Kolmogorov-Arnold Networks for 3D Human Motion and Time Series Prediction

Md Zahidul Hasan

A Thesis

in

The Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montreal, QC, Canada

December 2025

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Md Zahidul Hasan

Entitled: Kolmogorov-Arnold Networks for 3D Human Motion and Time Series Prediction

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Amr Youssef

_____ Examiner
Dr. Amr Youssef

_____ Examiner
Dr. Ali Ayub

_____ Thesis Supervisor(s)
Dr. Abdessamad Ben Hamza

_____ Thesis Supervisor(s)
Dr. Nizar Bouguila

Approved by _____
Dr. Chun Wang Chair of Department or Graduate Program Director

_____
Dean of Faculty

Abstract

Kolmogorov-Arnold Networks for 3D Human Motion and Time Series Prediction

Md Zahidul Hasan

Accurate modeling and forecasting of complex sequential data, spanning diverse applications from 3D human motion prediction to multivariate time series forecasting, remains a fundamental challenge in computer vision and machine learning. Existing approaches, including Transformer- and MLP-based models, have demonstrated strong empirical performance but often struggle with balancing prediction accuracy, computational complexity, and model interpretability. Transformers are constrained by quadratic complexity and positional encoding limitations, whereas MLPs suffer from spectral bias, hindering their ability to capture fine-grained temporal dynamics essential for high-fidelity predictions. This thesis addresses these limitations by leveraging the expressive power of Kolmogorov-Arnold Networks (KANs), which utilize polynomial-based learnable activation functions to enhance model efficiency and interpretability. The thesis contributions are twofold: The first contribution introduces LuKAN, an effective and computationally efficient model designed for 3D human motion prediction. LuKAN utilizes the Discrete Wavelet Transform to encode temporal information, and at its core, employs a Temporal Dependency Learner built on a KAN layer parameterized by Lucas polynomial activations. The second contribution introduces HaKAN, a versatile and lightweight framework for long-term multivariate time series forecasting. HaKAN is composed of a stack of Hahn-KAN blocks, where fixed activations are replaced with Hahn polynomial-based learnable functions to enhance both interpretability and adaptability, which not only mitigates spectral bias but also enables the model to effectively capture both global and local temporal patterns. HaKAN integrates channel independence, patching, a bottleneck structure, and residual connections, ensuring both strong generalization and a minimal architectural footprint. Extensive experiments on benchmark datasets for 3D motion prediction and long-term time series forecasting tasks demonstrate that both LuKAN and HaKAN consistently outperform competing state-of-the-art methods in terms of prediction accuracy and computational cost. Notably, their polynomial-based KAN architecture offers a unique advantage in interpretability and efficiency, as validated through comprehensive quantitative and ablation studies.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3D Three-Dimensional

3DPW 3D Poses in the Wild

AI Artificial Intelligence

AMASS Archive of Motion Capture as Surface Shapes

B-Spline Basis Spline

CD Channel Dependence

CI Channel Independence

CNN Convolutional Neural Network

DCT Discrete Cosine Transform

DFT Discrete Cosine Transform

DWT Discrete Wavelet Transform

EEG Electroencephalogram

ETT Electricity Transformer Temperature

GB Gigabytes

GCN Graph Convolutional Network

GNN Graph Neural Network

GPU Graphics Processing Unit

IDWT Inverse Discrete Wavelet Transform

KAN Kolmogorov-Arnold Network

KAT Kolmogorov Arnold Transformer

LN Layer Normalization

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MLP Multi-Layer Perceptron

MPJPE Mean Per Joint Position Error

MSE Mean Squared Error

PDE Partial Differential Equation

ResNet Residual Network

RevIN Reversible Instance Normalization

RNN Recurrent Neural Network

RTX Ray Tracing Texel eXtreme

Seq2Seq Sequence-to-sequence

SiLU Sigmoid Linear Unit

SMPL Skinned Multi-Person Linear

TCN Temporal Convolutional Network

ViT Vision Transformer

CHAPTER 1

Introduction

This chapter lays the foundation on spatiotemporal sequence modeling using Kolmogorov-Arnold Networks (KANs). It motivates the work by exposing the spectral bias plaguing Transformers, graph convolutional networks, and multi-layer perceptrons, which systematically underfit high-frequency details critical for human motion and fast-varying time series. The core problem is formalized as sequence-to-sequence prediction, with detailed treatment of two tasks: 3D human motion prediction and multivariate time series forecasting. Key objectives are to (1) build a unified, interpretable KAN framework that mitigates spectral bias via learnable orthogonal polynomials, (2) achieve improved performance with far fewer parameters, and (3) ensure model training stability through reversible instance normalization, residuals, and patching. Essential preliminaries cover the discrete cosine transform, discrete wavelet transform, normalization techniques, channel-independence vs. dependence, and the Kolmogorov-Arnold theorem. A concise literature review highlights limitations of existing KAN variants and forecasting architectures.

1.1 Background and Motivation

Learning effective representations of spatiotemporal data has been one of the most enduring challenges in modern machine learning and signal processing. From forecasting weather patterns and modeling financial time series to predicting 3D human motion and articulations, many real-world tasks require models that can learn complex dependencies over both space and time. Despite the proliferation of deep neural networks in recent years, designing architectures that generalize well across temporal scales while remaining computationally efficient and interpretable continues to be

an open research problem.

Traditional neural networks such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) architectures have been extensively applied to time-dependent data due to their ability to capture sequential correlations. However, these models often suffer from vanishing gradients, limited receptive fields, and high computational costs for long sequences. More recent architectures, such as Temporal Convolutional Networks (TCNs) and Transformers, have advanced temporal modeling through parallel processing and attention mechanisms. Nevertheless, these models still face the fundamental issue of **spectral bias**: a tendency to prioritize low-frequency components of the signal while underrepresenting the high-frequency details that often carry critical transient information. This bias results in smooth but imprecise reconstructions, particularly problematic for high-frequency signals such as human motion trajectories or fast-varying sensor data. Graph Convolutional Networks (GCNs) can learn spatial dependencies well but they smooth out temporal information of each variable. MLP-Mixers also suffer from spectral bias. Therefore, we need a foundational deep neural network architecture that are conceptually as simple as MLPs but can be superior in mitigating spectral bias.

1.2 Problem Formulation

The central objective of this thesis is to model temporal dynamics in high-dimensional data, where the goal is to infer future states of a system from a limited window of past observations. Such problems often manifest as *sequence-to-sequence* learning tasks, where the input and output are both sequences that evolve over time. This section formalizes the general sequence-to-sequence framework and discusses two specific instances of it studied in this thesis: *3D human motion prediction* and *multivariate time series forecasting*. Both problems share common challenges such as long-term dependency modeling, inter-variable dependencies, nonstationary dynamics, and the need for interpretable representations.

1.2.1 Sequence-to-Sequence Learning

Sequence-to-sequence (Seq2Seq) learning refers to a family of problems in which a model is trained to map an input sequence to an output sequence that may differ in length, resolution, or semantics. Given an input sequence $\mathbf{X}_{1:L} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L)^\intercal$, where $\mathbf{x}_t \in \mathbb{R}^{1\times M}$ represents the observation at time t, the goal is to predict a future or transformed sequence $\hat{\mathbf{X}}_{L+1:L+T} = (\hat{\mathbf{x}}_{L+1}, \dots, \hat{\mathbf{x}}_{L+T})^\intercal$. The mapping can be expressed as a parametric function

$\hat{\mathbf{X}}_{L+1:L+T} = f_\theta(\mathbf{X}_{1:L})$, where f_θ denotes a model with parameters θ learned through data-driven optimization.

Seq2Seq learning captures temporal causality and contextual dependency, making it applicable to diverse domains including language translation, speech synthesis, video generation, and sensor data forecasting. In contrast to static regression, where inputs and outputs are independent samples, sequence learning models must reason over temporal correlations, memory of past states, and the propagation of uncertainty across time. Key challenges include:

- **Temporal dependency:** capturing both short- and long-term correlations within sequences without loss of contextual information.
- **Nonstationarity:** adapting to time-varying distributions where the underlying data-generating process evolves.
- **High dimensionality:** learning joint representations for many correlated variables or spatially organized features.
- **Stability and interpretability:** ensuring smooth, consistent, and explainable predictions across time horizons.

In the following subsections, we discuss two representative Seq2Seq tasks that embody these challenges: 3D human motion prediction and multivariate time series forecasting, each emphasizing a different aspect of spatiotemporal reasoning.

1.2.2 3D Human Motion Prediction

3D human motion prediction involves forecasting future human poses from a given history of observed movements. Each motion sequence is represented as a series of 3D joint coordinates $\mathbf{x}_t = (x_{t1}, \ldots, x_{tK})^\intercal$, where $K = 3J$ and J is the number of body joints in the kinematic skeleton. Given a sequence of L observed poses $\mathbf{X}_{1:L} = (\mathbf{x}_1, \ldots, \mathbf{x}_L)^\intercal$, the goal is to predict the future sequence $\hat{\mathbf{X}}_{L+1:L+T}$ that continues the motion naturally and realistically.

Challenges. The task is inherently high-dimensional and structured. Human motion exhibits strong spatial correlations between joints (e.g., the movement of the shoulder affects the motion of the arm and hand) and long-range temporal dependencies that encode style, rhythm, and intention. The model must therefore capture both instantaneous pose relationships and global motion patterns extending across many frames.

Two key difficulties arise:

1. **Spatial structure:** The human body is a kinematic tree with articulated connections between joints. Learning the geometry and constraints of this structure is crucial to avoid implausible poses or self-intersections.

2. **Temporal continuity:** Future poses must evolve smoothly in both space and time while reflecting the stochastic nature of real movement. Overly deterministic models often lead to motion freezing or convergence to mean poses.

1.2.3 Multivariate Time Series Forecasting

Multivariate time series forecasting seeks to predict future values of multiple correlated variables based on their past behavior.A multivariate time series can be represented as $\mathbf{X_{1:L}} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L)^\intercal \in \mathbb{R}^{L \times M}$, where each vector $\mathbf{x}_t = (x_{t1}, \ldots, x_{tM})^\intercal \in \mathbb{R}^{1 \times M}$ contains M variables recorded at time t. The forecasting task involves estimating $\hat{\mathbf{X}}_{L+1:L+T}$, which represents the expected evolution of the system over the next T steps.

Characteristics of Multivariate Temporal Data. In real-world settings, multivariate data often exhibit:

- **Cross-variable dependency:** Strong correlations or causal interactions among variables (e.g., temperature, humidity, and pressure in weather data, or stock prices in financial systems).

- **Nonstationarity:** The statistical properties of the series, such as mean, variance, and correlation, evolve over time.

- **Multi-scale behavior:** Temporal dynamics occur at different frequencies, from fast local oscillations to slow global trends.

Problem Setting. The goal is to construct a function f_θ that maps past observations to future predictions:

$$\hat{\mathbf{X}}_{L+1:L+T} = f_\theta(\mathbf{X}_{1:L}).$$

Rather than relying solely on autoregressive modeling, modern approaches learn latent representations that encode both temporal and inter-variable dependencies. These representations must generalize across varying sampling rates, variable correlations, and forecasting horizons.

1.2.4 Shared Challenges and Unified Perspective

Both 3D motion prediction and multivariate time series forecasting can be understood as structured sequence-to-sequence problems. Although their data modalities differ, geometric skeletons versus numeric variables, both domains involve:

- learning rich temporal dependencies across multiple time scales,
- preserving structure (spatial or inter-variable) in the learned representation, and
- generating smooth and consistent predictions over extended horizons.

The overarching challenge is to design architectures that effectively combine spatial reasoning, temporal modeling, and interpretability. This thesis investigates these challenges through the lens of functional representation learning, providing a unified framework that treats sequential prediction as learning continuous transformations across space, time, and variable dimensions.

1.3 Objectives

The primary goal of this thesis is to develop efficient, interpretable, and generalizable neural architectures for modeling structured spatiotemporal data through the lens of functional representation learning. Specifically, this research aims to bridge the gap between traditional multilayer perceptrons (MLPs) and functional approximators by leveraging the Kolmogorov-Arnold representation theorem, which expresses multivariate functions as compositions of univariate functions and summations. The proposed work investigates how this principle can be effectively applied to sequence modeling tasks to enhance interpretability, stability, and efficiency. The specific objectives of this thesis are as follows:

- **Develop a unified functional learning framework** based on Kolmogorov-Arnold Networks that can model both spatial and temporal dependencies in structured data and learn frequency domain representations for 3D human motion prediction. The framework should maintain high expressive power while being computationally lightweight.
- **Design an efficient forecasting architecture for multivariate time series** that can effectively capture long-term dependencies via a unified orthogonal polynomial formulation of KANs.

- **Incorporate normalization and residual mechanisms** (such as layer normalization, reversible instance normalization, and skip connections) to enhance model training stability and mitigate the vanishing gradient problem.
- **Evaluate the proposed models on benchmark datasets** from both computer vision and time series domains, and demonstrate improvements in prediction accuracy, interpretability, and computational efficiency over recent state-of-the-art baselines.

1.4 Preliminaries

This section presents the mathematical and architectural concepts underlying the proposed seq2seq learning frameworks.We first review key frequency and scale-domain transforms: (i) Discrete Cosine Transform (DCT) and (ii) Discrete Wavelet Transform (DWT) - which motivate our spectral decomposition approach [8]. We then describe normalization and residual strategies used in modern deep architectures, including Layer Normalization and Reversible Instance Normalization (RevIN). Then, we discuss patching techniques that bridge sequence modeling and vision architectures by partitioning continuous inputs into local, learnable segments. Finally, we discuss Kolmogorov-Arnold Network which lies at the foundation of our models.

1.4.1 Frequency and Scale Representations

Discrete Cosine Transform (DCT). The Discrete Cosine Transform (DCT) expresses a finite sequence of data points as a weighted sum of cosine basis functions oscillating at different frequencies. For a 1-D signal x of length N, the DCT-II formulation is most commonly used in deep learning and signal compression:

$$X[k] = \alpha_k \sum_{n=0}^{N-1} x[n] \cos\left[\frac{\pi(2n+1)k}{2N}\right], \quad \alpha_k = \begin{cases} \sqrt{\frac{1}{N}}, & k = 0, \\ \sqrt{\frac{2}{N}}, & 1 \le k \le N-1. \end{cases} \tag{1.1}$$

The inverse transform reconstructs the original sequence by summing over cosine modes:

$$x[n] = \sum_{k=0}^{N-1} \alpha_k X[k] \cos\left[\frac{\pi(2n+1)k}{2N}\right]. \tag{1.2}$$

DCT compacts signal energy into low-frequency coefficients, enabling efficient representation of smooth trends. In time series forecasting and motion modeling, DCT encodes long-term temporal dynamics using relatively few coefficients, allowing the network to focus on meaningful

slow-varying patterns. Unlike the Discrete Fourier Transform (DFT), the DCT avoids complex arithmetic and enforces even symmetry, which reduces redundancy and stabilizes optimization in real-valued neural networks.

