximation capabilities and training efficiency. It has been shown that KAN models outperform MLP models in time-series forecasting, both in terms of accuracy and computational efficiency [20, 22]. Furthermore, when KAN layers are incorporated within recurrent neural networks (RNNs) and transformer architectures, they excel in multi-horizon forecasting tasks with reduced overfitting issues [3, 4].

While these approaches validate the effectiveness of KAN models in time-series prediction and outperforms traditional models specialized in sequential data (e.g., RNN and GRU), they involve integrating KAN into complex architectures, which can increases model complexity and computational demands. In this study, however, we propose an alternative methodology that combines KAN models with HiPPO transformation. By integrating KAN with the HiPPO transformation, we construct a simpler model architecture that retains high predictive performance without relying on complex recurrent or transformer structures.

3 HiPPO-KAN

Building upon the HiPPO framework, we consider a univariate time series $u_{1:L} \in \mathbb{R}^L$. The HiPPO transformation maps this time series into a coefficient vector $\mathbf{c}^{(L)} \in \mathbb{R}^N$ via the mapping

$$\text{hippo}_L : \mathbb{R}^L \to \mathbb{R}^N, \quad u_{1:L} \mapsto \mathbf{c}^{(L)} = \text{hippo}_L(u_{1:L}), \tag{15}$$

where N is the dimension of the hidden state. In our proposed method, the KAN is utilized to model the mapping between coefficient vectors corresponding to time series of length L and $L+1$. Specifically, KAN transforms the coefficient vector $\mathbf{c}^{(L)}$ into a new coefficient vector $\mathbf{c}^{(L+1)}$:

$$\text{KAN} : \mathbb{R}^N \to \mathbb{R}^N, \quad \mathbf{c}^{(L)} \mapsto \mathbf{c}^{(L+1)} = \text{KAN}(\mathbf{c}^{(L)}). \tag{16}$$

The resultant coefficient vector $\mathbf{c}^{(L+1)}$ represents the encoded state of the time series extended to length $L+1$. Given the coefficient $\mathbf{c}^{(L+1)}$, we can easily construct a time series data of length $L+1$. Let this process be denoted as hippo^{-1}:

$$\text{hippo}^{-1}_{L+1} : \mathbb{R}^N \to \mathbb{R}^{L+1}, \ \mathbf{c}^{(L+1)} \mapsto u'_{1:L+1} = \text{hippo}^{-1}_{L+1}(\mathbf{c}^{(L+1)}), \tag{17}$$

where $u_{1:L}$ and $u'_{1:L+1}$ are different time series. This process effectively extends the original time series by one time step, generating a prediction for the next value in the sequence. By operating within the fixed-dimensional coefficient space $\mathbb{R}^N$, where N is independent of the sequence length L, our approach maintains parameter efficiency and scalability. The use of KAN in this context allows for the modeling of complex nonlinear relationships between the coefficients, capturing the underlying dynamics of the time series.

3.1 Definition of HiPPO-KAN

We define the HiPPO-KAN model as a sequence-to-sequence (seq2seq) mapping that integrates the HiPPO transformations with the KAN mapping. Formally, the HiPPO-KAN model is defined as:

$$\text{HiPPO-KAN} \equiv \text{hippo}^{-1}_{L+1} \circ \text{KAN} \circ \text{hippo}_L. \tag{18}$$

This composite mapping takes the original time series $\{u_t\}_{t=1}^L$ as input and produces an extended time series $\{u_t\}_{t=1}^{L+1}$ as output:

$$\text{HiPPO-KAN} : \mathbb{R}^L \to \mathbb{R}^{L+1}, \quad \{u_t\}_{t=1}^L \mapsto \{u'_t\}_{t=1}^{L+1}. \tag{19}$$

In other words, HiPPO-KAN maps a time series of length L to a different time series of length $L+1$, effectively predicting the next value in the sequence while retaining the original sequence. By integrating these components, HiPPO-KAN effectively captures long-term dependencies and complex temporal patterns in time-series data. Operating within the coefficient space $\mathbb{R}^N$ ensures that the model remains parameter-efficient and scalable, as the dimensionality N does not depend on the sequence length L.

