Time Series Forecasting with Hahn Kolmogorov-Arnold Networks

Md Zahidul Hasan **A. Ben Hamza** **Nizar Bouguila**
Concordia Institute for Information Systems Engineering
Concordia University, Montreal, QC, Canada

Abstract

Recent Transformer- and MLP-based models have demonstrated strong performance in long-term time series forecasting, yet Transformers remain limited by their quadratic complexity and permutation-equivariant attention, while MLPs exhibit spectral bias. We propose HaKAN, a versatile model based on Kolmogorov-Arnold Networks (KANs), leveraging Hahn polynomial-based learnable activation functions and providing a lightweight and interpretable alternative for multivariate time series forecasting. Our model integrates channel independence, patching, a stack of Hahn-KAN blocks with residual connections, and a bottleneck structure comprised of two fully connected layers. The Hahn-KAN block consists of inter- and intra-patch KAN layers to effectively capture both global and local temporal patterns. Extensive experiments on various forecasting benchmarks demonstrate that our model consistently outperforms recent state-of-the-art methods, with ablation studies validating the effectiveness of its core components.

1 INTRODUCTION

Time series forecasting is widely used as a critical tool in diverse domains ranging from retail, energy and transportation to healthcare and finance (Wang et al., 2025a; Zhang et al., 2025). However, this task poses significant challenges due to the need to effectively capture complex temporal patterns and long-range dependencies, while maintaining computational efficiency.

Recent advances in multivariate time series forecasting have explored Transformer- and MLP-based models to address these challenges. Transformer-based methods (Zhou et al., 2021; Wu et al., 2021; Zhou et al., 2022; Liu et al., 2024) rely on attention mechanisms to capture long-range dependencies, with simple strategies such as channel independence and patching (Nie et al., 2023) contributing to improved efficiency and predictive performance. However, Transformers often suffer from high computational complexity, quadratic in sequence length, and their permutation-equivariant attention also contradicts the causal nature of time series data. On the other hand, MLP-based methods (Zeng et al., 2023; Das et al., 2023) offer a computationally lighter alternative by using linear layers to model temporal patterns, often incorporating the channel independence strategy to capture channel-specific patterns. Despite their efficiency, MLPs exhibit spectral bias (Rahaman et al., 2019), which limits their ability to model high-frequency components in time series, and struggle with capturing non-linear temporal dynamics due to their reliance on linear transformations, leading to suboptimal performance on datasets, where non-linear patterns dominate. More recently, Kolmogorov-Arnold Networks (KANs) (Liu et al., 2025b; Wang et al., 2025b) have emerged as a viable alternative to MLPs, offering a promising solution to the aforementioned limitations by replacing fixed activation functions with learnable functions, parameterized using splines. Rooted in the Kolmogorov-Arnold representation theorem (Braun and Griebel, 2009; Schmidt-Hieber, 2021), KANs are interpretable and mitigate spectral bias by enabling flexible function approximation, allowing the model to capture both low- and high-frequency components in the data (Wang et al., 2025b). This adaptability makes KANs particularly well-suited for long-term forecasting, where diverse temporal patterns, ranging from short-term fluctuations to long-term trends, must be modeled accurately and efficiently.

Proposed Work and Contributions. We propose Hahn Kolmogorov-Arnold Network (HaKAN)[1],

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

[1]Code: `https://github.com/zadidhasan/HaKAN`

a novel framework for multivariate long-term time series forecasting, where each KAN layer is parameterized using Hahn Polynomials (Koekoek et al., 2010), enabling flexible and efficient function approximation. Unlike Transformer-based models, HaKAN avoids the computational overhead of attention mechanisms by using inter- and intra-patch KAN layers to model temporal relationships. Compared to MLP-based models, HaKAN employs learnable activation functions based on Hahn polynomials to capture non-linear temporal dynamics, overcoming the limitations of linear transformations. HaKAN also incorporates channel independence, patching, and a bottleneck structure to enhance robustness and efficiency, making it well-suited for diverse forecasting datasets and across various prediction horizons. The proposed framework combines the flexibility of KANs with a hierarchical patch-based design, enabling our model to capture both global and local temporal patterns while maintaining interpretability through learnable activation functions. The key contributions of this paper can be summarized as follows: (i) We introduce HaKAN, an effective framework for multivariate long-term time series forecasting that leverages the expressive power of KANs; (ii) we design a novel architecture featuring an Hahn-KAN block that integrates inter- and intra-patch KAN layers to effectively capture both global and local temporal patterns, respectively; and (iii) we demonstrate through extensive experiments that our model consistently outperforms strong baselines.

2 RELATED WORK

Transformer-based Models. A sizable body of research has focused on designing Transformer-based methods for long-term time series forecasting (Liu et al., 2021; Zhou et al., 2021; Wu et al., 2021; Zhou et al., 2022; Liu et al., 2024; Nie et al., 2023). For instance, Informer (Zhou et al., 2021) enhances Transformer efficiency with a ProbSparse self-attention mechanism, self-attention distilling, and a generative decoder. Autoformer (Wu et al., 2021) introduces a decomposition architecture with an auto-correlation mechanism that leverages series periodicity for dependency discovery and representation aggregation. FEDformer (Zhou et al., 2022) integrates seasonal-trend decomposition with Fourier and Wavelet transforms to capture global time series characteristics, while iTransformer (Liu et al., 2024) inverts the traditional Transformer architecture by embedding entire time series of individual variates as tokens, using attention to capture multivariate correlations and feed-forward networks to learn series representations. PatchTST (Nie et al., 2023) segments time series into subseries-level patches as input tokens and employs channel-independence. The channel-independence strategy improves robustness and adaptability by enabling distinct attention paths for each channel, in contrast to channel-mixing methods. Our HaKAN framework also adopts this channel-independent approach to preserve the unique temporal dynamics of each variable of the multivariate time series. Despite the success of Transformer-based methods in time series forecasting, their self-attention mechanism is, however, permutation-equivariant, meaning that it does not naturally preserve the temporal order, potentially compromising the modeling of time-dependent information.

MLP- and KAN-based Models. Various MLP-based models have been adopted for long-term time series forecasting (Chen et al., 2023; Challu et al., 2023; Wang et al., 2024; Zeng et al., 2023) due to their architectural and computational efficiency. For instance, TSMixer (Chen et al., 2023) captures temporal patterns and cross-variate information by interleaving time-mixing and feature-mixing MLPs., while DLinear (Zeng et al., 2023) enhances long-term time series forecasting by decomposing input data into trend and seasonal components. While MLP-based models offer greater structural simplicity and faster computation compared to Transformer-based models, they often struggle to capture global temporal dependencies and typically require longer input sequences to match the performance of more expressive architectures. More recently, TimeKAN (Huang et al., 2025) introduces a KAN-based architecture that decomposes multivariate time series into multiple frequency bands using cascaded frequency decomposition and moving averages. Similarly, TsKAN (Chen et al., 2025) presents a KAN-based approach that incorporates a multi-scale patching module to extract temporal and cross-dimensional features across scales. Our proposed HaKAN framework differs from these KAN-based models by using inter-patch KAN layers to capture global dependencies, overcoming MLPs' reliance on long input sequences, and from Transformer-based models by using an efficient Mixer-like structure that leverages KAN layers with Hahn polynomials for flexible function approximation. Its advantages include effective modeling of global and local temporal patterns, mitigation of spectral bias, and computational efficiency.

