KAN4Drift: Are KAN Effective for Identifying and Tracking Concept Drift in Time Series?

Kunpeng Xu, Lifei Chen, Shengrui Wang
Department of Computer Science
Université de Sherbrooke
Sherbrooke, QC, Canada

Abstract

Dynamic concepts in time series are crucial for understanding complex systems such as financial markets, healthcare, and online activity logs. These concepts help reveal structures and behaviors in sequential data for better decision-making and forecasting. Existing models struggle with detecting and tracking concept drift due to limitations in interpretability and adaptability. This paper introduces Kolmogorov-Arnold Networks (KAN) into time series and proposes WormKAN, a KAN-based auto-encoder to address concept drift in co-evolving time series. WormKAN integrates the KAN-SR module, in which the encoder, decoder, and self-representation layer are built on KAN, along with a temporal constraint to capture concept transitions. These transitions, akin to passing through a "wormhole", are identified by abrupt changes in the latent space. Experiments show that KAN and KAN-based models (WormKAN) effectively segment time series into meaningful concepts, enhancing the identification and tracking of concept drifts.

1 Introduction

Time series analysis plays a crucial role in various fields such as finance, healthcare, and meteorology. Recently, deep learning models have made significant strides in forecasting tasks. Notable advancements include CARD [34], which aligns channels within transformers for accuracy, and TimeMixer [33], which employs a multiscale mixing approach. MSGNet [4] leverages inter-series correlations for multivariate forecasting, while Crossformer [45] exploits cross-dimensional dependencies. DeepTime [36] improves forecasting by integrating time indices into its structure. Additionally, large language models (LLMs) like Time-LLM [18] and UniTime [23] have been integrated into time series, opening new avenues for zero-shot and cross-domain forecasting.

Despite these advancements, a critical challenge remains – the ability to detect and track concept drift, particularly in co-evolving time series where multiple series exhibit interdependent behavior over time. Concept drift – changes in a series' statistical properties – can significantly degrade model performance. This is crucial, particularly in fields like finance, where shifts in market regimes and nonlinear relationships are just as important as prediction accuracy for decision-makers. While recent methods like CluStream [1] and DenStream [5] improve scalability, they often fail to capture temporal dependencies and dynamic transitions. Deep learning models such as OneNet [35] and FSNet [29] tackle concept drift but prioritize predictive accuracy over understanding the underlying concepts.

Kolmogorov-Arnold Networks (KAN) [25] offer a promising solution to the challenges of concept drift in time series analysis. Inspired by the Kolmogorov-Arnold representation theorem [21, 20], KAN replaces linear weights with spline-parametrized univariate functions, allowing the model to learn more complex relationships while improving both accuracy and interpretability. A notable advantage of KAN is its ability to refine spline grids, offering deeper insights into how inputs influence outputs, making the network's decision-making process more transparent. However, while KAN has

NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

demonstrated strong performance with smaller network sizes across various tasks, its effectiveness in identifying and tracking concept drift within the time series domain remains unexplored[1].

To this end, our goal is to propose a KAN-based model for addressing concept drift in time series and evaluate its effectiveness. We introduce WormKAN, a concept-aware KAN-based auto-encoder for co-evolving time series. WormKAN leverages the KAN-SR module, which consists of a KAN-based encoder and decoder, along with a self-representation layer and a temporal smoothness constraint to detect dynamic concept transitions. The key innovation lies in identifying these transitions by detecting abrupt changes in the latent space, metaphorically described as "passing through a wormhole." These transitions mark shifts to new behavior concepts, providing clear boundaries between different segments. Through experiments, we demonstrate that both the original KAN model and WormKAN effectively identify and track concept drift in time series.

2 WormKAN

We introduce WormKAN, a framework for concept-aware deep representation learning in co-evolving time sequences. The model uses Kolmogorov-Arnold Networks (KAN) in both the encoder and decoder, with a self-representation layer implemented as a 2-layer KAN. Combined with a temporal smoothness constraint, this architecture captures dynamic concepts and their transitions. To process co-evolving sequences, we segment the multivariate time series $\mathbf{S}$ using a sliding window, resulting in segments $\mathbf{W} = \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n$, where each $\mathbf{w}_i$ contains multiple time steps and channels. The framework is illustrated in Figure 1.

2.1 Kolmogorov-Arnold Networks

Kolmogorov-Arnold Networks (KAN) leverage the Kolmogorov-Arnold representation theorem [21], which states that any multivariate continuous function can be represented using univariate functions. Specifically, a function $f(x_1, x_2, \ldots, x_n)$ can be expressed as $f(x_1, x_2, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right)$, where $\varphi_{q,p}$ and Φ_q are univariate functions. KAN replaces linear weights with learnable univariate functions, often parametrized using splines, allowing complex nonlinear relationships to be modeled with fewer parameters and greater interpretability. Inputs x_p are transformed by $\varphi_{q,p}(x_p)$, aggregated, and passed through Φ_q. Stacking multiple KAN layers allows the network to capture intricate patterns while maintaining interpretability, with the deeper architecture described as $\text{KAN}(x) = (\Phi_{L-1} \circ \cdots \circ \Phi_0)(x)$, where L is the total number of layers.

2.2 KAN-Based Deep Representation Learning

WormKAN utilizes KAN to learn robust representations of co-evolving sequences through the KAN-SR module, which comprises an **Encoder**, a **Self-Representation Layer**, and a **Decoder**.

