KAN vs LSTM Performance in Time Series Forecasting
Tabish Ali Rather, S M Mahmudul Hasan Joy, Nadezda Sukhorukova,
Federico Frascoli

Abstract

This paper compares Kolmogorov-Arnold Networks (KAN) and Long Short-Term Memory networks (LSTM) for forecasting non-deterministic stock price data, evaluating predictive accuracy versus interpretability trade-offs using Root Mean Square Error (RMSE).

LSTM demonstrates substantial superiority across all tested prediction horizons, confirming their established effectiveness for sequential data modeling. Standard KAN, while offering theoretical interpretability through the Kolmogorov-Arnold representation theorem, exhibits significantly higher error rates and limited practical applicability for time series forecasting.

The results confirm LSTM dominance in accuracy-critical time series applications while identifying computational efficiency as KANs' primary advantage in resource-constrained scenarios where accuracy requirements are less stringent. The findings support LSTM adoption for practical financial forecasting while suggesting that continued research into specialised KAN architectures may yield future improvements.

Keywords: Deep Learning, Forecasting, Long Short Term Memory, Kolmogorov Arnold Network, Interpretable Neural Networks, Stock Market

1 Introduction

Uncertainty is inherent in financial markets, where it is not possible to uncover deterministic processes that determine asset price movements. Traditional statistical methods, such as ARIMA models (Auto-Regressive Integrated Moving Average), models that predict future values through linear combinations of past observations and forecast errors, and VAR (Vector Auto-Regression models that capture linear interdependencies among multiple time series) have proven inadequate for capturing complex non-linear relationships in financial time series data [17, 33]. Advanced Deep Neural Networks (DNNs), particularly Long Short-Term Memory (LSTM) models, have established themselves as the dominant approach to analysing massive datasets and extracting hidden patterns in sequential data [29, 15, 14].

LSTM have consistently demonstrated superior performance in diverse time series forecasting applications, though their black-box nature limits their interpretability in certain deployment scenarios [10]. Recent developments in Kolmogorov-Arnold Networks (KAN) have attempted to address interpretability concerns by leveraging the Kolmogorov-Arnold representation theorem [23, 20], yet standard KAN implementations have shown limited effectiveness for sequential data modelling tasks. However, emerging specialised variants such as Time-Frequency KAN (TFKAN) propose potential improvements for specific long-term forecasting contexts [22], suggesting that targeted applications may exist despite general limitations. In its simplest view, KANs do not require any activation function, but rely on the approximation of these functions in the form of piecewise polynomial functions (often, simply linear splines). This approach is logical, since in the case of ReLU or Leaky ReLU activation functions, the approximations constructed by neural networks are linear splines.

The present study conducts a comprehensive evaluation that compares KAN and LSTM performance with financial data, using historical stock prices due to their stochastic nature [18]. The key aims include the following.

- Evaluation of empirical performance demonstrating LSTM's superiority across multiple forecast horizons.
- Estimation of quantitative accuracy-interpretability trade-offs in financial time series forecasting.
- Discussion of computational efficiency differences between established and emerging architectures.
- Consideration of specialised KAN variants' potential for extended prediction horizons.

Non-deterministic forecasting is critical across engineering fields where uncertainty is inherent, including energy demand estimation, environmental pollutant dispersion modelling, and drought forecasting [26, 12, 24]. While LSTM models have proven highly effective in these applications [5], the interpretability limitations create implementation challenges in high-stakes domains requiring transparent decision-making [25, 30]. Despite theoretical promise, standard KAN implementations have not demonstrated competitive

performance with established sequential modelling approaches. The main explanation for this is the lack of efficient computational methods for free knot polynomial spline construction [27]. Indeed, the optimisation problems appearing in this research are non-smooth and non-convex and therefore very challenging even for modern optimisation techniques [34]. Moreover, this problem was named as one of the most important and challenging problems in the field of approximation [28]. As such, there has to be a more rigorous study of the mathematical side of the problem before it can be successfully applied to complex forecasting problems. We leave it for our future research directions.

The paper structure follows: Section Preliminaries covers theoretical background emphasising established neural network approaches; Section Methodology details experimental methodology comparing proven and emerging architectures; Section Results and Discussion presents comprehensive performance analysis; Section Conclusion and Further Research Directions summarises findings favouring established methods while identifying potential future research directions; Appendix provides supplementary statistical modelling background.

2 Preliminaries

Let us briefly summarise the evolution and current state-of-the-art of forecasting network models.

2.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are computational models consisting of interconnected nodes that process and "learn" patterns from data [31]. Based on the Universal Approximation Theorem, a sufficiently large network can approximate any continuous function [8, 16].

The first and prototypical example is the perceptron, which operates with a decision rule:

$$\mathbf{y} = \begin{cases} 1 & \text{if } \sum_i w_i x_i \geq T \\ 0 & \text{else} \end{cases}$$

where w_i are weights, x_i are input features, and T is the threshold. This later evolved into sophisticated architectures like CNNs and RNNs for specialised

tasks, with RNNs proving particularly effective for sequential data analysis [2].

