Forecasting VIX using interpretable Kolmogorov-Arnold networks

So-Yoon Cho[a], Sungchul Lee[a], Hyun-Gyoon Kim[b,*]

[a]Department of Mathematics, Yonsei University, Seoul 03722, Republic of Korea

[b]Department of Financial Engineering, Ajou University, Gyeonggi-do 16499, Republic of Korea

Abstract

This paper presents the use of Kolmogorov-Arnold Networks (KANs) for forecasting the CBOE Volatility Index (VIX). Unlike traditional MLP-based neural networks that are often criticized for their black-box nature, KAN offers an interpretable approach via learnable spline-based activation functions and symbolification. Based on a parsimonious architecture with symbolic functions, KAN expresses a forecast of the VIX as a closed-form in terms of explanatory variables, and provide interpretable insights into key characteristics of the VIX, including mean reversion and the leverage effect. Through in-depth empirical analysis across multiple datasets and periods, we show that KANs achieve competitive forecasting performance while requiring significantly fewer parameters compared to MLP-based neural network models. Our findings demonstrate the capacity and potential of KAN as an interpretable financial time-series forecasting method.

Keywords: Kolmogorov-Arnold network; Interpretable neural network; Time-series forecasting; Volatility; VIX

1. Introduction

Deep learning based on the universal approximation theorem and implemented via multi-layer perceptrons (MLPs) has substantially improved over the past few decades and has been applied across a wide range of fields and industries. This progress can be attributed to its advantages in, for example, feature extraction and representation [1], function approximation [2], and forecasting performance [3]. However, in finance where *interpretability* is crucial for decision-making, the black-box nature of MLPs often makes decision-makers hesitant to fully adopting deep learning. In particular, for financial time-series forecasting, the lack of interpretability raises concerns about their robustness to future market dynamics.

In financial time-series data, volatility is regarded as one of the most important measures along with returns. Among the various measures of volatility, the Chicago Board Options Exchange (CBOE) Volatility Index (VIX) represents an important indicator of expected market fluctuation. The VIX, often referred to as the "fear index", reflects the market's expectations of volatility for the S&P 500 index over the next 30 days. It is calculated by combining the weighted prices of S&P 500 index call and put options across various strike prices, thereby providing a forward-looking indicator of market uncertainty. Unlike historical volatility measures, which rely entirely on past data, options-implied volatility measures such as the VIX incorporate the forward-looking views of option market participants. As a result, the VIX often provides practitioners with a more valuable perspective on market risk than historical backward-looking metrics.

Beyond reflecting the market's expectation of volatility, the VIX also serves as the underlying asset for various tradable products following the launch of VIX derivatives such as VIX futures and VIX options in the 2000s. Notably,

---

the well-documented negative correlation between volatility and stock market returns, often called the leverage effect, makes these volatility-based instruments appealing hedging tools in portfolio management. Consequently, accurate VIX forecasting is critical not only for understanding market participants' expectations but also for preparing for and effectively managing upcoming risks.

Extensive research has been conducted on modeling and forecasting the VIX, building on empirical stylized facts of volatility such as volatility clustering [4], long-range dependence [5], and mean reversion [6]. Commonly used traditional approaches in the financial time-series modeling literature include ARMA-type and GARCH-type models [7]. The ARMA model [8] integrates autoregressive (AR) and moving average (MA) components, leveraging past values and errors to effectively describe time series data. Similarly, the ARIMA model [8] extends ARMA by incorporating differencing to address non-stationarity [9, 10]. Another notable method for modeling volatility is the Heterogeneous Autoregressive (HAR) model [11] which is based on the heterogeneous market hypothesis [12]. The HAR framework presumes that future volatility depends on averages of past volatility over multiple time horizons (e.g., daily, weekly, monthly) to capture the mean-reverting behavior and long-range dependence [11]. Empirical findings suggest that the HAR model often exhibits strong forecasting performances, as demonstrated in [13, 14, 15], where its additive cascade structure aligns well with real-world market data and improves predictive accuracy.

The generalized autoregressive conditional heteroskedasticity (GARCH) models [16] are also widely employed to capture conditional variance in financial time series. However, direct application of GARCH for the VIX may not be appropriate for two main reasons. First, GARCH relies on past stock returns to infer future volatility, whereas the VIX is estimated from option prices and thus captures forward-looking expectations of market participants[1]. Second, GARCH models are typically defined under the real (physical) probability measure, whereas the VIX is extracted from option, which are priced under the risk-neutral measure. In other words, GARCH-based estimates and the VIX are not in the same measure space, and therefore cannot be directly comparable.

In recent years, deep learning approaches, particularly those based on MLPs and recurrent neural networks (RNNs), have demonstrated notable success in volatility forecasting [17]. As noted by Ge et al. [17], many deep learning methods rely on MLPs [18, 19, 20] or RNN-based Long Short-Term Memory (LSTM) networks [21, 22, 23] to predict historical or realized volatility. For VIX forecasting, Lahmiri [24] proposed the VMD-GRNN model, combining variational mode decomposition (VMD) with a general regression neural network (GRNN). Similarly, Litimi et al. [25] investigated chaotic map-based models and found that neural networks outperformed other methods in out-of-sample VIX forecasting across multiple markets. Despite these encouraging results, MLP-based models have often been criticized for their black-box nature. This presents a serious disadvantage particularly in finance as decision-makers may hesitate to trust models whose inner workings are not clearly understood. Although a model may perform well on historical data, the lack of interpretability raises concerns about its robustness to future market dynamics.

To address these limitations, we adopt Kolmogorov-Arnold Networks (KANs) [26] for VIX forecasting. Based on the Kolmogorov-Arnold representation theorem [27, 28], KANs offer a promising alternative to conventional MLPs by employing learnable spline-based univariate functions along each network edge. This structure provides both interpretability and reliable forecasting accuracy. Specifically, KANs allow researchers to visualize how individual splines and activation functions evolve during training and thereby uncover intricate data patterns. Additionally, KANs can approximate complex target functions with potentially fewer parameters than typical MLP-based deep learning models, making them a more efficient approach to VIX forecasting.

Despite being introduced only in 2024, KANs have already garnered considerable attention across a range of time-series forecasting tasks [29, 30, 31]. In finance, KANs have been utilized to predict, for example, implied volatility [32], cryptocurrency absolute returns [33], and trading volume [34, 33, 35], and have demonstrated competitive or even superior performance compared to conventional MLP or LSTM models. Most existing studies, however, integrate KAN modules into more elaborate architectures (e.g., RNNs, convolutional networks, or gated residual networks) to enhance predictive power. Although these designs can improve accuracy, they also compromise KAN's key strength:

---

[1]Although both GARCH and ARMA-type models depend on historical data, ARMA-type approaches can directly model the VIX itself by using its past values (i.e., past VIX data) to forecast future VIX. By contrast, GARCH's dependence on past stock returns does not capture the volatility implied by option data.

interpretability. In contrast, our aim is to overcome the black-box limitations of MLPs and produce interpretable financial time-series forecasts by employing a pure KAN structure. Rather than blending KANs with complicated modules that dilute its transparency, we utilize KAN's inherent capacity for visual identification and mathematical characterization of time-series features in the data.

The remainder of this paper is organized as follows. Section 2 reviews the Kolmogorov-Arnold representation theorem and describes the structure and training procedure of KANs. Section 3 analyzes empirical results based on the interpretability of KAN and evaluates its forecasting performances through comparisons with traditional time-series models and MLP-based neural networks. Section 4 concludes the paper.

2. Kolmogorov-Arnold Networks: Representation, Structure, and Training

2.1. Kolmogorov-Arnold Representation Theorem

The Kolmogorov-Arnold representation theorem [27, 28] provides a foundational framework for expressing complex multivariate functions through compositions of univariate functions and additive operations. Formally, the theorem states that any continuous multivariable function $f : [0, 1]^n \to \mathbb{R}$ can be represented as a finite sum of compositions of continuous univariate functions:

$$f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right), \tag{2.1}$$

where $\phi_{q,p} : [0, 1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$ are continuous univariate functions. Note that these functions may exhibit non-smooth or fractal-like behavior.