3 METHOD

3.1 Problem Description and Preliminaries

Problem Statement. Time series forecasting refers to the process of predicting future values over a period of time using historical data. Let $\mathbf{X}_{1:L} = (\mathbf{x}_1, \cdots, \mathbf{x}_L)^\intercal \in \mathbb{R}^{L \times M}$ be a history sequence of L mul-

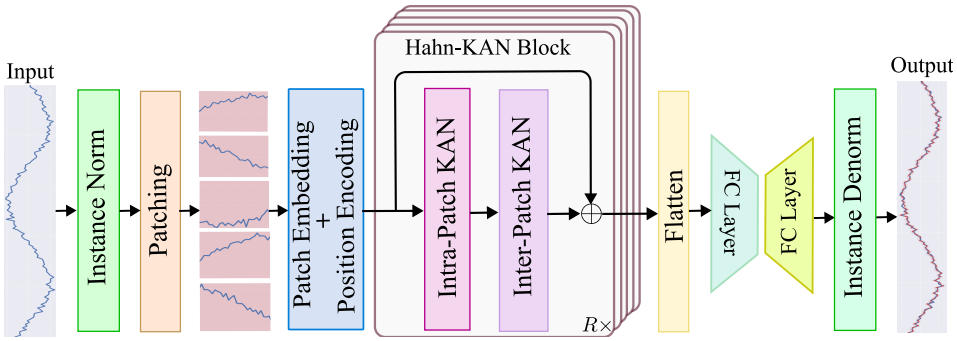

Figure 1: **HaKAN Architecture**. The model integrates channel independence, reversible instance normalization, and patching, followed by patch and position embeddings. A stack of R Hahn-KAN blocks, each with intra-patch and inter-patch KAN layers using Hahn polynomials, processes the embedded sequence to capture temporal patterns. The output is mapped through a bottleneck structure with two fully connected layers to produce the final forecast.

tivariate time series, where for any time step t, each row $\mathbf{x}_t = (x_{t1}, \ldots, x_{tM}) \in \mathbb{R}^{1\times M}$ is a multivariate vector consisting of M variables or channels. The goal of multivariate time series forecasting is to predict a sequence $\hat{\mathbf{X}}_{L+1:L+T} = (\hat{\mathbf{x}}_{L+1}, \ldots, \hat{\mathbf{x}}_{L+T})^\intercal \in \mathbb{R}^{T\times M}$ for the future T timesteps.

Kolmogorov-Arnold Networks. KANs are inspired by the Kolmogorov-Arnold representation theorem (Braun and Griebel, 2009; Schmidt-Hieber, 2021), which states that any continuous multivariate function on a bounded domain can be represented as a finite composition of continuous univariate functions of the input variables and the binary operation of addition. A KAN layer, a fundamental building block of KANs (Liu et al., 2025b), is defined as a matrix of 1D functions $\mathbf{\Phi} = (\phi_{q,p})$, where each trainable activation function $\phi_{q,p}$ is defined as a weighted combination, with learnable weights, of a sigmoid linear unit function and a spline function. Given an input vector $\mathbf{x}$, the output of an L-layer KAN is given by

$$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}^{(\mathsf{L}-1)} \circ \cdots \circ \mathbf{\Phi}^{(1)} \circ \mathbf{\Phi}^{(0)})\mathbf{x}, \qquad (1)$$

where $\mathbf{\Phi}^{(\ell)}$ is a matrix of learnable functions associated with the ℓ-th KAN layer.

3.2 Proposed HaKAN Framework

The proposed HaKAN model processes a multivariate time series $\mathbf{X}_{1:L} \in \mathbb{R}^{L\times M}$ to predict the future sequence $\hat{\mathbf{X}}_{L+1:L+T} \in \mathbb{R}^{T\times M}$. As illustrated in Figure 1, the model architecture consists of the following key components:

Channel Independence. Channel independence (CI) is a strategy that treats each feature or variable in a multivariate time series separately (Nie et al., 2023). Instead of combining information across channels, this strategy preserves the unique characteristics of each variable by maintaining their independence. Specifically, the input time series $\mathbf{X}_{1:L} = (\mathbf{x}_1, \ldots, \mathbf{x}_L)^\intercal$ is split into M univariate series $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_L^{(i)})^\intercal \in \mathbb{R}^L$, where $\mathbf{x}^{(i)}$ is the ith column of $\mathbf{X}_{1:L}$. Each of these univariate series is fed into the model backbone. Our HaKAN model takes $\mathbf{x}^{(i)}$ as input and returns a T-dimensional vector of predictions $\hat{\mathbf{x}}^{(i)} = (\hat{x}_{L+1}^{(i)}, \ldots, \hat{x}_{L+T}^{(i)})^\intercal$.

Normalization. Each input series is normalized using the reversible instance normalization (RevIN) technique (Kim et al., 2022), which addresses challenges related to shifts in data distributions over time. RevIN consists of two main steps: normalization and denormalization. In the first step, the input undergoes normalization to standardize its distribution in terms of mean and variance. After the model generates output sequences, RevIN reverses the normalization process by denormalizing these outputs.

Patching. Each normalized univariate series is partitioned into a sequence of patches to improve computational efficiency and capture local temporal patterns (Nie et al., 2023). The series is divided into patches $\mathbf{X}_p^{(i)} \in \mathbb{R}^{N\times P}$, where P is the patch length, $N = \lfloor \frac{L-P}{S} \rfloor + 2$ is the number of patches, and S is the stride of the sliding window. Patches are generated by sliding a window of size P over the series with stride S. If the last patch has fewer than P time steps, the final time step of the normalized univariate series is repeated to pad the patch. Patching offers several advantages, including improved retention of local semantic information, enhanced computational and memory efficiency, and access to a broader historical context.

Patch and Position Embeddings. Each patch in $\mathbf{X}_p^{(i)} \in \mathbb{R}^{N\times P}$ is projected into a D-dimensional em-

bedding using a temporal linear projection with a trainable weight matrix $\mathbf{W}_p \in \mathbb{R}^{P\times D}$. To retain the temporal order of the patches, which is critical for time series forecasting, a learnable positional embedding matrix $\mathbf{W}_{\text{pos}} \in \mathbb{R}^{N\times D}$ is added:

$$\mathbf{X}_d^{(i)} = \mathbf{X}_p^{(i)}\mathbf{W}_p + \mathbf{W}_{\text{pos}}, \tag{2}$$

where $\mathbf{X}_d^{(i)} \in \mathbb{R}^{N\times D}$ is the embedded sequence for the i-th channel. Each row of $\mathbf{X}_d^{(i)}$, referred to as a temporal patch-level token, represents the embedded features of a single patch from the i-th channel, maintaining the channel independence of the CI strategy. The positional embeddings ensure the model captures the sequential nature of the patches, addressing the causal structure of time series data. The embedded sequence serves as input to the Hahn-KAN block.

Hahn-KAN Block. The core component of our model architecture is the Hahn-KAN block, which processes the embedded sequence $\mathbf{X}_d^{(i)} \in \mathbb{R}^{N\times D}$ to capture both global and local temporal patterns. Each block consists of two KAN layers with Hahn Polynomials, structured with a residual connection:

$$\mathbf{X}_k^{(i)} = \text{KAN}(\text{KAN}(\mathbf{X}_d^{(i)})^\intercal)^\intercal + \mathbf{X}_d^{(i)}, \tag{3}$$

where $\mathbf{X}_k^{(i)} \in \mathbb{R}^{N\times D}$ is the output of the block, and each KAN$(\cdot)$ operation corresponds to a single KAN layer with univariate functions parameterized by Hahn polynomials. Specifically, each trainable univariate function $\phi_{q,p}$ of the KAN layer is parameterized using Hahn polynomials (Koekoek et al., 2010) to provide flexibility in function approximation:

$$\phi_{q,p}(x_p) = \sum_{r=0}^{d} \gamma_{q,p,r} P_r(x_p), \tag{4}$$

where x_p represents the p-th element of the KAN input vector, and $\gamma_{q,p,r}$ is the learnable coefficient of the r-th Hahn polynomial $P_r(x_p)$ for the q-th output element. The r-th Hahn polynomial $P_r(x) = \text{Hahn}(a, b, n)$, with parameters a, b and n, is defined by the recurrence relation

$$AP_r(x) = (A + B - x)P_{r-1}(x) - BP_{r-2}(x), \tag{5}$$

with coefficients:

$$A = \frac{(r+a+b)(r+a)(n-r+1)}{(2r+a+b-1)(2r+a+b)}, \tag{6}$$

$$B = \frac{(r-1)(r+b-1)(r+a+b+n)}{(2r+a+b-2)(2r+a+b-1)}, \tag{7}$$

and initial conditions $P_0(x) = 1$, $P_1(x) = 1 - \frac{a+b+2}{(a+1)n}x$.