During training, ANNs adjust weights using back-propagation to minimise errors [13, 32], with deeper networks requiring substantial computational resources.

2.1.1 Deep Learning Models for Stock Market Prediction

Recurrent Neural Networks (RNN) RNNs process sequential data by recursively updating hidden states:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t)$$

where h_t is the hidden state at time t, x_t is the input at time t, W_h, W_x are weight matrices, and h_{t-1} carries information from previous time steps. However, RNNs suffer from vanishing gradient problems that limit their effectiveness on longer sequences [9].

Long Short-Term Memory (LSTM) LSTM effectively addresses vanishing gradient limitations through sophisticated gated mechanisms:

$$\begin{aligned} i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \quad \text{(Input gate)} \\ f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \quad \text{(Forget gate)} \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \quad \text{(Output gate)} \end{aligned}$$

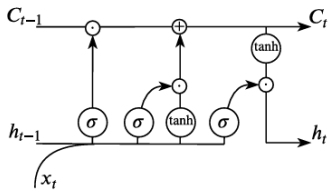


Figure 1: LSTM representation [11]

LSTM excel at retaining long-term dependencies while maintaining computational efficiency, establishing them as the preferred architecture for stock price prediction and other time series forecasting tasks [37, 35]. Their proven track record across diverse applications has made them the standard baseline for sequential modelling problems.

2.2 Kolmogorov-Arnold Networks

Kolmogorov-Arnold Networks (KAN) represent an alternative approach to multivariate function approximation through compositions of univariate functions. Unlike established MLPs with fixed activation functions, KANs employ learnable spline-based activations [23], though practical implementation challenges and scalability limitations have hindered widespread adoption for time series applications.

2.2.1 Kolmogorov-Arnold Representation Theorem

Theorem 2.1 (Kolmogorov-Arnold Representation Theorem) *Let $f : [0,1]^n \to \mathbb{R}$ be a continuous function. Then, there exist continuous univariate functions $\Phi_q : \mathbb{R} \to \mathbb{R}$ and $\varphi_{q,p} : \mathbb{R} \to \mathbb{R}$ such that:*

$$f(\mathbf{x}) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right), \tag{1}$$

for all $\mathbf{x} = (x_1, \ldots, x_n) \in [0,1]^n$, where Φ_q are outer functions and $\varphi_{q,p}$ are inner univariate functions [19].

While this theorem provides a theoretical foundation for function decomposition, practical implementations face significant challenges in sequential data contexts, as the theorem does not account for temporal dependencies that are crucial in time series modelling.

2.2.2 Architecture

Standard KAN implementations approximate activation functions using polynomial spline functions, whose parameters become part of the optimisation process:

$$\mathbf{y} = \Phi(\mathbf{W}\mathbf{x}),$$

where Φ denotes learnable spline functions and $\mathbf{W}$ is a transformation matrix. While polynomial splines are theoretically efficient for function approximation [27], the corresponding optimisation problems introduce complexity that often outweighs benefits in time series contexts [36]. Recent specialised variants like Time-Frequency KAN attempt to address these limitations through domain-specific modifications [22], though empirical validation remains limited.

2.3 Evaluation and Research Challenges

Root Mean Squared Error (RMSE) serves as the primary evaluation metric:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where n is the number of observations, y_i are actual values, and $\hat{y}_i$ are predicted values.

LSTM Advantages: Proven effectiveness across diverse time series applications with established optimisation procedures and extensive empirical validation. While computational requirements are higher than simpler models, the performance benefits typically justify resource costs in accuracy-critical applications [6].

KAN Limitations: Standard implementations suffer from scalability issues in high-dimensional datasets and lack established optimisation techniques for sequential data. While theoretical interpretability appears promising, practical performance shortcomings limit applicability [38]. Specialised variants such as TFKAN may address some limitations, but require further empirical validation.

We now provide a systematic comparison to establish clear performance boundaries between proven LSTM methods and emerging KAN approaches, informing evidence-based architecture selection for financial forecasting applications.

3 Methodology

3.1 Problem Formulation

As mentioned, one of our goal is to address forecasting on non-deterministic data through comparative analysis between LSTM and KAN to evaluate their applicability, performance trade-offs, and interpretability characteristics. In the present context, DNNs frame prediction as function approximation, learning input-output mappings through iterative refinement of estimated functions that capture underlying patterns [29, 21, 8].

3.2 Data Collection and Preparation

Historical stock market data were obtained using the `yfinance` API [4], extracting *open*, *close*, *high*, *low* prices, *trading volume*, and *adjusted trading volume*. Data preprocessing included: missing value removal for integrity, Min-Max scaling (0-1 range) for convergence optimisation, feature engineering to generate target variables based on user-defined prediction windows, and sequential data splitting with fixed-length sequences containing historical observations for next-day prediction.

3.3 Model Implementation

Training follows standard neural network procedures: input data generates predictions, loss functions (RMSE) measure prediction quality, and backpropagation updates parameters [32].