Following the definition of the HiPPO-KAN model, we derive its explicit output formulation by integrating the HiPPO transformations with the KAN mapping. Applying the hippo_L transformation to the input time series $\{u_t\}_{t=1}^L$, the function $f(s)$ can be approximately represented in terms of orthogonal basis functions:

$$f(s) \approx \sum_{n=1}^{N} c_n \, p_n(L, s), \tag{20}$$

where $c_n \in \mathbb{R}$ are the coefficients, and $p_n(L, s)$ are the HiPPO basis functions evaluated at time L for all $s \le L$.

Utilizing the KAN mapping, we update the coefficients to incorporate the system dynamics:

$$c'_n = \sum_{m=1}^{N} \Phi_{nm}(c_m), \tag{21}$$

where Φ_{nm} are the elements of the KAN matrix $\Phi \in \mathbb{R}^{N \times N}$. We defined hippo^{-1} as

$$\begin{aligned} u'_{1:L+1} &= \sum_{n=1}^{N} \left(c'_n + Bu_L\right) p_n(L+1, s) \\ &= \sum_{n=1}^{N} \left(\sum_{m=1}^{N} \Phi_{nm}(c_m) + Bu_L \right) p_n(L+1, s), \end{aligned} \tag{22}$$

where $B \in \mathbb{R}^N$ is learnable parameters. This is analogous to the $\mathbf{B}u(t)$ term in the state-space model's state equation. Evaluating at $s = L + 1$, the final output for the next time step becomes:

$$u'_{L+1} = \sum_{n=1}^{N} \left(\sum_{m=1}^{N} \Phi_{nm}(c_m) + Bu_L \right) p_n(L+1, L+1). \tag{23}$$

In the case of Leg-S, from the definition of the basis functions, we have $p_n(L+1, L+1) = \sqrt{2n+1}$ [7, 10]. Therefore, we obtain:

$$u'_{L+1} = \sum_{n=1}^{N} \sqrt{2n+1} \left(\sum_{m=1}^{N} \Phi_{nm}(c_m) + Bu_L \right). \tag{24}$$

This methodology resembles an autoencoder architecture, where the encoder (HiPPO transformation) compresses the input time series into a latent coefficient vector $\mathbf{c}^{(L)}$, whose dynamics are modeled by the KAN layers in our HiPPO-KAN model. The decoder (inverse HiPPO transformation) reconstructs the extended time series from $\mathbf{c}^{(L+1)}$. The fixed-dimensional latent space acts as a bottleneck, promoting efficient learning.

3.2 Methodology

3.2.1 Task Definition. In this study, we address the problem of time-series forecasting in the context of cryptocurrency markets, specifically focusing on the BTC-USDT trading pair. The objective is to predict the next price point given a historical sequence of observed prices. Formally, let $u_t t = 1^L$ denote a univariate time series representing the BTC-USDT prices at discrete time steps $t = 1, 2, \ldots, L$, where L is the window size. The forecasting task aims to estimate the subsequent value $uL + 1$ based on the given window of past observations.

Mathematically, the prediction function can be expressed as:

$$\hat{u}_{L+1} = f(u_1, u_2, \ldots, u_L), \tag{25}$$

where $f : \mathbb{R}^L \to \mathbb{R}$ is a mapping from the past L observations to the predicted next value $\hat{u}_{L+1}$. The challenges inherent in this task include:

- **Non-Stationarity:** Cryptocurrency prices exhibit high volatility and non-stationary behavior, making it difficult to model underlying patterns using traditional statistical methods.
- **Long-Term Dependencies:** Capturing long-term dependencies is essential, as market trends and cycles can influence future prices over extended periods.
- **Computational Efficiency:** Handling long sequences efficiently without a proportional increase in computational complexity or model parameters is critical for scalability.