Discrete Wavelet Transform (DWT). While DCT captures global periodic components, it lacks temporal localization.The Discrete Wavelet Transform (DWT) resolves this limitation by decomposing a signal into localized frequency bands through a cascade of filters.Given a discrete signal x, the one-level DWT produces an *approximation* (low-frequency) component $A[n]$ and a *detail* (high-frequency) component $D[n]$:

$$A[n] = \sum_k x[k]\, g[2n-k], \qquad D[n] = \sum_k x[k]\, h[2n-k], \tag{1.3}$$

where $g[\cdot]$ and $h[\cdot]$ are low-pass and high-pass wavelet filters, respectively. By recursively applying this operation to the approximation coefficients, the signal is decomposed into multi-resolution sub-bands capturing both global trends and local variations.

DWT offers an interpretable time–frequency trade-off: coarse levels capture slow temporal variations, while finer levels represent transient or abrupt changes.In neural architectures, DWT is used to provide hierarchical features for nonstationary signals such as motion sequences, weather data, or EEG recordings.The inverse DWT reconstructs the original signal via up-sampling and convolution with the synthesis filters.

1.4.2 Normalization and Residual Mechanisms

Layer Normalization. Normalization stabilizes model training by controlling the distribution of activations across layers. Layer Normalization (LayerNorm) [9] normalizes the activations of each sample across its feature dimension. Given an input vector $\mathbf{x} \in \mathbb{R}^d$, the normalized output $\hat{\mathbf{x}}$ is computed as:

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}}, \quad \mu = \frac{1}{d}\sum_{i=1}^{d} x_i, \quad \sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2. \tag{1.4}$$

A learnable affine transformation restores expressivity:

$$\mathrm{LayerNorm}(\mathbf{x}) = \gamma \odot \hat{\mathbf{x}} + \beta, \tag{1.5}$$

where γ and β are trainable scale and bias parameters. LayerNorm is invariant to batch size and input order, making it especially effective for Transformer-based and time series model architectures.

Residual and Skip Connections. Residual or skip connections were first popularized in ResNet architectures to mitigate vanishing gradients in deep networks. The key idea is to allow information

to bypass nonlinear transformations, facilitating stable gradient flow. Given an input $\mathbf{x}$ and a nonlinear function $\mathcal{F}(\cdot)$ (e.g., a sequence of layers), the residual output is:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}) + \mathbf{x}. \tag{1.6}$$

This additive identity mapping enables the network to learn perturbations of the input rather than entirely new mappings, simplifying optimization and improving convergence. In temporal models, residual paths ensure that baseline trends or slowly varying components are preserved across prediction horizons. Variants such as gated residuals or cross-block skip connections further control the degree of information blending.

Reversible Instance Normalization (RevIN). RevIN [10] was designed for nonstationary time series forecasting, where the input distribution may shift over time. Unlike standard normalization that irreversibly modifies input scale, RevIN normalizes each instance and later restores its original statistics. Given a time series $\mathbf{x} \in \mathbb{R}^L$ with mean μ and standard deviation σ, the normalization and recovery steps are:

$$\text{Normalization:} \quad \tilde{\mathbf{x}} = \frac{\mathbf{x} - \mu}{\sigma}, \tag{1.7}$$

$$\text{Recovery:} \quad \hat{\mathbf{y}} = \sigma \odot f_\theta(\tilde{\mathbf{x}}) + \mu. \tag{1.8}$$

During training, the model operates on the normalized data $\tilde{\mathbf{x}}$, while during inference the predicted output $f_\theta(\tilde{\mathbf{x}})$ is rescaled back to the original domain via the stored μ and σ. This reversibility prevents loss of absolute scale information, improving generalization to distribution shifts and unseen domains.

1.4.3 Patching Strategies in Vision and Time Series

Patch Embedding in Vision Models. Patching transforms high-dimensional structured inputs into compact tokens suitable for attention-based processing. In Vision Transformers (ViTs), an image $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$ is divided into N non-overlapping patches of size $P \times P$, flattened, and linearly projected:

$$\mathbf{z}_i = \mathbf{W}_p \, \mathrm{vec}(\mathbf{I}_{(i)}), \quad i = 1, \ldots, N, \tag{1.9}$$

where $\mathbf{W}_p \in \mathbb{R}^{(P^2C) \times D}$ projects each patch into a D-dimensional latent token. This operation converts spatial data into a sequence of embeddings, enabling Transformer layers to model spatial relationships through self-attention.

Temporal Patching for Time Series. The patching concept extends naturally to time-series forecasting [11]. Given an input sequence $\mathbf{x} = [x_1, x_2, \ldots, x_L]$, we segment it into N overlapping or

non-overlapping sub-sequences (patches) of length P:

$$\mathbf{p}_i = [x_{(i-1)S+1},\, x_{(i-1)S+2},\, \ldots,\, x_{(i-1)S+P}], \tag{1.10}$$

where S denotes the stride. Each patch is then linearly projected or processed by a small encoder to produce a patch embedding $\mathbf{z}_i \in \mathbb{R}^D$. These patch tokens capture local temporal patterns, while inter-patch dependencies are modeled by attention or KAN-based mixing layers. This hierarchical representation improves long-term forecasting performance and allows parallel computation across patches.

1.4.4 Channel-Dependence vs Channel-Independence

In multivariate time series forecasting, an essential design choice concerns *how* correlations among different variables (channels) are modeled. Given a temporal sequence $\mathbf{X}_{1:L} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L)^\intercal$, where each vector $\mathbf{x}_t \in \mathbb{R}^{1\times M}$ contains M variables measured at time t, one may choose to either model each variable (or channel) independently, or learn a joint representation across all channels. These two perspectives are known respectively as *channel-independent* and *channel-dependent* forecasting approaches. Understanding their conceptual differences and practical implications is critical for designing architectures that balance scalability, generalization, and interpretability.

Channel-Independent Forecasting. Channel-independent (CI) models treat each variable in a multivariate time series as an individual univariate sequence. The forecasting function is factorized across channels:

$$\hat{\mathbf{x}}^{(i)}_{L+1:L+T} = f^{(i)}_\theta\big(\mathbf{x}^{(i)}_{1:L}\big), \quad i = 1, \ldots, M, \tag{1.11}$$

where $f^{(i)}_\theta$ denotes a predictor dedicated to the i-th variable. Sometimes, the same predictor is used for all the variables. Each model captures temporal dependencies within a single channel but does not explicitly account for interactions among different variables.

Advantages:

- **Scalability:** Each model operates independently, making training parallelizable across channels and memory-efficient for large N.
- **Robustness to noise:** Uncorrelated noise or irregular sampling in one channel does not propagate to others.
- **Simplified optimization:** Independent models are less prone to overfitting inter-variable correlations, especially when training data per channel is abundant.

Limitations:

- **No cross-variable reasoning:** Interactions or causal influences among variables are ignored, which can severely limit performance when variables are correlated.
- **Parameter redundancy:** Each model must learn similar temporal dynamics separately, increasing total parameter count and reducing generalization.

Channel-independent modeling is commonly used in benchmark studies to provide a lower bound for model performance or to isolate the temporal learning ability of the architecture without the confounding effects of cross-channel dependencies. Many patch-based or convolutional architectures for long-term forecasting begin with channel-wise encoders for efficiency and modularity.

Channel-Dependent Forecasting. Channel-dependent (CD) models, in contrast, treat the entire multivariate sequence as a single high-dimensional input. They learn shared temporal representations that jointly encode correlations among channels:

$$\hat{\mathbf{X}}_{L+1:L+T} = f_\theta(\mathbf{X}_{1:L}), \tag{1.12}$$

where f_θ models both intra-channel (temporal) and inter-channel (spatial or cross-variable) relationships simultaneously.

Advantages:

- **Capturing cross-variable dependencies:** Joint modeling allows the network to exploit statistical and causal interactions among variables (e.g., how pressure affects temperature or one joint influences another).
- **Parameter sharing:** Shared parameters enable efficient representation learning and improve data efficiency when inter-variable correlations are strong.
- **Richer temporal context:** The model can disambiguate ambiguous local patterns by referencing global correlations across channels.

Limitations:

- **Increased complexity:** Modeling full joint dependencies scales quadratically with the number of variables in attention-based architectures, leading to high computational cost.
- **Overfitting risk:** Spurious correlations among channels may mislead the model, especially in noisy or heterogeneous datasets.

- **Reduced interpretability:** Learned joint embeddings are harder to interpret, as individual contributions of variables become entangled.

1.4.5 Theoretical Foundation of Kolmogorov-Arnold Networks

Kolmogorov-Arnold Networks (KANs) [12] are a recently proposed class of neural architectures inspired by the *Kolmogorov-Arnold representation theorem*. Unlike conventional multilayer perceptrons (MLPs) that rely on fixed, node-wise activation functions and linear weights, KANs learn *univariate functional transformations* along the edges of the network. This design replaces scalar weights with learnable continuous functions, leading to a highly expressive and interpretable model family that bridges function approximation theory and deep learning.

Theoretical Foundation. The Kolmogorov-Arnold representation theorem states that any continuous multivariate function $f : [0, 1]^n \to \mathbb{R}$ can be expressed as a finite superposition of univariate functions:

$$f(x_1, x_2, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \Psi_{q,p}(x_p) \right), \tag{1.13}$$

where Φ_q and $\Psi_{q,p}$ are continuous univariate functions. This remarkable result implies that the curse of dimensionality can, in principle, be mitigated through structured compositions of one-dimensional mappings. KANs operationalize this theorem by parameterizing each function $\Psi_{q,p}$ (edge function) and Φ_q (node aggregation) as trainable components within a neural network. Each edge between two neurons represents a learnable univariate transformation $\psi_{i,j}(\cdot)$, while the nodes aggregate incoming signals through summation. Formally, for a neuron j in layer $\ell + 1$, its output is given by:

$$h_j^{(\ell+1)} = \sum_{i=1}^{n_\ell} \psi_{i,j}^{(\ell)}(h_i^{(\ell)}), \tag{1.14}$$

where $\psi_{i,j}^{(\ell)}$ is a univariate function usually parameterized by smooth basis functions such as B-splines. This eliminates the need for activation functions as a means of non-linearity in neural networks.

1.5 Literature Review

1.5.1 Kolmogorov-Arnold Networks

In [12], the learnable univariate functions are defined as a weighted combination of the B-spline function and the SiLU function:

$$\phi(x) = w_b \,\mathrm{silu}(x) + w_s \,\mathrm{spline}(x), \qquad \mathrm{silu}(x) = \frac{x}{1+e^{-x}}, \quad \mathrm{spline}(x) = \sum_i c_i B_i(x). \quad (1.15)$$

This design yields highly interpretable, edge-wise nonlinear mappings that can be directly visualized, helping reveal how different input dimensions contribute to the learned transformation. Empirically, smaller KANs match or surpass MLPs in data-fitting and PDE-solving accuracy, while demonstrating faster neural scaling laws and better interpretability. However, the canonical KAN is computationally heavy because every edge stores its own set of spline parameters, which can become prohibitive for high-dimensional data. To improve efficiency, several studies introduced regularization and architectural optimizations. DropKAN [13] masks a random subset of post-activations within the computation graph, mitigating overfitting with negligible overhead. MatrixKAN [14] reformulates spline evaluations as batched matrix multiplications, achieving up to $40\times$ speedups for high-degree splines. Parameter-efficient versions such as PRKAN [15] and AF-KAN [16] introduce shared or mixed activations, while ReLU-KAN [17] approximates spline functions with ReLU-based surrogates for GPU-friendly implementation. ChebyKAN [18] use Chebyshev polynomials of the first kind defined on $[-1, 1]$, providing smoother gradients and faster convergence than B-spline KANs [12]. However, Chebyshev polynomials suffer from boundary oscillations and gradient amplification near ± 1, which can destabilize training on unbounded or discrete data. The authors also do not discuss a generalized framework for all discrete and continuous orthogonal polynomial families or their computational benefits.

Several KAN variants exploit frequency-domain or multi-resolution representations. KAF [19] integrates KAN layers with Fourier series or Random Fourier Features, directly encoding periodic components while retaining learnable nonlinear modulation. WavKAN [20] employs wavelet bases for localized time-frequency decomposition, offering a powerful tool for nonstationary signals.

KANs have also been extended beyond Euclidean data to graph and sequence domains. GraphKAN [21] introduces edge-wise functional weights within message-passing networks, enhancing nonlinear relation modeling between connected nodes. SigKAN [22] incorporates path-signature features to better encode sequential dynamics, while TKAN [23] and KAN4TSF [24] demonstrate the framework's scalability to long-horizon forecasting and anomaly detection tasks. These extensions highlight the flexibility of KANs as functional message-passing or sequence-mixing operators, unifying classical graph and time-series architectures under a continuous functional lens.

In machine learning, Physics-Informed KANs such as KINN [25] embed physical constraints within functional layers, allowing solutions of differential equations that respect boundary or conservation laws. CoxKAN [26] adapts the KAN framework to survival analysis, preserving interpretability while achieving competitive accuracy in biomedical applications. State-Space KAN (SS-KAN) [27] integrates KAN representations within state-space models for interpretable nonlinear system identification, whereas Deep Operator-Learning KANs (DeepOKAN) [28] generalize Fourier Neural Operators by learning functional kernels from data rather than fixing them in advance. Theoretical studies show that KANs exhibit universal approximation capability comparable to ReLU networks, but with lower spectral bias and smoother gradient propagation, making them favorable for physics-inspired learning. To enhance scalability, hybrid KANs integrate with existing deep backbones. Kolmogorov-Arnold Transformer (KAT) [29] replaces MLP blocks in Transformers with KAN layers, providing stronger expressivity and stability on large-scale vision and language tasks and proposed rational-polynomial-based KANs. Low-rank and shared-coefficient KANs compress basis expansions, reducing memory overhead while preserving functional richness. Together, these methods address the main practical bottlenecks: parameter growth, memory cost, and training instability without sacrificing interpretability. However, most of these KANs are not parallelizable, parameter efficient, computationally efficient and highly accurate all at the same time. Our proposed orthogonal polynomial-based formulation of KANs achieve all of that.

1.5.2 3D Human Motion Prediction

RNN-based Methods. RNNs were among the earliest models for human motion prediction due to their ability to capture temporal dependencies in sequential data [30–33]. However, they struggle with long-term dependencies and suffer from gradient instability, limiting their performance on complex motion sequences.

GCN- and MLP-based Methods. GCNs represent human poses as graphs with joints as nodes and bones as edges, effectively encoding spatial dependencies [34, 35]. Mao *et al.* [1] applied DCT to motion sequences and used learnable adjacency matrices to capture joint relations. Guo *et al.* [7] extended this idea using MLPs on DCT-transformed inputs, though fixed activations limit flexibility and DCT constrains locality. Feng *et al.* [36] proposed MotionWavelet, leveraging 2D wavelet transforms but with high computational cost due to its diffusion-based refinement.

Recent research has explored the frequency-domain and polynomial-based representations to capture hierarchical motion features with fewer parameters. Yet, most existing models rely on fixed activation functions or handcrafted bases, which restrict adaptability and increase computation for

fine-grained motion dynamics. This motivates exploring functional architectures capable of learning both temporal and spectral dependencies in a data-driven manner.

1.5.3 Multivariate Time Series Forecasting

Transformer-based Models. Transformer architectures have achieved strong results in long-term time series forecasting [37–42]. Informer [38] improves efficiency with ProbSparse attention and a generative decoder, while Autoformer [39] introduces decomposition and auto-correlation mechanisms to capture periodicity. FEDformer [40] leverages Fourier and Wavelet transforms for seasonal-trend modeling, and iTransformer [41] embeds each variate as a token to learn inter-series relations. PatchTST [42] segments inputs into patches and adopts channel independence, enabling separate attention paths for each variate. Despite their success, Transformer-based models remain computationally heavy, and their permutation-equivariant attention may also weaken temporal order modeling.