This theorem is often regarded as highly promising in the field of deep learning, as it reduces the challenge of approximating multivariable functions to the optimization of univariate functions. Nevertheless, its original formulation has certain limitations, notably a shallow two-layer structure and restricted flexibility in the intermediate nodes.

2.2. Kolmogorov-Arnold Networks

KANs are neural networks inspired by the Kolmogorov-Arnold representation theorem. They address the constraints of the original design−which is limited to two layers and $2n + 1$ nodes in the middle layer− by stacking multiple KAN layers, each with an arbitrary number of nodes. Formally, a single KAN layer is defined as

$$\Phi = \{\phi_{q,p}\}, \quad q = 1, 2, \cdots, n_{\text{in}}, \quad p = 1, 2, \cdots, n_{\text{out}},$$

where the functions $\phi_{q,p}$ are univariate trainable activation functions, and n_{in} and n_{out} denote the dimensions of its inputs and outputs, respectively. Given an n_{in}-dimensional input vector $\mathbf{x} = (x_1, x_2, \cdots, x_{n_{in}})$, the output of this KAN layer is defined by

$$\Phi(\mathbf{x}) = \left(\sum_{q=1}^{n_{in}} \phi_{q,1}\left(x_q\right), \sum_{q=1}^{n_{in}} \phi_{q,2}\left(x_q\right), \cdots, \sum_{q=1}^{n_{in}} \phi_{q,n_{out}}\left(x_q\right) \right) \in \mathbb{R}^{n_{out}}.$$

Then a generalized KAN with L layers is expressed as:

$$KAN(\mathbf{x}) = \left(\Phi^{(L-1)} \circ \cdots \circ \Phi^{(0)} \right)(\mathbf{x}), \tag{2.2}$$

where each $\Phi^{(l)} := \{\phi_{q,p}^{(l)}\}$ represents the l-th KAN layer. Notably, the structure derived from the Kolmogorov-Arnold representation theorem can be seen as a special case of a KAN with two layers and $2n + 1$ nodes in the middle layer. By allowing arbitrary depths and widths, KANs substantially extend this framework's flexibility.

Activation functions of KANs. Conventional MLP-based neural networks typically employ a single activation function (e.g., ReLU, Sigmoid) shared across all nodes after a global linear transformation. In contrast, KANs introduce residual activation functions that combine a smooth basis function $b(x)$ with a B-spline component. Specifically, each node's activation function in KAN is given by

$$\phi(x) = w_b b(x) + w_s \text{spline}(x), \tag{2.3}$$

Algorithm 1: Training procedure of KANs

Input: Explanatory variables $x_{<t}$ for forecasting V_t
1. Train $KAN(x_{<t})$ to predict V_t using the loss function (2.5).
2. Prune low-importance edges and nodes.
3. Symbolify the trained B-spline-based activation functions, and incorporate affine transforms.
4. Fine-tune the affine parameters added in Step 3.
return *The symbolic formula for $\hat{V}_t$*

---

where w_b and w_s are trainable scaling coefficients. The basis function $b(x)$ is typically chosen to be the SiLU function,

$$b(x) = \frac{x}{1 + e^{-x}},$$

and spline(x) is a linear combination of B-spline basis functions given by

$$\text{spline}(x) = \sum_i c_i B_i(x),$$

where the coefficients c_i are trainable parameters, and each B-spline basis function B_i is of order k with grid G. In practice, these additional parameters $\{w_b, w_s, c_i\}$ are learned through standard backpropagation, providing robust flexibility while preserving interpretability. Moreover, this activation function design facilitates locally precise approximations of complex relationships and enhances the model's capacity for capturing highly nonlinear behavior.

Training KANs. While the overall training procedure of KANs closely resembles that of MLP-based networks, it involves additional steps, such as node pruning and symbolification, to enhance interpretability. Specifically, we consider the VIX at time t, denoted by V_t, as our target variable, and seek to predict it from explanatory variables $x_{<t}$ observed prior to time t. In this setting, the predictive loss $\mathcal{L}_{\text{pred}}$ is defined as

$$\mathcal{L}_{\text{pred}} = \sum_t \left(V_t - KAN\left(x_{<t} \right) \right)^2, \tag{2.4}$$

in the same manner to the mean-squared error used in conventional MLP-based neural network training.

According to the design of KANs, the number of nodes and layers can be arbitrarily large; however, in many scientific and real-world tasks, most functions often have sparse compositional structures [26]. Motivated by the importance of interpretability in such contexts, KANs employ L_1-norm and entropy regularizations, both defined at the layer level, to facilitate sparse representations. Specifically, the L_1-norm of a KAN layer Φ is defined by $\|\Phi\|_1 = \sum_{i,j} \|\phi_{ij}\|_1$ where $\|\phi_{ij}\|_1$ denotes the average magnitude of the activation function $\phi_{q,p}$ and is computed as $\|\phi_{q,p}\|_1 = \frac{1}{N_p} \sum_{s=1}^{N_p} \left| \phi_{q,p}(x^{(s)}) \right|$ for N_p input data points $x^{(s)}$. The entropy of Φ is given by $S(\Phi) = -\sum_{q,p} \frac{\|\phi_{q,p}\|_1}{\|\Phi\|_1} \log \left(\frac{\|\phi_{q,p}\|_1}{\|\Phi\|_1} \right)$. Hence, the total training loss is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} + \lambda \left(\mu_1 \sum_{l=0}^{L-1} \left\| \Phi^{(l)} \right\|_1 + \mu_2 \sum_{l=0}^{L-1} S\left(\Phi^{(l)} \right) \right), \tag{2.5}$$

where λ controls the overall regularization strength, and μ_1, μ_2 determine the weights for the L_1-norm and entropy terms, respectively.

Once trained with these regularization terms, nodes whose incoming and outgoing values remain below a certain threshold are considered unimportant and are pruned. The pruned KAN then undergoes a symbolification step, in which B-spline-based activation functions are replaced by elementary (e.g., polynomial, exponential, sine) or domain-specific functions (e.g., Gaussian, Bessel) that closely approximate the learned activation functions. At this step, one can visually identify the learned activation function, and a user with domain knowledge can manually replace them with a symbolic function. Alternatively, a symbolic function can be automatically selected from a prespecified set of candidate functions. Finally, an additional affine transformation is applied and these affine parameters are fine-tuned through further training. The overall training procedure is summarized in Algorithm 1. For further details on the theoretical backgrounds and implementation, we refer to [26].

3. VIX Forecasting using KAN

3.1. Data

In this paper, we focus on forecasting the CBOE VIX using KANs. Since the VIX calculation method was revised in September 2003[2], we employ the VIX data from January 2004 to December 2023 to ensure intertemporal consistency. Alongside the VIX data, we also utilize the S&P 500 daily excess returns over the same period, using the 1-month U.S. Treasury bill yield obtained from the Federal Reserve Economic Data (FRED) as the risk-free rate. To align the scale with the VIX, the S&P 500 daily excess returns are multiplied by 100.

We construct three datasets from historical VIX data to use as explanatory variables, each designed to examine whether particular characteristics of volatility can be observed. The first dataset uses five consecutive lagged values, $\{V_{t-1}, V_{t-2}, V_{t-3}, V_{t-4}, V_{t-5}\}$, to predict the next VIX value V_t. By focusing on the preceding five trading days, this dataset highlights short-term momentum. The second dataset consists of non-uniform lags, $\{V_{t-1}, V_{t-5}, V_{t-10}, V_{t-21}\}$, to investigate whether the short- (one day), medium- (one or two weeks), or long- (one month) range dependencies remain and provide predictability. Finally, the third dataset comprises the previous VIX value, V_{t-1}, with weekly average V^w, monthly average V^w, and quarterly average V^q, defined by

$$V_{t-1}^w = \frac{1}{5} \sum_{i=1}^{5} V_{t-i}, \quad V_{t-1}^m = \frac{1}{21} \sum_{i=1}^{21} V_{t-i}, \quad V_{t-1}^q = \frac{1}{63} \sum_{i=1}^{63} V_{t-i}, \tag{3.1}$$

respectively. We use this aggregated-statistics approach to determine whether the mean-reverting property of volatility across multiple horizons can be captured and, if so, to clarify the effects of short-, medium-, long-, and very long-term averages on volatility dynamics. For convenience, we denote these datasets as *Dataset 1*, *Dataset 2*, and *Dataset 3*, respectively, in the remainder of this paper.