Our approach utilizes the HiPPO-KAN model to effectively tackle these challenges by encoding the input time series into a fixed-dimensional coefficient vector using the HiPPO transformation. This allows the model to process long sequences while maintaining a constant parameter count, facilitating efficient learning and improved predictive accuracy.

3.2.2 Data Normalization. We evaluated the performance of HiPPO-KAN using BTC-USDT 1-minute futures data from January 1st to February 1st, comprising univariate time-series data. Prior to training, we normalized the raw time-series data using the formula, $(u_t - \mu)/\mu$, where μ denotes the mean value of the data within each window. This normalization serves several critical purposes in the context of time-series modeling. First, it centers the data around zero, which helps stabilize the training process and accelerate convergence by mitigating biases introduced by varying data scales. Second, scaling by the mean adjusts for fluctuations in the magnitude of the data across different windows, ensuring that the model's learning is not skewed by windows with larger absolute values.

By normalizing each window individually, we effectively address the non-stationarity inherent in financial time-series data, where statistical properties such as mean and variance can change over time. This window-specific normalization allows the model to focus on learning the underlying patterns and dynamics within each window without being influenced by shifts in the data scale. Consequently, this approach enhances the robustness of the model and improves its ability to generalize across different segments of the time series.

3.2.3 Loss Function for Model Training. Training the HiPPO-KAN model involves optimizing the network parameters to minimize the discrepancy between the predicted values and the actual observed values in the time-series data. We employ the Mean Squared Error (MSE) as the loss function, a standard choice for regression tasks in time-series forecasting due to its sensitivity to large errors.

The MSE loss function is defined as:

$$\mathcal{L}(\theta) = \frac{1}{D} \sum_{i=1}^{D} \left(u_{L+1}^{(i)} - \hat{u}_{L+1}^{(i)} \right)^2, \tag{26}$$

where θ represents the model parameters, D is the number of samples in the training set, $u^{(i)}L + 1$ is the true next value in the time series for the i-th sample, and $\hat{u}^{(i)}L + 1$ is the corresponding prediction made by the model.

Minimizing the MSE loss encourages the model to produce predictions that are, on average, as close as possible to the actual values, with larger errors being penalized more heavily due to the squaring operation. The choice of MSE as the loss function aligns with the evaluation metrics used in our experiments—namely, the Mean Squared Error (MSE) and Mean Absolute Error (MAE)—facilitating a consistent assessment of the model's performance during training and testing.

3.2.4 Experimental Results. The experimental results are presented in Tables 1 to facilitate a clear and concise comparison of model performances. Table 1 summarizes the results for a prediction horizon of 1. Each table includes the model name, window size, network width (architecture), Mean Squared Error (MSE), Mean Absolute Error (MAE), and the number of parameters used in the model.

By organizing the results in tabular form, we provide a straightforward means to compare the effectiveness of HiPPO-KAN against baseline models such as HiPPO-MLP, KAN, LSTM, and RNN across different configurations. This structured presentation highlights the consistency and scalability of HiPPO-KAN, especially in terms of parameter efficiency and predictive accuracy over varying window sizes and prediction horizons. The tables clearly demonstrate that HiPPO-KAN achieves superior performance with fewer parameters, emphasizing the advantages of integrating HiPPO transformations with KAN mappings in time series forecasting tasks.

Table 1: Performance comparison of models for prediction horizon 1. Best models are highlighted in bold.