MLP-based Models. MLP architectures [43–47] offer a lightweight alternative, focusing on mixing strategies for efficiency. TSMixer [43] interleaves time- and feature-mixing layers, while Time Series MLP [45] captures short-term dependencies via a precurrent mechanism. FreTS [46] uses frequency-domain processing to learn global dependencies, and DLinear [47] models trend–seasonal components separately. Although MLP-based models provide faster training and simpler structures, they often lack capacity for long-range dependency modeling, motivating the need for architectures that balance global context with computational efficiency.

1.6 Research Challenges

The design of effective spatiotemporal models involves addressing several challenges:

1. **Spectral Bias Mitigation:** Most neural models tend to underfit high-frequency components. Both LuKAN and HaKAN are designed to counter this bias through polynomial-based learnable nonlinearities.

2. **Multi-Scale Temporal Learning:** Time-dependent processes exhibit dynamics at multiple scales. Integrating DWT-based multi-resolution representations with KAN's adaptive functional mappings allows efficient handling of both short-term and long-term dependencies.

3. **Spatial-Temporal Decoupling:** In motion prediction, spatial interactions (joint dependencies) must be modeled separately yet coherently with temporal evolution.

4. **Generalization and Efficiency:** Achieving high accuracy while maintaining parameter efficiency and interpretability remains an ongoing pursuit. The proposed orthogonal KANs seek to balance these trade-offs through compact functional representations.

1.7 Overview and Contributions

The remainder of the thesis is organized as follows:

- **Chapter 2** presents LuKAN, a Kolmogorov-Arnold Network framework for 3D human motion prediction. The model introduces Lucas polynomial activations and integrates Discrete Wavelet Transform encoding to capture localized temporal variations in motion. A spatial projection and temporal dependency learner jointly model spatial and temporal correlations in the human body. Extensive experiments on benchmark datasets such as Human3.6M, AMASS, and 3DPW demonstrate LuKAN's superior performance and computational efficiency compared to existing methods.

- **Chapter 3** introduces HaKAN, a Hahn polynomial-based KAN model for multivariate time series forecasting. It employs patching strategies, reversible instance normalization, and separate intra- and inter-patch KAN layers to model channel-independent temporal dependencies. HaKAN achieves state-of-the-art forecasting accuracy while maintaining parameter efficiency and robustness across diverse datasets. Comprehensive ablation studies and complexity analyses are conducted to validate the model's design choices.

- **Chapter 4** concludes the thesis by summarizing the key findings and highlighting the theoretical and empirical contributions of orthogonal polynomial-based KANs. It discusses their implications for spatiotemporal modeling, identifies current limitations, and outlines future research directions, including theoretical analysis, architectural extensions, and application-driven developments.

1.8 Summary of Publications

The contents of Chapter 2 consist of research ideas and results taken from:

> **Md Zahidul Hasan**, A. Ben Hamza, Nizar Bouguila. "LuKAN: A Kolmogorov-Arnold Network Framework for 3D Human Motion Prediction," *Proc. British Machine Vision Conference (BMVC)*, 2025.

The contents of Chapter 3 consist of research ideas and results taken from:

> **Md Zahidul Hasan**, A. Ben Hamza, Nizar Bouguila. "Time Series Forecasting with Hahn Kolmogorov-Arnold Networks ," *Submitted*, 2025.

CHAPTER 2

LuKAN: A Kolmogorov-Arnold Network Framework for 3D Human Motion Prediction

The goal of 3D human motion prediction is to forecast future 3D poses of the human body based on historical motion data. Existing methods often face limitations in achieving a balance between prediction accuracy and computational efficiency. In this chapter, we introduce LuKAN, an effective model based on Kolmogorov-Arnold Networks (KANs) with Lucas polynomial activations. Our model first applies the discrete wavelet transform to encode temporal information in the input motion sequence. Then, a spatial projection layer is used to capture inter-joint dependencies, ensuring structural consistency of the human body. At the core of LuKAN is the Temporal Dependency Learner, which employs a KAN layer parameterized by Lucas polynomials for efficient function approximation. These polynomials provide computational efficiency and an enhanced capability to handle oscillatory behaviors. Finally, the inverse discrete wavelet transform reconstructs motion sequences in the time domain, generating temporally coherent predictions. Extensive experiments on three benchmark datasets demonstrate the competitive performance of our model compared to strong baselines, as evidenced by both quantitative and qualitative evaluations. Moreover, its compact architecture coupled with the linear recurrence of Lucas polynomials, ensures computational efficiency. Code is available at: https://github.com/zadidhasan/LuKAN

2.1 Introduction

The task of 3D human motion prediction is to forecast the future 3D poses of a human body over a specified time horizon based on historical motion data. It empowers diverse applications requiring dynamic and responsive interaction with human movements, including human-object interaction [48], animation [49], and autonomous driving [50, 51]. In recent years, substantial progress has been made in 3D human motion prediction [1, 3, 4, 6, 7, 52–54], yet accurately forecasting future motion remains a major challenge due to the intrinsic complexity and variability of human movements. The spatio-temporal nature of human motion further compounds these challenges, requiring models to effectively capture both spatial inter-joint relationships and temporal dynamics across sequential frames.

State-of-the-art methods have embraced diverse neural network architectures tailored to the spatio-temporal nature of motion data, including Recurrent Neural Networks (RNNs) [30–32], Graph Convolutional Networks (GCNs) [1–4, 55], Transformers [2, 56, 57], and Multi-Layer Perceptrons (MLPs) [6, 7]. RNNs excel at modeling sequential dependencies but struggle with long-term sequences. GCN-based approaches capture spatial relationships through graph convolutions, but are prone to oversmoothing. Transformers, leveraging the self-attention mechanism, have quadratic computational complexity with respect to sequence length, requiring substantial computation for effective training. MLP-based models achieve reduced computational overhead, but use fixed activation functions and require deep architectures to model complex relationships. More recently, Kolmogorov-Arnold networks (KANs) have emerged as a compelling alternative to MLPs, demonstrating superior performance in function representation across various tasks, including regression [12], while mitigating spectral bias [58]. Unlike MLPs, KANs leverage learnable activation functions on the edges. Existing GCN- and MLP-based approaches employ the discrete cosine transform (DCT) to encode motion in the frequency domain [1, 7]. However, the reliance on DCT may limit their flexibility in capturing localized motion patterns. Moreover, most GCN- and Transformer-based models incorporate MLPs as their core components for feature learning, inheriting a fundamental drawback of MLPs, namely spectral bias [59].

Proposed Work and Contributions. In this chapter, we introduce LuKAN, a robust model for 3D human motion prediction based on KANs. It integrates a KAN layer that learns univariate functions parameterized by Lucas polynomials to capture interactions between temporal patterns across joints, and spatial projections that model inter-joint relationships. We summarize our contributions as follows: (1) We propose a novel architecture, leveraging KANs and the discrete wavelet transform to encode temporal information in the motion sequence by decomposing the trajectory of each body joint into low-frequency (coarse-scale) components and high-frequency components (fine-

scale). Wavelet functions excel at capturing transient and rapidly changing features in a signal, offering a significant advantage over DCT, particularly for motion data where localized variations and dynamic changes are crucial. For instance, rapid hand gestures (high-frequency components) can be captured at fine scales, while slower, more gradual movements like walking (low-frequency components) can be captured at coarser scales; (2) We design a Temporal Dependency Learner to model both localized motion variations and global trends in human motion; (3) We conduct extensive experiments on benchmark datasets, showing that LuKAN achieves competitive performance with minimal computational overhead.

2.2 Related Work

RNN-based Methods. RNNs have been extensively used in the early stages of human motion prediction research due to their ability to model temporal dependencies in sequential data [30–33]. These models excel at capturing temporal patterns, making them suitable for tasks where the sequence order and history play a vital role. However, RNN-based methods are often limited by their inability to effectively capture long-term dependencies and are prone to gradient instability during training, particularly for complex motion sequences.

GCN- and MLP-based Methods. GCNs represent human poses as graphs, with joints as nodes and bones as edges. This graph structure enables GCN-based methods to encode inter-joint dependencies naturally. Mao *et al.* [1] proposed a spatio-temporal network that applies DCT to input motion sequences to encode the temporal dynamics of joint coordinates in the trajectory space. The network uses GCNs with learnable adjacency matrices to capture spatial dependencies between body joints. Guo *et al.* [7] introduced an effective approach using MLPs on the spatial and temporal dimensions of the DCT-transformed input. However, relying on DCT may constrain the flexibility of these models in capturing localized motion patterns effectively. Moreover, MLPs use fixed activation functions at their nodes, limiting their flexibility to adapt to diverse data patterns. Feng *et al.* [36] introduced MotionWavelet, leveraging 2D wavelet transforms to model human motion patterns in the spatial-frequency domain. However, its reliance on diffusion models with guidance mechanisms to control prediction refinement results in higher computational cost. Our proposed LuKAN framework differs from existing methods in that it employs learnable 1D functions on its edges, allowing the network to adaptively model complex temporal dependencies in motion data. It also employs DWT to encode temporal dependencies in the joint trajectory, allowing the model to capture both coarse and fine-grained motion patterns. While both our model and MotionWavelet leverage wavelet transforms for human motion prediction, they differ significantly

in terms of their architectural design and learning methodology. Unlike MotionWavelet [36], which modifies motion signals repeatedly through the diffusion process, LuKAN retains high-frequency details. Moreover, using a KAN layer parameterized with Lucas polynomials provides flexibility and computational efficiency, as they are more efficient to evaluate than the piecewise construction of B-splines used in standard KANs.

2.3 Method

In this section, we first describe the task at hand. Next, we provide a preliminary background on KANs [12, 58]. Then, we introduce the key building blocks of our network architecture.

Problem Description. Let $\mathbf{X}_{1:L} = (\mathbf{x}_1, \ldots, \mathbf{x}_L)^\intercal \in \mathbb{R}^{L \times K}$ be a history motion sequence of L consecutive 3D human poses, where L is the look-back window, $K = 3J$ in the feature dimension, and J is the total number of body joints. At each time step t, each pose $\mathbf{x}_t \in \mathbb{R}^{1 \times K}$ is a flattened vector formed by concatenating the 3D coordinates of all joints in a single frame. The objective is to construct a predictive model that estimates a motion sequence $\hat{\mathbf{X}}_{L+1:L+T} = (\hat{\mathbf{x}}_{L+1}, \ldots, \hat{\mathbf{x}}_{L+T}) \in \mathbb{R}^{T \times K}$ for the subsequent T timesteps. To this end, we design an efficient model based on Kolmogorov-Arnold networks [12].

Kolmogorov-Arnold Networks. KANs are inspired by the Kolmogorov-Arnold representation theorem [60, 61], 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 is a fundamental building block of KANs [12], and 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 (SiLU) function and a spline function. Given an input vector $\mathbf{x}$, the output of an L-layer KAN is given by

$$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}^{(\mathsf{L}-1)} \circ \cdots \circ \mathbf{\Phi}^{(1)} \circ \mathbf{\Phi}^{(0)})\mathbf{x}, \tag{2.1}$$

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

Approach Overview. The proposed model architecture begins with applying DWT to the input 3D motion sequence. Unlike the discrete cosine transform, which creates a representation of the joint trajectory using cosine waves that oscillate indefinitely, wavelet functions are compact and designed such that their oscillations diminish over time, thereby allowing not only efficient access of localized information about the trajectory, but also capturing rapidly changing features in a trajectory, which DCT cannot address as efficiently. A spatial projection follows, explicitly modeling the spatial structure of the human body by analyzing inter-joint relationships. The core of the

architecture is the Temporal Dependency Learner, which consists of a single KAN layer parameterized by Lucas polynomials, a layer normalization, and a residual skip connection to effectively capture both local and global temporal patterns. To ensure accurate spatial representation, a second spatial projection is applied after the temporal learner. Then, the data is transformed back to the time domain using the inverse discrete wavelet transform, reconstructing a sequence of predicted 3D human poses.

Model Architecture. The overall framework of our network architecture is depicted in Figure 2.1. LuKAN is designed to efficiently predict 3D human motion by modeling both spatial relationships and temporal dependencies in motion data.

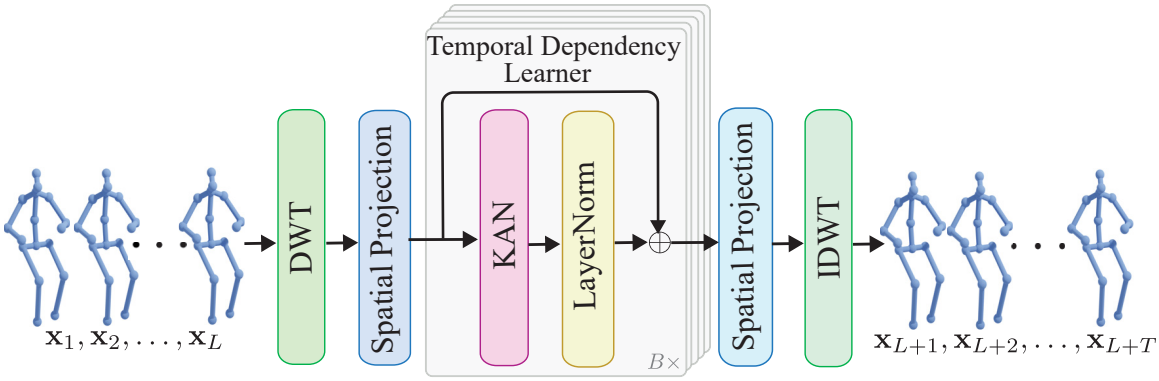

Figure 2.1: **Overview of Model Architecture**. LuKAN processes input 3D motion data by applying DWT to encode temporal information. A spatial projection is applied both before and after the Temporal Dependency Learner block (repeated B times). Each block consists of a KAN layer, LayerNorm, and a residual skip connection. The inverse DWT (IDWT) reconstructs the motion in the time domain, outputting a sequence of predicted 3D poses.

2.3.1 Temporal Encoding

Joint Trajectory. The ith column of the history motion sequence $\mathbf{X}_{1:L}$, denoted as $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_L^{(i)})^\intercal$, represents the trajectory of the i-th skeleton joint over the L consecutive frames in the sequence. The coordinates $x_\ell^{(i)}$ at each time step ℓ represent the position of the i-th joint in 3D space at that specific moment. This representation allows for capturing the motion of each joint individually over the observed time window.