Encoder: The encoder employs a KAN to map input segments $\mathbf{W}$ into a latent representation space. Specifically, the encoder performs a nonlinear transformation $\mathbf{Z}_{\Theta_e} = \text{KAN}_{\Theta_e}(\mathbf{W})$, where KAN_{Θ_e} is the encoding function implemented using KAN, and $\mathbf{Z}_{\Theta_e}$ represents the latent representations.

Self-Representation Layer: Implemented as a 2-layer KAN (only input and output layers), this layer captures intrinsic relationships among the latent representations and enforces that each latent representation can be expressed as a combination of others: $\mathbf{Z}_{\Theta_e} = \mathbf{Z}_{\Theta_e}\mathbf{\Theta_s}$, where $\mathbf{\Theta_s} \in \mathbb{R}^{n \times n}$ is the self-representation coefficient matrix learned by the KAN. Each column $\boldsymbol{\theta}_{s,i}$ of $\mathbf{\Theta_s}$ represents the weights used to reconstruct the i-th latent representation from all latent representations. To promote sparsity in $\mathbf{\Theta_s}$ and highlight the most significant relationships, we introduce an ℓ_1 norm regularization: $\mathcal{L}_{\text{self}}(\mathbf{\Theta_s}) = \|\mathbf{\Theta_s}\|_1$.

Decoder: The decoder reconstructs the input segments from the refined latent representations using another KAN network: $\hat{\mathbf{W}}_\Theta = \text{KAN}_{\Theta_d}(\hat{\mathbf{Z}}_{\Theta_e})$, where KAN_{Θ_d} is the decoding function implemented using KAN, and $\hat{\mathbf{W}}_\Theta$ represents the reconstructed time series segments.

Temporal Smoothness Constraint: To ensure that the latent representations vary smoothly over time, we incorporate a temporal smoothness constraint on $\mathbf{\Theta_s}$. We define a difference matrix

[1]Due to the space limit, more related works in the Appendix A.

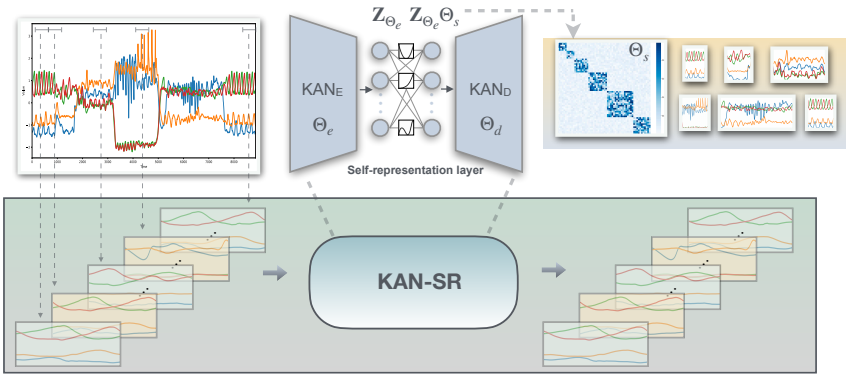

Figure 1: WormKAN. From the parameters of the self-representation layer, we construct an affinity matrix, which we use to get the concept.

$\mathbf{R} \in \mathbb{R}^{n\times(n-1)}$, where $\mathbf{R}_{i,i} = -1$ and $\mathbf{R}_{i,i+1} = 1$. The product $\mathbf{\Theta_s R}$ captures the differences between consecutive columns: $\mathbf{\Theta_s R} = [\boldsymbol{\theta}_{s,2} - \boldsymbol{\theta}_{s,1},\ \boldsymbol{\theta}_{s,3} - \boldsymbol{\theta}_{s,2},\ \ldots,\ \boldsymbol{\theta}_{s,n} - \boldsymbol{\theta}_{s,n-1}]$. The temporal smoothness constraint is defined as $\mathcal{L}_{\text{smooth}}(\mathbf{\Theta_s}) = \|\mathbf{\Theta_s R}\|_{1,2}$, where $\|\cdot\|_{1,2}$ denotes the sum of the ℓ_2 norms of the columns. This constraint penalizes large deviations, promoting smooth transitions and effectively capturing dynamic concept changes, see details in Appendix B.

Loss Function: Training involves minimizing a loss function that combines reconstruction loss, self-representation regularization, and the temporal smoothness constraint:

$$\mathcal{L}(\Theta) = \tfrac{1}{2}\|\mathbf{W} - \hat{\mathbf{W}}_\Theta\|_F^2 + \lambda_1\|\mathbf{\Theta_s}\|_1 + \lambda_2\|\mathbf{Z}_{\Theta_e} - \mathbf{Z}_{\Theta_e}\mathbf{\Theta_s}\|_F^2 + \lambda_3\|\mathbf{\Theta_s R}\|_{1,2}, \tag{1}$$

where $\mathbf{\Theta} = \{\mathbf{\Theta_e}, \mathbf{\Theta_s}, \mathbf{\Theta_d}\}$ includes all learnable parameters, with λ_1, λ_2, and λ_3 balancing the different loss components. Specifically, λ_1 promotes sparsity in the self-representation Θ_s, λ_2 preserves the self-representation property by minimizing the difference between $\mathbf{Z}_{\Theta_e}$ and $\mathbf{Z}_{\Theta_e}\mathbf{\Theta_s}$, and λ_3 ensures temporal smoothness by reducing deviations in the temporal difference matrix $\mathbf{\Theta_s R}$.

2.3 Concept Transition Detection

Once the self-representation coefficient matrix $\mathbf{\Theta_s}$ is obtained, the next step is to segment the matrix to identify concept transitions. Unlike most methods, we do not require the number of concepts to be known beforehand.