Model	Window Size	Width	MSE	MAE	Parameters
HiPPO-KAN	120	[16, 16]	3.40×10^{-7}	4.14×10^{-4}	4,384
HiPPO-KAN	500	[16, 16]	3.34×10^{-7}	$\mathbf{3.95 \times 10^{-4}}$	4,384
HiPPO-KAN	1200	[16, 16]	$\mathbf{3.26 \times 10^{-7}}$	4.00×10^{-4}	4,384
HiPPO-MLP	120	[32, 64, 64, 32, 32]	2.33×10^{-6}	1.04×10^{-3}	9,792
HiPPO-MLP	500	[32, 64, 64, 32, 32]	2.68×10^{-5}	3.84×10^{-3}	9,792
HiPPO-MLP	1200	[32, 64, 64, 32, 32]	5.87×10^{-6}	1.96×10^{-3}	9,792
KAN	120	[120, 1]	8.9×10^{-7}	6.82×10^{-4}	1,680
KAN	500	[500, 1]	1.66×10^{-6}	9.62×10^{-4}	7,000
KAN	1200	[1200, 1]	4.03×10^{-6}	1.56×10^{-3}	16,800
LSTM	120	–	4.69×10^{-7}	4.99×10^{-4}	4,513
LSTM	500	–	6.50×10^{-7}	6.00×10^{-4}	4,513
LSTM	1200	–	9.21×10^{-7}	7.21×10^{-4}	4,513
RNN	120	–	1.14×10^{-6}	8.60×10^{-4}	12,673
RNN	500	–	1.09×10^{-6}	7.70×10^{-4}	12,673
RNN	1200	–	1.18×10^{-6}	7.79×10^{-4}	12,673

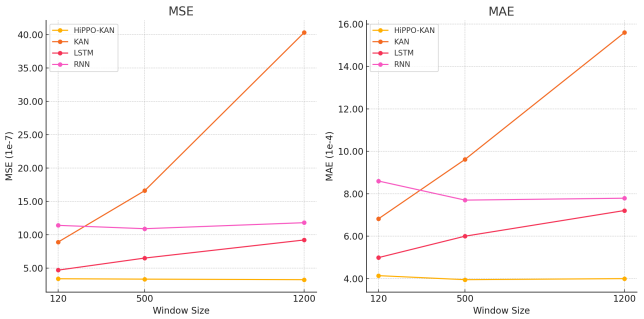

Figure 4: MSE and MAE comparisons for various models (HiPPO-KAN, KAN, LSTM, RNN) using different window sizes (120, 500, 1200). The results show the performance of each model in terms of error metrics as the window size increases.

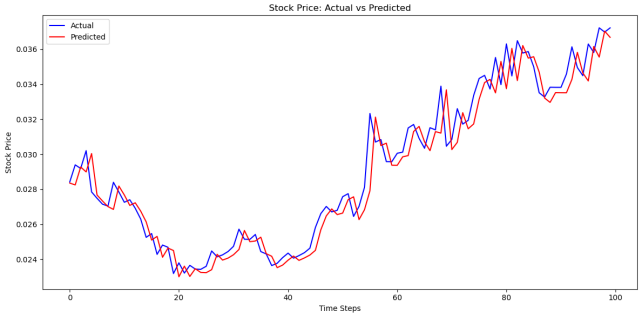

Figure 5: Lagging Effect in KAN, RNN, and LSTM Models. These models exhibit a tendency to produce outputs that closely mimic the preceding values, indicating an inability to capture rapid changes in the data effectively.

3.3 Lagging problem

While the result presented above are impressive, we observed that the model still suffers from the lagging problem when examining the plots of the predictions. The lagging problem refers to the phenomenon where the model's predictions lag behind the actual time series, failing to capture sudden changes promptly [14]. This issue is particularly detrimental in time series forecasting, where timely and accurate predictions are crucial.