The Hahn-KAN block consists of two nested layers: an intra-patch KAN layer (feature-mixing) and an inter-patch KAN layer (patch-mixing), both parameterized by Hahn polynomials. The inter-patch layer focuses on cross-patch relationships to capture global temporal patterns across the entire look-back window, such as patterns spanning the look-back window timesteps, while the intra-patch layer refines the features by focusing on local patterns within each patch. The latter captures fine-grained patterns within each patch, such as sudden changes in a short time window. The residual connection ensures training stability by allowing the Hahn-KAN block to learn incremental updates to the input.

The use of Hahn Polynomials in both intra-KAN and inter-KAN layers enhances the model's ability to approximate complex temporal functions, mitigating the spectral bias of traditional MLPs and providing interpretability through learnable activation functions. To capture hierarchical temporal patterns, the Hahn-KAN block is repeated R times in a stack, with each block taking the output of the previous block as its input, starting with the embedded sequence $\mathbf{X}_d^{(i)}$. The output of the r-th block, $\mathbf{X}_{k,r}^{(i)} \in \mathbb{R}^{N\times D}$, becomes the input to the $(r+1)$-th block. After R blocks, the final output $\mathbf{X}_k^{(i)} \in \mathbb{R}^{N\times D}$ is flattened into a feature vector $\mathbf{x}_f^{(i)} \in \mathbb{R}^{ND}$, where ND is the total feature dimension. This stacking mechanism enables the model to iteratively refine the features, capturing patterns at multiple temporal scales, from short-term fluctuations to long-term trends.

Why KAN with Hahn Polynomials? In a standard KAN layer with d_{in}-dimensional inputs and d_{out}-dimensional outputs, a B-spline of order d and grid size G is used as a learnable activation function. Unlike standard KANs, our proposed Hahn polynomial-based KANs offer superior computation and parameter efficiency. First, Hahn polynomials eliminate the need for grid discretization, removing the dependency on grid size G, a key factor in the complexity of standard KANs. Second, while standard KANs incur a time complexity of $\mathcal{O}(d_{\text{in}}d_{\text{out}}[9d(G+1.5d) + 2G - 2.5d + 3])$ (Yang and Wang, 2025), our Hahn KANs achieve a simplified complexity of $\mathcal{O}(d_{\text{in}}d_{\text{out}}d)$, where d is the Hahn polynomial degree (typically $d = 3$). This is comparable to the $\mathcal{O}(d_{\text{in}}d_{\text{out}})$ complexity of MLPs. Third, Hahn KANs require only $(d_{\text{in}}d_{\text{out}}(d+1))$ parameters, significantly fewer than the $(d_{\text{in}}d_{\text{out}}(G+d+3) + d_{\text{out}})$ parameters of standard KANs (Yang and Wang, 2025). This efficient design, coupled with polynomial-time evaluation and full parallelizability, makes our proposed HaKAN model a lightweight framework for time series forecasting.

Output Layer with Bottleneck Structure. The flattened vector $\mathbf{x}_f^{(i)} \in \mathbb{R}^{ND}$ is passed through an output layer consisting of two fully connected layers that form a bottleneck structure, mapping the features to the prediction horizon T. The first layer is a down-projection, which reduces the dimensionality of the feature vector to a bottleneck middle dimension H, using a weight matrix $\mathbf{W}_{\text{down}} \in \mathbb{R}^{H \times ND}$:

$$\mathbf{h}^{(i)} = \mathbf{W}_{\text{down}} \mathbf{x}_f^{(i)}, \tag{8}$$

where $\mathbf{h}^{(i)} \in \mathbb{R}^H$. This compression reduces both the risk of overfitting and the computational cost of the output layer.

The second layer is an up-projection, which expands the compressed features to the prediction horizon T, using a weight matrix $\mathbf{W}_{\text{up}} \in \mathbb{R}^{T \times H}$:

$$\hat{\mathbf{x}}^{(i)} = \mathbf{W}_{\text{up}} \mathbf{h}^{(i)}, \tag{9}$$

where $\hat{\mathbf{x}}^{(i)} \in \mathbb{R}^T$ is the forecasted sequence for the i-th channel.

The bottleneck structure ensures efficient mapping to the prediction horizon, especially for large T, by first compressing the features before expanding them. The forecasted sequences for all M channels are combined to form the final output $\hat{\mathbf{X}}_{L+1:L+T} \in \mathbb{R}^{T \times M}$. Finally, RevIN denormalization is applied to $\hat{\mathbf{x}}^{(i)}$ for each channel, using the stored mean and standard deviation, to restore the original data scale.

Algorithm 1 HaKAN: Time series forecasting

Require: Input multivariate time series $\mathbf{X}_{1:L} \in \mathbb{R}^{L \times M}$ with look-back L and M channels; forecast horizon T
Ensure: Forecasted sequence $\hat{\mathbf{X}}_{L+1:L+T} \in \mathbb{R}^{T \times M}$
for $i = 1$ to M **do** ▷ *Channel independence*
  Using RevIN, normalize the channel univariate series $\mathbf{x}^{(i)} = (\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_L^{(i)})^\intercal \in \mathbb{R}^L$
  Partition the normalized channel univariate series into N patches of size P to generate $\mathbf{X}_p^{(i)} \in \mathbb{R}^{N \times P}$
  Embed patches: $\mathbf{X}_d^{(i)} = \mathbf{X}_p^{(i)} \mathbf{W}_p + \mathbf{W}_{\text{pos}} \in \mathbb{R}^{N \times D}$
  Initialize $\mathbf{X}_k^{(i)} = \mathbf{X}_d^{(i)}$
  for $r = 1$ to R **do** ▷ *Hahn-KAN blocks*
    $\mathbf{X}_k^{(i)} = \text{KAN}(\text{KAN}(\mathbf{X}_k^{(i)})^\intercal)^\intercal + \mathbf{X}_k^{(i)}$
  end for
  Flatten $\mathbf{X}_k^{(i)} \in \mathbb{R}^{N \times D} \to \mathbf{x}_f^{(i)} \in \mathbb{R}^{ND}$
  Bottleneck mapping:
    $\mathbf{h}^{(i)} = \mathbf{W}_{\text{down}} \mathbf{x}_f^{(i)} \in \mathbb{R}^H$
    $\hat{\mathbf{x}}^{(i)} = \mathbf{W}_{\text{up}} \mathbf{h}^{(i)} \in \mathbb{R}^T$
  Denormalize $\hat{\mathbf{x}}^{(i)}$ via RevIN
end for
Combine the channels: $\hat{\mathbf{X}}_{L+1:L+T} = (\hat{\mathbf{x}}^{(1)}, \ldots, \hat{\mathbf{x}}^{(M)})$

3.3 Model Training

The parameters of our HaKAN model are learned by minimizing the following training objective function

$$\mathcal{L} = \frac{1}{MT} \sum_{i=1}^{M} \sum_{\tau=L+1}^{L+T} \left\| \mathbf{x}_\tau^{(i)} - \hat{\mathbf{x}}_\tau^{(i)} \right\|^2, \tag{10}$$

where $\mathbf{x}_\tau^{(i)}$ and $\hat{\mathbf{x}}_\tau^{(i)}$ are the ground-truth and prediction, respectively, $\tau \in \{L+1, \ldots, L+T\}$, L is the look-back window, T is the prediction horizon, and M is the number of time series variables.

The main algorithmic steps of the proposed HaKAN framework are summarized in Algorithm 1.