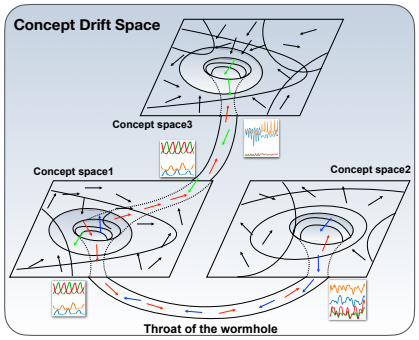

Figure 2: Concept transition through "wormhole".

Concept transitions can be viewed as passing through "wormholes," where boundaries between concepts indicate rapid shifts in behavior. These boundaries are detected by analyzing the self-representation matrix $\mathbf{\Theta_s R}$. Significant deviations in $\mathbf{\Theta_s R}$ signal concept changes, with large shifts indicating transitions between disconnected concept spaces. Within a segment, columns of $\mathbf{\Theta_s R}$ should be near zero, while deviations suggest boundaries between segments, akin to transitioning through a wormhole to a new concept space (Figure 2). To pinpoint these transitions, we compute the absolute value matrix $\mathbf{B} = |\mathbf{\Theta_s R}|$ and calculate the vector of the column-wise means. A peak-finding algorithm is then applied to the vector, where peaks correspond to concept transition points, effectively detecting dynamic changes in the time series structure (see Appendix C for details).

3 Experiments

In this section, we first evaluate the original KAN model's effectiveness in detecting concept drift, followed by experiments validating the performance of WormKAN for concept-aware deep representation learning in co-evolving time series.

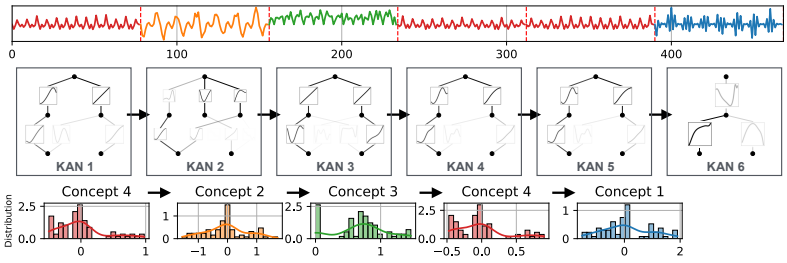

Figure 3: Training original KAN and detecting concept drift.

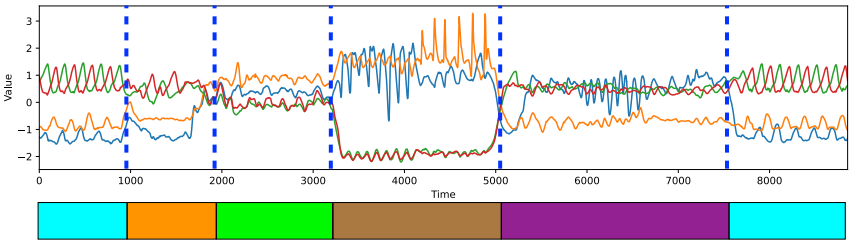

Figure 4: WormKAN identifies concepts and transitions on co-evolving motion series.

Table 1: Comparison of WormKAN and baseline models.

Dataset	Metric	StreamScope	TICC	AutoPlait	WormKAN
Motion Capture Data	F1-Score	0.84	0.48	0.87	**0.90**
	ARI	0.60	0.22	0.60	**0.65**
Stock Market Data	F1-Score	0.75	0.32	0.80	**0.86**
	ARI	0.62	0.20	0.74	**0.82**
Online Activity Logs	F1-Score	0.92	0.80	0.90	**0.94**
	ARI	0.85	0.75	0.83	**0.90**

3.1 Original KAN Model Evaluation

As illustrated in Figure 3, we use a sliding window to traverse the synthetic time series, creating input-output pairs for training the KAN model. For simplicity, two historical time steps predict the next step. Different KAN structures and activation functions represent different concepts, and observing variations in KAN can reveal concept drift. For example, the learnable activation functions in KAN1, KAN4, and KAN5 behave consistently, indicating they are within the same concept. Details on the synthetic data used, the results on real-world datasets, as well as the KAN model's parameter settings, can be found in the Appendix D.1.

3.2 WormKAN Performance and Baseline Comparison

We evaluated WormKAN using three co-evolving time series datasets: human motion, financial markets, and online activity logs (details in the Appendix D.2). WormKAN was compared against StreamScope [19], TICC [15], and AutoPlait [27], which are methods for discovering patterns in co-evolving series. The results in Table 1 show that WormKAN outperforms all baselines.

We also visualized the concept transitions detected by WormKAN on the Motion Capture Data. Figure 4 illustrates time series segments with marked transitions between different motion types, such as walking and dragging. These visualizations highlight WormKAN's ability to accurately detect boundaries between different types of activity, reinforcing its effectiveness in identifying dynamic changes in co-evolving sequences.

4 Conclusion

This work demonstrates the effectiveness of Kolmogorov-Arnold Networks (KANs) in detecting concept drift in time series. We introduced WormKAN, which outperforms baseline models in identifying concept transitions across various datasets. Our results highlight KANs' potential for robust, adaptive modeling in dynamic time series environments.

References

[1] Charu C Aggarwal, S Yu Philip, Jiawei Han, and Jianyong Wang. A framework for clustering evolving data streams. In *Proceedings 2003 VLDB conference*, pages 81–92. Elsevier, 2003.

[2] Guangji Bai, Chen Ling, and Liang Zhao. Temporal domain generalization with drift-aware dynamic neural networks. *arXiv preprint arXiv:2205.10664*, 2022.