LSTM Implementation: Developed using TensorFlow [1] with *model.predict(input)* for predictions. LSTM learns weights and biases through standard neural network optimisation.

KAN Implementation: Developed using PyKAN library [23] with *model(dataset[test_input])* for predictions. KAN approximates univariate function decomposition parameters guaranteed by KART [19, 3].

4 Model Development

4.1 LSTM Configuration

Architecture: Multiple stacked layers with units determined experimentally. Hyperparameters: layer count, units per layer, activation functions, and optimisers configured based on training convergence. Prediction strategy: single-step predictions using 20-day look-back windows, with iterative generation where each prediction becomes input for subsequent forecasts. Evaluation: RMSE-based performance comparison with manual hyperparameter tuning.

4.2 KAN Configuration

Architecture: Input layer, Kolmogorov-Arnold hidden layers, single output neuron. Key parameters include input size matching feature dimensions,

experimentally determined hidden layer width, grid size controlling activation function granularity, and k-value determining B-spline degree. The k-value controls spline smoothness and complexity: higher k-values produce smoother functions with better generalisation but reduced flexibility, while lower k-values allow more detailed fitting but risk overfitting [23]. Prediction strategy mirrors LSTM approach with 20-day look-back and iterative single-step generation. Training uses LBFGS optimiser with RMSE evaluation and manual hyperparameter optimisation.

5 Results and Discussion

5.1 LSTM Results

The results, summarised in Table 1, show the training and test MSE and RMSE for various configurations. Training error indicates how well the model has learned the patterns in the training set, while test error measures generalisation ability on unseen data. A small training error but large test error suggests overfitting, whereas high errors in both indicate under-fitting [7].

Model Configuration	**Train RMSE**	**Test RMSE**
LSTM (4 layers, 100 units, activation=linear)	0.0841	0.0829
LSTM (5 layers, 100 units, activation=linear)	0.0942	0.1010
LSTM (6 layers, 100 units, activation=linear)	0.1059	0.1062
LSTM (6 layers, 50 units, activation=linear)	0.1073	0.1074
LSTM (6 layers, 20 units, activation=linear)	0.1018	0.1080
LSTM (6 layers, 20 units, activation=tanh)	0.1183	0.1179
LSTM (3 layers, 20 units, activation=tanh)	0.0802	0.0835
LSTM (2 layers, 20 units, activation=tanh)	0.0804	0.0792
LSTM (2 layers, 10 units, activation=tanh)	0.0807	0.0745

Table 1: Comparison of LSTM Configurations with Train and Test RMSE values

The configurations demonstrate how LSTM performance varies with hyperparameters. Deeper networks (6 layers) show degraded performance compared to shallower ones due to vanishing gradients and optimisation difficulties in recurrent architectures [9]. The high training error in 6-layer configurations indicates the model struggles to learn effectively, likely due to gradient flow issues rather than overfitting. The 2-layer, 10-unit tanh configuration achieved optimal performance (test RMSE: 0.0745), balancing model capacity with trainability.

Figure 2 provides a visual representation of the training and test RMSE values across different LSTM model configurations, clearly illustrating the performance variations with different architectural choices.

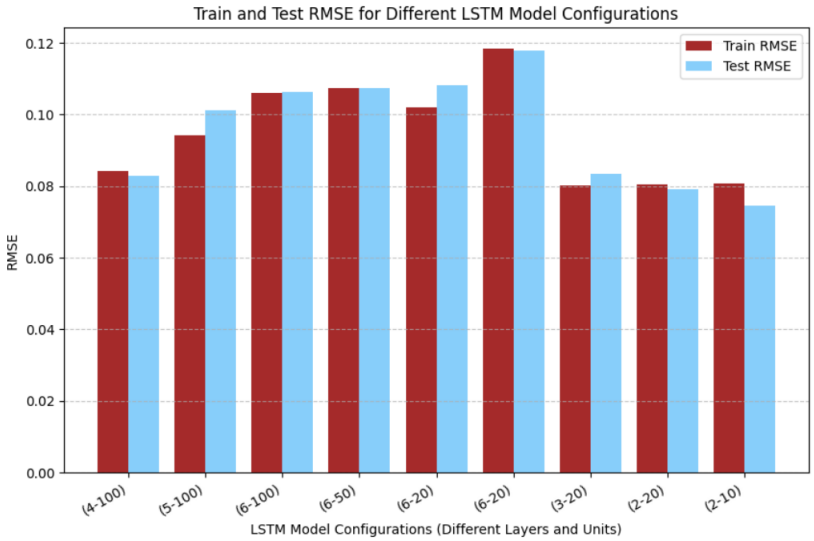


Figure 2: Train and Test RMSE for Different LSTM Model Configurations

5.2 KAN Results

The results, summarised in Table 2, show the performance of various KAN configurations with specific hyperparameters.