To ensure the robustness of our experiments, we vary the proportions of training, validation, and test sets across three different configurations. *Period 1* allocates the data from 2004 to 2015 for training, 2016–2017 for validation, and 2018–2023 for testing, corresponding to a 6:1:3 ratio. *Period 2* applies a 7:1:2 ratio, and *Period 3* uses 8:1:1. Each of the three datasets is tested over all three periods, resulting in nine total experimental setups. We denote a setup with *Dataset i* and *Period j* by D_iP_j, where $i \in \{1, 2, 3\}$ and $j \in \{1, 2, 3\}$.

3.2. Empirical Results

3.2.1. Experimental Setup

Before examining our empirical findings in detail, we describe the hyperparameters used for KAN in this study. All experiments, across every dataset and period, are conducted with the same hyperparameter settings. Specifically, we use a two-layer KAN with two hidden nodes in the hidden layer. We adopt this parsimonious configuration because increasing the depth or the number of hidden nodes does not lead to any significant improvement or even any noticeable difference after the pruning and symbolification steps. An analysis of the number of hidden nodes and layers is provided briefly in Appendix A. The activation functions include B-spline components of order $k = 3$ with a grid size of 3. For optimization, we employ the L-BFGS algorithm with a learning rate of 0.04. If the validation loss fails to improve for five consecutive epochs, we reduce the learning rate by a factor of 0.1; if it does not improve for ten consecutive epochs, we stop training activation functions and prune unimportant edges and nodes. We then replace the activation functions with symbolic functions that incorporate learnable affine parameters. Finally, these affine parameters are trained for 30 epochs with a learning rate of 0.0004. The regularization coefficient λ for sparsification is set to zero during training, as unimportant edges and nodes are pruned effectively even without the help of this regularization.

For comparison, we employ both statistical time-series models and MLP-based neural networks as benchmark methods. The statistical models consist of a naive forward-filling method, ARMA and ARIMA approaches, and two

---

[2]The CBOE revised the calculation method for the VIX by switching its underlying asset from OEX to SPX and incorporating out-of-the-money options [36].

variants of the HAR model. The naive forward-filling method predicts V_t simply by using the previous value (i.e., $\hat{V}_t = V_{t-1}$), motivated by the high persistence [37] and clustering effects [4] observed in volatility. Despite its simplicity, this naive approach can show competitive performance in many scenarios. The ARMA model combines AR and MA components to describe time-series behavior based on past values and errors, and the ARIMA model extends ARMA by applying differencing to model changes in the data rather than levels. To choose the AR parameter p, MA parameter q, and the order of differencing d, we conduct a grid search over $p, q \in \{0, 1, 2, 3, 4, 5\}$ and $d \in \{0, 1, 2\}$. The optimal values of p and q are selected based on the Akaike Information Criterion (AIC), and d is determined using the KPSS test [38]. Through this process, we found that the ARMA(1,1) model and the ARIMA(1,1,1) model provides the optimal fit for our in-sample dataset. We also compare with the HAR model, which utilizes averages of past volatility over multiple horizons. Specifically, we employ two HAR models. One of these two model uses the previous VIX data, and weekly and monthly averages. This model is expected to capture the predictability induced by the short-, medium-, and long-term averages as in [11]. The other model additionally includes a quarterly average to reflect the effect of very-long-term average. These two HAR models are denoted by HAR(3) and HAR(4), respectively, and given by

$$\hat{V}_t = c + \beta_1 V_{t-1} + \beta_2 V_{t-1}^w + \beta_3 V_{t-1}^m + \beta_4 V_{t-1}^q, \tag{3.2}$$

where $\beta_4 = 0$ in HAR(3), and $\beta_4 \neq 0$ in HAR(4). Note that, to predict V_t, the ARMA and ARIMA models employ *Dataset 1*, and the HAR models use *Dataset 3*. Thus, it is worthwhile to compare these models to KAN with *Dataset 1* and *Dataset 3*.

On the other hand, we also implement a MLP and a LSTM network as benchmarks. An MLP, which can be regarded as a nonlinear regressor [39], is an artificial neural network composed of input, hidden, and output layers. In our experiments, the MLPs are built with three hidden layers, each containing 100 units, where each linear transformation is followed by a LeakyReLU activation function. We train this model using the Adam optimizer with a learning rate of 1×10^{-3}. Meanwhile, the LSTM network, an extension of the RNN architecture, incorporates memory cells to learn long-term dependencies, which makes it more suitable for complex time-series forecasting. The LSTM models in our experiments consist of two stacked LSTM layers, each with 70 hidden units. The final hidden states are passed through a two-layer fully connected network with a Sigmoid activation. We then scale its output by multiplying by 100. This design allows the LSTM to handle intricate temporal features that may be difficult for simpler MLP models to capture.

Lastly, all the experiment in Section 3 are performed using out-of-sample datasets.

3.2.2. Analysis of KAN forecasting results

We begin by investigating the results of the trained KANs and the effects of symbolification without yet comparing them to other benchmark models.

Figure 1 illustrates the results of training KAN on three different datasets under *Period 3*. The top row shows the learned activation functions before symbolification (i.e., the functions are still expressed as B-spline-based activation functions), while the bottom row illustrates the activation functions after symbolification (i.e., the functions have been replaced by symbolic functions). From left to right, each column corresponds to *Dataset 1*, *Dataset 2*, and *Dataset 3*. Since the outcomes for *Periods 1* and *2* are visually similar, they are presented in Appendix B for brevity.

A notable observation from the top-row figures is that, in all cases including other periods in Appendix B, KAN's B-spline-based activation functions are trained to exhibit nearly linear shapes. Recall that KAN's activation functions (the black curves in the top-row figures) are designed to approximate arbitrary univariate continuous functions via splines. However, under the configurations and explanatory variables used in this study, linear forms consistently emerge. This suggests that, for predicting the VIX, every explanatory variable we tested primarily exerts *linear* influence on forecast performance. It also aligns with prior research indicating that simpler and more parsimonious models often suffice to capture essential characteristics of noisy financial time-series data [40, 8, 41]. Moverover, this observation indirectly supports the linearity assumptions commonly made by statistical time-series models such as ARMA, ARIMA, and HAR, while simultaneously highlighting KAN's distinctive capability. Whereas these statistical models presuppose linearity, KAN identifies such structures directly from data, imposing no prior assumptions on its functional form. This advantage allows researchers to uncover data-driven insights.

6

Table 1: Comparison of R^2 of KAN results before and after symbolification across different *Dataset*s and *Period*s.

Dataset	Period	KAN before symbolification	KAN after symbolification
	1	0.9302	**0.9306**
1	2	0.9285	**0.9286**
	3	0.9398	**0.9400**
	1	0.9287	**0.9291**
2	2	0.9262	**0.9264**
	3	**0.9397**	0.9392
	1	0.9303	**0.9307**
3	2	0.9285	**0.9286**
	3	0.9403	**0.9403**

Another important observation is the automatic pruning of less influential edges and nodes. In these figures, edges are drawn more vividly when their impact is high and faintly if their impact is low. Nodes and edges with importance below a certain threshold are pruned and removed from the final network. Notably, in *Dataset 2* and *Dataset 3*, the network prunes unimportant nodes in the middle layer so that only a single influential node remains; similarly, for V_{t-21} in *Dataset 2* and V_q in *Dataset 3*, the pruned edges starting from these nodes make them disconnected from the prediction results. This automatic selection of meaningful edges and nodes highlights KAN's capacity to return a sparse and parsimonious representation of the data.

Finally, based on the visual identification of these activation functions, we specify a set of candidate symbolic functions: $y = x$ and $y = 0$. During symbolification, the trained activation functions are replaced by the closest symbolic function, followed by the incorporation of affine transformations. These affine parameters are fine-tuned afterward. The bottom-row figures (in red) depict these final activation functions, demonstrating how symbolification simplifies the learned representation.