[3] Jürgen Braun and Michael Griebel. On a constructive proof of kolmogorov's superposition theorem. *Constructive approximation*, 30:653–675, 2009.

[4] Wanlin Cai, Yuxuan Liang, Xianggen Liu, Jianshuai Feng, and Yuankai Wu. Msgnet: Learning multi-scale inter-series correlations for multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11141–11149, 2024.

[5] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006.

[6] Chia-Yuan Chang, Yu-Neng Chuang, Zhimeng Jiang, Kwei-Herng Lai, Anxiao Jiang, and Na Zou. Coda: Temporal domain generalization via concept drift simulator. *arXiv preprint arXiv:2310.01508*, 2023.

[7] Rongbo Chen, Mingxuan Sun, Kunpeng Xu, Jean-Marc Patenaude, and Shengrui Wang. Clustering-based cross-sectional regime identification for financial market forecasting. In *International Conference on Database and Expert Systems Applications*, pages 3–16. Springer, 2022.

[8] Rongbo Chen, Kunpeng Xun, Jean-Marc Patenaude, and Shengrui Wang. Dynamic cross-sectional regime identification for financial market prediction. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 295–300. IEEE, 2022.

[9] Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting–full version. *arXiv preprint arXiv:2204.13767*, 2022.

[10] Chang Dong, Liangwei Zheng, and Weitong Chen. Kolmogorov-arnold networks (kan) for time series classification and robust analysis. *arXiv preprint arXiv:2408.07314*, 2024.

[11] Wei Fan, Pengyang Wang, Dongkun Wang, Dongjie Wang, Yuanchun Zhou, and Yanjie Fu. Dish-ts: a general paradigm for alleviating distribution shift in time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7522–7529, 2023.

[12] Remi Genet and Hugo Inzirillo. A temporal kolmogorov-arnold transformer for time series forecasting. *arXiv preprint arXiv:2406.02486*, 2024.

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

[14] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[15] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 215–223, 2017.

[16] Hongbin Huang, Minghua Chen, and Xiao Qiao. Generative learning for financial time series with irregular and scale-invariant patterns. In *The Twelfth International Conference on Learning Representations*, 2023.

[17] Zhaojing Huang, Jiashuo Cui, Leping Yu, Luis Fernando Herbozo Contreras, and Omid Kavehei. Abnormality detection in time-series bio-signals using kolmogorov-arnold networks for resource-constrained devices. *medRxiv*, pages 2024–06, 2024.

[18] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[19] Koki Kawabata, Yasuko Matsubara, and Yasushi Sakurai. Automatic sequential pattern mining in data streams. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1733–1742, 2019.

[20] Andrei Nikolaevich Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*, pages 953–956. Russian Academy of Sciences, 1957.

[21] Andrei Nikolaevich Kolmogorov. *On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables*. American Mathematical Society, 1961.

[22] Rudolf Kruse, Sanaz Mostaghim, Christian Borgelt, Christian Braune, and Matthias Steinbrecher. Multi-layer perceptrons. In *Computational intelligence: a methodological introduction*, pages 53–124. Springer, 2022.

[23] Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. Unitime: A language-empowered unified model for cross-domain time series forecasting. In *Proceedings of the ACM on Web Conference 2024*, pages 4095–4106, 2024.

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

[25] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[26] Yasuko Matsubara and Yasushi Sakurai. Dynamic modeling and forecasting of time-evolving data streams. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 458–468, 2019.

[27] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. Autoplait: Automatic mining of co-evolving time sequences. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 193–204, 2014.

[28] Kohei Miyaguchi and Hiroshi Kajino. Cogra: Concept-drift-aware stochastic gradient descent for time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4594–4601, 2019.

[29] Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven CH Hoi. Learning fast and slow for online time series forecasting. *arXiv preprint arXiv:2202.11672*, 2022.

[30] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

[31] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

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

[33] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*, 2024.

[34] Xue Wang, Tian Zhou, Qingsong Wen, Jinyang Gao, Bolin Ding, and Rong Jin. Card: Channel aligned robust blend transformer for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2023.

[35] Qingsong Wen, Weiqi Chen, Liang Sun, Zhang Zhang, Liang Wang, Rong Jin, Tieniu Tan, et al. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. *Advances in Neural Information Processing Systems*, 36, 2024.

[36] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Learning deep time-index models for time series forecasting. In *International Conference on Machine Learning*, pages 37217–37237. PMLR, 2023.

[37] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

[38] Kunpeng Xu, Lifei Chen, Jean-Marc Patenaude, and Shengrui Wang. Kernel representation learning with dynamic regime discovery for time series forecasting. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 251–263. Springer, 2024.

[39] Kunpeng Xu, Lifei Chen, Jean-Marc Patenaude, and Shengrui Wang. Rhine: A regime-switching model with nonlinear representation for discovering and forecasting regimes in financial markets. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pages 526–534. SIAM, 2024.

[40] Kunpeng Xu, Lifei Chen, and Shengrui Wang. Data-driven kernel subspace clustering with local manifold preservation. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 876–884. IEEE, 2022.

[41] Kunpeng Xu, Lifei Chen, and Shengrui Wang. A multi-view kernel clustering framework for categorical sequences. *Expert Systems with Applications*, 197:116637, 2022.

[42] Kunpeng Xu, Lifei Chen, and Shengrui Wang. Are kan effective for identifying and tracking concept drift in time series? *arXiv preprint arXiv:2410.10041*, 2024.

[43] Kunpeng Xu, Lifei Chen, and Shengrui Wang. Kolmogorov-arnold networks for time series: Bridging predictive power and interpretability. *arXiv preprint arXiv:2406.02496*, 2024.