To address this issue, we modified the loss function used during training and put $B = 0$. Instead of computing the MSE between the inverse-HiPPO-transformed outputs $\hat{u}_{1:L+1} = \text{hippo}_{L+1}^{-1}\left(\hat{\mathbf{c}}^{(L+1)}\right)$ and the actual time series $u_{1:L+1}$, we computed the MSE directly on the coefficient vectors in the HiPPO domain. Specificaly, the loss function is defined as:

$$\mathcal{L}(\theta) = \frac{1}{D}\sum_{i=1}^{D}\left|\mathbf{c}_{\text{true}}^{(L+1)(i)} - \hat{\mathbf{c}}^{(L+1)(i)}\right|^2 \tag{27}$$

where θ represents the model parameters, D is the number of samples in the training set. If the coefficient vector $\mathbf{c}$ has a 'true' subscript, it represents the true value obtained by applying the HiPPO transformation to the actual time series, whereas if the vector has a hat, it represents the predicted value output by the KAN model.

By training the model using this modified loss function, we aimed to align the learning process more closely with the underlying representation in the coefficient space, where the HiPPO transformation captures the essential dynamics of the time series. This approach emphasizes learning the progression of the coefficient directly, which may help the model respond more promptly to changes in the input data.

3.3.1 Interpretation of the Coefficient-Based Loss Function. When obtaining the coefficient vector $\mathbf{c}$, it is important to recognize that $\mathbf{c}$ does not correspond to a single, unique function. Instead, it encapsulates an approximation of the original time-series function within a finite-dimensional subspace spanned by the first N basis

functions. The true function $f_{\text{true}}(s)$ can be expressed as:

$$f_{\text{true}}(s) = \sum_{i=1}^{N} c_i p_i(t,s) + \sum_{i=N+1}^{\infty} c_i p_i(t,s)$$
$$= f(s) + \sum_{i=N+1}^{\infty} c_i p_i(t,s) \tag{28}$$

where $p_i(t,s)$ are the orthogonal basis functions of the HiPPO transformation, and c_i are the corresponding coefficients. Here, $f(s) = \sum_{i=1}^{N} c_i p_i(t,s)$ represents the finite-dimensional approximation of $f_{\text{true}}(s)$. The finite sum over $i = 1$ to N captures the primary components of the function, while the infinite sum over $i = N+1$ to ∞ represents the residual components not captured due to truncation at N. This implies that $\mathbf{c}$ corresponds to a class of functions sharing the same coefficients for the first N basis functions but potentially differing in higher-order terms. By working with this finite-dimensional approximation, the model focuses on the most significant features of the time series, enabling efficient learning and generalization.

By minimizing the average loss across the batch, the model effectively converges toward aligning the primary components (from 1 to N) of the coefficient vectors across different samples within each batch. This convergence of the primary components means that the model is effectively converging to a specific class of functions within the function space defined by the finite-dimensional basis. Each batch adjusts the model parameters to reduce discrepancies in these primary components, reinforcing the shared underlying dynamics among the batch samples.

This idea is the key to solving the lagging problem commonly observed in time-series forecasting models. By focusing on the convergence of the primary components, the model captures the essential dynamics of the time series more accurately and promptly. This emphasis allows the model to respond quickly to sudden changes in the data, effectively mitigating the lagging effect where models fail to predict abrupt shifts in the time series.

The batch averaging acts as a mechanism to align the model's predictions with the shared features across different time-series segments, guiding it toward a consensus representation. As a result, the model captures the dominant patterns and trends that are consistent across the batch, enhancing its ability to generalize and reducing the likelihood of overfitting to specific instances. The averaging effect smooths out idiosyncratic variations in individual samples, promoting the learning of robust features pertinent to the forecasting task. This convergence toward a specific function helps the model produce more accurate and reliable predictions, particularly when dealing with complex and noisy time-series data.

4 Conclusion

In this study, we introduced HiPPO-KAN, a novel model that integrates the HiPPO framework with the KAN model to enhance time series forecasting. By encoding time series data into a fixed-dimensional coefficient vector using the HiPPO transformation, and then modeling the progression of these coefficients with KAN, HiPPO-KAN efficiently performed time-series prediction task.