We next assess the effects of symbolification on forecasting performance, as given in Table 1. The table compares the R^2 values of KAN forecasts before and after symbolification for each *Dataset* and *Period*. Notably, for all cases except D_2P_3, the R^2 value improves slightly once the B-spline-based activation functions are replaced by simpler symbolic functions. This finding reinforces the notion that a more parsimonious model can work effectively for financial time-series data that exhibit a low signal-to-noise ratio. In other words, activation functions may have been subtly overfitted and deviating from a linear shape during training, but symbolification corrects these deviations and enhances predictability in most cases.

We also note that *Dataset 3* consistently yields the highest R^2 across all three periods, with *Dataset 1* a close second and *Dataset 2* somewhat lower. This result suggests that recent VIX values and weekly, monthly, and quarterly averages (*Dataset 3*) provide more stable and robust predictive information than non-consecutive lagged values (*Dataset 2*). By aggregating VIX data over multiple horizons, *Dataset 3* partially averages out daily noise, making its signal clearer and capturing longer-range dynamics more reliably. We further confirm these findings with additional benchmark models and performance metrics in Section 3.2.5.

The most significant aspect of symbolification is that it provides a closed-form expression composed of symbolic functions, thereby enabling interpretable forecasting. Table 2 presents the closed-form symbolic expressions of $\hat{V}_t$ across different *Dataset*s and *Period*s, which correspond to the bottom row plots in Figure 1. In each case, the B-spline-based activation functions are replaced by simpler functional forms, and the corresponding coefficients for each explanatory variable are given explicitly.

A consistent pattern emerges across all cases: V_{t-1} exerts a dominant influence, typically with a coefficient exceeding 0.8. This overall observation indicates that KAN successfully captures the well-known high persistence and clustering effect of the VIX. Turning specifically to *Dataset 1*, the coefficient of V_{t-5} consistently shows the second-largest magnitude, whereas V_{t-2}, V_{t-3}, and V_{t-4} appear less consistent and persistent. Nonetheless, the coefficients of these three intermediate variables retain the same signs, and only V_{t-4} affects the prediction negatively. In *Dataset*

Table 2: Closed-form expressions $\hat{V}_t$ across different *Dataset*s and *Period*s

Dataset	Period	Closed-form expression
1	1	$\hat{V}_t = 0.8482 \cdot V_{t-1} + 0.0536 \cdot V_{t-2} + 0.0421 \cdot V_{t-3} - 0.0447 \cdot V_{t-4} + 0.0816 \cdot V_{t-5} + 0.3779$
	2	$\hat{V}_t = 0.8785 \cdot V_{t-1} + 0.0160 \cdot V_{t-2} + 0.0561 \cdot V_{t-3} - 0.0321 \cdot V_{t-4} + 0.0865 \cdot V_{t-5} + 0.2345$
	3	$\hat{V}_t = 0.8584 \cdot V_{t-1} + 0.0206 \cdot V_{t-2} + 0.0632 \cdot V_{t-3} - 0.0374 \cdot V_{t-4} + 0.0801 \cdot V_{t-5} + 0.2793$
2	1	$\hat{V}_t = 0.8661 \cdot V_{t-1} + 0.0634 \cdot V_{t-5} + 0.0655 \cdot V_{t-10} + 0.0712$
	2	$\hat{V}_t = 0.8800 \cdot V_{t-1} + 0.0733 \cdot V_{t-5} + 0.0359 \cdot V_{t-10} + 0.1950$
	3	$\hat{V}_t = 0.8840 \cdot V_{t-1} + 0.0742 \cdot V_{t-5} + 0.0344 \cdot V_{t-10} + 0.1099$
3	1	$\hat{V}_t = 0.8297 \cdot V_{t-1} + 0.1477 \cdot V_{\mathrm{w}} + 0.4756$
	2	$\hat{V}_t = 0.8189 \cdot V_{t-1} + 0.1660 \cdot V_{\mathrm{w}} + 0.2749$
	3	$\hat{V}_t = 0.8255 \cdot V_{t-1} + 0.1292 \cdot V_{\mathrm{w}} + 0.0377 \cdot V_{\mathrm{m}} + 0.1025$

(a) D_1P_3 before symbolification

(b) D_2P_3 before symbolification

(c) D_3P_3 before symbolification

(d) D_1P_3 after symbolification

(e) D_2P_3 after symbolification
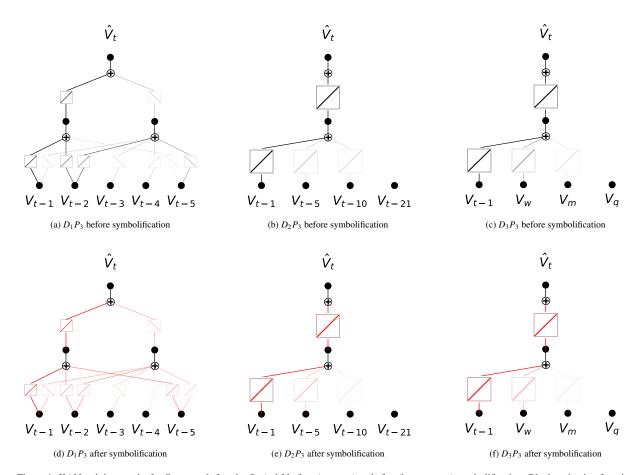
(f) D_3P_3 after symbolification

Figure 1: KAN training results for *Datasets 1–3* under *Period 3* before (top row) and after (bottom row) symbolification. Black activation functions indicate the trained B-spline-based structure, whereas red activation functions depict the replaced symbolic functions. Vivid edges represent stronger effects, while fainter edges mean weaker ones.

2, the term V_{t-21} is pruned in every period. Meanwhile, in *Dataset 3*, the very-long-term average V_q is pruned in all *Period*s, and even the long-term average V_m is pruned in certain periods. These observations suggest that long- or very-long-range dependence may contribute less to forecasting the VIX than short- and medium-range variables.

It is also noteworthy that in *Dataset 3*, the contribution of V_{t-1} may appear smaller than in the other datasets. However, this is because V_{t-1} is partially embedded in the weekly or monthly averages. For instance, in *Period 1* of *Dataset 3*, the coefficient of the weekly average V_w is estimated at 0.1477. Thus, the total contribution from V_{t-1} is calculated as $0.8290 + (0.1472/5) = 0.8584$, which is comparable with the results of *Dataset 1*. Additionally, the fact that the coefficient of V_w in *Dataset 3* consistently exceeds 0.1 and surpasses all other variables except V_{t-1} demonstrates that aggregated averages provide significant predictability.

Lastly, we empirically demonstrate that the VIX can be modeled with mean-reverting properties using the closed-form expression derived from *Dataset 3*. Let us first examine *Period 1* with *Dataset 3*. Here, we denote the difference between the predicted and previous values, $\hat{V}_t - V_{t-1}$, as ΔV_t which is given by

$$
\begin{aligned}
\Delta V_t &= (0.8290V_{t-1} + 0.1472V_w + 0.4866) - V_{t-1} \\
&= 0.1472\,(V_w - V_{t-1}) + (-0.0238V_{t-1} + 0.4866)
\end{aligned}
\tag{3.3}
$$

The historical average of the VIX during the training period of *Period 1* (2004–2015) is estimated to be 20.34, and the mean of the last term $(-0.0238V_{t-1} + 0.4866)$ is calculated to be 0.0025. Similarly, for *Period 2* and *Period 3*, the mean values of the corresponding terms in the equations for ΔV_t are calculated as -0.0164 and -0.0177, respectively, indicating that the the value of these terms is nearly zero. Based on our empirical observations, we can propose the following model of ΔV_t capturing the mean-reverting property of the VIX as

$$
\Delta V_t = \kappa(\theta - V_{t-1}) + \epsilon_t,
\tag{3.4}
$$

where κ and θ are constants. Here, ϵ_t may or may not depend on V_{t-1}, and is assumed to have zero-mean. A stochastic process whose dynamics is given as (3.4) is known to have a mean-reverting property, and κ and θ represent the mean-reverting speed and the mean-level, respectively (e.g., see [42, 43]).