Discrete Wavelet Transform Encoding. To encode temporal information of the human motion in the trajectory, we employ DWT, which decomposes a signal into its approximate and detail components using wavelets, ensuring that localized temporal variations in the motion sequence are captured effectively. Specifically, given a wavelet (e.g., Daubechies wavelet), applying a single-

level DWT to the i-th joint trajectory $\mathbf{x}^{(i)}$ yields

$$\mathbf{c}^{(i)} = \text{DWT}(\mathbf{x}^{(i)}), \tag{2.2}$$

where $\mathbf{c}^{(i)} = (\mathbf{a}^{(i)}, \mathbf{d}^{(i)})^\intercal$ is an $(L_a + L_d)$-dimensional vector of wavelet coefficients that describe the signal's approximation and detail components. The approximation coefficients $\mathbf{a}^{(i)} \in \mathbb{R}^{L_a}$ represent the low-frequency (coarse-scale) components of the trajectory, while the detail coefficients $\mathbf{d}^{(i)} \in \mathbb{R}^{L_d}$ represent the high-frequency (fine-scale) variations in the trajectory. Unlike cosine waves, which oscillate indefinitely, wavelet functions are compact, with oscillations that diminish over time, enabling them to localize effectively and capture transient or rapidly changing features in a trajectory, which DCT cannot address as efficiently. The original trajectory can be reconstructed from its wavelet coefficients using the Inverse Discrete Wavelet Transform (IDWT) as follows:

$$\hat{\mathbf{x}}^{(i)} = \text{IDWT}(\mathbf{a}^{(i)}, \mathbf{d}^{(i)}), \tag{2.3}$$

which takes as input an $(L_a + L_d)$-dimensional vector of wavelet coefficients and returns an L-dimensional reconstructed trajectory, ensuring that essential motion characteristics are preserved while enabling a more localized representation of human motion sequences.

2.3.2 Spatial Projection

The spatial projection maps the DWT-transformed history motion sequence into an embedding space of dimension D, capturing inter-joint dependencies and providing an expressive representation of the spatial structure of the human body. Its output is an $(L_a + L_d) \times D$ matrix given by

$$\mathbf{Z}_1 = \text{DWT}(\mathbf{X}_{1:L})\mathbf{W}_1 \tag{2.4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{K \times D}$ is a learnable weight matrix, which defines a linear projection along the spatial (i.e., joint) dimension, and D is the embedding dimension. For notational simplicity, the bias term is omitted here and throughout the following subsections.

2.3.3 Temporal Dependency Learner

The Temporal Dependency Learner is a core component of LuKAN, designed to capture temporal relationships within the motion sequence data. It operates as a sequence modeling block, emphasizing both local and global temporal dependencies to effectively predict future motion, while maintaining computational efficiency. This component consists of three key elements: a single KAN layer, LayerNorm, and a residual skip connection. The design choices are motivated as

follows: (1) unlike deep MLPs, a shallow KAN effectively captures dependencies with its Lucas polynomials as learnable activation functions, providing flexibility in modeling both localized variations (such as fast changes in pose) and global trends (like slow transitions in motion), while reducing the need for excessively deep architectures; (2) LayerNorm helps stabilize training and ensures feature consistency across different motion sequences; and (3) a residual skip connection enhances gradient flow and prevents information loss, mitigating the limitations of purely feedforward architectures.

KAN Layer. We employ a single KAN layer, with associated matrix $\mathbf{\Phi} = (\phi_{q,p})$ whose (q,p)-th entry is a function with learnable parameters. Each trainable function $\phi_{q,p}$ is parameterized by a weighted linear combination of Lucas polynomials

$$\phi_{q,p}(x_p) = \sum_{r=0}^{R} \gamma_{q,p,r} P_r(x_p), \tag{2.5}$$

where x_p represents the p-th element of the joint trajectory vector, and $\gamma_{q,p,r}$ is the learnable coefficient of the r-th Lucas polynomial $P_r(x_p)$ for the q-th output element. These learnable parameters are adjusted during training to optimize the network's performance with the aim of improving the accuracy of the function approximation. In it important to mention that Lucas polynomials are defined recursively, making them computationally efficient to evaluate [62]. Specifically, Lucas polynomials $P_r(x)$ are defined by the linear recurrence relation

$$P_r(x) = xP_{r-1}(x) + P_{r-2}(x), \tag{2.6}$$

with initial conditions $P_0(x) = 2$ and $P_1(x) = x$. The degree of $P_r(x)$ is equal to r.

Layer Normalization (LN). LN is applied immediately after the KAN layer to standardize the output by normalizing feature activations.

Residual Skip Connection. This skip connection links the input of KAN directly to its output, creating a residual pathway. Specifically, the output of the Temporal Dependency Learner is an $(L_a + L_d) \times D$ matrix given by

$$\mathbf{Z_2} = \text{LN}(\text{KAN}(\mathbf{Z_1})) + \mathbf{Z_1}, \tag{2.7}$$

where KAN and LN are applied along the temporal dimension.

2.3.4 Spatial Projection and Inverse Discrete Wavelet Transform

The spatial projection, applied after the Temporal Dependency Learner, refines the spatial relationships between human body joints, ensuring structural consistency in the predicted poses. It models

inter-joint dependencies, complementing the initial spatial projection. On the other hand, IDWT maps the temporally processed data back to the time domain. Together, the spatial projection and IDWT refine joint relationships and reconstruct the motion sequence in the time domain, resulting in an $L \times K$ output expressed as:

$$\mathbf{Z}_3 = \text{IDWT}(\mathbf{Z}_2\mathbf{W}_2), \tag{2.8}$$

where $\mathbf{W}_2 \in \mathbb{R}^{D\times K}$ is a learnable weight matrix. As pointed out in Subsection 2.3.1, IDWT restores the temporal length from $L_a + L_d$ to the original L, thereby generating an $L \times K$ output $\mathbf{Z}_3$. The spatial projection corrects and reinforces joint relationships after KAN has processed the temporal dependencies, while IDWT ensures that these relationships are translated back into the time domain for motion reconstruction.

Model Prediction. The predicted sequence is a $T \times K$ matrix given by

$$\hat{\mathbf{X}}_{L+1:L+T} = \widetilde{\mathbf{Z}}_3 + \mathbf{X}_L, \tag{2.9}$$

where T is the prediction horizon, $\widetilde{\mathbf{Z}}_3$ consists of the first T rows of $\mathbf{Z}_3$, and $\mathbf{X}_L \in \mathbb{R}^{T\times K}$ is constructed by replicating the final pose $\mathbf{x}_L$ of the historical motion sequence T times.

Model Training. We train our model using the following loss function

$$\mathcal{L} = \frac{1}{T}\sum_{t=L+1}^{L+T}(\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_2 + \|\mathbf{v}_t - \hat{\mathbf{v}}_t\|_2), \tag{2.10}$$

where $\|\cdot\|$ denotes the ℓ_2-norm, $\hat{\mathbf{x}}_t$ and $\mathbf{x}_t$ are the predicted and ground truth poses for the t-th predicted frame, $\mathbf{v}_t$ and $\hat{\mathbf{v}}_t$ are the associated velocities, respectively.

2.4 Experiments

2.4.1 Experimental Setup

Datasets. We conduct experimental evaluations on three standard datasets: Human3.6M [63], Archive of Motion Capture as Surface Shapes (AMASS) [64], and 3D Pose in the Wild dataset (3DPW) [65]. We follow standard protocols [2] for data preprocessing and splitting. Details about the datasets are provided below:

Human3.6M. is among the most extensive and widely adopted datasets for 3D human motion analysis, comprising over 3.6 million frames of human activities recorded in a controlled indoor environment. It features 7 actors performing 15 everyday activities, including Walking, Sitting,

Eating, Greeting, Discussing, and Taking Photos. The dataset is preprocessed to represent poses as 3D coordinates with 22 joints per frame. For evaluation, Subject 11 (S11) is reserved for validation, Subject 5 (S5) is used for testing, and the remaining 5 subjects are employed for training.

AMASS is a comprehensive motion database, consolidating multiple optical marker-based motion capture datasets into a standardized framework via the Skinned Multi-Person Linear (SMPL) model. This standardization ensures uniform parameterization across diverse motion recordings. Comprising 51K frames of human activities in indoor and outdoor environments, each pose is represented by 23 joints. For evaluation, the AMASS-BMLrub dataset is used for testing.

3DPW includes a diverse range of challenging human activities, spanning both controlled indoor settings and natural outdoor environments. To assess the robustness and generalization of our model trained on AMASS, we evaluate its performance on the 3DPW test set, where each pose is represented by 18 body joints.

Evaluation Metric and Baselines. We assess the model's performance using the Mean Per Joint Position Error (MPJPE), measured in millimeters, where lower values correspond to better prediction performance. We benchmark LuKAN against several state-of-the-art approaches for 3D human motion prediction, including ConvSeq2Seq [33], Learning Trajectory Dependencies (LTD) [1], History repeats (Hisrep) [2], Dynamic Multiscale Graph Neural Networks (DMGNN) [66], MultiScale Residual Graph Convolution Network (MSR-GCN) [3], Spatial and Temporal Dense Graph Convolutional Network (ST-DGCN) [4], Context-based Interpretable Spatio-Temporal Graph Convolutional Network (CIST-GCN) [5], MotionMixer [6], Skeleton-Parted Graph Scattering Networks (SPGSN) [52], and Simple Multi-Layer Perceptron (SiMLPe) [7].

Implementation Details. All experiments are performed on a single NVIDIA RTX 3070 GPU with 8GB of memory using PyTorch. Our model is trained for 50K epochs on Human3.6M and 115K epochs on AMASS, using a batch size of 128. We use Adam optimizer [67] with a weight decay of 10^{-4}. The learning rate is initialized at 3×10^{-4} and decayed to 10^{-5} after 30K epochs. The look-back window is set to $L = 50$, with a prediction horizon of $T = 10$ for Human3.6M, and $T = 25$ for AMASS and 3DPW. We employ Daubechies wavelets with 4 vanishing moments in both DWT and IDWT, and we set the number of levels of decomposition to 3. We also set the number of temporal dependency learner blocks to $B = 48$.

2.4.2 Results and Analysis

Results on Human3.6M. We report the MPJPE errors averaged across all time steps in Table 2.1 for both short-term (80ms - 400ms) and long-term (560ms - 1000ms) predictions. The results demonstrate the effectiveness of LuKAN compared to the best-performing baseline, SiMLPe. LuKAN consistently achieves lower MPJPE errors across all time steps, with notable relative error reductions. For instance, at the 720ms prediction horizon, LuKAN achieves an MPJPE of 89.9mm compared to 90.1mm for SiMLPe, yielding a relative error reduction of approximately 0.22%. Similarly, at the 1000ms horizon, LuKAN reduces the MPJPE to 109.3mm from SiMLPe's 109.4mm, resulting in a relative error reduction of approximately 0.09%. These results highlight LuKAN's capability to improve upon the state-of-the-art, while maintaining its simple and efficient architecture.

Table 2.1: Average MPJPE results of our model and baseline methods on Human3.6M for different prediction time steps in milliseconds (ms) ranging from 80ms to 1000ms. These MPJPE errors, measured in millimeters (mm), are averaged across all different actions in the dataset. The best results are shown in **bold**, and the second best results are underlined.

	MPJPE (mm)↓							
	80	160	320	400	560	720	880	1000
ConvSeq2Seq [33]	16.6	33.3	61.4	72.7	90.7	104.7	116.7	124.2
LTD-10-10 [1]	11.2	23.4	47.9	58.9	78.3	93.3	106.0	114.0
Hisrep [2]	10.4	22.6	47.1	58.3	77.3	91.8	104.1	112.1
DMGNN [66]	17.0	33.6	65.9	79.7	103	-	-	137.2
MSR-GCN [3]	11.3	24.3	50.8	61.9	80.0	-	-	112.9
ST-DGCN [4]	10.6	23.1	47.1	57.9	76.3	90.7	102.4	109.7
SPGSN [52]	10.4	22.3	47	58.2	77.4	-	-	109.6
CIST-GCN [5]	10.5	23.2	47.9	59.0	77.2	-	-	110.3
MotionMixer [6]	11	23.6	47.8	59.3	77.8	91.4	106	111
SiMLPe [7]	9.6	21.7	46.3	57.3	**75.7**	90.1	101.8	109.4
LuKAN (ours)	**9.4**	**21.5**	**46.2**	**57.2**	**75.7**	**89.9**	**101.6**	**109.3**

Results on AMASS and 3DPW. We train our model on the AMASS dataset and test it on on the AMASS-BMLrub and 3DPW datasets, adhering to the standard evaluation protocol outlined in [2]. The results in Table 2.2 provide a comprehensive comparison of our model against strong baseline methods on the AMASS-BMLrub and 3DPW datasets, evaluated in terms of MPJPE across different prediction horizons. On AMASS-BMLrub, LuKAN achieves competitive results, particularly excelling in short-term predictions. At 80ms and 160ms, LuKAN matches the best-performing LTD-10-10 with MPJPEs of 10.6mm and 19.3mm, respectively. For longer horizons,

LuKAN consistently demonstrates robust performance, achieving the second-best MPJPE scores, such as 34.4mm at 320ms and 66.4mm at 1000ms. Compared to SiMLPe at 320ms, for example, LuKAN yields comparable performance, highlighting its ability to stay on par with state-of-the-art models. On the more challenging 3DPW dataset, which evaluates the generalization ability of prediction models, LuKAN consistently outperforms all baselines across all prediction horizons. For instance, at 320ms, LuKAN achieves an MPJPE of 37.9mm, outperforming SiMLPe's 38.1mm with a relative error reduction of 0.52%. At 1000ms, LuKAN achieves an MPJPE of 72.2mm, matching SiMLPe and further underscoring its robustness in generalization. Overall, the combination of competitive performance in short-term predictions and robust results in long-term horizons highlights LuKAN's versatility and ability to balance prediction accuracy and efficiency across different time horizons.

Table 2.2: **Performance comparison of our model and baselines on AMASS-BMLrub and 3DPW** for various prediction horizons.

	AMASS-BMLrub								3DPW							
	80	160	320	400	560	720	880	1000	80	160	320	400	560	720	880	1000
ConvSeq2Seq [33]	20.6	36.9	59.7	67.6	79.0	87.0	91.5	93.5	18.8	32.9	52.0	58.8	69.4	77.0	83.6	87.8
LTD-10-10 [1]	**10.3**	**19.3**	36.6	44.6	61.5	75.9	86.2	91.2	12.0	22.0	38.9	46.2	59.1	69.1	76.5	81.1
LTD-10-25 [1]	11.0	20.7	37.8	45.3	57.2	65.7	71.3	75.2	12.6	23.2	39.7	46.6	57.9	65.8	71.5	75.5
Hisrep [2]	11.3	20.7	35.7	42.0	51.7	58.6	63.4	67.2	12.6	23.1	39.0	45.4	56.0	63.6	69.7	73.7
SiMLPe [7]	10.8	19.6	**34.3**	**40.5**	**50.5**	**57.3**	**62.4**	**65.7**	12.1	22.1	38.1	44.5	**54.9**	62.4	68.2	**72.2**
Ours	10.6	**19.3**	34.4	40.8	50.9	57.6	62.7	66.4	**11.9**	**21.8**	**37.9**	**44.4**	**54.9**	**62.2**	**68.1**	**72.2**

Action-Wise Performance. The results in Table 2.3 demonstrate that our LuKAN model consistently outperforms or matches the performance of the best-performing baselines, MotionMixer and SiMLPe, across most actions and prediction horizons in terms of MPJPE. For instance, in the Eating action at 400ms, LuKAN achieves an MPJPE of 35.5mm compared to 36.1mm for SiMLPe, resulting in a relative error reduction of 1.7%. Similarly, for the Greeting action at 400ms, LuKAN reduces the MPJPE to 76.2mm compared to SiMLPe's 77.3mm, yielding a relative error reduction of approximately 1.42%, highlighting LuKAN's ability to generate accurate predictions in challenging scenarios. For the Waiting action at 400ms, LuKAN achieves an MPJPE of 53.3mm, which is comparable to SiMLPe's best result of 53.2mm, showcasing LuKAN's ability to deliver competitive performance even in actions where SiMLPe excels. LuKAN's performance is particularly notable in the Sitting Down action at 320ms, where it achieves an MPJPE of 58.7mm, outperforming MotionMixer's MPJPE of 61.4mm with a relative error reduction of 4.4%. This significant improvement underscores LuKAN's capability to handle challenging motions with complex joint interactions.