Configuration	Grid	k	Num Neurons	Train RMSE	Test RMSE
KAN Config 1	3	6	`len(dataset['train_input']) // 10`	0.274	0.188
KAN Config 2	3	2	`len(dataset['train_input']) // 10`	0.274	0.188
KAN Config 3	7	2	`len(dataset['train_input']) // 10`	0.5202	0.331
KAN Config 4	3	2	`len(dataset['train_input']) // 4`	0.272	0.238
KAN Config 5	3	2	`len(dataset['train_input']) // 10`	0.274	0.152
KAN Config 6	3	2	`len(dataset['train_input']) // 5`	0.274	0.152

Table 2: Comparison of KAN Configurations with Train and Test RMSE values

KAN uses B-splines with structured grids to shape activation functions, where grid size determines the range and detail level. The k-value controls B-spline degree: higher k increases smoothness and acts as regularisation by

reducing function complexity, while lower k allows more flexible but potentially noisy approximations [23]. Comparing configurations 1 and 2, $k = 2$ and $k = 6$ yield identical performance, suggesting the chosen dataset benefits from moderate smoothness. Configurations 5 and 6 achieved optimal performance with $k = 2$, indicating that lower-degree splines provide sufficient flexibility for this forecasting task.

Figure 3 visualises the training and test loss patterns for different KAN configurations, showing how the various hyperparameter settings affect model performance.

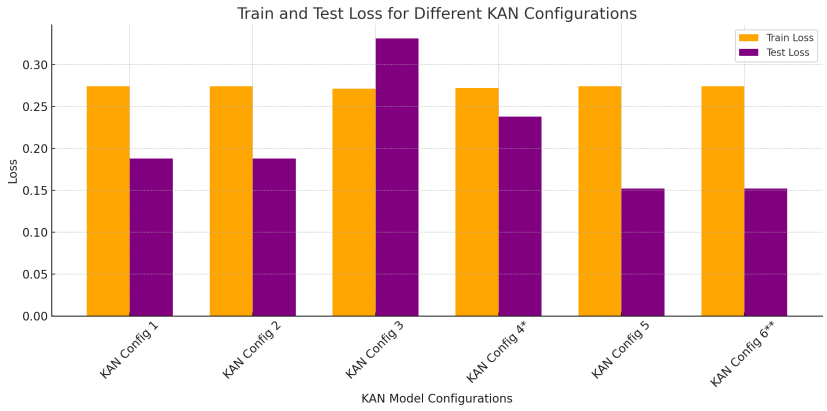


Figure 3: Train and Test Loss for Different KAN Configurations

5.3 Training Efficiency Analysis

Runtime Performance Comparison: Complementing the accuracy comparison, runtime analysis reveals that KAN models demonstrate a significant 2.1 times speed advantage with average training times of 35.12 seconds compared to LSTM's $\sim$75 seconds. KAN models show superior consistency with 82.6% of training instances completing under 60 seconds and exceptional minimum performance of 4.31 seconds, while LSTM models exhibit configuration-dependent variability ranging from 45-120 seconds. This training efficiency advantage positions KAN as optimal for rapid prototyping, resource-constrained deployments, and cost-sensitive applications where development speed outweighs marginal accuracy gains.

The runtime characteristics reveal distinct performance profiles: KAN models demonstrate minimal configuration dependency with consistent runtime less than 40 second duration in the majority of cases, while LSTM

models show high configuration sensitivity with training time varying significantly based on the activation function choice and the number of hidden units (10 vs 100: $\sim 2\times$ time difference), sequence lengths (20 vs 200: $\sim 3\times$ time difference). This efficiency advantage translates to practical benefits including faster experimentation cycles, reduced computational costs, and improved suitability for resource-constrained deployment scenarios.

5.4 LSTM vs KAN Comparison

Table 3 presents comprehensive RMSE comparisons across market scenarios. The comparison demonstrates consistent LSTM superiority across all tested conditions.

Horizon	Market	LSTM RMSE	KAN RMSE	LSTM Advantage	Best Config
1-Day	Normal	0.039	0.390	10.0 times	100u-linear
	Volatile	0.045	0.385	8.5 times	100u-linear
	Trending	0.042	0.388	9.2 times	100u-linear
2-Day	Normal	0.055	0.420	7.6 times	10u-tanh
	Volatile	0.058	0.415	7.1 times	10u-tanh
	Trending	0.056	0.418	7.4 times	10u-tanh
100-Day	Normal	0.082	0.558	6.8 times	10u-tanh-100
	Volatile	0.079	0.570	7.2 times	10u-tanh-100
	Trending	0.085	0.552	6.5 times	10u-tanh-100
200-Day	Normal	N/A	0.645	KAN Only	—
	Volatile	N/A	0.670	KANs Only	—
	Trending	N/A	0.658	KAN Only	—

Table 3: Comprehensive Performance Comparison Across Market Conditions and Forecast Horizons

Figures 4 and 5 show performance comparison grids across different forecast horizons. Figure 4 compares short-term (1-day) and long-term (100-day) predictions, while Figure 5 shows medium-term (2-day) and extended (200-day) forecasts, demonstrating LSTM's consistent superiority across all time horizons.