4 EXPERIMENTS

4.1 Experimental Setup

Datasets. We evaluate HaKAN on several benchmark datasets: Weather, Electricity, Illness, and four ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2) (Wu et al., 2021). **Weather** records 21 meteorological indicators every 10 minutes throughout 2020. **Traffic** comprises hourly road occupancy data from sensors across San Francisco Bay area freeways. **Electricity** tracks hourly electricity usage for 321 customers from 2012 to 2014. **ETT** includes transformer load and oil temperature data, sampled hourly for ETTh datasets and every 15 minutes for ETTm datasets, spanning July 2016 to July 2018. **Illness** contains weekly records of patient counts and influenza-like illness ratios.

Baselines and Evaluation Metrics. We evaluate the performance of our model against various recent state-of-the-art methods, including S-Mamba (Wang et al., 2025c), TimeKAN (Huang et al., 2025), Timer-XL (Liu et al., 2025a), TsKAN (Chen et al., 2025), iTransformer (Liu et al., 2024), PatchTST (Nie et al., 2023), TimesNet (Wu et al., 2023), Crossformer (Zhang and Yan, 2023), DLinear and RLinear (Zeng et al., 2023), N-HiTS (Challu et al., 2023), TiDE (Das et al., 2023), MICN (Wang et al., 2023), and FEDformer (Zhou et al., 2022). PatchTST includes 2 variants, PatchTST/42 and PatchTST/64, with the latter being the best performing model. Performance is evaluated using mean squared error (MSE) and mean absolute error (MAE).

Implementation Details. All experiments are conducted on a linux machine with a single NVIDIA RTX 4090 GPU 24GB. The HaKAN model is implemented in PyTorch, and Adam (Kingma and Ba, 2015) is used as optimizer. For the KAN layers, we use Hahn polynomials of the form $\text{Hahn}(a, b, n)$, where $a = 1$, $b = 1$, and $n = 7$, with the polynomial degree fixed at $d = 3$. The number of Hahn-KAN blocks is set to $R = 5$, and

Table 1: Time series forecasting results across prediction lengths $T \in \{24, 36, 48, 60\}$ for the Illness dataset and $T \in \{96, 192, 336, 720\}$ for the other datasets. The best results are highlighted in **bold**, and the second-best are underlined. For each method, multiple look-backs $L \in \{96, 192, 336, 720\}$ are evaluated, with the best-performing look-back reported. The dash (-) indicates no reported results in the baselines' papers.

Method		**HaKAN (ours)**		TsKAN (2025)		Timer-XL (2025a)		TimeKAN (2025)		PatchTST/64 (2023)		N-HiTS (2023)		DLinear (2023)		MICN (2023)		TimesNet (2023)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.369	0.394	0.376	0.395	**0.364**	0.397	0.367	0.395	0.379	0.401	0.378	0.436	0.375	0.399	0.413	0.442	0.421	0.440
	192	**0.406**	**0.414**	0.419	0.426	0.405	0.424	0.414	0.420	0.413	0.429	0.427	0.436	0.405	0.420	0.451	0.462	0.511	0.498
	336	**0.402**	**0.421**	0.449	0.450	0.427	0.439	0.445	0.434	0.435	0.436	0.458	0.484	0.439	0.443	0.556	0.528	0.484	0.478
	720	**0.443**	**0.459**	0.464	0.475	0.439	0.459	0.444	0.459	0.446	0.464	0.472	0.551	0.472	0.490	0.658	0.607	0.554	0.527
	Avg.	**0.405**	**0.422**	0.427	0.436	0.409	0.430	0.417	0.427	0.418	0.432	0.434	0.477	0.423	0.438	0.519	0.510	0.492	0.486
ETTh2	96	**0.260**	**0.328**	0.282	0.342	0.277	0.343	0.290	0.340	0.274	0.337	0.274	0.345	0.289	0.353	0.303	0.364	0.366	0.417
	192	**0.319**	**0.373**	0.361	0.391	0.348	0.391	0.375	0.392	0.332	0.380	0.353	0.401	0.383	0.418	0.403	0.446	0.426	0.447
	336	**0.318**	**0.380**	0.407	0.427	0.375	0.418	0.423	0.435	0.363	0.397	0.382	0.425	0.448	0.465	0.603	0.550	0.406	0.435
	720	0.394	0.432	0.415	0.448	0.409	0.458	0.443	0.449	**0.393**	**0.430**	0.625	0.557	0.605	0.551	1.106	0.852	0.427	0.457
	Avg.	**0.323**	**0.378**	0.366	0.402	0.352	0.402	0.383	0.404	0.341	0.386	0.408	0.432	0.431	0.447	0.604	0.553	0.406	0.439
ETTm1	96	0.289	0.345	0.310	0.356	**0.290**	**0.341**	0.322	0.361	0.293	0.346	0.302	0.350	0.299	0.343	0.308	0.360	0.356	0.385
	192	0.329	0.370	0.350	0.378	**0.337**	**0.369**	0.357	0.383	0.333	0.370	0.347	0.383	0.335	0.365	0.343	0.384	0.452	0.428
	336	**0.360**	**0.391**	0.368	0.394	0.374	0.392	0.382	0.401	0.369	0.392	0.369	0.402	0.369	0.386	0.395	0.411	0.419	0.425
	720	0.418	**0.416**	0.433	0.440	0.437	0.428	0.445	0.435	**0.416**	0.420	0.431	0.441	0.425	0.421	0.427	0.434	0.452	0.451
	Avg.	**0.349**	**0.380**	0.365	0.392	0.359	0.382	0.377	0.395	0.353	0.382	0.362	0.394	0.357	0.379	0.368	0.397	0.420	0.422
ETTm2	96	**0.166**	**0.255**	0.173	0.262	0.175	0.257	0.174	0.255	**0.166**	0.256	0.176	**0.255**	0.167	0.260	0.169	0.268	0.188	0.276
	192	**0.222**	**0.293**	0.231	0.305	0.242	0.301	0.239	0.299	0.223	0.296	0.245	0.305	0.224	0.303	0.247	0.333	0.242	0.310
	336	**0.265**	**0.323**	0.294	0.339	0.293	0.337	0.301	0.340	0.274	0.326	0.295	0.346	0.281	0.342	0.290	0.351	0.300	0.346
	720	**0.346**	**0.375**	0.392	0.398	0.376	0.390	0.395	0.396	0.362	0.385	0.401	0.413	0.397	0.421	0.417	0.434	0.391	0.403
	Avg.	**0.250**	**0.311**	0.272	0.326	0.271	0.321	0.277	0.323	0.256	0.316	0.279	0.330	0.267	0.332	0.281	0.346	0.280	0.334
Weather	96	0.148	0.198	**0.143**	0.205	0.157	0.205	0.162	0.208	0.149	0.198	0.158	**0.195**	0.176	0.237	0.178	0.249	0.163	0.219
	192	**0.190**	**0.240**	0.201	0.264	0.207	0.249	0.194	0.241	0.211	0.247	0.220	0.282	0.243	0.269	0.211	0.259	0.275	0.329
	336	0.242	0.282	0.256	0.301	0.259	0.291	**0.206**	**0.250**	0.263	0.290	0.245	**0.282**	0.274	0.300	0.265	0.319	0.278	0.338
	720	0.317	**0.333**	0.326	0.347	0.337	0.344	0.338	0.340	**0.314**	0.334	0.401	0.413	0.323	0.362	0.320	0.360	0.359	0.363
	Avg.	**0.224**	0.263	0.231	0.279	0.240	0.272	0.225	**0.260**	0.234	0.267	0.256	0.293	0.254	0.292	0.243	0.297	0.269	0.312
Traffic	96	0.365	0.252	-	-	**0.340**	**0.238**	-	-	0.360	0.249	0.402	0.282	0.410	0.282	0.473	0.293	0.595	0.318
	192	0.391	0.262	-	-	**0.360**	**0.247**	-	-	0.379	0.256	0.420	0.297	0.423	0.287	0.483	0.298	0.615	0.326
	336	0.407	0.272	-	-	**0.377**	**0.256**	-	-	0.392	0.264	0.448	0.313	0.436	0.296	0.491	0.303	0.616	0.326
	720	0.447	0.291	-	-	**0.418**	**0.279**	-	-	0.432	0.286	0.539	0.353	0.466	0.315	0.559	0.327	0.655	0.353
	Avg.	0.403	0.269	-	-	**0.374**	**0.255**	-	-	0.391	0.264	0.452	0.311	0.434	0.295	0.502	0.305	0.620	0.331
Electricity	96	**0.128**	**0.222**	-	-	-	-	0.174	0.266	0.129	**0.222**	0.147	0.249	0.140	0.237	0.157	0.266	0.178	0.284
	192	**0.146**	**0.240**	-	-	-	-	0.182	0.273	0.147	**0.240**	0.167	0.269	0.153	0.249	0.175	0.287	0.187	0.289
	336	**0.162**	**0.256**	-	-	-	-	0.197	0.286	0.163	0.259	0.186	0.290	0.169	0.267	0.200	0.308	0.208	0.307
	720	0.202	0.292	-	-	-	-	0.236	0.320	**0.197**	**0.290**	0.243	0.340	0.203	0.301	0.228	0.338	0.245	0.321
	Avg.	0.160	**0.253**	-	-	-	-	0.197	0.286	**0.159**	**0.253**	0.186	0.287	0.166	0.264	0.190	0.300	0.204	0.300
Illness	24	**1.183**	**0.685**	-	-	-	-	-	-	1.319	0.754	1.862	0.869	2.215	1.081	2.345	1.043	2.157	0.978
	36	**1.261**	**0.746**	-	-	-	-	-	-	1.579	0.870	2.071	0.934	1.963	0.963	2.330	1.001	2.318	1.031
	48	**1.406**	0.818	-	-	-	-	-	-	1.553	**0.815**	2.134	0.932	2.130	1.024	2.386	1.051	2.121	1.005
	60	1.540	0.851	-	-	-	-	-	-	**1.470**	**0.788**	2.137	0.968	2.368	1.096	2.616	1.131	1.975	0.975
	Avg.	**1.347**	**0.775**	-	-	-	-	-	-	1.480	0.807	2.051	0.926	2.169	1.041	2.419	1.056	2.143	0.997