When considering the average errors reported in the bottom-right corner of the table, the results indicate a consistent improvement in accuracy across all prediction horizons. The relative error reductions for the average MPJPE of LuKAN compared to SiMLPe at 320ms and 400ms are approximately 0.22% and 0.17%, respectively. Overall, LuKAN shows consistent superiority or parity with the best performing baselines across most actions and prediction horizons, reinforcing its effectiveness in predicting 3D human motion.

Qualitative Results. In Figure 2.2, we present a comparison of our predicted poses with those generated by SiMLPe for the Directions and Eating actions from Human3.6M. To facilitate visual assessment, the predicted frames are overlaid on the ground truth poses, highlighting any deviations. For both actions, our model demonstrates superior alignment with the ground truth, particularly for the Directions action. Notably, the predicted leg positions from our model are closer to the ground truth compared to those predicted by SiMLPe.

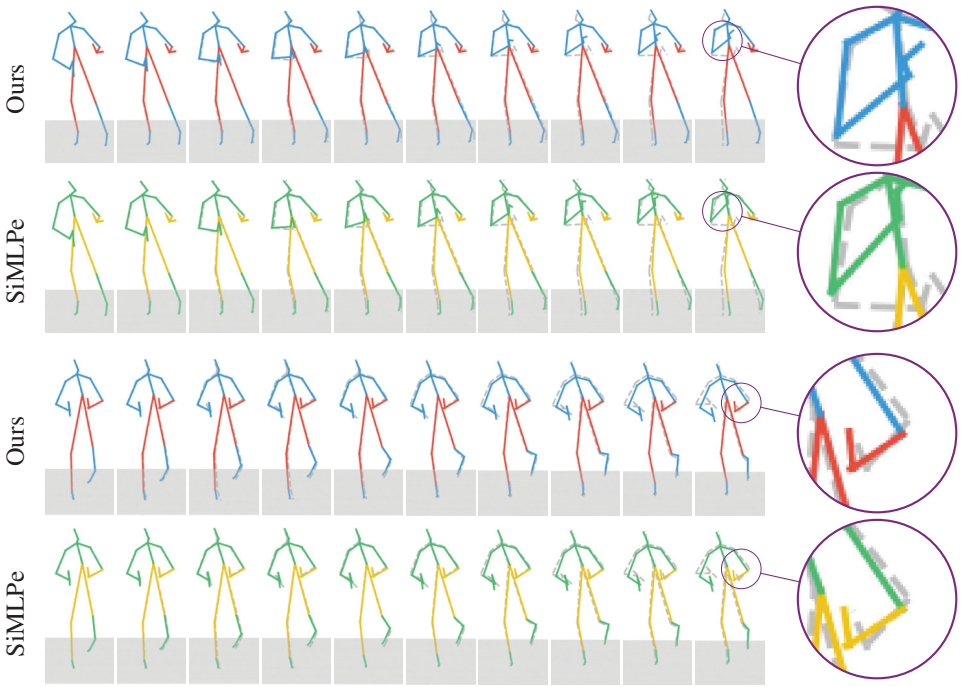

Figure 2.2: **Visual comparison results of our model and the SiMLPe baseline** on two actions: Directions (top) and Eating (bottom). Predicted poses from our model are depicted in red and blue, while those from SiMLPe are shown in yellow and green. Ground truth poses, represented by dashed lines, are overlaid with the predictions to highlight deviations.

Table 2.3: **Action-wise performance comparison of our model and baseline methods on Human3.6M** for different prediction horizons ranging from 80ms to 400ms. MPJPE errors are in mm. The average errors are reported in the bottom-right corner of the table.

	Walking				Eating				Smoking				Discussion			
	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
ConvSeq2Seq [33]	17.7	33.5	56.3	63.6	11.0	22.4	40.7	48.4	11.6	22.8	41.3	48.9	17.1	34.5	64.8	77.6
DMGNN [66]	17.3	30.7	54.6	65.2	11.0	21.4	36.2	43.9	9.0	17.6	32.1	40.3	17.3	34.8	61.0	69.8
MSR-GCN [3]	12.2	22.7	38.6	45.2	8.4	17.1	33.0	40.4	8.0	16.3	31.3	38.2	12.0	26.8	57.1	69.7
LTD-10-10 [1]	11.1	21.4	37.3	42.9	7.0	14.8	29.8	37.3	7.5	15.5	30.7	37.5	10.8	24	52.7	65.8
Hisrep [2]	10.0	19.5	**34.2**	39.8	6.4	14.0	28.7	36.2	7.0	14.9	29.9	36.4	10.2	23.4	52.1	65.4
MotionMixer [6]	10.8	22.4	36.5	42.4	7.7	14.0	**27.3**	36.1	7.1	**14.0**	**29.1**	36.8	10.2	22.5	51.0	64.1
SiMLPe [7]	9.9	-	-	**39.6**	5.9	-	-	36.1	6.5	-	-	**36.3**	9.4	-	-	64.3
Ours	**9.8**	**19.2**	34.3	40	**5.8**	**13.3**	28.1	**35.5**	**6.4**	14.1	29.3	36.4	**9.1**	**22.2**	**50.4**	**63.7**

	Directions				Greeting				Phoning				Posing			
	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
ConvSeq2Seq [33]	13.5	29.0	57.6	69.7	22.0	45.0	82.0	96.0	13.5	26.6	49.9	59.9	16.9	36.7	75.7	92.9
DMGNN [66]	13.1	24.6	64.7	81.9	23.3	50.3	107.3	132.1	12.5	25.8	48.1	58.3	15.3	29.3	71.5	96.7
MSR-GCN [3]	8.6	19.7	43.3	53.8	16.5	37.0	77.3	93.4	10.1	20.7	41.5	51.3	12.8	29.4	67.0	85.0
LTD-10-10 [1]	8.0	18.8	43.7	54.9	14.8	31.4	65.3	79.7	9.3	19.1	39.8	49.7	10.9	25.1	59.1	75.9
Hisrep [2]	7.4	18.4	44.5	56.5	13.7	30.1	63.8	78.1	8.6	18.3	39.0	49.2	10.2	24.2	58.5	75.8
MotionMixer [6]	8.3	18.1	43.8	**53.4**	12.8	33.4	62.3	82.2	10.0	20.1	**37.4**	51.1	11.7	23.3	62.4	79.5
SiMLPe [7]	**6.5**	-	-	55.8	12.4	-	-	77.3	8.1	-	-	48.6	8.8	-	-	**73.8**
Ours	**6.5**	**17.3**	**43.1**	54.9	**12.1**	**28.6**	**62.2**	**76.2**	**7.9**	**17.5**	38.3	**48.1**	**8.5**	**22.5**	**57.3**	74.1

	Purchases				Sitting				Sitting Down				Taking Photo			
	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
ConvSeq2Seq [33]	20.3	41.8	76.5	89.9	13.5	27.0	52.0	63.1	20.7	40.6	70.4	82.7	12.7	26.0	52.1	63.6
DMGNN [66]	21.4	38.7	75.7	92.7	11.9	25.1	44.6	50.2	15.0	32.9	77.1	93.0	13.6	29.0	46.0	58.8
MSR-GCN [3]	14.8	32.4	66.1	79.6	10.5	22.0	46.3	57.8	16.1	31.6	62.5	76.8	9.9	21.0	44.6	56.3
LTD-10-10 [1]	13.9	30.3	62.2	75.9	9.8	20.5	44.2	55.9	15.6	31.4	59.1	71.7	8.9	18.9	41.0	51.7
Hisrep [2]	13.0	29.2	60.4	73.9	9.3	20.1	44.3	56.0	14.9	30.7	59.1	72.0	8.3	18.4	40.7	51.5
MotionMixer [6]	14.6	31.3	62.8	76.1	10.0	20.9	**43.7**	**54.5**	**12.0**	31.4	61.4	74.5	9.0	18.9	41.0	51.6
SiMLPe [7]	**11.7**	-	-	**72.4**	**8.6**	-	-	55.2	13.0	-	-	**70.8**	7.8	-	-	**50.8**
Ours	**11.7**	**27.8**	**59.2**	72.9	**8.6**	**19.4**	43.9	55.7	13.6	**29.5**	**58.7**	71.9	**7.7**	**17.7**	**40.2**	51.1

	Waiting				Walking Dog				Walking Together				Average			
	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
ConvSeq2Seq [33]	14.6	29.7	58.1	69.7	27.7	53.6	90.7	103.3	15.3	30.4	53.1	61.2	16.6	33.3	61.4	72.7
DMGNN [66]	12.2	24.2	59.6	77.5	47.1	93.3	160.1	171.2	14.3	26.7	50.1	63.2	17.0	33.6	65.9	79.7
MSR-GCN [3]	10.7	23.1	48.3	59.2	20.7	42.9	80.4	93.3	10.6	20.9	37.4	43.9	12.1	25.6	51.6	62.9
LTD-10-10 [1]	9.2	19.5	43.3	54.4	20.9	40.7	73.6	86.6	9.6	19.4	36.5	44	11.2	23.4	47.9	58.9
Hisrep [2]	8.7	19.2	43.4	54.9	20.1	40.3	73.3	86.3	8.9	18.4	35.1	41.9	10.4	22.6	47.1	58.3
MotionMixer [6]	10.2	21.1	45.2	56.4	20.5	42.8	75.6	87.8	10.5	20.6	38.7	43.5	11.0	23.6	47.8	59.3
SiMLPe [7]	7.8	-	-	**53.2**	18.2	-	-	83.6	8.4	-	-	41.2	9.6	21.7	46.3	57.3
Ours	**7.6**	**17.9**	**42**	53.3	**18.1**	**38.1**	**71**	**83.3**	**8.2**	**17.6**	**34.4**	**41.1**	**9.4**	**21.5**	**46.2**	**57.2**

2.4.3 Ablation Study

Effect of Temporal Encoding. The results in Table 2.4 compare the performance of DWT and DCT for temporal encoding across Human3.6M, AMASS, and 3DPW datasets in terms of MPJPE. On Human3.6Mt, DWT achieves an MPJPE of 89.9mm at 720ms, outperforming DCT's 90.2mm with a relative error reduction of 0.33%. Similarly, at 1000ms, DWT achieves a lower MPJPE of 109.3mm compared to DCT's 109.5mm, yielding a relative error reduction of 0.18%. On AMASS, DWT consistently outperforms DCT across all time steps. For instance, at 400ms, DWT achieves an MPJPE of 40.8mm compared to 41.5mm for DCT, resulting in a relative error reduction of 1.69%. At 1000ms, DWT reduces the MPJPE to 66.4mm compared to DCT's 66.9mm, with a relative error reduction of 0.75%. On 3DPW, the difference between DWT and DCT is less pronounced, but DWT achieves slightly better results for most time steps. At 320ms, DWT achieves an MPJPE of 37.9mm compared to 38.4mm for DCT, yielding a relative error reduction of 1.3%. Overall, DWT demonstrates consistent improvements over DCT across all datasets, particularly in short-term predictions.

Table 2.4: Ablation study on the choice of temporal encoding: DWT vs. DCT across all datasets for various prediction horizons. DWT consistently outperforms DCT.

		MPJPE (mm)↓							
		80	160	320	400	560	720	880	1000
Human3.6M	DCT	**9.4**	**21.4**	**45.8**	**56.8**	**75.7**	90.2	101.8	109.5
	DWT	**9.4**	21.5	46.2	57.2	**75.7**	**89.9**	**101.6**	**109.3**
AMASS	DCT	10.9	19.7	34.9	41.5	51.6	58.7	63.6	66.9
	DWT	**10.6**	**19.3**	**34.4**	**40.8**	**50.9**	**57.6**	**62.7**	**66.4**
3DPW	DCT	12.2	22.2	38.4	44.9	55.1	62.3	**68.1**	**72.2**
	DWT	**11.9**	**21.8**	**37.9**	**44.4**	**54.9**	**62.2**	**68.1**	**72.2**

Effect of Polynomial Basis. The results in Table 2.5 highlight the superior performance of Lucas polynomials compared to B-splines, used in standard KANs, and other polynomial bases. At 400ms, Lucas polynomials outperform the next best basis, Hermite, yielding a relative error reduction of 0.17%. Similarly, at 1000ms, Lucas polynomials achieve an MPJPE of 109.3mm, outperforming Hermite's 110.1mm by a relative reduction of 0.73%. In comparison to B-splines, the improvements are more pronounced, yielding a relative error reduction of 2.5%. At 320ms, Lucas polynomials achieve an MPJPE of 46.2mm compared to 49.0mm for B-splines, resulting in a relative error reduction of 5.71%. These results demonstrate that Lucas polynomials yield

significant improvements over B-splines and other polynomial bases, for both short- and long-term predictions.

Table 2.5: Ablation study on the choice of the polynomial basis in KAN for various prediction horizons. Lucas polynomials yield significant improvements over B-splines.

Polynomials	MPJPE (mm)↓							
	80	160	320	400	560	720	880	1000
B-Splines	10.3	23.3	49.0	60.1	78.7	92.7	104.5	112.1
Chebyshev	9.7	22.1	47.2	58.3	77.1	91.7	103.7	111.6
Legendre	9.6	21.8	46.8	57.9	76.3	90.4	102.4	110.1
Hermite	9.5	21.6	46.3	57.3	76.0	90.3	102.2	110.1
Lucas	**9.4**	**21.5**	**46.2**	**57.2**	**75.7**	**89.9**	**101.6**	**109.3**

Effect of Embedding Dimension. Table 2.6 reports the effect of varying the embedding dimension D of the spatial projection on model prediction in terms of MPJPE across different time steps using the Human3.6M dataset. In this dataset, each 3D human pose comprises 22 joints, represented by 3 coordinates (x, y, z), resulting in an input spatial dimension of $D = 66$. This input is processed by the spatial projection layer, which maps the poses into different embedding dimensions to optimize the representation for the 3D human motion prediction task. By varying the embedding dimension D, the spatial projection layer refines the inter-joint dependencies, ensuring the spatial relationships are effectively captured for accurate motion prediction. The best performance is observed at $D = 200$, which achieves an MPJPE of 109.3mm at 1000ms, outperforming $D = 66$ with 110.3mm. The consistent improvement across all time steps demonstrates that setting $D = 200$ provides a balanced representation, enhancing the model's ability to capture spatial dependencies effectively, while larger or smaller dimensions introduce insufficient expressiveness.

Table 2.6: Ablation study on the embedding dimension D of the spatial projection for various prediction horizons. The best performance is achieved with $D = 200$.