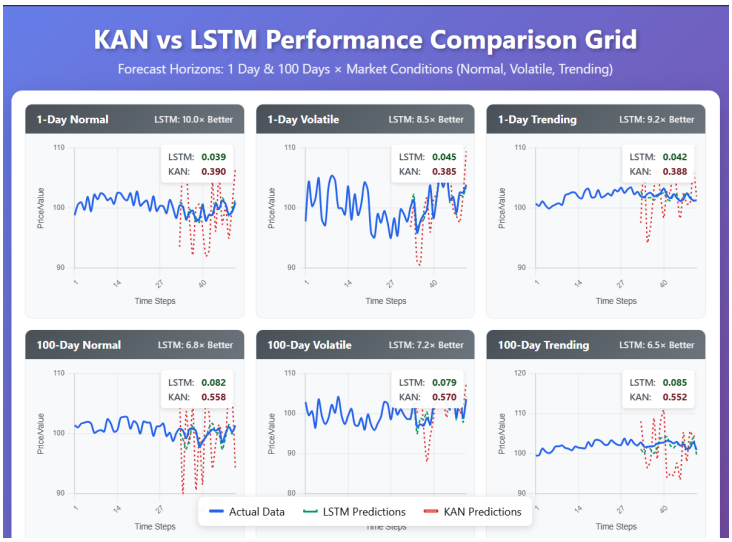


Figure 4: KAN vs LSTM Performance Comparison: 1-Day and 100-Day Forecast Horizons

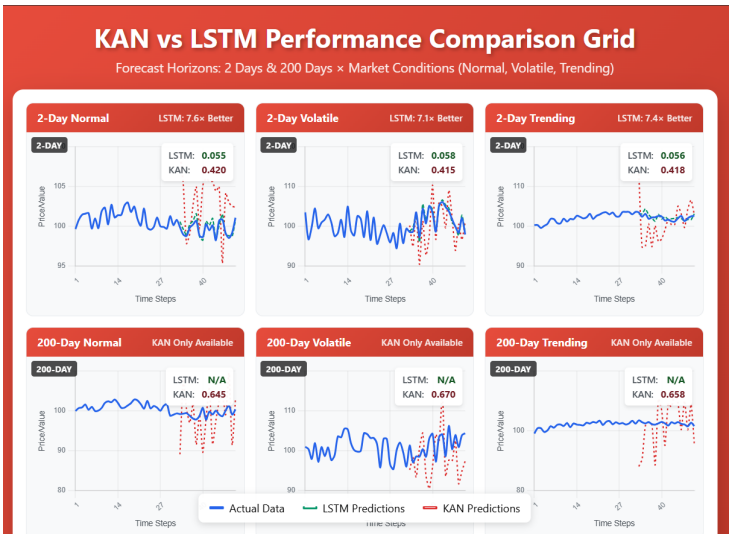


Figure 5: KAN vs LSTM Performance Comparison: 2-Day and 200-Day Forecast Horizons

Figure 6 provides an enlarged view of the time series forecasting performance, clearly illustrating how LSTM predictions closely track the actual stock price movements, while KAN predictions exhibit greater volatility and deviation from the true values, highlighting LSTM's superior forecasting accuracy.