[44] En Yu, Jie Lu, Bin Zhang, and Guangquan Zhang. Online boosting adaptive learning under concept drift for multistream classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16522–16530, 2024.

[45] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2022.

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

Supplementary Materials

In this document, we have gathered all the results and discussions that, due to page limitations, were not included in the main manuscript.

Appendix

A Extended Related Work

A.1 Kolmogorov-Arnold Networks (KAN)

Kolmogorov-Arnold Networks (KAN) represent a recent innovation proposed by the MIT team [25]. KAN is a novel neural network architecture designed as a potential alternative to MLPs, inspired by the Kolmogorov-Arnold representation theorem [21, 20, 3]. Unlike MLPs, KAN applies activation functions on the connections between nodes, with these functions being capable of learning and adapting during training. By replacing linear weights with spline-parametrized univariate functions along the network edges, KAN enhances both the accuracy and interpretability of the network. A significant advantage of KAN is that a spline can be made arbitrarily accurate to a target function by refining the grid. This design not only improves network performance but also enables them to achieve comparable or superior results with smaller network sizes across various tasks [25, 32, 13].

Theoretical Foundation. The Kolmogorov-Arnold representation theorem states that any multivariate continuous function can be decomposed into a finite sum of compositions of univariate functions [21]. Formally, the theorem is expressed as:

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

where $\varphi_{q,p}$ are univariate functions that map each input variable x_p, and Φ_q are continuous functions. This allows KAN to model complex interactions in high-dimensional data through compositions of simpler univariate functions.

Network Architecture. KAN leverages the Kolmogorov-Arnold representation theorem by replacing traditional linear weights in neural networks with spline-parametrized univariate functions. Unlike conventional Multi-Layer Perceptrons (MLPs), which use fixed activation functions at the nodes, KAN applies adaptive, learnable activation functions on the edges between nodes. These functions are parametrized as B-spline curves, which adjust dynamically during training to better capture the underlying data patterns. This unique structure enables KAN to effectively capture complex nonlinear relationships within the data. Formally, a KAN layer can be defined as $\Phi = \{\varphi_{q,p}\}, \quad p = 1, 2, \ldots, n_{\text{in}}, \quad q = 1, 2, \ldots, n_{\text{out}},$, where $\varphi_{q,p}$ are parametrized functions with learnable parameters. This structure allows KAN to capture complex nonlinear relationships within the data more effectively than traditional Multi-Layer Perceptrons (MLPs).

To extend the capabilities of KAN, deeper network architectures have been developed. A deeper KAN is essentially a composition of multiple KAN layers [25], enhancing its ability to model more complex functions. The architecture of a deeper KAN can be described as:

$$KAN(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \ldots \circ \Phi_0)(x), \tag{3}$$

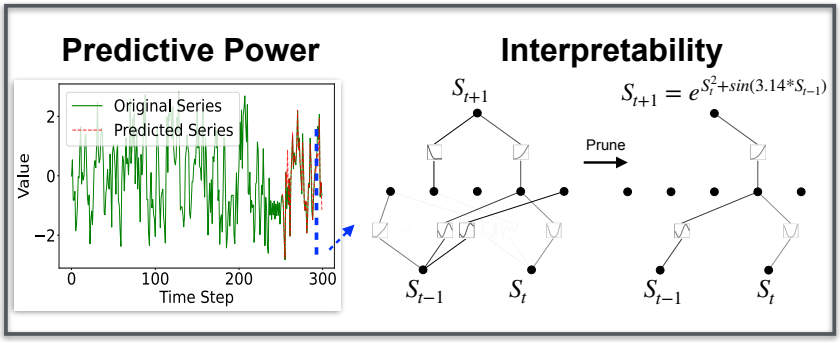

Figure 5: Predictive and Interpretable Capabilities of KAN in Time Series.

where each Φ_l represents a KAN layer. The depth of the network (i.e., the number of layers) allows it to capture more intricate patterns and dependencies in the data. Each layer l transforms the input x through a series of learnable functions $\varphi_{q,p}$, making the network highly adaptable and powerful.

Symbolic Regression for Interpretability. Symbolic regression is incorporated into KAN to enhance interpretability by fitting mathematical expressions to the learnable activation functions. This approach allows us to generate human-readable models that explain the underlying patterns in the data. In our time series application of KAN, symbolic regression decodes the nonlinear relationships between predictions and prior time steps, significantly improving interpretability, see Figure 5.

Limitation of KAN. Despite the advantages, the implementation of KAN is typically 10 times slower to train compared to MLPs with the same number of parameters. This inefficiency arises because KAN' diverse activation functions cannot fully leverage batch processing. A potential solution is to group activation functions into "heads" that share the same function, balancing between KAN and MLPs. For fast training needs, MLPs are preferable, but KAN is valuable for tasks requiring interpretability and complex relationship modeling.

KAN on Time Series. Kolmogorov-Arnold Networks (KAN) have been increasingly applied to time series analysis, demonstrating their adaptability and effectiveness in modeling complex temporal patterns. Vaca-Rubio et al. [32] explored the potential of KAN for time series analysis, showing how KAN's structure enhances interpretability while maintaining predictive power. Xu et al. [43, 42] further advanced KAN's application by bridging the gap between prediction accuracy and model interpretability, providing a balanced approach to time series forecasting. Genet and Inzirillo introduced the Temporal Kolmogorov-Arnold Network (TKAN) [13], leveraging the unique architecture of KAN to model temporal dependencies. Their work demonstrated that TKAN could outperform traditional methods in forecasting tasks. They further extended this concept with the Kolmogorov-Arnold Transformer (KAT) [12], which combines KAN with transformer-based architectures to boost the performance of time series forecasting models. Dong et al. [10] applied KAN to time series classification and robust analysis, highlighting KAN's strengths in handling noisy and incomplete data, while Huang et al. [17] focused on its use in resource-constrained environments. Their study on abnormality detection in bio-signal time series demonstrated the lightweight nature of KAN, making it suitable for edge devices in healthcare applications.