Embed Dim. (D)	MPJPE (mm)↓							
	80	160	320	400	560	720	880	1000
66	9.8	22.2	47.0	58.1	76.7	90.7	102.5	110.3
100	9.6	21.8	46.5	57.4	76.1	90.5	102.4	110.1
150	9.5	21.7	46.5	57.5	76.0	90.3	101.9	109.6
200	**9.4**	**21.5**	**46.2**	**57.2**	**75.7**	**89.9**	**101.6**	**109.3**
220	9.6	21.7	46.3	57.2	76.0	90.6	102.6	110.4

2.4.4 Model Complexity Analysis

In this section, we analyze the time and memory complexity of LuKAN by considering its main architectural components: spatial projections, DWT and its IDWT, and the Temporal Dependency Learner based on KAN with Lucas polynomial activations.

Time Complexity. Each spatial projection involves a matrix multiplication of complexity $\mathcal{O}(DJL)$, where J is the number of joints, D is the embedding dimension, and L is the length of the input sequence. DWT and its inverse are applied along the temporal dimension. As these are linear-time operations per sequence and per feature, their total complexity is $\mathcal{O}(JL)$. The core component of LuKAN is a B-layer KAN with Lucas polynomial activations, where B is the total number of blocks. Its time complexity is $\mathcal{O}(BDRL^2)$, where R is the degree of the Lucas polynomial. Hence, the time complexity of LuKAN is $\mathcal{O}(DJL + BDRL^2)$.

Memory Complexity. In terms of memory complexity, the model maintains a lightweight parameter count. Each spatial projection require $\mathcal{O}(JD)$ parameters, while the B-layer KAN contributes $\mathcal{O}(BRL^2)$ parameters, giving a total parameter complexity of $\mathcal{O}(JD + BRL^2)$. During runtime, memory is also allocated for storing intermediate activations and for evaluating the polynomial basis, yielding a total runtime memory complexity of $\mathcal{O}(DL + JL)$. Overall, LuKAN achieves a compelling balance between expressive power and computational efficiency.

2.4.5 Performance and Model Size Comparison

A significant advantage of our LuKAN model lies in its simplicity, which distinguishes it from many existing methods for 3D human motion prediction. LuKAN's simple design eliminates unnecessary complexity, focusing instead on robust modeling of temporal and spatial dependencies. Despite its straightforward architecture, LuKAN achieves state-of-the-art performance, demonstrating reduced prediction errors compared to strong baseline methods. As illustrated in Figure 2.3, LuKAN achieves competitive performance by reducing Mean Per Joint Position Error (MPJPE), outperforming strong baseline methods such as LTD [1], Hisrep [2], MSR-GCN [3], ST-DGCN [4], CIST-GCN [5], MotionMixer [6], and SiMLPe [7]. This comparison, conducted on the widely used Human3.6M dataset, highlights LuKAN's ability to deliver accurate motion predictions while maintaining a compact model size. The simplicity of LuKAN translates directly into computational efficiency, making it an ideal solution for applications requiring lightweight models without compromising prediction performance.

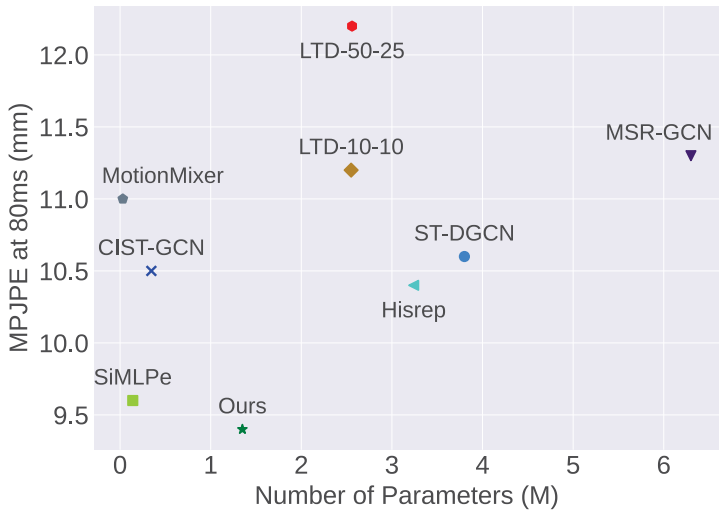

Figure 2.3: **Comparison of performance and model complexity.** Our model is benchmarked against state-of-the-art methods, including LTD [1], Hisrep [2], MSR-GCN [3], ST-DGCN [4], CIST-GCN [5], MotionMixer [6], and SiMLPe [7]. Performance is assessed using the Mean Per Joint Position Error (MPJPE), where lower values indicate superior prediction performance. All evaluations are performed on the Human3.6M dataset.

2.5 Discussion

This work presents LuKAN, an effective and interpretable model for 3D human motion prediction that strikes a balance between simplicity and accuracy while effectively capturing spatio-temporal dependencies. Despite its promising contributions, several limitations and avenues for future research deserve consideration.

Limitations. While LuKAN achieves competitive performance and computational efficiency, its robustness on large-scale and highly diverse motion datasets containing noisy or incomplete data requires further investigation. In addition, although the model is lightweight, further optimization could be explored enhance its suitability for real-time applications, especially in resource-constrained environments.

Future Work. Extending LuKAN to multi-person scenarios or incorporating contextual factors such as environmental constraints could improve its performance in complex, real-world settings. Future efforts could also focus on scaling the model to handle longer motion sequences or higher-dimensional data, expanding its applicability. Exploring alternative polynomial bases may uncover configurations that further enhance accuracy and generalization. Furthermore, integrating LuKAN with unsupervised or semi-supervised learning paradigms could reduce its dependence on large-scale annotated datasets, broadening its accessibility for diverse applications.

Social Impact. LuKAN offers substantial potential for positive societal impact through accurate, efficient, and interpretable 3D human motion prediction. In healthcare, LuKAN could improve physical rehabilitation systems by predicting patient movements, leading to better therapy outcomes. In autonomous systems, LuKAN could enhance human-robot interaction by enabling robots to anticipate human actions, improving safety and usability. In the entertainment and animation industries, its lightweight design could support real-time character animation, lowering production costs and increasing accessibility for smaller studios. In adddition, the interpretability of LuKAN fosters trust and transparency, ensuring ethical use in sensitive domains such as surveillance and monitoring.

CHAPTER 3

HaKAN: Time Series Forecasting with Hahn Kolmogorov-Arnold Networks

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.

3.1 Introduction

Time series forecasting leverages historical data to predict future values, and is widely used as a critical tool in diverse domains ranging from retail, energy and transportation to healthcare and finance [68, 69]. 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 [38–41] rely on attention mechanisms to capture long-range dependencies, with simple strategies such as channel independence and patching [42] 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 [47, 70] 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 [59], 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) [12, 58] 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 [60, 61], 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 [58]. 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. Motivated by the expressive power of KANs, we propose HaKAN, a novel framework for multivariate long-term time series forecasting, where each KAN layer is parameterized using Hahn Polynomials [71], 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 chapter 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.

3.2 Related Work

Transformer-based Models. A sizable body of research has focused on designing Transformer-based methods for long-term time series forecasting [37–42]. For instance, Informer [38] enhances Transformer efficiency with a ProbSparse self-attention mechanism, self-attention distilling, and a generative decoder. Autoformer [39] introduces a decomposition architecture with an auto-correlation mechanism that leverages series periodicity for dependency discovery and representation aggregation. FEDformer [40] integrates seasonal-trend decomposition with Fourier and Wavelet transforms to capture global time series characteristics, while iTransformer [41] 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 [42] 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-based Models. Various MLP-based models have been adopted for long-term time series forecasting [43–47] due to their architectural and computational efficiency. TSMixer [43] captures temporal patterns and cross-variate information by interleaving time-mixing and feature-mixing MLPs. which effectively integrates cross-variate and auxiliary information. Time Series MLP [45] introduces a lightweight MLP with a precurrent mechanism to capture local temporal dependencies and variate interactions. FreTS [46] leverages frequency-domain MLPs to capture global dependencies and compact signal energy. DLinear [47] enhances long-term time series forecasting by decomposing input data into trend and seasonal components using a moving average kernel, applying separate linear layers to each, and summing the results. 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. Our proposed HaKAN framework differs from MLP-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.

3.3 Method

3.3.1 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 multivariate 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 [60, 61], 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 [12], 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

$$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}^{(\mathsf{L}-1)} \circ \cdots \circ \mathbf{\Phi}^{(1)} \circ \mathbf{\Phi}^{(0)})\mathbf{x}, \tag{3.1}$$

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

3.3.2 Approach Overview

As depicted in Figure 3.1, HaKAN is a lightweight architecture designed for multivariate time series forecasting, leveraging KANs to predict future values over long horizons while addressing the computational and causal limitations of Transformer- and MLP-based models. Given an input time series, represented as a matrix of timestamps and variables, HaKAN first applies the

channel-independence strategy, treating each channel separately to capture channel-specific temporal dynamics with reduced complexity. Reversible instance normalization is then applied to stabilize data distribution shifts, standardizing each channel independently and ensuring robustness across varying conditions. The normalized sequence is segmented into overlapping patches via a sliding window, preserving local dependencies while reducing the sequence length for improved efficiency. Each patch is projected into a latent space through a temporal linear embedding, with positional embeddings added to maintain temporal order. The embedded patch tokens are processed by a stack of Hahn-KAN blocks, each consisting of an inter-patch KAN layer, which models global patterns across patches using Hahn Polynomial-based activation functions, and an intra-patch KAN layer, which refines local relationships within patches. Residual connections stabilize learning, allowing hierarchical temporal features to emerge across different scales. After passing through multiple Hahn-KAN blocks, the sequence is flattened and passed through a lightweight bottleneck formed by two fully connected layers, producing the final forecast for each channel over the prediction horizon. A final denormalization step restores the outputs to the original scale.

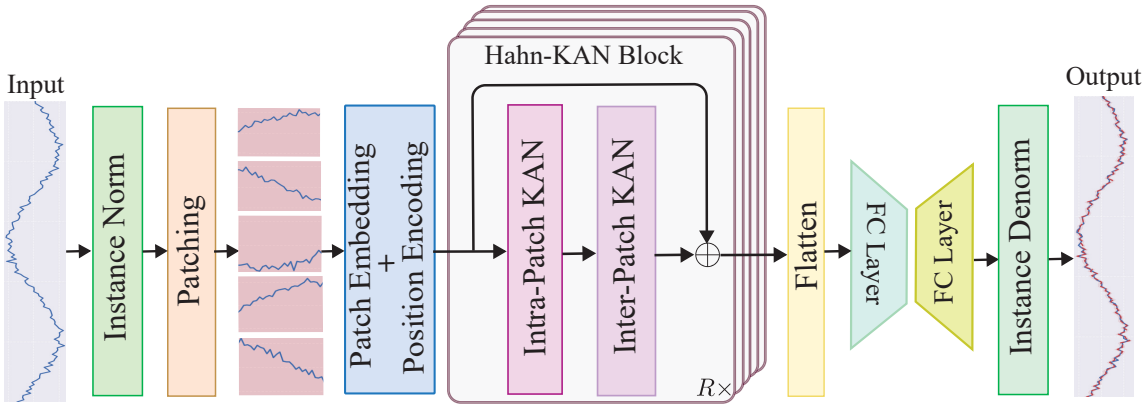

Figure 3.1: **HaKAN Architecture**. The model integrates channel independence, reversible instance normalization, and patching, followed by patch and position embeddings. A stack of R Hahn-KAN blocks, each with intra-patch and inter-patch KAN layers using Hahn polynomials, processes the embedded sequence to capture temporal patterns. The output is mapped through a bottleneck structure with two fully connected layers to produce the final forecast.

3.3.3 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 3.1, 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 [42]. 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 [72], 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 [42]. 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 embedding 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:

$$\mathbf{X}_d^{(i)} = \mathbf{X}_p^{(i)} \mathbf{W}_p + \mathbf{W}_{\text{pos}}, \tag{3.2}$$

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:

$$\mathbf{X}_k^{(i)} = \text{KAN}(\text{KAN}(\mathbf{X}_d^{(i)})^\intercal)^\intercal + \mathbf{X}_d^{(i)}, \tag{3.3}$$

where $\mathbf{X}_k^{(i)} \in \mathbb{R}^{N \times D}$ is the output of the block, and each $\text{KAN}(\cdot)$ 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 [71] to provide flexibility in function approximation:

$$\phi_{q,p}(x_p) = \sum_{r=0}^{d} \gamma_{q,p,r} P_r(x_p), \tag{3.4}$$

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

$$AP_r(x) = (A + B - x)P_{r-1}(x) - BP_{r-2}(x), \tag{3.5}$$

with coefficients:

$$A = \frac{(r+a+b)(r+a)(n-r+1)}{(2r+a+b-1)(2r+a+b)}, \tag{3.6}$$

$$B = \frac{(r-1)(r+b-1)(r+a+b+n)}{(2r+a+b-2)(2r+a+b-1)}, \tag{3.7}$$

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 $\mathcal{O}(d_{\text{in}}d_{\text{out}}[9d(G+1.5d)+2G-2.5d+3])$ [29], 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 [29]. 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 $\mathbf{W}_{\text{down}} \in \mathbb{R}^{H\times ND}$:

$$\mathbf{h}^{(i)} = \mathbf{W}_{\text{down}}\mathbf{x}_f^{(i)}, \tag{3.8}$$

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 $\mathbf{W}_{\text{up}} \in \mathbb{R}^{T\times H}$:

$$\hat{\mathbf{x}}^{(i)} = \mathbf{W}_{\text{up}}\mathbf{h}^{(i)}, \tag{3.9}$$

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.

3.3.4 Model Training

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

$$\mathcal{L} = \frac{1}{MT} \sum_{i=1}^{M} \sum_{\tau=L+1}^{L+T} \left\| \mathbf{x}_{\tau}^{(i)} - \hat{\mathbf{x}}_{\tau}^{(i)} \right\|^2, \tag{3.10}$$

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.

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 $\mathbf{X}_k^{(i)} \in \mathbb{R}^{N \times D} \rightarrow \mathbf{x}_f^{(i)} \in \mathbb{R}^{ND}$
  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: $\hat{\mathbf{X}}_{L+1:L+T} = (\hat{\mathbf{x}}^{(1)}, \ldots, \hat{\mathbf{x}}^{(M)})$

3.4 Experiments

3.4.1 Experimental Setup

Datasets. We evaluate HaKAN on several benchmark datasets: Weather, Electricity, Illness, and four ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2) [39]. **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. Dataset statistics are summarized in Table 3.1.

Table 3.1: 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

Baselines and Evaluation Metrics. We evaluate the performance of our model against various recent state-of-the-art methods, including S-Mamba [73], TimeKAN [74], Timer-XL [75], TsKAN [76], iTransformer [41], PatchTST [42], TimesNet [77], Crossformer [78], DLinear and RLinear [47], N-HiTS [44], TiDE [70], MICN [79], and FEDformer [40]. 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 [67] is used as optimizer. For the KAN layers, we use Hahn polynomials of the form 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 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 [42]. 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. Code is available at: `https://anonymous.4open.science/r/HPKAN-F82C`. A detailed list of hyperparameters for each experiment is shown below. For both look-backs $L = 336$ and $L = 96$,

Table 3.2: 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.

Method		**HaKAN**		TsKAN		Timer-XL		TimeKAN		PatchTST/64		N-HiTS		DLinear		MICN		TimesNet	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
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
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
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 number of Hahn-KAN blocks and maximum degree of Hahn polynomials are set to 3. For Hahn polynomial basis Hahn(a, b, n), we set $a = 1, b = 1, n = 7$.