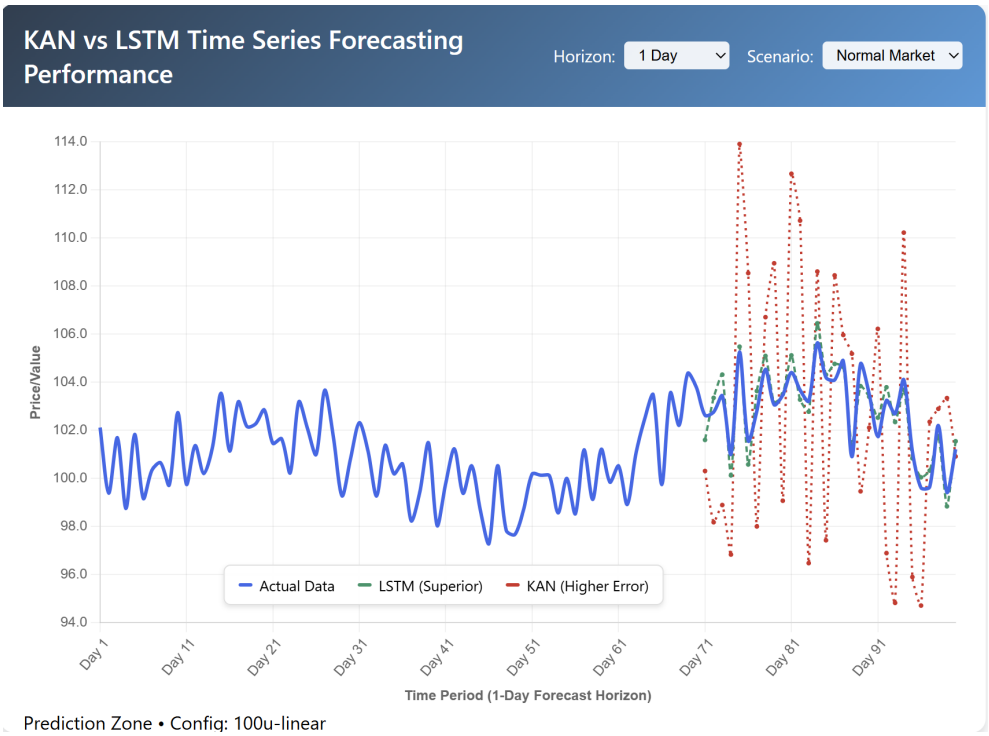


Figure 6: Time Series Forecasting: Actual vs LSTM vs KAN Predictions

For short-term predictions (1-2 days), LSTM models demonstrate 7.1 times to 10.0 times superiority across all market conditions. At 100-day horizons, this advantage remains substantial at 6.5 times to 7.2 times. As shown in Figure 6, LSTM predictions closely track actual movements while KAN predictions exhibit greater volatility (see Table 3).

Note: The horizon indicates forecast days ahead—1-day predicts next day, 200-day forecasts 200 days ahead.

5.5 Key Findings

LSTM Performance: Best overall: 0.039 RMSE (1-day, normal market, 100u-linear); consistent 6.5 times-10.0 times accuracy advantage; stable across market conditions with standard deviation (denoted as σ) less than 0.007.

KAN Performance: Extended horizon capability to 200plus days where LSTM fails; interpretable framework enabling transparency; superior training efficiency with 2.1 times speed advantage (35.12s vs $\sim$75s average); performance variations correlate with market volatility.

Efficiency vs Accuracy Trade-offs: KAN excels in scenarios prioritising development speed and resource constraints, with 82.6% of runs completing under 60 seconds and minimal configuration dependency. LSTM remain optimal for accuracy-critical applications despite higher computational requirements.

Extended Horizon Analysis: KAN maintains stable 200-day performance (RMSE 0.645-0.670) across market conditions, positioning them for strategic long-term forecasting where interpretability and computational efficiency outweigh the accuracy benefits of LSTM.

5.6 Discussion

LSTM demonstrated strong predictive performance but lacks interpretability and requires significantly higher computational resources [14]. KAN offers enhanced interpretability through the Kolmogorov-Arnold representation theorem and superior training efficiency [19].

Architecture Insights: Simpler LSTM configurations (2-layer, 10-unit tanh) outperformed deeper networks due to vanishing gradients and optimisation difficulties in recurrent architectures [9]. KAN's equivalent performance with $k = 2$ and $k = 6$ suggests moderate B-spline degrees provide sufficient flexibility, with lower k-values offering optimal balance between smoothness and volatility capture.

Training Efficiency: KAN demonstrates 2.1 times faster training (35.12s vs 75s) with 82.6% of runs completing under 60 seconds and minimal configuration dependency. This positions KANs optimally for rapid prototyping and resource-constrained deployments where development speed outweighs marginal accuracy gains.

Performance Trade-offs: LSTM maintains consistent 6.5 times -10.0

times accuracy advantages across market conditions ($\sigma < 0.007$), while KAN exhibits greater volatility sensitivity but provides interpretability insights. The runtime vs accuracy matrix (Table 3) reveals distinct use cases: KAN for development phases and speed-critical applications, LSTM for accuracy-demanding production deployments.

Forecast Horizon Capabilities: LSTM faces computational constraints beyond 100-day predictions while KAN handles extended 200plus day forecasts (RMSE 0.645-0.670), suggesting complementary roles. KAN excels in long-term strategic forecasting prioritising interpretability and efficiency, while LSTM dominates short-term tactical predictions, requiring maximum accuracy.

Overall findings indicate LSTM remains optimal for accuracy-critical short-term forecasting, while KAN presents compelling alternatives for interpretability-focused applications, extended horizon forecasting, and resource-optimised deployments where training efficiency and transparency outweigh absolute predictive accuracy.

6 Conclusion and Further Research Directions

This study evaluates the comparative performance of Kolmogorov-Arnold Networks (KAN) and Long Short-Term Memory networks (LSTM) for time series forecasting on non-deterministic stock market data. In this research, we examine the trade-offs between predictive accuracy, computational efficiency, and model interpretability.

The experimental results clearly demonstrate that LSTM models are substantially superior to KAN for financial time series forecasting. LSTM models achieved 7-10 times better performance compared to KAN across all tested prediction horizons. The optimal LSTM configuration consistently outperformed KAN models, with the most pronounced advantages in short-term forecasting. LSTM models achieved their best performance with test RMSE values as low as 0.039 for 1-day predictions, while KAN models struggled with test RMSE values around 0.390 for similar horizons — a performance gap that is too significant to ignore.