Our experimental results, as presented in Table 1, demonstrate that HiPPO-KAN consistently outperforms traditional KAN and other baseline models such as HiPPO-MLP, LSTM, and RNN across various window sizes and prediction horizons. Notably, HiPPO-KAN maintains a constant parameter count regardless of sequence length, highlighting its parameter efficiency and scalability. For example, at a window size of 1,200 and a prediction horizon of 1, HiPPO-KAN achieved an MSE of 3.26×10^{-7} and an MAE of 4.00×10^{-4}, compared to KAN's MSE of 4.03×10^{-6} and MAE of 1.56×10^{-3}, with fewer parameters.

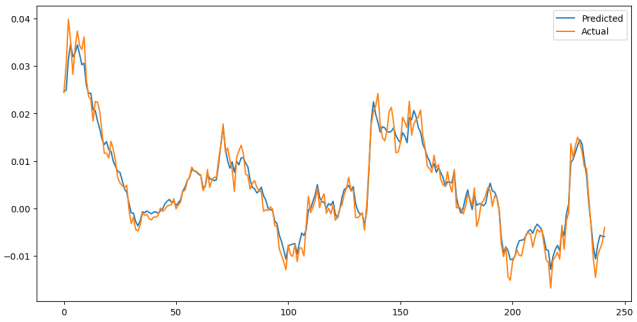

Figure 6: The modified loss function effectively resolves the lagging problem, resulting in predictions that closely follow the actual time series data. This result is based on a randomly selected segment of BTC-USDT 1-minute interval data, using a KAN architecture with a width of [16, 2, 16].

The integration of HiPPO theory into the KAN framework provides a powerful approach for handling long sequences without increasing the model size. By operating within a fixed-dimensional latent space, HiPPO-KAN not only improves predictive accuracy but also offers better interpretability of the model's internal workings. The use of KAN allows for modeling complex nonlinear relationships between the HiPPO coefficients, capturing the underlying dynamics of the time series more effectively than traditional methods. These promising results position HiPPO-KAN as a significant advancement in time-series forecasting, offering a scalable and efficient solution that could potentially revolutionize applications across various domains, from financial modeling to climate prediction.

Additionally, we addressed the lagging problem commonly encountered in time series forecasting models. By modifying the loss function to compute the MSE directly on the coefficient vectors in the HiPPO domain, we significantly improved the model's ability to capture sudden changes in the data without delay. This adjustment aligns the learning process more closely with the underlying dynamics of the time series, allowing HiPPO-KAN to produce predictions that closely follow the actual data, as illustrated in Figure 6.

4.1 Future Work

4.1.1 Integration with Graph Neural Networks for Multivariate Time Series. To extend the HiPPO-KAN model to handle multivariate time-series data, we propose integrating it with Graph Neural Networks (GNNs) [2]. In this framework, each variable or time series in the multivariate dataset is represented as a node within a graph structure. At each node, the HiPPO transformation encodes the

local time-series data into a fixed-dimensional coefficient vector, serving as a localized representation of the temporal dynamics.

These coefficient vectors act as local embeddings of the time series at each node. The edges of the graph define the interactions between nodes, capturing the dependencies and relationships among different variables in the dataset. By modeling these interactions, we can define functions that operate on pairs or groups of coefficient vectors, effectively allowing information to flow across the graph and capturing the multivariate dependencies.

This integration leverages the strength of HiPPO-KAN in modeling individual time series efficiently while utilizing the relational modeling capabilities of GNNs to handle the interconnectedness of multivariate data. Future work could focus on developing this combined HiPPO-KAN-GNN architecture, investigating how the interactions between nodes can be effectively modeled, and exploring the impact on forecasting accuracy and interpretability. This approach has the potential to address complex systems where variables are interdependent, such as in financial markets, climate modeling, and social network analysis.

References

[1] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.

[2] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velickovic. 2021. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *arXiv preprint arXiv:2104.13478* (2021).

[3] Remi Genet and Hugo Inzirillo. 2024. A Temporal Kolmogorov-Arnold Transformer for Time Series Forecasting. *arXiv preprint arXiv:2406.02486* (2024).