If we let $(-0.0238V_{t-1} + 0.4866)$ in *Period 1* be ϵ_t, then the equation (3.3) becomes identical to (3.4). Consequently, the closed-form expressions derived by KAN with *Dataset 3* can be interpreted as representing a typical mean-reverting process reverting to its medium- or long-term average. Note that this interpretation arises from KAN's ability to provide symbolic closed-form expressions, while such interpretability is generally unattainable in black-box MLP-based models.

3.2.3. Statistical assessment of KAN as a forecasting model

To examine the statistical properties and significance of our fitted prediction model, we employ two tests commonly used in financial time-series forecasting−the Mincer–Zarnowitz test [44] and the Durbin–Watson test [45, 46]−and present the results in Table 3. The Mincer–Zarnowitz approach examines whether a forecast is unbiased by regressing

$$
V_t = \alpha + \beta\hat{V}_t + u_t,
\tag{3.5}
$$

for residual u_t, and constant α and β, and testing whether $\alpha = 0$ and $\beta = 1$ [44]. In our setting, the F-statistics remain well below the usual critical thresholds, and p-values consistently exceed conventional significance levels (e.g., 0.10), indicating insufficient evidence to reject the null hypothesis $(\alpha, \beta) = (0, 1)$. In other words, there is no strong indication that $\alpha \neq 0$ or $\beta \neq 1$. Meanwhile, the Durbin–Watson statistic ranges between approximately 1.799 and 2.180 for all datasets and forecast periods. This statistic typically falls between 0 and 4, with values near 2 indicating no first-order autocorrelation. Empirical studies often use 1.5–2.5 as a practical threshold range for concluding that no substantial first-order serial correlation remains. In our results, the statistic falls comfortably within that interval, suggesting no problematic first-order autocorrelation in the residuals. Overall, these results suggest that the model's forecasts do not deviate significantly from unbiasedness, nor do the residuals exhibit problematic autocorrelation.

Table 3: Results of the Mincer–Zarnowitz test and Durbin–Watson test are presented. The Mincer–Zarnowitz columns display the F-statistic and its p-value under the null hypothesis $(\alpha, \beta) = (0, 1)$. The Durbin–Watson statistic within [1.5, 2.5] typically indicates no significant first-order residual autocorrelation.

| Dataset | Period | Mincer–Zarnowitz | | Durbin–Watson statistic |
		F-statistic	p-value	
1	1	1.9642	0.1406	2.098
	2	1.6492	0.1927	2.178
	3	0.1914	0.8258	1.799
2	1	1.7725	0.1703	2.167
	2	1.5861	0.2052	2.157
	3	1.1493	0.3177	1.824
3	1	0.1938	0.8238	2.156
	2	1.9326	0.1453	2.180
	3	0.1279	0.8800	1.806

3.2.4. Capturing the leverage effect

Having established that KAN effectively replicates volatility clustering and mean-reverting behavior, we next examine its capacity to capture the leverage effect. In particular, we aim to determine whether excess returns of the underlying asset (S&P 500) help explain remaining variability in VIX forecasts once past VIX data have been accounted for. To this end, we consider the augmented model:

$$V_t = \tilde{V}_t + e_t, \quad \text{where} \quad \tilde{V}_t := KAN(\hat{V}_t, R^e_{t-1}). \tag{3.6}$$

Here, $\hat{V}_t$ is the KAN-based forecast derived exclusively from past VIX observations using *Datasets 1–3*, R^e_{t-1} is the excess return of the underlying asset at $t - 1$, and e_t is a residual term associated with $\tilde{V}_t$.

Under the hypothesis that the KAN models in Section 3.2.2 are trained optimally on their given explanatory variables (i.e., $\hat{V}_t$ represents the best possible prediction from historical VIX alone), we attempt to explain the remaining variability stemming from the stock market via R^e_{t-1}. If this hypothesis holds, then the KAN model in (3.6) would return $\hat{V}_t$ unchanged and the effect of the excess return would be expressed through a suitable function during training. In other words, the VIX may be written as

$$V_t = \tilde{V}_t + e_t = \left(\hat{V}_t + \psi(R^e_{t-1})\right) + e_t, \tag{3.7}$$

where ψ is a function KAN learns for excess returns, and e_t remains the irreducible noise. From the perspective of the Mincer-Zarnowitz test results in Section 3.2.3, which did not reject the null hypothesis $\alpha = 0$ and $\beta = 1$ in (3.5), this expression suggests that the error term u_t in (3.5) can be decomposed into $\psi(R^e_{t-1})$ and the irreducible error e_t.

We design a straightforward KAN to explore how R^e_{t-1} affects the remaining forecast error. Specifically, we feed both $\hat{V}_t$ and R^e_{t-1} into a one-layer KAN, where each passes through its own B-spline-based activation function to construct $\tilde{V}_t$. Figure 2 illustrates the learned activation functions for all three *Datasets* across three *Periods*. Notably, $\hat{V}_t$ consistently exhibits an almost-linear shape whereas the activation functions for R^e_{t-1} tend to decrease in every case except for *Dataset 2* in *Period 1*. This downward slope is a hallmark of the leverage effect as lower excess returns correspond to a higher predicted VIX. Occasionally, for extremely large positive returns, the functions flatten or even curve upward. This behavior implies that the model predicts high volatility following such extreme returns, which seems a reasonable outcome given the typical market reactions to drastic price movements.

Table 4 reports the R^2 values for $\tilde{V}_t$ and their improvements over $\hat{V}_t$. In every case, incorporating R^e_{t-1} robustly enhances forecasting accuracy across all *Datasets* and *Periods*. These improvements suggest that latent information in excess return data helps KAN to refine VIX estimates. To further quantify the contribution of excess returns, we replace the activation functions for R^e_{t-1} with a simple linear function $y = -x$. The resulting closed-form expressions
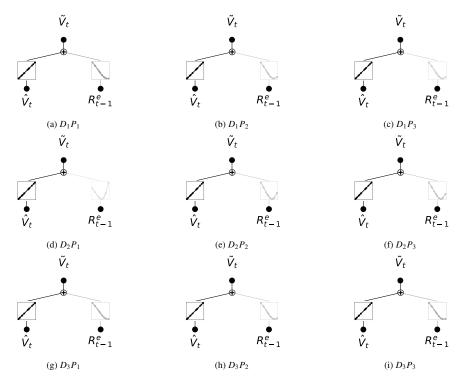
Figure 2: Training results of $KAN(\hat{V}_t, R^e_{t-1})$ across *Dataset*s and *Period*s. To clearly see the activation functions, the contrast of the figures is increased by three times compared to Figure 1.

are presented in Table 4. Although we acknowledge that the trained activation functions for R^e_{t-1} cannot be strictly regarded as $y = -x$ and that this replacement may introduce errors, our objective is to provide a quantitative assessment of how R^e_{t-1} influences $\tilde{V}_t$. As shown in the table, the coefficient of $\hat{V}_t$ remains close to one in all cases, aligning with our theoretical expectations in (3.7). Additionally, the coefficients for R^e_{t-1} range from -0.02 to -0.07 in most cases except for *Dataset 2* in *Period 1*. This result suggests a nonnegligible negative impact of R^e_{t-1} on forecasting the VIX, and further demonstrates KAN's ability to capture the leverage effect effectively.

The anomalous positive coefficient of R^e_{t-1} for *Dataset 2* in *Period 1* is attributed to the corresponding activation function before symbolification (see Figure 2). In the figure, the KAN is trained to predict substantially higher V_t values for large positive R^e_{t-1}. As a result, the learned activation function becomes nonlinear and does not align with

Table 4: R^2 values of $\tilde{V}_t$ and their improvement over $\hat{V}_t$ in parenthesis, and closed-form expressions after symbolification.