A.2 Concept Drift in Time Series

Concept drift has been a significant challenge in time series analysis, particularly in streaming data environments where the underlying data distributions may change over time [39]. Traditional models such as Hidden Markov Models (HMM) and Autoregression (AR) have been widely used but often lack adaptability in the presence of continuous data streams. Recent advancements, such as OrbitMap [26] and TKAN [43], have improved scalability but still face challenges in capturing temporal dependencies and dynamic transitions. Additionally, models like Cogra [28] and Dish-TS [11] have introduced techniques to address concept drift by incorporating stochastic gradient descent and distribution shift alleviation, respectively.

Co-Evolving Sequences and Dynamic Concept Identification. The identification of dynamic concepts in co-evolving sequences is crucial for understanding complex temporal patterns. Various methods have been proposed to segment time series data into meaningful patterns, including the use of hierarchical HMM-based models like AutoPlait [27], Kernel-based models [40, 38, 41] and Clustering-based methods [7, 8, 15].

Deep Representation Learning for Time Series. Deep learning techniques have gained traction in temporal data analysis, offering powerful tools for representation learning. OneNet [35] and FSNet [29] are notable examples that enhance time series forecasting by adapting to concept drift. However, these models primarily aim to improve predictive accuracy rather than offering insights into the concept identification process. Informer [46], TIMESNET [37], Triformer [9], and Non-stationary Transformers [24] further extend the capabilities of deep learning models in handling long sequence time-series forecasting and non-stationary behaviors in time series.

Concept-Aware Models. The idea of concept-aware models, which can detect transitions between different behaviors or patterns, has been explored in various domains. StreamScope [19] and the Generative Learning model [16] for financial time series have contributed to this area by automatically discovering patterns in co-evolving data streams. Other approaches such as online boosting adaptive learning [44], temporal domain generalization via concept drift simulation [6], and drift-aware dynamic neural networks [2] have also been proposed to handle concept drift in temporal data. However, these models do not explicitly address the interpretability of the learned representations, a gap that Wormhole seeks to fill by providing clear demarcations of concept transitions, enhancing the understanding of dynamic temporal patterns.

B Expanded Explanation of Temporal Smoothness Constraint

In the main text, we introduced the temporal smoothness constraint, which ensures that the latent representations vary smoothly over time. This constraint is critical in capturing gradual transitions and dynamic concept changes in time series data. Here, we provide a more detailed explanation of the underlying components and their formulation.

Definition of the Difference Matrix:

The key element of the temporal smoothness constraint is the difference matrix $\mathbf{R}$. This matrix captures the differences between consecutive columns of the self-representation matrix $\mathbf{\Theta_s}$, ensuring smooth transitions across time steps. Formally, $\mathbf{R} \in \mathbb{R}^{n\times(n-1)}$ is defined as:

$$\mathbf{R} = \begin{bmatrix} -1 & 0 & 0 & \cdots & 0 \\ 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Each row of $\mathbf{R}$ captures the difference between two consecutive columns in $\mathbf{\Theta_s}$. By multiplying $\mathbf{\Theta_s}$ with $\mathbf{R}$, we compute the differences between consecutive time steps as follows:

$$\mathbf{\Theta_s R} = [\boldsymbol{\theta}_{s,2} - \boldsymbol{\theta}_{s,1},\ \boldsymbol{\theta}_{s,3} - \boldsymbol{\theta}_{s,2},\ \ldots,\ \boldsymbol{\theta}_{s,n} - \boldsymbol{\theta}_{s,n-1}].$$

This constraint measures how the latent representations change over time and is used to enforce smoothness.

C Concept Identification

Once a solution to $\mathbf{\Theta_s}$ has been found, the next step is to segment the coefficient matrix $\mathbf{\Theta_s}$ to find the concept. We discuss methods for segmentation depending on amount and type of prior knowledge about the original data. Unlike most methods, we do not require the number of concepts to be known beforehand.

- If we assume that concept is drawn from a set of disconnected subspaces, i.e., $\mathbf{\Theta_s}$ is block diagonal, we can use the information encoded by $\mathbf{\Theta_s R}$ to identify the boundaries between

concepts. Ideally, columns of $\mathbf{\Theta_s R}$, such as $\boldsymbol{\theta}_{s,i} - \boldsymbol{\theta}_{s,i-1}$, that are within a segment should be close to the zero vector, as columns from the same subspace exhibit similarity. Columns of $\mathbf{\Theta_s R}$ that significantly deviate from the zero vector indicate the boundary of a segment, as this deviation suggests a lower degree of similarity. First, let $\mathbf{B} = (|\mathbf{\Theta_s R}_{ij}|)$ represent the absolute value matrix of $\mathbf{\Theta_s R}$. Then, let $\mu^{\mathbf{B}}$ be the vector of column-wise means of $\mathbf{B}$. We then apply a peak-finding algorithm to $\mu^{\mathbf{B}}$ to locate the segment boundaries. This method, which we refer to as "intrinsic segmentation," effectively identifies transitions between distinct segments within the time series.