the bottleneck dimension is set to $H = 336$. We set a patch length of $P = 16$, a stride of $S = 8$, and a patch embedding dimension to $D = 128$. We follow the standard data partitioning protocols (Nie et al., 2023). Specifically, for the ETT datasets, we use the first 12 months of data for training, the subsequent 4 months for validation, and the final 4 months for testing. This split ensures that if the model fails to generalize to months 13-16, it is unlikely to improve for months 17-20. For the remaining datasets, we adopt a split consisting of 70% training, 10% validation, and 20% testing. HaKAN is trained for up to 100 epochs, with early stopping and patience 10. The learning rate is set to 0.0025 for the Illness dataset, and to 0.0001 for all other datasets.

4.2 Results and Analysis

Optimized Look-back Window. To ensure a fair comparison, each baseline is run with look-back windows $L \in \{96, 192, 336, 720\}$, and the best-performing look-back is chosen to avoid underestimating their performance. For the proposed HaKAN model, we similarly evaluate across the same look-back windows and find that the best results are achieved with $L = 336$, which aligns with the optimal look-backs selected for PatchTST (Nie et al., 2023) and DLinear (Zeng et al., 2023), ensuring consistency in the comparison. All

Table 2: Long-term time series forecasting results for various prediction lengths $T \in \{96, 192, 336, 720\}$. The look-back is set to 96.

Method		**HaKAN (ours)**		S-Mamba (2025c)		iTransformer (2024)		RLinear (2023)		PatchTST/64 (2023)		Crossformer (2023)		TiDE (2023)		TimesNet (2023)		FEDformer (2022)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.383	**0.395**	0.386	0.405	0.386	0.405	0.386	**0.395**	0.414	0.419	0.423	0.448	0.479	0.464	0.384	0.402	**0.376**	0.419
	192	0.434	**0.421**	0.443	0.437	0.441	0.436	0.437	0.424	0.460	0.445	0.471	0.474	0.525	0.492	0.436	0.429	**0.420**	0.448
	336	0.473	**0.439**	0.489	0.468	0.487	0.458	0.479	0.446	0.501	0.466	0.570	0.546	0.565	0.515	0.491	0.469	**0.459**	0.465
	720	**0.469**	**0.461**	0.502	0.489	0.503	0.491	0.481	0.470	0.500	0.488	0.653	0.621	0.594	0.558	0.521	0.500	0.506	0.507
	Avg.	**0.439**	**0.429**	0.455	0.450	0.454	0.447	0.446	0.434	0.469	0.454	0.529	0.522	0.541	0.507	0.458	0.450	0.440	0.460
ETTh2	96	**0.277**	**0.332**	0.296	0.348	0.297	0.349	0.288	0.338	0.302	0.348	0.745	0.584	0.400	0.440	0.340	0.374	0.358	0.397
	192	**0.358**	**0.384**	0.376	0.396	0.380	0.400	0.374	0.390	0.388	0.400	0.877	0.656	0.528	0.509	0.402	0.414	0.429	0.439
	336	**0.342**	**0.382**	0.424	0.431	0.428	0.432	0.415	0.426	0.426	0.433	1.043	0.731	0.643	0.571	0.452	0.452	0.496	0.487
	720	**0.416**	**0.436**	0.426	0.444	0.427	0.445	0.420	0.440	0.431	0.446	1.104	0.763	0.874	0.679	0.462	0.468	0.463	0.474
	Avg.	**0.348**	**0.383**	0.381	0.405	0.383	0.407	0.374	0.398	0.387	0.407	0.942	0.684	0.611	0.550	0.414	0.427	0.437	0.449
ETTm1	96	**0.328**	0.368	0.333	0.368	0.334	0.368	0.355	0.376	0.329	**0.367**	0.404	0.426	0.364	0.387	0.338	0.375	0.379	0.419
	192	**0.365**	**0.385**	0.376	0.390	0.377	0.391	0.391	0.392	0.367	**0.385**	0.450	0.451	0.398	0.404	0.374	0.387	0.426	0.441
	336	**0.388**	**0.404**	0.408	0.413	0.426	0.420	0.424	0.415	0.399	0.410	0.532	0.515	0.428	0.425	0.410	0.411	0.445	0.459
	720	0.457	0.442	0.475	0.448	0.491	0.459	0.487	0.450	**0.454**	**0.439**	0.666	0.589	0.487	0.461	0.478	0.450	0.543	0.490
	Avg.	**0.384**	**0.399**	0.398	0.405	0.407	0.410	0.414	0.407	0.387	0.400	0.513	0.496	0.419	0.419	0.400	0.406	0.448	0.452
ETTm2	96	0.176	0.260	0.179	0.263	0.180	0.264	0.182	0.265	**0.175**	**0.259**	0.287	0.366	0.207	0.305	0.187	0.267	0.203	0.287
	192	**0.240**	**0.301**	0.250	0.309	0.250	0.309	0.246	0.304	0.241	0.302	0.414	0.492	0.290	0.364	0.249	0.309	0.269	0.328
	336	**0.299**	**0.339**	0.312	0.349	0.311	0.348	0.307	0.342	0.305	0.343	0.597	0.542	0.377	0.422	0.321	0.351	0.325	0.366
	720	**0.392**	**0.394**	0.411	0.406	0.412	0.407	0.407	0.398	0.402	0.400	1.730	1.042	0.558	0.524	0.408	0.403	0.421	0.415
	Avg.	**0.276**	**0.324**	0.288	0.332	0.288	0.332	0.286	0.327	0.281	0.326	0.757	0.610	0.358	0.404	0.291	0.333	0.305	0.349

models are evaluated on the Weather, Traffic, Electricity, ETTh1, ETTh2, ETTm1, ETTm2, and Illness datasets for prediction lengths $T \in \{96, 192, 336, 720\}$, using MSE and MAE as evaluation metrics. As reported in Table 1, HaKAN consistently outperforms most of the baselines, achieving the best MSE in 18 out of 32 cases and the best MAE in 19 out of 32 cases, with notable relative average MSE and MAE reductions of 8.98% and 3.96% on Illness. It excels particularly on datasets with smooth trends like ETT, for instance, achieving a relative average MSE and MAE error reductions of 5.28% and 2.07% on ETTh2.