Dataset	Period	$\tilde{V}_t$ $(\tilde{V}_t - \hat{V}_t)$	Closed-form expression after symbolification
1	1	0.9314 (0.0007)	$\tilde{V}_t = 0.9974 \cdot \hat{V}_t - 0.0662 \cdot R^e_{t-1} - 0.0393$
	2	0.9297 (0.0011)	$\tilde{V}_t = 1.0017 \cdot \hat{V}_t - 0.0339 \cdot R^e_{t-1} - 0.0977$
	3	0.9401 (0.0002)	$\tilde{V}_t = 1.0035 \cdot \hat{V}_t - 0.0284 \cdot R^e_{t-1} - 0.0870$
2	1	0.9301 (0.0010)	$\tilde{V}_t = 1.0041 \cdot \hat{V}_t + 0.0043 \cdot R^e_{t-1} - 0.0982$
	2	0.9281 (0.0017)	$\tilde{V}_t = 1.0026 \cdot \hat{V}_t - 0.0237 \cdot R^e_{t-1} - 0.0737$
	3	0.9396 (0.0004)	$\tilde{V}_t = 0.9885 \cdot \hat{V}_t - 0.0294 \cdot R^e_{t-1} + 0.2487$
3	1	0.9318 (0.0011)	$\tilde{V}_t = 0.9921 \cdot \hat{V}_t - 0.0505 \cdot R^e_{t-1} + 0.0665$
	2	0.9295 (0.0009)	$\tilde{V}_t = 1.0041 \cdot \hat{V}_t - 0.0355 \cdot R^e_{t-1} - 0.1278$
	3	0.9407 (0.0004)	$\tilde{V}_t = 1.0006 \cdot \hat{V}_t - 0.0523 \cdot R^e_{t-1} - 0.0212$

the linear $y = -x$, leading to a less meaningful outcome upon symbolification. Moreover, automatically symbolifying them using the closest symbolic function can undermine interpretability[3]. For this reason, we exclude these adjusted models from forecasting performance comparisons with other benchmarks, as interpretability remains a primary motivation for adopting KAN in this study.

3.2.5. Comparison with traditional and deep learning benchmark methods

We compare forecasting performances of KAN and various benchmark methods: traditional statistical time-series models (forward filling, ARMA, ARIMA, HAR), and neural network models (MLP, LSTM). For notational convenience, the neural network model (KAN, MLP, LSTM) trained on *Dataset i* is denoted as *model-D_i* for $i = 1, 2, 3$. To ensure a robust and comprehensive assessment of model performance, we employ several metrics: mean squared error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), R^2, and quasi-likelihood (QLIKE). Their definitions are given by

$$\text{MSE} = \frac{1}{N_{test}} \sum_{t=1}^{N_{test}} \left(V_t - \hat{V}_t \right)^2, \quad \text{MAE} = \frac{1}{N_{test}} \sum_{t=1}^{N_{test}} \left| V_t - \hat{V}_t \right|, \quad \text{MAPE} = \frac{1}{N_{test}} \sum_{t=1}^{N_{test}} \left| \frac{V_t - \hat{V}_t}{V_t} \right| \times 100,$$

$$R^2 = 1 - \frac{\sum_{t=1}^{N_{test}} \left(V_t - \hat{V}_t \right)^2}{\sum_{t=1}^{N_{test}} \left(V_t - \bar{V} \right)^2}, \qquad \text{QLIKE} = \frac{1}{N_{test}} \sum_{t=1}^{N_{test}} \left[\frac{V_t}{\hat{V}_t} - \ln \frac{V_t}{\hat{V}_t} - 1 \right],$$

where N_{test} is the number of test data points, $\bar{V}$ denotes the mean of the VIX index in the test period, and V_t and $\hat{V}_t$ are the actual and forecasted VIX values at time t, respectively.

Tables 5–7 summarize the out-of-sample forecasting accuracy for KAN and various benchmark methods. Across the three *Period*s, KAN consistently delivers competitive results, often achieving the best or second-best performances. Although the naive forward filling method performs surprisingly well−likely owing to the strong persistence

Table 5: Forecasting accuracy comparison of various models (*Period 1*). **Bold** and <u>underline</u> values indicate the best and second-best performances, respectively.

Model	MSE	MAE	MAPE	R^2	QLIKE	# params
Forward filling	4.5302	1.2425	5.5466	0.9285	<u>0.0143</u>	-
ARMA(1, 1)	6.9365	1.6635	7.6243	0.8906	0.5433	4
ARIMA(1, 1, 1)	7.1332	1.6745	7.6344	0.8875	0.5465	4
HAR(3)	4.4337	1.2228	5.4967	0.9301	0.0144	4
HAR(4)	4.4347	1.2229	5.4970	0.9300	0.0144	5
MLP-D_1	4.4262	1.2317	5.5294	0.9302	0.0154	20901
MLP-D_2	4.6329	<u>1.2180</u>	<u>5.4044</u>	0.9269	0.0151	20801
MLP-D_3	4.7031	**1.2116**	**5.3792**	0.9258	0.0159	20801
LSTM-D_1	4.8150	1.2396	5.5124	0.9240	0.0163	63841
LSTM-D_2	4.8059	1.2908	5.7796	0.9242	0.0156	63561
LSTM-D_3	4.9712	1.2552	5.5850	0.9216	0.0157	63561
KAN-D_1	<u>4.3977</u>	1.2254	5.5063	<u>0.9306</u>	0.0144	72
KAN-D_2	4.4970	1.2280	5.5008	0.9291	0.0145	60
KAN-D_3	**4.3957**	1.2219	5.5263	**0.9307**	**0.0142**	60

---

[3]In some trials, the activation functions were replaced by a sine function or error function, yet these replacements seem meaningless in practice.

Table 6: Forecasting accuracy comparison of various models (*Period 2*). **Bold** and underline values indicate the best and second-best performances, respectively.

Model	MSE	MAE	MAPE	R^2	QLIKE	# params
Forward filling	5.2130	1.3274	5.2876	0.9266	**0.0124**	-
ARMA(1, 1)	7.6826	1.7531	7.1646	0.8919	0.4931	4
ARIMA(1, 1, 1)	7.9619	1.7741	7.2251	0.8879	0.4945	4
HAR(3)	5.1215	1.3029	5.2172	0.9279	0.0125	4
HAR(4)	5.1222	1.3029	5.2172	0.9279	0.0125	5
MLP-D_1	**5.0205**	1.2938	5.1739	**0.9293**	0.0130	20901
MLP-D_2	5.4638	1.3046	<u>5.1737</u>	0.9231	0.0131	20801
MLP-D_3	5.2307	**1.2706**	**5.0659**	0.9264	0.0130	20801
LSTM-D_1	5.4018	1.3172	5.2062	0.9240	0.0133	63841
LSTM-D_2	5.5741	1.3694	5.3861	0.9215	0.0134	63561
LSTM-D_3	5.3274	1.3890	5.5304	0.9250	0.0129	63561
KAN-D_1	<u>5.0711</u>	1.3051	5.2162	<u>0.9286</u>	<u>0.0125</u>	72
KAN-D_2	5.2314	1.3155	5.2499	0.9264	0.0127	60
KAN-D_3	5.0755	<u>1.3003</u>	5.2037	<u>0.9286</u>	**0.0124**	60

Table 7: Forecasting accuracy comparison of various models (*Period 3*). **Bold** and underline values indicate the best and second-best performances, respectively.

Model	MSE	MAE	MAPE	R^2	QLIKE	# params
Forward filling	1.9713	1.0065	4.5349	0.9399	<u>0.0074</u>	-
ARMA(1, 1)	3.7467	1.4055	6.3387	0.8858	0.3298	4
ARIMA(1, 1, 1)	3.7772	1.4257	6.4268	0.8849	0.3318	4
HAR(3)	1.9582	0.9939	4.5057	0.9403	<u>0.0074</u>	4
HAR(4)	1.9579	0.9938	4.5057	<u>0.9403</u>	<u>0.0074</u>	5
MLP-D_1	1.9846	<u>0.9869</u>	<u>4.4459</u>	0.9395	0.0076	20901
MLP-D_2	1.9902	**0.9844**	**4.4250**	0.9393	0.0076	20801
MLP-D_3	2.0092	1.0015	4.5294	0.9388	0.0076	20801
LSTM-D_1	1.9884	1.0167	4.6120	0.9394	0.0075	63841
LSTM-D_2	2.0544	1.0143	4.5548	0.9374	0.0076	63561
LSTM-D_3	**1.9333**	0.9961	4.5186	**0.9411**	**0.0073**	63561
KAN-D_1	1.9702	0.9968	4.5112	0.9400	0.0075	72
KAN-D_2	1.9955	1.0096	4.5586	0.9392	0.0076	60
KAN-D_3	<u>1.9575</u>	0.9934	4.5012	<u>0.9403</u>	<u>0.0074</u>	60

inherent in the VIX−ARMA(1,1) and ARIMA(1,1,1) reveal the limitations of a purely linear framework devoid of longer-range dependence or multiple timescale averages. Meanwhile, the HAR(3) and HAR(4) models, which assume a linear additive structure over short-, medium-, long-, and very-long-term averages, generally achieve decent results.