- Alternatively, if $\mathbf{\Theta_s}$ is block diagonal and noiseless, we can analyze the eigenspectrum of $\mathbf{\Theta_s}$ to determine the number and size of each concept segment. Using the eigengap heuristic, we identify a set of explanatory eigenvalues, where the number of eigenvalues indicates the number of concept segments, and the magnitude of each eigenvalue represents the segment size. If $\mathbf{\Theta_s}$ contains noise, the eigengap heuristic may fail to accurately determine the segment sizes, but the number of concept segments will still be correct.
- If the number of segments is known beforehand or estimated through eigenspectrum analysis, we recommend using Ncut [31] to perform the segmentation. Ncut has proven to be robust in subspace segmentation tasks and is considered state-of-the-art. In cases where $\mathbf{\Theta_s}$ is not block diagonal or contains significant noise, Ncut will provide more accurate segmentation than previous methods.

D Additional Experiments Details

D.1 Detailed Evaluation of the Original KAN Model

In this subsection, we provide additional details on the evaluation of the original KAN model using both synthetic data and real-world datasets.

Detail of Datasets. We utilized a financial time series dataset for our experiments due to its complex and often unpredictable nature, characterized by high volatility and a lack of consistent periodicities. This dataset comprises daily OHCLV (open, high, close, low, volume) data spanning from January 4, 2012, to June 22, 2022[2]. Our primary objective was to predict the implied volatility of each stock. Since true volatility cannot be directly observed, we approximated it using an estimator based on realized volatility. The conventional volatility estimator is defined as: $\mathcal{V}_t = \sqrt{\sum_{t=1}^{n}(r_t)^2}$, where $r_t = \ln(c_t/c_{t-1})$ and c_t represents the closing price at time t. We also crafted a Synthetic `SyD` dataset comprising 500 simulated time series, each generated by a combination of following five nonlinear functions. The synthetic dataset allows the controllability of the structures/numbers of concepts and the availability of ground truth. To make a 780-steps long time series, we randomly choose one of the five functions ten times; every time, this function produces 78 sequential values – which are considered as a concept.

$$\begin{cases} g_1(t) = \cos\left(\frac{4\pi t}{5}\right) + \cos(\pi(t-50)) + \frac{t}{100} \\ g_2(t) = \sin\left(\frac{\pi t}{3} - 3\right) - \sin\left(\frac{\pi t}{6}\right) + \frac{t}{100} \\ g_3(t) = 1 - \sin\left(\frac{\pi t}{2} - 3\right) \times \cos\left(\frac{\pi(t-3)}{6}\right) \times \cos(\pi(t-13)) + \frac{t}{100} \\ g_4(t) = \sin\left(\frac{\pi t}{2} - 3\right) \times \cos\left(\frac{\pi(t-3)}{6}\right) \times \cos(\pi(t-13)) + \frac{t}{100} \\ g_5(t) = \cos\left(\frac{3\pi t}{5}\right) + \sin\left(\frac{2\pi t}{5} - t\right) + \frac{t}{100} \end{cases} \tag{4}$$

Experimental setup. For the Original KAN Model, we created two variants, T-KAN and MT-KAN, designed for univariate and multivariate time series, respectively. Both T-KAN and MT-KAN are implemented using a spline-based parameterization for the univariate functions. Both T-KAN and MT-KAN are implemented using a spline-based parameterization for the univariate functions. For the stock datasets, we use a 21-step window, with four such windows (totaling 84 steps) as inputs to predict the volatility for the next 21 days. We adopt the network structure from the original KAN paper, utilizing a simple two-layer (the input layer is not accounted as a layer per se) architecture with only 5 hidden neurons, i.e., [84, 5, 21]. For MT-KAN, we group 5 variables together for input,

[2] https://ca.finance.yahoo.com/

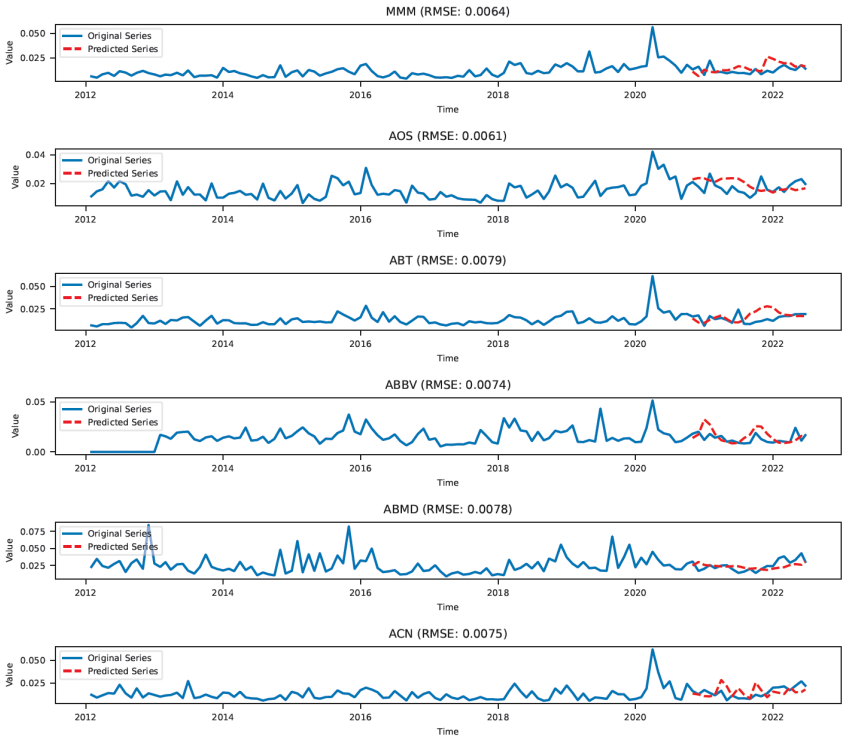

Figure 6: Predicted results using a simple T-KAN: true (blue) and forecasted (red) values for the first six series.

resulting in a structure of [84*5, 5, 21*5]. This configuration allows MT-KAN to model and predict the interactions between multiple time series more effectively, especially for stock with $m = 503$ variables. We set the training steps to 20 iterations, followed by a pruning step with a threshold of 5×10^{-2}, and then another 20 training iterations. This ensures that the models can effectively learn and adapt to the data while maintaining computational efficiency.