Fixed Look-back Window. A number of baselines, such as S-Mamba (Wang et al., 2025c) and iTransformer (Liu et al., 2024), report the MSE and MAE values for a fixed look-back window of $L = 96$. We also compare our HaKAN model with recent baselines using a fixed look-back $L = 96$. As reported in Table 2, HaKAN achieves the best average MSE and MAE across prediction lengths $T \in \{96, 192, 336, 720\}$ on five benchmarks (ETTm1, ETTm2, ETTh1, ETTh2), outperforming strong baselines with notable relative error reductions, though PatchTST and Crossformer remain competitive at shorter horizons. On ETTm1, HaKAN's average MSE/MAE (0.384/0.399) yield relative error reductions of 7.2%/2.0% over RLinear, leading at $T = 96, 192, 336$, while PatchTST slightly outperforms at $T = 720$. On ETTm2, HaKAN achieves average MSE/MAE of 0.276/0.324 with relative error reductions of 3.5%/1.2%, excelling at $T = 192, 336, 720$, though PatchTST leads at $T = 96$. On ETTh1, HaKAN's average MSE/MAE (0.439/0.429) achieve relative error reductions of 1.6%/1.2% over RLinear, leading at $T = 720$ despite FEDformer's advantage at early horizons. On ETTh2, HaKAN dominates with average MSE/MAE (0.348/0.383), offering relative error reductions of 6.9%/3.8%, leading across all prediction lengths.

4.3 Ablation Study

In the ablation experiments, we consider six datasets $\mathcal{D} = \{$ETTh1, ETTh2, ETTm1, ETTm2, Weather, Illness$\}$ and four prediction horizons $\mathcal{T} = \{96, 192, 336, 720\}$ for the first five datasets and $\mathcal{T} = \{24, 36, 48, 60\}$ for Illness. Given a look-back window L, we define the average MSE and MAE over all datasets and across all prediction horizons as follows:

$$\overline{\text{MSE}} = \frac{1}{|\mathcal{T}||\mathcal{D}|} \sum_{\delta \in \mathcal{D}} \sum_{T \in \mathcal{T}} \text{MSE}(\mathbf{X}^{\delta}_{L+1:L+T}, \hat{\mathbf{X}}^{\delta}_{L+1:L+T}), \tag{11}$$

$$\overline{\text{MAE}} = \frac{1}{|\mathcal{T}||\mathcal{D}|} \sum_{\delta \in \mathcal{D}} \sum_{T \in \mathcal{T}} \text{MSE}(\mathbf{X}^{\delta}_{L+1:L+T}, \hat{\mathbf{X}}^{\delta}_{L+1:L+T}), \tag{12}$$

where $\mathbf{X}^{\delta}_{L+1:L+T}$ and $\hat{\mathbf{X}}^{\delta}_{L+1:L+T}$ are the ground-truth and predicted sequences, respectively, for the dataset $\delta \in \mathcal{D}$.

Polynomial Basis. We conduct an ablation study to assess how different polynomial bases (Koekoek et al., 2010) affect the performance of HaKAN, with results summarized in Table 3. The findings show that the choice of basis functions significantly impacts forecasting performance, with Hahn outperforming alter-

Table 3: Impact of basis.

Basis	$\overline{\text{MSE}}$	$\overline{\text{MAE}}$	Avg.
Hahn	**0.507**	**0.431**	**0.469**
Lucas	0.531	0.435	0.482
Chebyshev	0.539	0.439	0.488
B-Splines	0.548	0.443	0.495

Table 4: Impact of number of blocks.

#Blocks	$\overline{\text{MSE}}$	$\overline{\text{MAE}}$	Params (K)
1	0.526	0.436	**635**
3	0.534	0.438	767
5	**0.507**	**0.431**	899
20	0.549	0.442	1891

Table 5: Impact of hidden dimension.

H	$\overline{\text{MSE}}$	$\overline{\text{MAE}}$	Params (K)
200	0.536	0.438	695
336	**0.507**	**0.431**	899
800	0.523	0.435	1598
1000	0.543	0.440	1899

natives across all metrics.

Number of Hahn-KAN Blocks. Table 4 demonstrates a clear trade-off between model performance and parameter efficiency (measured in thousands of learnable parameters) across $R \in \{1, 3, 5, 20\}$. The configuration with $R = 5$ provides the best balance, achieving the lowest errors.

Bottleneck Dimension. The bottleneck dimension controls the number of parameters introduced by the down- and up-projection layers in the bottleneck structure of HaKAN. Table 5 summarizes the results, which indicate that a bottleneck dimension of 336 provides the best balance between model size and predictive performance.

HaKAN vs. MLP-Based Variant. Figure 2 provides a comparative analysis of the forecasting performance of HaKAN against its MLP-based counterpart, where each KAN layer in the HaKAN block is replaced with a fully connected layer. The comparison is conducted across five datasets, with the look-back window fixed at $L = 96$, and the average MSE over prediction horizons $T \in \{96, 192, 336, 720\}$ is used as the evaluation metric. The figure shows that HaKAN consistently outperforms the MLP-based variant across all datasets, achieving the lowest average MSE. KAN layers parameterized with Hahn polynomials offer expressive, learnable activation functions for modeling complex temporal dynamics, making our model an efficient alternative to MLP-based models for long-term time series forecasting.

Intra-Patch and Inter-Patch. Table 6 evaluates the individual and combined contributions of the intra-patch and inter-patch KAN layers, which form the cornerstone of our network architecture, using six ablation datasets with a fixed look-back window of $L = 96$ timesteps. This analysis highlights how each component influences the model's ability to capture local and global temporal patterns, respectively. The complete model, integrating both intra-patch and inter-patch layers, achieves the best overall performance, with lowest average MSE and MAE errors, demonstrating the synergistic effect of these components. Notably, removing the intra-patch KAN layer results in the most substantial performance degradation, with errors rising to 0.559 (MSE) and 0.447 (MAE), highlighting its critical role in refining local feature representations within patches. Conversely, omitting the inter-patch KAN layer increases errors to 0.520 (MSE) and 0.435 (MAE), indicating its importance in modeling cross-patch relationships for global context, though the impact is less severe than the intra-patch removal. These findings emphasize the hierarchical design's effectiveness, where the intra-patch layer's focus on fine-grained patterns complements the inter-patch layer's broader temporal perspective, enhancing the model's forecasting accuracy across diverse datasets.

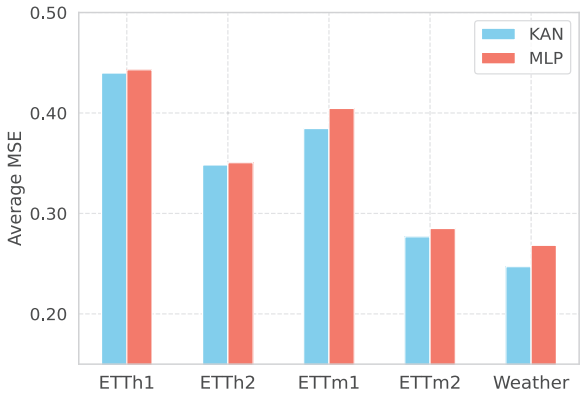

Figure 2: Performance comparison between HaKAN and its MLP-based variant across multiple datasets. The look-back window is fixed at $L = 96$, and the average MSE over prediction horizons $T \in \{96, 192, 336, 720\}$ is used as the evaluation metric.

Table 6: Effect of intra- and inter-patch KAN layers on model performance.