[4] Remi Genet and Hugo Inzirillo. 2024. Tkan: Temporal kolmogorov-arnold networks. *arXiv preprint arXiv:2405.07344* (2024).

[5] F. A. Gers, J. Schmidhuber, and F. Cummnins. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation* 12, 10 (2000), 2451–2471.

[6] I. Goodfellow, Y. Bengjo, and A. Courville. 2016. *Deep Learning*. The MIT Press.

[7] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré. 2020. HiPPO: Recurrent Memory with Optimal Polynomial Projections. *arXiv preprint arXiv:2008.07669* (2020).

[8] A. Gu, K. Goel, and C. Ré. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. *arXiv preprint arXiv:2111.00396* (2022).

[9] A. Gu, A. Gupta, K. Goel, and C. Ré. 2022. On the Parametrization and Initialization of Diagonal State Space Models. *arXiv preprint arXiv:2206.11893* (2022).

[10] A. Gu, I. Johnson, A. Timalsina, A. Rudra, and C. Re. 2022. How to Train Your HiPPO: State Space Models with Generalized Orthogonal Basis Projections. *arXiv preprint arXiv:2206.12037* (2022).

[11] H. S. Hippert, C. E. Pedreira, and R. C. Souza. 2001. Neural Networks for Short-term Load Forecasting: A review and evaluation. *IEEE Transactions on Power Systems* 16, 1 (2001), 44–55.

[12] H. Inzirillo. 2024. Deep State Space Recurrent Neural Networks for Time Series Forecasting. *arXiv preprint arXiv:2407.15236* (2024).

[13] Y. Kong, Z. Wang, Y. Nie, T. Zhou, S. Zohren, Y. Liang, P. Sun, and Q. Wen. 2024. Unlocking the Power of LSTM for Long Term Time Series Forecasting. *arXiv preprint arXiv:2408:10006* (2024).

[14] J. Li, L. Song, D. Wu, J. Shui, and T. Wang. 2023. Lagging problem in financial time series forecasting. *Neural Computing and Applications* 35 (2023), 20819–20839.

[15] L. Li, J. Yan, X. Yang, and Y. Jin. 2021. Learning Interpretable Deep State Space Model for Probabilistic Time Series Forecasting. *arXiv preprint arXiv:2102.00397* (2021).

[16] Z. C. Lipton, J. Berkowitz, and C. Elkan. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv preprint arXiv:1506.00019* (2015).

[17] Z. Liu, P. Ma, Y. Wang, W. Matusik, and M. Tegmark. 2024. KAN 2.0: Kolmogorov-Arnold Networks Meet Science. *arXiv preprint arXiv:2408.10205* (2024).

[18] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljiacic, T. Y. Hou, and M. Tegmark. 2024. KAN: Kolmogorov-Arnold Networks. *arXiv preprint arXiv:2404.19756* (2024).

[19] S. S. Rangapuram, M. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. 2018. Deep State Space Models for Time Series Forecasting. In *Advances in Neural Information Processing Systems*. 7796–7805.

[20] Cristian J Vaca-Rubio, Luis Blanco, Roberto Pereira, and Màrius Caus. 2024. Kolmogorov-arnold networks (kans) for time series analysis. *arXiv preprint arXiv:2405.08790* (2024).

[21] A. R. Voelker, I. Kajic, and C. Eliasmith. 2019. Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS 2019)*. Vancouver, Canada.

[22] Kunpeng Xu, Lifei Chen, and Shengrui Wang. 2024. Kolmogorov-Arnold Networks for Time Series: Bridging Predictive Power and Interpretability. *arXiv preprint arXiv:2406.02496* (2024).

[23] G. P. Zhang. 2001. An Investigation of Neural Networks for Linear Time-series Forecasting. *Computers & Operations Research* 28, 12 (2001), 1183–1202.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009