Regarding neural-network models, MLP and LSTM often demonstrate strong predictive power. However, they require approximately 300 or 1,000 times more parameters compared to KAN while offering comparable accuracy. Moreover, a closer look at the metrics suggests that KAN tends to excel in MSE rather than MAE or MAPE. This indicates that KAN produces fewer extreme errors or outliers. Remarkably, KAN's parsimonious design consistently achieves performance on par with these more parameter-intensive MLP-based networks, underscoring its robustness and stability in noisy financial time-series data.

Turning to the different dataset configurations, *Dataset 3* typically yields the best results for KAN, MLP, LSTM, and HAR models alike. Notably, KAN–D_3 demonstrates particularly strong accuracy in all three periods. This suggests that effectively capturing key characteristics of the VIX, such as mean-reversion and high-persistence, leads to reliable forecasting results. Models using *Dataset 1* also perform reasonably well. The decent results when using *Dataset 1* are presumably because consecutive lags capture crucial short-term dependencies (e.g., volatility clustering, momentum). By contrast, *Dataset 2* often fails to provide significant multi-horizon features.

Taken together, these findings confirm that KAN offers a robust and interpretable alternative to both classical time-series approaches and parameter-intensive MLP-based neural networks. This illustrates that a parsimonious structure can yield high forecasting accuracy in noisy financial time-series such as the VIX.

4. Conclusion

MLP-based neural networks have exhibited remarkable forecasting performances in numerous studies, but their black-box nature often restricts their practical use in financial time-series forecasting such as the VIX. In this study, we employ Kolmogorov–Arnold Networks (KANs) to address these limitations. Each edge in KAN extracts important features from the data and describes them with symbolic functions, enabling the output of the network to be expressed as a closed-form in terms of explanatory variables. This interpretability allows us to gain valuable insights into the VIX's inherent characteristics such as mean reversion and the leverage effect. Furthermore, our empirical analysis across multiple periods and datasets indicates that KAN achieves robust and competitive accuracy compared to MLPs and LSTMs while requiring significantly fewer parameters and maintaining interpretability. These findings suggest that KAN can serve as a powerful and parsimonious framework for analyzing noisy financial time-series data, including the VIX, and may be extended to broader contexts where clear, data-driven insights are essential.

Appendix A. Training results for deep and wide KAN

In this paper, all experiments were conducted with a two-layer KAN using two nodes in the hidden layer. This section examines the impact of increasing either the depth or the number of nodes. Specifically, we analyze two cases: (1) a two-layer KAN with 10 hidden nodes, and (2) a three-layer KAN with 4 hidden nodes each. The training results are presented in Figure A.1. While depth and width can be adjusted to various values, the results remain consistent and are therefore omitted. As shown in the figure, even with increased width or depth, the learned activation functions for each layer predominantly exhibit linear characteristics. The actual number of active nodes in the figure may appear smaller (e.g., 2 or 6) than the predefined configuration with a larger number of nodes (e.g., 10). This is because, during training, unimportant nodes are pruned as part of the optimization process, leaving only significant nodes to contribute to the final model. All these activation functions are replaced with linear functions $y = x$ or $y = -x$ at the symbolification step, and summed at the node level. Consequently, the outcomes of each layer become a linear combination of input variables. Even after passing through multiple layers, the result remains a linear combination. Thus, we chose the simplest configuration, $[n_i, 2, 1]$, where $n_i = 4$ for *Dataset 1* and $n_i = 5$ for *Dataset 2* and *Dataset 3*, as the structure of the KAN model.
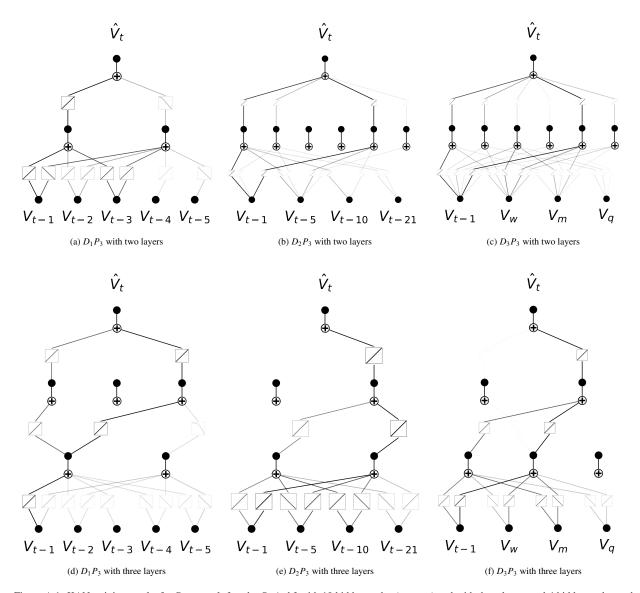
Figure A.1: KAN training results for *Datasets 1–3* under *Period 3* with 10 hidden nodes (top row) and with three layers and 4 hidden nodes each (bottom row). The trained activation functions predominantly exhibit a linear form, leading to the KAN model being expressed as a sum of linear terms. This structure is preserved across all *Dataset*s.

Appendix B. KAN training result before and after symbolification in *Period 1* and *Period 2*

In Section 3.2.2, only the training results of KANs in *Period 3* are presented. The results in *Period 1* and *Period 2* are illustrated in Figure B.2 and Figure B.3, respectively. As shown in the figures, it is evident that the trained activation functions before symbolification exhibit a linear form across all *Dataset*s and *Period*s.

Acknowledgement

(a) D_1P_1

(b) D_2P_1

(c) D_3P_1

(d) D_1P_1
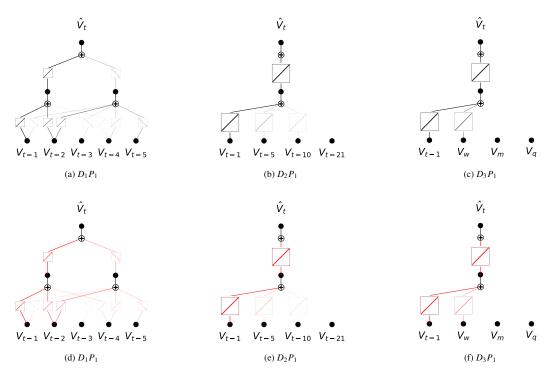
(e) D_2P_1

(f) D_3P_1

Figure B.2: KAN training results for *Datasets 1–3* under *Period 1* before (top row) and after (bottom row) symbolification. Black activation functions indicate the trained B-spline-based structure, whereas red activation functions depict the replaced symbolic functions. Vivid edges represent stronger effects, while fainter edges mean weaker ones.
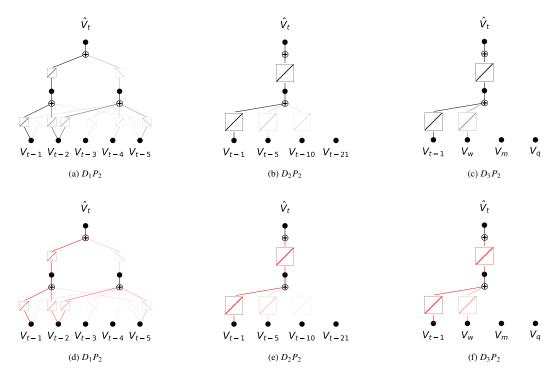

(a) D_1P_2

(b) D_2P_2

(c) D_3P_2

(d) D_1P_2

(e) D_2P_2

(f) D_3P_2

Figure B.3: KAN training results for *Datasets 1–3* under *Period 2* before (top row) and after (bottom row) symbolification. Black activation functions indicate the trained B-spline-based structure, whereas red activation functions depict the replaced symbolic functions. Vivid edges represent stronger effects, while fainter edges mean weaker ones.