Component		Metric	
Intra-Patch	Inter-Patch	$\overline{\text{MSE}}$	$\overline{\text{MAE}}$
✗	✓	0.559	0.447
✓	✗	0.520	0.435
✓	✓	**0.507**	**0.431**

Patch Length. Figure 3 displays the average MSE and MAE errors for varying patch length $P \in \{4, 8, 16, 24, 32\}$ across all the six ablation datasets. In this experiment, the look-back window is fixed at $L = 96$ timesteps, and the stride S is dynamically set to $S = P/2$ to ensure overlapping patches that balance computational efficiency and temporal

coverage. The results, depicted in the figure, show that our model achieves the best performance with a patch length of $P = 16$, where both average MSE and MAE reach their lowest values, indicating the best trade-off between local pattern capture and global context preservation. This optimal setting suggests that $P = 16$ effectively segments the time series into patches that are sufficiently detailed to capture fine-grained temporal dynamics while maintaining enough overlap (stride $S = 8$) to support robust forecasting across the datasets.

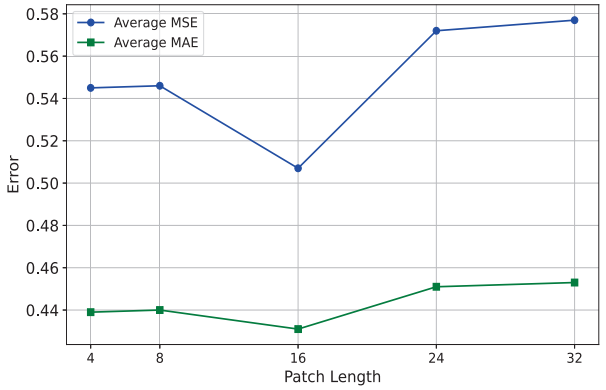

Figure 3: Average MSE and MAE results across the six ablation datasets for a varying patch length.

4.4 Computational Complexity Analysis

Time Complexity. For each channel, the temporal embedding requires $\mathcal{O}(NPD)$ operations. Each Hahn-KAN block comprises an intra-patch KAN layer with time complexity $\mathcal{O}(ND^2)$ and an inter-patch KAN layer with time complexity $\mathcal{O}(N^2D)$, yielding a total of $\mathcal{O}(R(N^2D + ND^2))$ for R blocks per channel. The bottleneck head adds $\mathcal{O}(NDH + HT)$ operations per channel. As channels are processed independently, the overall time complexity is $\mathcal{O}(M[R(N^2D+ND^2)+ NDH + HT])$. In contrast, a Transformer-based encoder incurs a time complexity of $\mathcal{O}(M[RL^2D + NDH + HT])$, dominated by $\mathcal{O}(L^2D)$ for the self-attention term, making HaKAN more efficient, especially for long sequences where $N \ll L$, due to patching and the compact Hahn polynomial representation.

Space Complexity. HaKAN stores parameters for each intra-patch KAN layer ($\mathcal{O}(D^2(d+1))$ per block) and inter-patch KAN layer ($\mathcal{O}(N^2(d+1))$ per block), totaling $\mathcal{O}(R(N^2(d+1) + D^2(d+1)))$ per channel, plus $\mathcal{O}(NDH + HT)$ for the bottleneck head, $\mathcal{O}(PD)$ for patch embedding, $\mathcal{O}(ND)$ for positional encoding, $\mathcal{O}(M)$ for RevIN, and $\mathcal{O}(MND)$ for activation memory. Thus, the simplified total space complexity is $\mathcal{O}(M[R(N^2 + D^2) + NDH + HT])$.

5 DISCUSSION

While our proposed HaKAN framework builds upon established concepts, its novelty lies in the integration of Hahn polynomial-based KAN layers within a hierarchical patch-based architecture. The primary motivation for Hahn polynomials is computational efficiency without sacrificing expressiveness. Unlike B-splines, which require grid discretization and incur complexity dependent on grid size, Hahn polynomials eliminate grid dependency, offer closed-form recurrence relations for fast evaluation. HaKAN introduces a *unique integration and optimization strategy* that differentiates it from prior works in three key aspects: (1) *Hahn Polynomial Parameterization:* Unlike TimeKAN (Huang et al., 2025) and TsKAN (Chen et al., 2025), which rely on frequency decomposition or spline-based activations, HaKAN leverages Hahn polynomials, an orthogonal basis on discrete domains, providing global approximation capability and parameter efficiency without grid discretization; (2) *Dual-Layer Hahn-KAN Block:* HaKAN introduces a novel hierarchical design with intra-patch and inter-patch KAN layers, enabling simultaneous modeling of fine-grained local patterns and global temporal dependencies; and (3) *Lightweight Complexity Profile:* HaKAN achieves near-MLP efficiency while retaining KAN's flexibility and and interpretability.

Despite its effectiveness, HaKAN assumes channel independence, which limits its performance on datasets with strong inter-variable correlations such as Traffic.

6 CONCLUSION

In this paper, we introduced HaKAN, a novel framework for multivariate time series forecasting that leverages Kolmogorov-Arnold Networks with Hahn polynomials, effectively capturing both local and global temporal patterns while maintaining computational efficiency. Comparative experiments on several benchmark datasets demonstrated that HaKAN consistently outperforms state-of-the-art baselines across various prediction horizons. This superior performance can be attributed to the KAN layers, which enable the model to approximate complex temporal functions more effectively than MLP- or Transformer-based architectures. Our ablation studies also confirmed the efficacy of key design choices, which collectively minimize forecasting error while balancing model complexity. For future work, we plan to explore the integration of HaKAN with frequency-domain techniques to further enhance its ability to model periodic patterns.

Acknowledgments. This work was supported in part by NSERC Discovery and FRQNT Team Grants.

Bibliography

Jürgen Braun and Michael Griebel. On a constructive proof of Kolmogorov's superposition theorem. *Constructive Approximation*, 30:653–675, 2009.

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. N-HiTS: Neural hierarchical interpolation for time series forecasting. In *Proc. AAAI Conference on Artificial Intelligence*, pages 6989–6997, 2023.

Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecasting. *Transactions on Machine Learning Research*, 2023.

Zechuan Chen, TianMing Sha, Ziyi Tang, and Keze Wang. TsKAN: A transparent architecture for improving the interpretability of multivariate time series forecasting. In *International Conference on Learning Representations Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2025.

Abhimanyu Das, Weihao Kong, Andrew Leach, Rajat Sen, and Rose Yu. Long-term forecasting with TiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023.

Songtao Huang, Zhen Zhao, Can Li, and LEI BAI. TimeKAN: KAN-based frequency decomposition learning architecture for long-term time series forecasting. In *International Conference on Learning Representations*, 2025.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Roelof Koekoek, Peter A. Lesky, and René F. Swarttouw. *Hypergeometric Orthogonal Polynomials and Their q-Analogues*. Springer, 2010.

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

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted Transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-XL: Long-context transformers for unified time series forecasting. In *International Conference on Learning Representations*, 2025a.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov-arnold networks. In *International Conference on Learning Representations*, 2025b.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: long-term forecasting with Transformers. In *International Conference on Learning Representations*, 2023.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proc. International Conference on Maching Learning*, 2019.

Johannes Schmidt-Hieber. The Kolmogorov-Arnold representation theorem revisited. *Neural Networks*, 137:119–126, 2021.

Dongjing Wang, Gangming Guo, Tianpei Ouyang, Dongjin Yu, Haiping Zhang, Bao Li, Rong Jiang, Guandong Xu, and Shuiguang Deng. A lightweight spatio-temporal neural network with sampling-based time series decomposition for traffic forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 2025a.

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. MICN: Multi-scale local and global context modeling for long-term series forecasting. In *International Conference on Learning Representations*, 2023.

Yixuan Wang, Jonathan W. Siegel, Ziming Liu, and Thomas Y. Hou. On the expressiveness and spectral bias of KANs. In *International Conference on Learning Representations*, 2025b.