Table 3.3: 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 3.4: 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

Table 3.5: 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**		S-Mamba		iTransformer		RLinear		PatchTST/64		Crossformer		TiDE		TimesNet		FEDformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
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
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

3.4.2 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 [42] and DLinear [47], ensuring consistency in the comparison. All 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 3.2, HaKAN consistently outperforms the baselines, achieving the best MSE in 19 out of 32 cases and the best MAE in 24 out of 32 cases, with notable relative error reductions such as 16% on Weather and 46.5% on Illness. It excels particularly on datasets with smooth trends like Weather, Electricity, and ETTm2, for instance, achieving a relative error reduction of 29% on ETTh2.

Fixed Look-back Window. A number of baselines, such as S-Mamba [73] and iTransformer [41], 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 3.5, 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.

3.4.3 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 the Illness dataset. Given a look-back window L, we define the

average MSE and MAE over all datasets and across all prediction horizons as follows:

$$\overline{\text{MSE}} = \frac{1}{|\mathcal{T}||\mathcal{D}|} \sum_{S \in \mathcal{D}} \sum_{T \in \mathcal{T}} \text{MSE}(\mathbf{X}^S_{L+1:L+T}, \hat{\mathbf{X}}^S_{L+1:L+T}), \tag{3.11}$$

$$\overline{\text{MAE}} = \frac{1}{|\mathcal{T}||\mathcal{D}|} \sum_{S \in \mathcal{D}} \sum_{T \in \mathcal{T}} \text{MSE}(\mathbf{X}^S_{L+1:L+T}, \hat{\mathbf{X}}^S_{L+1:L+T}), \tag{3.12}$$

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

Polynomial Basis. We conduct an ablation study to assess how different polynomial bases [71] affect the performance of HaKAN, with results summarized in Table 3.6. The findings show that the choice of basis functions significantly impacts forecasting performance, with Hahn outperforming alternatives across all metrics.

Table 3.6: Impact of the polynomial basis.

	Evaluation Metric (↓)		
Polynomial Basis	$\overline{\text{MSE}}$	$\overline{\text{MAE}}$	Avg.
Hahn(1, 1, 7)	**0.508**	**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
Monomials	0.566	0.443	0.504

Number of Hahn-KAN Blocks. Table 3.7 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.

Table 3.7: Impact of the number of Hahn-KAN blocks.

Number of Blocks	$\overline{\text{MSE}}$	$\overline{\text{MAE}}$	Params (K)
1	0.526	0.436	**635**
3	0.534	0.438	767
5	**0.508**	**0.431**	899
20	0.549	0.442	1891

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 3.8 summarizes the results, which indicate that a bottleneck dimension of 336 provides the best balance between model size and predictive performance.

Table 3.8: Impact of the bottleneck dimension.

Bottleneck Dimension	$\overline{\text{MSE}}$	$\overline{\text{MAE}}$	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

HaKAN vs. MLP-Based Variant. Figure 3.2 provides a comparative analysis of the forecasting performance of the HaKAN model 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.

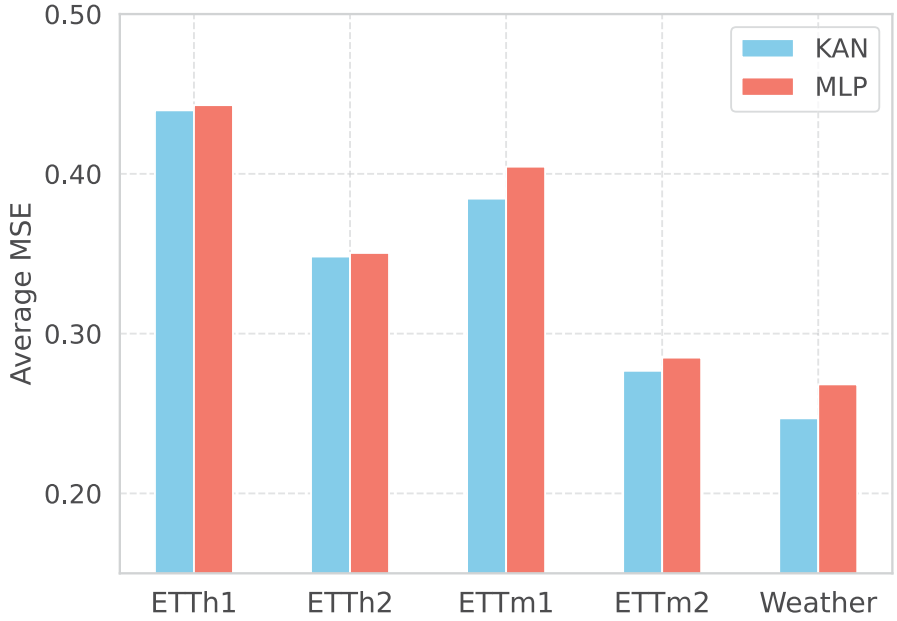

Figure 3.2: Performance comparison between HaKAN and its MLP-based variant across multiple datasets. The look-back window is fixed at $L = 96$, and the average MSE over prediction horizons $T \in \{96, 192, 336, 720\}$ is used as the evaluation metric.

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 3.3 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.

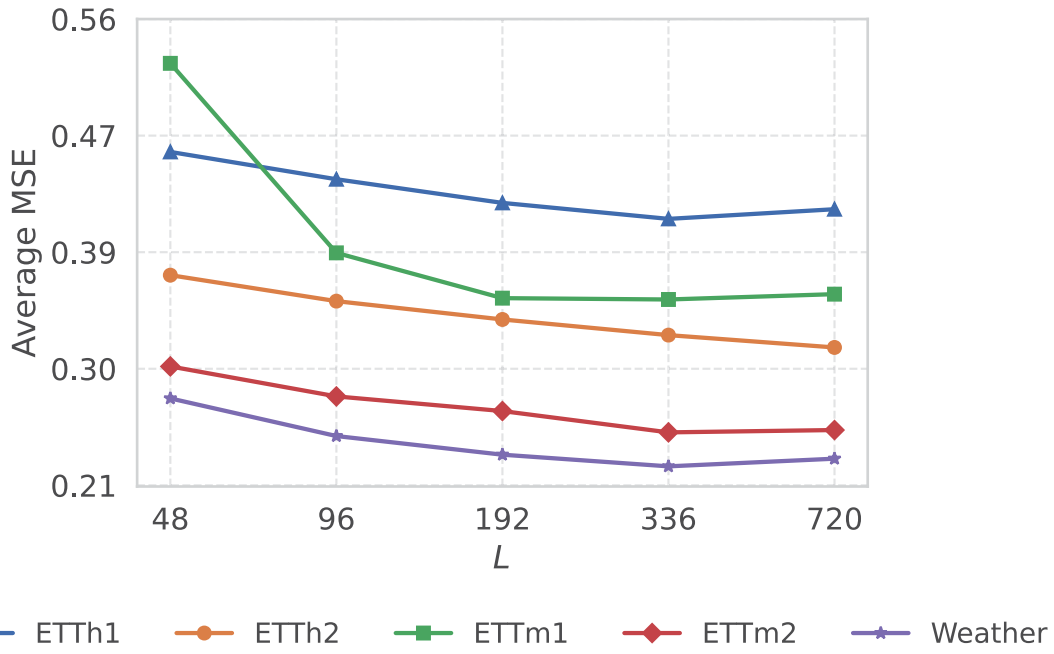

Figure 3.3: Evaluation of long-term forecasting performance across different look-back window lengths on multiple datasets, using the average MSE over prediction horizons $T \in \{96, 192, 336, 720\}$ as the evaluation metric.

Effect of Patch Length. Figure 3.4 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.

Effect of Intra-Patch and Inter-Patch. Table 3.9 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

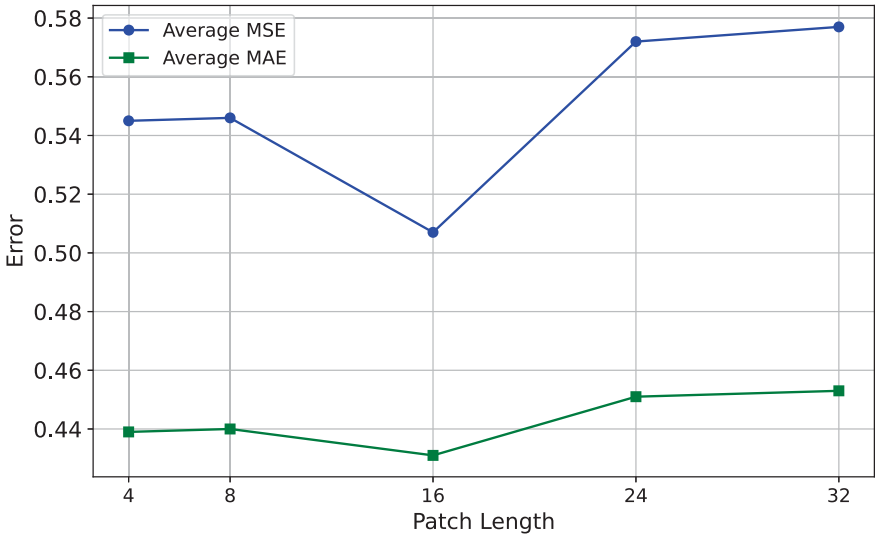

Figure 3.4: Average MSE and MAE results across the six ablation datasets for a varying patch length.

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.

Table 3.9: Effect of intra- and inter-patch KAN layers on model performance.

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

3.4.4 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 3.10: Average $\pm$ 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

3.4.5 Computational Complexity Analysis

Recall that M denotes the number of channels, L the input sequence length, P the patch length, S the stride, $N = \left\lfloor \frac{L-P}{S} \right\rfloor + 2$ the number of patches, D the patch embedding dimension, R the number of HP-KAN blocks, H the bottleneck dimension, and d the degree of Hahn polynomials (typically $d = 3$).

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 $\mathcal{O}(M[R(N^2+D^2)+NDH+HT])$.

3.5 Discussion

This work introduces HaKAN, a Hahn polynomial-based Kolmogorov-Arnold Network designed for efficient and interpretable multivariate time series forecasting. HaKAN demonstrates strong generalization across diverse datasets and forecasting horizons while maintaining parameter efficiency. Its modular patching design, together with reversible instance normalization (RevIN) and intra/inter-patch KAN layers, enables effective modeling of both local and global temporal dependencies. Despite its advantages, several limitations and opportunities for future research remain.

Limitations. Although HaKAN achieves state-of-the-art accuracy on benchmark datasets, its performance in an unsupervised or self-supervised setting has not yet been explored. Also, highly irregular or sparse time series has not been explored. The current framework assumes uniformly sampled data and relies on supervised training, which may limit adaptability to missing or asynchronous signals. Another limitation is that, HaKAN cannot utilize the potential of inter-channel dependence.

Future Work. Future extensions could explore data-driven or adaptive polynomial selection, allowing the model to automatically learn the most suitable basis for a given dataset. Integrating self-supervised or contrastive pre-training objectives could improve robustness under limited labeled data. Applying HaKAN to domains such as financial forecasting, energy management, and physiological monitoring offers further potential for real-world impact. It is worth exploring a channel-dependent forecasting approach using orthogonal polynomials to see how they can learn inter-variable dependencies.

Social Impact. By enabling accurate and interpretable time series forecasting, HaKAN contributes positively to data-driven decision-making in critical sectors. In healthcare, it could assist in early detection of anomalies in patient vitals. In energy and environmental monitoring, its lightweight design can support real-time edge deployment for sustainability analytics. In finance

and logistics, improved forecasting stability can enhance resource planning and risk mitigation. The interpretability afforded by orthogonal polynomial activations also fosters transparency and trust in automated predictive systems, ensuring ethical use in sensitive applications.

CHAPTER 4

Conclusions and Future Work

This thesis presented two novel Kolmogorov-Arnold Network (KAN)-based frameworks, namely LuKAN and HaKAN, for 3D human motion prediction and long-term time series forecasting, respectively. Both models were designed for a distinct, but conceptually related domain. LuKAN addresses structured 3D human motion prediction, while HaKAN focuses on multivariate time series forecasting. Despite their different application contexts, both models stem from a common theoretical foundation: learnable univariate functions parameterized by orthogonal polynomials can provide adaptive, interpretable, and spectrally balanced feature representations. Throughout this work, we have investigated how polynomial-activated KANs bridge function approximation and modern deep learning. The temporal dependency learner of LuKAN and the Hahn-KAN-block of HaKAN show that KANs are at least as powerful as Transformer-based methods in capturing temporal dependencies. We have also demonstrated that in addition to superior performance, orthogonal polynomial-based KANs are comparable to MLPs in terms of time and parameter complexity.

4.1 Summary of Thesis Contributions

4.1.1 LuKAN: Lucas Polynomial KAN for 3D Motion Prediction

In Chapter 2, we proposed LuKAN, an effective model for predicting 3D human motion, inspired by Kolmogorov-Arnold networks. Our model captures both localized temporal dependencies and complex motion dynamics effectively. The model's spatial projections ensure that LuKAN main-

tains structural consistency while remaining computationally efficient. Through extensive experiments on three benchmark datasets, we demonstrated that our model achieves competitive or superior prediction performance compared to state-of-the-art methods, with significantly fewer parameters and lower computational cost. Notably, LuKAN strikes a good balance between prediction accuracy, efficiency, and model simplicity.

4.1.2 HaKAN: Hahn Polynomial KAN for Time Series Forecasting

In Chapter 3, 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.

4.2 Discussion and Implications

The findings of this thesis carry several important implications for the future of deep learning in spatiotemporal analysis:

1. **Bridging Classical and Modern Paradigms:** The proposed models reveal that frequency-domain reasoning and neural function approximation are not mutually exclusive. Orthogonal polynomial KANs provide a principled framework to unify these two traditions, combining the interpretability of classical transforms with the flexibility of modern networks.

2. **Interpretability and Spectral Transparency:** The univariate nature of KAN activations allows direct visualization and interpretation of learned nonlinearities. Unlike opaque activation layers in conventional networks, the polynomial coefficients learned by LuKAN and HaKAN describe explicit functional mappings, enabling post-hoc analysis of frequency responses.

3. **Computational Efficiency:** Owing to their compact polynomial representations and edge-wise learning design, KAN-based architectures exhibit reduced parameter counts and lower

training costs compared to Transformer variants, making them suitable for embedded and real-time applications.

4.3 Limitations

Despite their advantages, the proposed models have several limitations that merit further attention:

- **Lack of Theoretical Guarantees:** Although empirical results suggest improved spectral balance, formal proofs on convergence rates, spectral bias theory and approximation error bounds under orthogonal activations are still absent.
- **Supervised Learning Only:** So far, both of the problems were solved in a supervised setting. It's worth seeing how our models do in masked-reconstruction, or contrastive learning or how they learn latent representation by minimizing reconstruction loss.
- **Lack of Real World Complex Data:** Future efforts could also focus on scaling the model to handle longer motion sequences or higher-dimensional data, expanding its applicability.
- **Single Agent Motion Prediction:** Extending LuKAN to multi-person scenarios or incorporating contextual factors such as environmental constraints could improve its performance in complex, realworld settings.