16

CRediT authorship contribution statement

So-Yoon Cho: Conceptualization, Software, Writing - Original Draft. **Sungchul Lee**: Writing - Review & Editing, Validation. **Hyun-Gyoon Kim**: Conceptualization, Methodology, Writing - Original Draft, Supervision.

Data availability

The data that support the findings of this study are available from DataGuide. Restrictions apply to the availability of these data, which were used under license for this study. Data are available from Yonsei University with the permission of DataGuide.

Declaration of competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT in order to improve the grammatical quality and readability. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.

[2] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals and Systems 2 (1989) 303–314.

[3] G. Zhang, B. E. Patuwo, M. Y. Hu, Forecasting with artificial neural networks: The state of the art, International Journal of Forecasting 14 (1998) 35–62.

[4] R. Cont, Empirical properties of asset returns: stylized facts and statistical issues, Quantitative Finance 1 (2001) 223.

[5] R. T. Baillie, Long memory processes and fractional integration in econometrics, Journal of Econometrics 73 (1996) 5–59.

[6] S. J. Taylor, Modelling financial time series, World Scientific, 2008.

[7] S. Satchell, J. Knight, Forecasting volatility in the financial markets, Elsevier, 2011.

[8] G. E. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, Time series analysis: forecasting and control, John Wiley & Sons, 2015.

[9] K. R. French, G. W. Schwert, R. F. Stambaugh, Expected stock returns and volatility, Journal of financial Economics 19 (1987) 3–29.

[10] M. Fernandes, M. C. Medeiros, M. Scharth, Modeling and predicting the CBOE market volatility index, Journal of Banking & Finance 40 (2014) 1–10.

[11] F. Corsi, A simple approximate long-memory model of realized volatility, Journal of Financial Econometrics 7 (2009) 174–196.

[12] U. A. Müller, M. M. Dacorogna, R. D. Davé, O. V. Pictet, R. B. Olsen, J. R. Ward, Fractals and intrinsic time-a challenge to econometricians, SSRN, 2008.

[13] T. G. Andersen, T. Bollerslev, F. X. Diebold, Roughing it up: Including jump components in the measurement, modeling, and forecasting of return volatility, The Review of Economics and Statistics 89 (2007) 701–720.

[14] A. J. Patton, K. Sheppard, Optimal combinations of realised volatility estimators, International Journal of Forecasting 25 (2009) 218–238.

[15] M. Izzeldin, M. K. Hassan, V. Pappas, M. Tsionas, Forecasting realised volatility using ARFIMA and HAR models, Quantitative Finance 19 (2019) 1627–1638.

[16] T. Bollerslev, Generalized autoregressive conditional heteroskedasticity, Journal of Econometrics 31 (1986) 307–327.

[17] W. Ge, P. Lalbakhsh, L. Isai, A. Lenskiy, H. Suominen, Neural network–based financial volatility forecasting: A systematic review, ACM Computing Surveys (CSUR) 55 (2022) 1–30.

[18] F. Mostafa, T. Dillon, E. Chang, Computational intelligence approach to capturing the implied volatility, in: Artificial Intelligence in Theory and Practice IV: 4th IFIP TC 12 International Conference on Artificial Intelligence, IFIP AI 2015, Held as Part of WCC 2015, Daejeon, South Korea, October 4-7, 2015, Proceedings 4, Springer, 2015, pp. 85–97.

[19] H. Kumar, S. B. Patil, Estimation & forecasting of volatility using arima, arfima and neural network based techniques, in: 2015 IEEE International Advance Computing Conference (IACC), IEEE, 2015, pp. 992–997.

[20] S. Pyo, J. Lee, Exploiting the low-risk anomaly using machine learning to enhance the black–litterman framework: Evidence from south korea, Pacific-Basin Finance Journal 51 (2018) 1–12.

[21] G. Petneházi, J. Gáll, Exploring the predictability of range-based volatility estimators using recurrent neural networks, Intelligent Systems in Accounting, Finance and Management 26 (2019) 109–116.

[22] H. Y. Kim, C. H. Won, Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models, Expert Systems with Applications 103 (2018) 25–37.

[23] Y. Liu, Novel volatility forecasting using deep learning–long short term memory recurrent neural networks, Expert Systems with Applications 132 (2019) 99–109.

[24] S. Lahmiri, A variational mode decompoisition approach for analysis and forecasting of economic and financial time series, Expert Systems with Applications 55 (2016) 268–273.

[25] H. Litimi, A. BenSaida, L. Belkacem, O. Abdallah, Chaotic behavior in financial market volatility, Journal of Risk, Forthcoming (2018).

[26] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, M. Tegmark, KAN: Kolmogorov-Arnold networks, arXiv preprint arXiv:2404.19756 (2024).

[27] A. N. Kolmogorov, On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition, in: Doklady Akademii Nauk, volume 114, Russian Academy of Sciences, 1957, pp. 953–956.

[28] A. N. 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.

[29] Y. Peng, M. He, F. Hu, Z. Mao, X. Huang, J. Ding, Predictive modeling of flexible ehd pumps using kolmogorov-arnold networks, arXiv preprint arXiv:2405.07488 (2024).

[30] C. J. Vaca-Rubio, L. Blanco, R. Pereira, M. Caus, Kolmogorov-arnold networks (kans) for time series analysis, arXiv preprint arXiv:2405.08790 (2024).

[31] D. Alves, F. Mendonça, S. S. Mostafa, F. Morgado-Dias, On the use of kolmogorov–arnold networks for adapting wind numerical weather forecasts with explainability and interpretability: application to madeira international airport, Environmental Research Communications 6 (2024) 105008.

[32] K. Xu, L. Chen, S. Wang, Kolmogorov-arnold networks for time series: Bridging predictive power and interpretability, arXiv preprint arXiv:2406.02496 (2024).

[33] H. Inzirillo, R. Genet, Sigkan: Signature-weighted kolmogorov-arnold networks for time series, arXiv preprint arXiv:2406.17890 (2024).

[34] R. Genet, H. Inzirillo, Tkan: Temporal kolmogorov-arnold networks, arXiv preprint arXiv:2405.07344 (2024).

[35] R. Genet, H. Inzirillo, A temporal kolmogorov-arnold transformer for time series forecasting, arXiv preprint arXiv:2406.02486 (2024).

[36] R. E. Whaley, Understanding the vix, Journal of Portfolio Management 35 (2009) 98–105.

[37] Z. Ding, C. W. Granger, R. F. Engle, A long memory property of stock market returns and a new model, Journal of Empirical Finance 1 (1993) 83–106.

[38] D. Kwiatkowski, P. C. Phillips, P. Schmidt, Y. Shin, Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?, Journal of Econometrics 54 (1992) 159–178.

[39] I. Goodfellow, Deep learning, 2016.

[40] F. X. Diebold, Elements of forecasting, Citeseer, 1998.

[41] J. D. Hamilton, Time series analysis, Princeton university press, 2020.

[42] S. L. Heston, A closed-form solution for options with stochastic volatility with applications to bond and currency options, The Review of Financial Studies 6 (1993) 327–343.

[43] L. O. Scott, Option pricing when the variance changes randomly: Theory, estimation, and an application, Journal of Financial and Quantitative Analysis 22 (1987) 419–438.

[44] J. A. Mincer, V. Zarnowitz, The evaluation of economic forecasts, in: Economic forecasts and expectations: Analysis of forecasting behavior and performance, NBER, 1969, pp. 3–46.

[45] J. Durbin, G. S. Watson, Testing for serial correlation in least squares regression. I, in: Breakthroughs in Statistics: Methodology and Distribution, Springer, 1992, pp. 237–259.

[46] J. Durbin, G. S. Watson, Testing for serial correlation in least squares regression. II, in: Breakthroughs in Statistics: Methodology and Distribution, Springer, 1992, pp. 260–266.