We compare our models against classical models, including MLP [22], RNN [30], and LSTM [14], rather than the most recent deep learning methods. Given the early stage of KAN development, it is fair to compare it as a potential alternative to MLPs and other classical methods. This strategy allows us to better evaluate the performance improvements brought by KAN relative to well-established architectures in time series forecasting. The evaluation metrics used in this experiment include Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean squared Error (RMSE).

Results on Real-world Dataset. The results are summarized in Table 2. The best results are highlighted in bold, and the suboptimal results are underlined. As shown in Table 2, both T-KAN and MT-KAN achieve competitive results. Specifically, T-KAN, with only a single layer of 5 hidden neurons, demonstrates efficiency and robustness comparable to other models that use additional hidden layers or increase the number of hidden neurons. This indicates that even with a simpler architecture, T-KAN can achieve results close to those of more complex models. MT-KAN leverages the nonlinear relationships in multivariate time series to improve prediction accuracy compared to T-KAN.

Furthermore, the efficiency of KAN-based models in terms of parameter count is evident. For instance, T-KAN has only 193 parameters, significantly fewer than the comparable LSTM model with [84, 50, 21] configuration, which has 11,671 parameters. Similarly, MT-KAN, with its [84*5, 5, 21*5] configuration, manages to outperform other models while using 2,132 parameters. This indicates that KAN-based models can achieve high accuracy with fewer parameters.

We visualized the original series (in blue) and the predicted results (in red) for the first 6 stocks (Nasdaq: MMM, AOS, ABT, ABBV, ABMD, ACN) from the dataset. As shown in Figure 6, despite

Table 2: Comparison of forecasting performance between T-KAN, MT-KAN, and baseline models.

Model	Configuration	MSE	MAE	RMSE	Parameters
MLP	[84,5,21]	0.0465	0.1774	0.2141	551
MLP	[84,50,21]	0.0002	0.0122	0.0157	5321
MLP	[84,200,21]	8.92e-5	0.0072	0.0088	21221
MLP	[84,5,5,21]	0.0504	0.1798	0.2230	581
MLP	[84,50,50,21]	0.0001	0.0103	0.0130	7871
RNN	[84,5,21]	0.0541	0.1737	0.2282	166
RNN	[84,50,21]	0.0001	0.0079	0.0098	3721
RNN	[84,200,21]	8.03e-5	0.0069	0.0083	44821
RNN	[84,5,5,21]	0.0497	0.1691	0.2185	226
RNN	[84,50,50,21]	0.0001	0.0079	0.0098	8821
LSTM	[84,5,21]	0.0132	0.0737	0.1105	286
LSTM	[84,50,21]	6.69e-5	0.0066	0.0078	11671
LSTM	[84,200,21]	6.52e-5	0.0064	0.0075	166621
LSTM	[84,5,5,21]	0.0136	0.0777	0.1124	526
LSTM	[84,50,50,21]	6.67e-5	0.0066	0.0076	32071
T-KAN	[84,5,21]	6.91e-5	0.0069	0.0078	193
MT-KAN	[84*5,5,21*5]	**6.37e-5**	**0.0062**	**0.0075**	2132

having only 2 layers (including the output layer), 5 hidden neurons, and training for only 40 steps, the predicted values align closely with the original time series.

D.2 Detailed Evaluation of WormKAN

Detail of Datasets. We evaluated WormKAN using three datasets representing different domains of co-evolving time series. The Motion Capture Streaming Data from the CMU database[3] captures various motions such as walking and dragging, making it ideal for analyzing transitions between different types of human activities. The Stock Market Data includes historical prices and financial indicators from 503 companies[4], providing a large-scale, high-dimensional dataset to test the model's performance in detecting concept changes in financial markets. Lastly, the Online Activity Logs from GoogleTrend event streams[5] include 20 time series of Google queries for a music player from 2004 to 2022, used to evaluate the model's ability to detect behavioral shifts in user interactions.

Comparison with Baseline Models. To thoroughly evaluate the effectiveness of our WormKAN model, we compared it with several baseline models. These included StreamScope[19], a scalable streaming algorithm for automatic pattern discovery in co-evolving data streams; TICC[15], which segments time series into interpretable clusters based on temporal dynamics; and AutoPlait[27], a hierarchical HMM-based model for automatic time series segmentation that identifies high-level patterns.

More detail of Table 1. Table 1 demonstrates that our model consistently outperforms the baseline models across all datasets. Notably, WormKAN achieves the highest F1-Score and ARI in both Motion Capture Data and Online Activity Logs, indicating its superior capability in detecting and segmenting concept transitions. In the Stock Market Data, despite the complexity and high dimensionality, WormKAN significantly outperforms the baselines, highlighting its robustness in handling large-scale co-evolving time series.

[3] MoCap:http://mocap.cs.cmu.edu/

[4] https://ca.finance.yahoo.com/

[5] http://www.google.com/trends/