4.4 Future Work

The exploration of orthogonal polynomial-based KANs opens several promising directions for future research:

4.4.1 Theoretical Analysis

- Developing rigorous mathematical analysis of the approximation properties of polynomial KANs, particularly their relationship to spline- and kernel-based methods.
- Quantifying spectral bias reduction analytically by examining the Fourier spectra of learned polynomial activations.
- Extending the theoretical framework to non-orthogonal or data-driven basis expansions, such as learned polynomial families or piecewise functional priors.

4.4.2 Architectural Extensions

- Incorporating dynamic polynomial adaptation, where the network learns not only polynomial coefficients but also the optimal order or family (e.g., Legendre, Chebyshev, Hermite) for each layer.
- Integrating KANs with graph-based models to better capture spatial dependencies in multi-agent systems, human-object interactions, or sensor networks.
- Exploring hybrid frequency-KAN architectures that combine DWT/DCT preprocessing with learnable polynomial functions for enhanced interpretability.

4.4.3 Application-Oriented Directions

- Deploying KAN-based forecasting systems in real-world scenarios such as energy demand prediction, physiological signal monitoring, and motion planning for robotics in a multi-agent environment.
- Investigating multimodal fusion, where KANs jointly model visual, textual, and auditory temporal streams.
- Adapting polynomial KANs for physics-informed and partial differential equation (PDE) modeling, where function interpretability aligns with governing dynamics.

References

[1] W. Mao, M. Liu, M. Salzmann, and H. Li, "Learning trajectory dependencies for human motion prediction," in *Proc. IEEE International Conference on Computer Vision*, 2019, pp. 9489–9497.

[2] W. Mao, M. Liu, and M. Salzmann, "History repeats itself: Human motion prediction via motion attention," in *Proc. European Conference on Computer Vision*, 2020, pp. 474–489.

[3] L. Dang, Y. Nie, C. Long, Q. Zhang, and G. Li, "MSR-GCN: Multi-scale residual graph convolution networks for human motion prediction," in *Proc. IEEE International Conference on Computer Vision*, 2021, pp. 11 447–11 456.

[4] T. Ma, Y. Nie, C. Long, Q. Zhang, and G. Li, "Progressively generating better initial guesses towards next stages for high-quality human motion prediction," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6437–6446.

[5] E. Medina, L. Loh, N. Gurung, K. H. Oh, and N. Heller, "Context-based interpretable spatio-temporal graph convolutional network for human motion forecasting," in *Proc. IEEE Winter Conference on Applications of Computer Vision*, 2024.

[6] A. Bouazizi, A. Holzbock, U. Kressel, K. Dietmayer, and V. Belagiannis, "MotionMixer: MLP-based 3D human body pose forecasting," in *Proc. International Joint Conference on Artificial Intelligence*, 2022, pp. 791–798.

[7] W. Guo, Y. Du, X. Shen, V. Lepetit, X. Alameda-Pineda, and F. Moreno-Noguer, "Back to MLP: A simple baseline for human motion prediction," in *Proc. IEEE Winter Conference on Applications of Computer Vision*, 2023, pp. 4809–4819.

[8] H. Krim and A. Ben Hamza, *Geometric methods in signal and image analysis*. Cambridge University Press, 2015.

[9] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[10] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *International Conference on Learning Representations*, 2022.

[11] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: long-term forecasting with Transformers," in *International Conference on Learning Representations*, 2023.

[12] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold Networks," in *International Conference on Learning Representations*, 2025.

[13] M. G. Altarabichi, "Dropkan: Regularizing kans by masking post-activations," *arXiv preprint arXiv:2407.13044*, 2024.

[14] C. Coffman and L. Chen, "MatrixKAN: Parallelized Kolmogorov-Arnold network," *arXiv preprint arXiv:2502.07176*, 2025.

[15] H.-T. Ta, D.-Q. Thai, A. Tran, G. Sidorov, and A. Gelbukh, "PRKAN: Parameter-reduced Kolmogorov-Arnold networks," *arXiv preprint arXiv:2501.07032*, 2025.

[16] H.-T. Ta and A. Tran, "AF-KAN: Activation function-based Kolmogorov-Arnold networks for efficient representation learning," *arXiv preprint arXiv:2503.06112*, 2025.

[17] Q. Qiu, T. Zhu, H. Gong, L. Chen, and H. Ning, "ReLU-KAN: New Kolmogorov-Arnold networks that only need matrix addition, dot multiplication, and ReLU," *arXiv preprint arXiv:2406.02075*, 2024.

[18] S. SS, K. AR, G. R, and A. KP, "Chebyshev polynomial-based Kolmogorov-Arnold networks: An efficient architecture for nonlinear function approximation," *arXiv preprint arXiv:2406.02075*, 2024.

[19] J. Zhang, Y. Fan, K. Cai, and K. Wang, "Kolmogorov-Arnold Fourier networks," *arXiv preprint arXiv:2502.06018*, 2025.

[20] Z. Bozorgasl and H. Chen, "Wav-KAN: Wavelet Kolmogorov-Arnold networks," *arXiv preprint arXiv:2405.12832*, 2024.

[21] F. Zhang and X. Zhang, "GraphKAN: Enhancing feature extraction with graph Kolmogorov-Arnold networks," *arXiv preprint arXiv:2406.13597*, 2024.

[22] H. Inzirillo and R. Genet, "SigKAN: Signature-weighted Kolmogorov-Arnold networks for time series," *arXiv preprint arXiv:2406.17890*, 2024.

[23] R. Genet and H. Inzirillo, "TKAN: Temporal Kolmogorov-Arnold networks," *arXiv preprint arXiv:2405.07344*, 2025.

[24] X. Han, X. Zhang, Y. Wu, Z. Zhang, and Z. Wu, "Are KANs effective for multivariate time series forecasting?" *arXiv preprint arXiv:2408.11306*, 2025.

[25] Y. Wang, J. Sun, J. Bai, C. Anitescu, M. S. Eshaghi, X. Zhuang, T. Rabczuk, and Y. Liu, "Kolmogorov–arnold-informed neural network: A physics-informed deep learning framework for solving forward and inverse problems based on kolmogorov–arnold networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 433, p. 117518, Jan. 2025.

[26] W. Knottenbelt, Z. Gao, R. Wray, W. Z. Zhang, J. Liu, and M. Crispin-Ortuzar, "Coxkan: Kolmogorov-arnold networks for interpretable, high-performance survival analysis," 2024.

[27] G. G. Cruz, B. Renczes, M. C. Runacres, and J. Decuyper, "State-space kolmogorov arnold networks for interpretable nonlinear system identification," *IEEE Control Systems Letters*, vol. 9, pp. 847–852, 2025.

[28] D. W. Abueidda, P. Pantidis, and M. E. Mobasher, "DeepOKAN: Deep operator network based on Kolmogorov-Arnold networks for mechanics problems," *arXiv preprint arXiv:2405.19143*, 2024.

[29] X. Yang and X. Wang, "Kolmogorov-Arnold Transformer," in *International Conference on Learning Representations*, 2025.

[30] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," *Proc. IEEE International Conference on Computer Vision*, pp. 4346–4354, 2015.

[31] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep learning on spatio-temporal graphs," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5308–5317.

[32] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4674–4683.

[33] C. Li, Z. Zhang, W. S. Lee, and G. H. Lee, "Convolutional sequence to sequence model for human dynamics," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5226–5234.

[34] T. Hassan and A. Ben Hamza, "Regular splitting graph network for 3D human pose estimation," *IEEE Transactions on Image Processing*, vol. 32, pp. 4212–4222, 2023.

[35] M. Mesgaran and A. Ben Hamza, "Graph fairing convolutional networks for anomaly detection," *Pattern Recognition*, vol. 145, 2023.

[36] Y. Fenga, Z. Dou, L.-H. Chen, Y. Liu, T. Li, J. Wang, Z. Cao, W. Wang, T. Komura, and L. Liu, "MotionWavelet: Human motion prediction via wavelet manifold learning," *arXiv:2411.16964*, 2024.

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

[38] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient Transformer for long sequence time-series forecasting," in *Proc. AAAI Conference on Artificial Intelligence*, 2021, pp. 11 106–11 115.

[39] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition Transformers with auto-correlation for long-term series forecasting," in *Advances in Neural Information Processing Systems*, 2021, pp. 22 419–22 430.

[40] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed Transformer for long-term series forecasting," in *Proc. International Conference on Machine Learning*, 2022, pp. 27 268–27 286.

[41] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "iTransformer: Inverted Transformers are effective for time series forecasting," in *International Conference on Learning Representations*, 2024.

[42] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: long-term forecasting with Transformers," in *International Conference on Learning Representations*, 2023.

[43] S.-A. Chen, C.-L. Li, N. Yoder, S. O. Arik, and T. Pfister, "TSMixer: An all-MLP architecture for time series forecasting," *Transactions on Machine Learning Research*, 2023.

[44] C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramirez, M. M. Canseco, and A. Dubrawski, "N-HiTS: Neural hierarchical interpolation for time series forecasting," in *Proc. AAAI Conference on Artificial Intelligence*, 2023, pp. 6989–6997.

[45] Z. Wang, S. Ruan, T. Huang, H. Zhou, S. Zhang, Y. Wang, L. Wang, Z. Huang, and Y. Liu, "A lightweight multi-layer perceptron for efficient multivariate time series forecasting," *Knowledge-Based Systems*, vol. 288, 2024.

[46] K. Yi, Q. Zhang, W. Fan, S. Wang, P. Wang, H. He, N. An, D. Lian, L. Cao, and Z. Niu, "Frequency-domain MLPs are more effective learners in time series forecasting," in *Advances in Neural Information Processing Systems*, 2023, pp. 76 656–76 679.

[47] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are Transformers effective for time series forecasting?" in *Proc. AAAI conference on Artificial Intelligence*, 2023, pp. 11 121–11 128.

[48] C. Diller and A. Dai, "CG-HOI: Contact-guided 3D human-object interaction generation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 888–19 901.

[49] S. Azadi, A. Shah, T. Hayes, D. Parikh, and S. Gupta, "Make-An-Animation: Large-scale text-conditional 3D human motion generation," in *Proc. IEEE International Conference on Computer Vision*, 2023, pp. 15 039–15 048.

[50] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, N. Singh, and J. Schneiders, "Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving," in *Proc. IEEE Winter Conference on Applications of Computer Vision*, 2020.

[51] P. Wu, S. Chen, and D. Metaxas, "MotionNet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 385–11 395.

[52] M. Li, S. Chen, Z. Zhang, L. Xie, Q. Tian, and Y. Zhang, "Skeleton-parted graph scattering networks for 3D human motion predictiont," in *Proc. European Conference on Computer Vision*, 2022, pp. 18–36.

[53] X. Sun, H. Sun, B. Li, D. Wei, W. Li, and J. Lu, "DeFeeNet: Consecutive 3D human motion prediction with deviation feedback," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5527–5536.

[54] D. Wei, H. Sun, X. Sun, and S. Hug, "NeRMo: Learning implicit neural representations for 3D human motion prediction," in *Proc. European Conference on Computer Vision*, 2024, pp. 409–427.

[55] J. Zhang, Y. Zhang, X. Cun, S. Huang, Y. Zhang, H. Zhao, H. Lu, and X. Shen, "T2M-GPT: Generating human motion from textual descriptions with discrete representations," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2023.

[56] Y. Cai, L. Huang, Y. Wang, T.-J. Cham, J. Cai, J. Yuan, J. Liu, X. Yang, Y. Zhu, X. Shen, D. Liu, J. Liu, and N. M. Thalmann, "Learning progressive joint propagation for human motion prediction," in *Proc. European Conference on Computer Vision*, 2020.

[57] E. Aksan, M. Kaufmann, P. Cao, and O. Hilliges, "A spatio-temporal transformer for 3D human motion prediction," in *Proc. International Conference on 3D Vision*, 2021, pp. 565–574.

[58] Y. Wang, J. W. Siegel, Z. Liu, and T. Y. Hou, "On the expressiveness and spectral bias of KANs," in *International Conference on Learning Representations*, 2025.

[59] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *Proc. International Conference on Maching Learning*, 2019.

[60] J. Braun and M. Griebel, "On a constructive proof of Kolmogorov's superposition theorem," *Constructive Approximation*, vol. 30, pp. 653–675, 2009.

[61] J. Schmidt-Hieber, "The Kolmogorov-Arnold representation theorem revisited," *Neural Networks*, vol. 137, pp. 119–126, 2021.

[62] Ömer Oruç, "A new algorithm based on lucas polynomials for approximate solution of 1D and 2D nonlinear generalized Benjamin-Bona-Mahony-Burgers equation," *Computers and Mathematics with Applications*, vol. 74, pp. 3042–3057, 2017.

[63] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2014.

[64] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "AMASS: Archive of motion capture as surface shapes," in *Proc. IEEE International Conference on Computer Vision*, 2019.

[65] T. von Marcard, R. Henschel, M. B. B. J., Rosenhahn, and G. Pons-Moll, "Recovering accurate 3D human pose in the wild using IMUs and a moving camera," in *Proc. European Conference on Computer Vision*, 2018.

[66] M. Li, S. Chen, Y. Zhao, Y. Zhang, Y. Wang, and Q. Tian, "Dynamic multiscale graph neural networks for 3D skeleton-based human motion prediction," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 215–223.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[68] D. Wang, G. Guo, T. Ouyang, D. Yu, H. Zhang, B. Li, R. Jiang, G. Xu, and S. Deng, "A lightweight spatio-temporal neural network with sampling-based time series decomposition for traffic forecasting," *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[69] X. Zhang, Z. Huang, Y. Wu, X. Lu, E. Qi, Y. Chen, Z. Xue, Q. Wang, P. Wang, and W. Wang, "Multi-period learning for financial time series forecasting," in *Proc. ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

[70] A. Das, W. Kong, A. Leach, R. Sen, and R. Yu, "Long-term forecasting with TiDE: Time-series dense encoder," *Transactions on Machine Learning Research*, 2023.

[71] R. Koekoek, P. A. Lesky, and R. F. Swarttouw, *Hypergeometric Orthogonal Polynomials and Their q-Analogues*. Springer, 2010.

[72] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *International Conference on Learning Representations*, 2022.

[73] Z. Wang, F. Kong, S. Feng, M. Wang, X. Yang, H. Zhao, D. Wang, and Y. Zhang, "Is Mamba effective for time series forecasting?" *Neurocomputing*, vol. 619, 2025.

[74] S. Huang, Z. Zhao, C. Li, and L. BAI, "TimeKAN: KAN-based frequency decomposition learning architecture for long-term time series forecasting," in *International Conference on Learning Representations*, 2025.

[75] Y. Liu, G. Qin, X. Huang, J. Wang, and M. Long, "Timer-XL: Long-context transformers for unified time series forecasting," in *International Conference on Learning Representations*, 2025.

[76] Z. Chen, T. Sha, Z. Tang, and K. Wang, "TsKAN: A transparent architecture for improving the interpretability of multivariate time series forecasting," in *ICLR 2025 Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2025.

[77] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "TimesNet: Temporal 2D-variation modeling for general time series analysis," in *International Conference on Learning Representations*, 2023.

[78] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *International Conference on Learning Representations*, 2023.

[79] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao, "MICN: Multi-scale local and global context modeling for long-term series forecasting," in *International Conference on Learning Representations*, 2023.