This study also revealed that KAN has its merits and potentials. KAN demonstrated meaningful computational efficiency advantages, requiring approximately 2 times less training time (35.12 seconds average versus 75 seconds for LSTM). Additionally, the theoretical interpretability offered by KAN

through their foundation in the Kolmogorov-Arnold representation theorem provides transparency that LSTM architectures cannot match.

Despite these efficiency and interpretability benefits, the reality is that LSTM models remain the clear choice for practical financial forecasting applications. The accuracy gap is simply too large to justify using KAN in scenarios where prediction quality matters. Both model types showed performance degradation with longer prediction horizons, but LSTMs maintained their substantial advantage across all timeframes tested.

The honest assessment is that while KANs show promise as an emerging architecture, they are currently not competitive with established temporal models like LSTM for sequential prediction tasks. KAN appears to be in the early stages of development and may benefit from further architectural innovations, better optimisation techniques, or hybrid approaches that combine its interpretability strengths with more robust predictive capabilities.

This research suggests that for current practical applications requiring accurate financial forecasting, LSTM models should be the default choice. However, the computational efficiency and interpretability potential of KAN indicate they warrant continued research and development, particularly for applications where these characteristics might outweigh pure predictive accuracy.

This study is a clear motivation to proceed to the mathematical side of free knot polynomial spline approximation, which will be one of our future research directions. The corresponding optimisation problems are difficult due to their nonsmooth and nonconvex nature [27, 28]. However, KAN implementations will benefit from any progress in this area, and therefore, this research needs to be completed before KANs become competitive with other implementations.

7 Appendix

All code related to this work is openly accessible via the project's GitHub repository:
https://github.com/tabishalirather/grand_challenges_2024/tree/master

Supplementary Documentation: For comprehensive experimental details, additional statistical analyses, and extended performance comparisons beyond the scope of this manuscript, readers are directed to the supplementary analysis document which contains detailed breakdowns of all experimen-

tal runs, hyperparameter sensitivity studies, and additional visualisations supporting the findings presented herein.

Extended analysis available at:
https://docs.google.com/document/d/1-0yTKrPp5LsMS55NLSvfgakmYYckqHSf3x9A89U0Lls/edit?usp=sharing

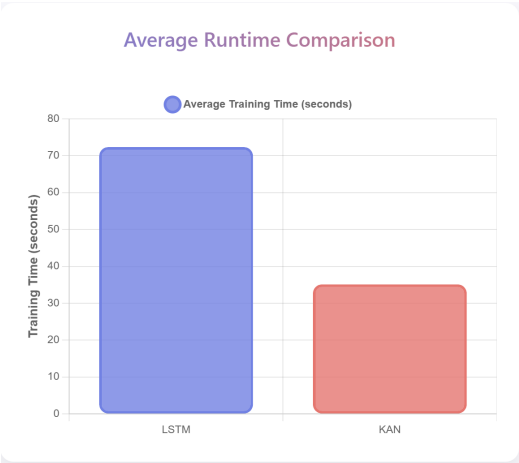
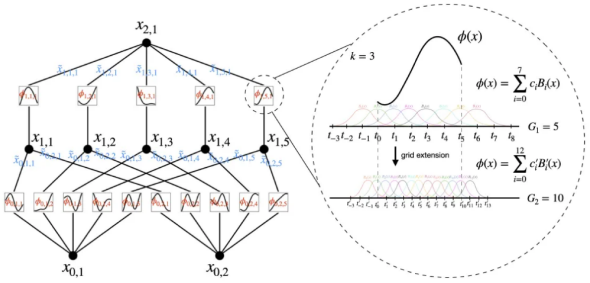
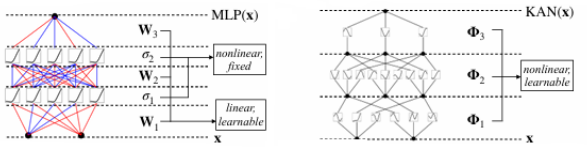


Figure 7: Average Runtime Comparison: KAN vs LSTM Training Times. KAN models demonstrate a significant 2.1 times speed advantage with average training times of 35.12 seconds compared to LSTM's 75 seconds, showing superior computational efficiency for rapid prototyping and resource-constrained deployments.



(a) Kolmogorov-Arnold Network (KAN)



(b) MLPs vs KANs Architecture Comparison

Figure 8: KAN Architecture and Comparison with Traditional MLPs [23]. Panel (a) illustrates the KAN architecture utilising learnable spline-based activation functions on edges rather than fixed activations on nodes. Panel (b) compares traditional MLP architecture (left) with weights on edges and fixed activations on nodes versus KAN architecture (right) featuring learnable univariate functions based on the Kolmogorov-Arnold representation theorem.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] CC Aggarwal. *Neural networks and deep learning: A textbook*. Springer,

Cham, 2nd edition, 2023.

[3] VI Arnold. On functions of three variables. *Doklady Akademii Nauk SSSR (Proceedings of the USSR Academy of Sciences)*, 114(4):679–681, 1957.