Zihan Wang, Fanheng Kong, Shi Feng, Ming Wang, Xiaocui Yang, Han Zhao, Daling Wang, and Yifei Zhang. Is Mamba effective for time series forecasting? *Neurocomputing*, 619, 2025c.

Ziqing Wang, Shaopeng Ruan, Tianyu Huang, Hao Zhou, Shiji Zhang, Yifan Wang, Lin Wang, Zhen Huang, and Yang Liu. A lightweight multi-layer perceptron for efficient multivariate time series forecasting. *Knowledge-Based Systems*, 288, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition Transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, pages 22419–22430, 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2D-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Xingyi Yang and Xinchao Wang. Kolmogorov-Arnold Transformer. In *International Conference on Learning Representations*, 2025.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are Transformers effective for time series forecasting? In *Proc. AAAI conference on Artificial Intelligence*, pages 11121–11128, 2023.

Xu Zhang, Zhengang Huang, Yunzhi Wu, Xun Lu, Erpeng Qi, Yunkai Chen, Zhongya Xue, Qitong Wang, Peng Wang, and Wei Wang. Multi-period learning for financial time series forecasting. In *Proc. ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient Transformer for long sequence time-series forecasting. In *Proc. AAAI Conference on Artificial Intelligence*, pages 11106–11115, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed Transformer for long-term series forecasting. In *Proc. International Conference on Machine Learning*, pages 27268–27286, 2022.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Material

A DATASET DETAILS

Dataset statistics are summarized in Table 7.

Table 7: Summary statistics of benchmark datasets.

Dataset	Features	Timesteps	Frequency
Weather	21	52,696	10 min
Traffic	862	17,544	1 hour
Electricity	321	26,304	1 hour
Illness	7	966	1 week
ETTh1, ETTh2	7	17,420	1 hour
ETTm1, ETTm2	7	69,680	15 min

B MODEL ROBUSTNESS AGAINST RANDOM SEEDS

To evaluate the robustness of our model across different random seeds, we conducted experiments using the seeds $\{2021, 2022, 2023\}$ and calculated the average and standard deviation of MSE and MAE across three runs.

Table 8: Average ± std of forecasting results (3 seeds) per dataset and horizon.

ETTh1	MSE	MAE	**ETTh2**	MSE	MAE
96	0.3663 ± 0.0015	0.3917 ± 0.0012	96	0.2610 ± 0.0000	0.3277 ± 0.0006
192	0.4047 ± 0.0012	0.4097 ± 0.0049	192	0.3163 ± 0.0006	0.3673 ± 0.0006
336	0.4210 ± 0.0010	0.4243 ± 0.0012	336	0.3077 ± 0.0006	0.3697 ± 0.0006
720	0.4490 ± 0.0026	0.4620 ± 0.0026	720	0.3887 ± 0.0025	0.4277 ± 0.0015
ETTm1			**ETTm2**		
96	0.2903 ± 0.0032	0.3460 ± 0.0017	96	0.1670 ± 0.0000	0.2557 ± 0.0012
192	0.3287 ± 0.0015	0.3700 ± 0.0010	192	0.2230 ± 0.0017	0.2943 ± 0.0023
336	0.3587 ± 0.0012	0.3897 ± 0.0015	336	0.2773 ± 0.0015	0.3293 ± 0.0015
720	0.4207 ± 0.0031	0.4210 ± 0.0056	720	0.3757 ± 0.0090	0.3870 ± 0.0000
Weather			**Electricity**		
96	0.1477 ± 0.0006	0.1977 ± 0.0006	96	0.1280 ± 0.0000	0.2227 ± 0.0006
192	0.1897 ± 0.0006	0.2400 ± 0.0000	192	0.1460 ± 0.0000	0.2393 ± 0.0006
336	0.2420 ± 0.0000	0.2807 ± 0.0012	336	0.1620 ± 0.0000	0.2560 ± 0.0000
720	0.3173 ± 0.0015	0.3330 ± 0.0000	720	0.2027 ± 0.0006	0.2920 ± 0.0000

C HYPERPARAMETERS

For both look-backs $L = 336$ and $L = 96$, the number of Hahn-KAN blocks and maximum degree of Hahn polynomials are set to 3. For Hahn polynomial basis $\text{Hahn}(a, b, n)$, we set $a = 1, b = 1, n = 7$.

Table 9: Hyperparameter configurations for each dataset. All experiments used a fixed look-back $L = 336$ (except for Illness: $L = 104$).

Dataset	D	Patch Length	Stride	Batch Size	Learning Rate	Training Epochs
Electricity	128	16	8	32	1e-4	100
ETTh1	128	16	8	256	1e-4	100
ETTh2	128	16	8	256	1e-4	50
ETTm1	128	16	8	1024	1e-4	100
ETTm2	128	16	8	1024	1e-4	100
Illness	16	24	2	64	2.5e-3	100
Traffic	128	16	8	6	1e-4	100
Weather	128	16	8	256	1e-4	100

Table 10: HaKAN hyperparameter configurations per dataset. Default look-back is $L = 96$.

Dataset	D	Patch Length	Stride	Batch Size	Learning Rate	Training Epochs
ETTh1	128	16	8	128	1e-4	100
ETTh2	128	16	8	512	1e-4	50
ETTm1	128	16	8	700	1e-4	100
ETTm2	128	16	8	128	1e-4	100

D ADDITIONAL ABLATION STUDIES

Effect of Look-back Window. The look-back window L plays an important role in the HP-KAN model's ability to capture temporal dependencies for long-term time series forecasting, having a direct impact on the number of model parameters due to the use of fixed patch size P and stride S. As L increases, the number of patches $N = \lfloor \frac{L-P}{S} \rfloor + 2$ also grows, resulting in a larger input sequence. Figure 4 illustrates the effect of varying look-back window lengths on the long-term forecasting performance of HP-KAN, with the average MSE across prediction horizons $T \in \{96, 192, 336, 720\}$ for each dataset. The figure reveals that performance consistently improves as L increases from 48 to 336, with the lowest average MSE achieved at $L = 336$, reflecting the benefit of increased temporal context.

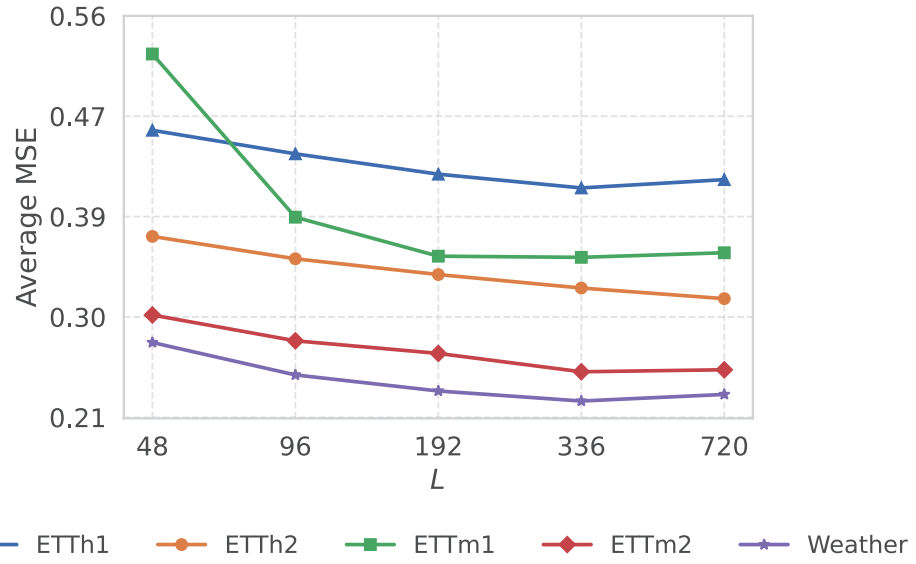

Figure 4: Evaluation of long-term forecasting performance across different look-back window lengths on multiple datasets, using the average MSE over prediction horizons $T \in \{96, 192, 336, 720\}$ as the evaluation metric.