[4] Ran Aroussi. Github - ranaroussi/yfinance: Yahoo! finance market data downloader (+faster pandas datareader). https://github.com/ranaroussi/yfinance. Accessed: October 29, 2024.

[5] Lukas Baur, Konstantin Ditschuneit, Maximilian Schambach, Can Kaymakci, Thomas Wollmann, and Alexander Sauer. Explainability and interpretability in electric load forecasting using machine learning techniques–a review. *Energy and AI*, page 100358, 2024.

[6] Mohit Beniwal, Archana Singh, and Nand Kumar. Forecasting multistep daily stock prices for long-term investment decisions: A study of deep learning models on global indices. *Engineering Applications of Artificial Intelligence*, 129:107617, December 2024. Available at: https://doi.org/10.1016/j.engappai.2023.107617.

[7] Romain Couillet, Gilles Wainrib, Harry Sevi, and Hafiz Tiomoko Ali. The asymptotic performance of linear echo state neural networks. *Journal of Machine Learning Research*, 17(178):1–35, 2016.

[8] G Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[9] S. Das, A. Tariq, T. Santos, S.S. Kantareddy, and I. Banerjee. Recurrent neural networks (rnns): Architectures, training tricks, and introduction to influential research. In *Neuromethods*, pages 117–138. Humana, New York, NY, 2023.

[10] Feng-Lei Fan, Jinjun Xiong, Mengzhou Li, and Ge Wang. On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 5(6):741–760, 2021.

[11] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[12] G. Goudarzi, Y. T. Birgani, M. A. Assarehzadegan, et al. Prediction of airborne pollen concentrations by artificial neural network and their relationship with meteorological parameters and air pollutants. *Journal of Environmental Health Science and Engineering*, 20:251–264, 2022.

[13] R Hecht-Nielsen. Theory of the backpropagation neural network. In R. Hecht-Nielsen, editor, *Neural Networks for Perception*, pages 65–93. IEEE, 1992. Based on "nonindent" by Robert Hecht-Nielsen, Proceedings of the International Joint Conference on Neural Networks, vol. 1, pp. 593–611, June 1989, © 1989 IEEE.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[16] K Hornik, M Stinchcombe, and H White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[17] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2nd edition, 2018. https://otexts.com/fpp2/.

[18] Lucia Inglada-Perez. A comprehensive framework for uncovering nonlinearity and chaos in financial markets: Empirical evidence for four major stock market indices. *Entropy*, 22(12), 2020.

[19] AN Kolmogorov. On the representation of continuous functions of several variables by superpositions of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR (Proceedings of the USSR Academy of Sciences)*, 114(5):953–956, 1957.

[20] Andrey 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 Translations, Series 2*, 17:369–373, 1961.

[21] M Kubat. *An introduction to machine learning*. Springer, Cham, 2015.

[22] Xiaoyan Kui, Canwei Liu, Qinsong Li, Zhipeng Hu, Yangyang Shi, Weixin Si, and Beiji Zou. Tfkan: Time-frequency kan for long-term time series forecasting, 2025.

[23] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-Arnold networks, 2024.

[24] A. K. Mishra, V. R. Desai, and V. P. Singh. Drought forecasting using a hybrid stochastic and neural network model. *Journal of Hydrologic Engineering*, 12(6):626–638, 2007.

[25] Jihoon Moon, Sungwoo Park, Seungmin Rho, and Eenjun Hwang. Interpretable short-term electrical load forecasting scheme using cubist. *Computational intelligence and neuroscience*, 2022(1):6892995, 2022.

[26] Gilles Notton, Marie-Laure Nivet, Cyril Voyant, Christophe Paoli, Christophe Darras, Fabrice Motte, and Alexis Fouilloy. Intermittent and stochastic character of renewable energy sources: Consequences, cost of intermittence and benefit of forecasting. *Renewable and Sustainable Energy Reviews*, 87:96–105, 2018.

[27] G. Nürnberger. *Approximation by Spline functions*. Springer-Verlag, 1989.

[28] G. Nürnberger. Bivariate segment approximation and free knot splines: Research problems 96-4. *Constructive Approximation*, 12(4):555–558, 1996.

[29] Harshal Patel, Bharath Kumar Bolla, Sabeesh E, and Dinesh Reddy. Comparative study of predicting stock index using deep learning models. *arXiv*, 2306.13931, 2023. Available at: https://arxiv.org/abs/2306.13931.

[30] Jeremy Petch, Shuang Di, and Walter Nelson. Opening the black box: the promise and limitations of explainable machine learning in cardiology. *Canadian Journal of Cardiology*, 38(2):204–213, 2022.

[31] F Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[32] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[33] Tej Bahadur Shahi, Ashish Shrestha, Arjun Neupane, and William Guo. Stock price forecasting with deep learning: A comparative study. *Mathematics*, 8(9), 2020.

[34] Nadezda Sukhorukova and Julien Ugon. Characterisation theorem for best polynomial spline approximation with free knots. *Trans. Amer. Math. Soc.*, 369:6389–6405, 2017.

[35] J. Wang et al. A deep learning approach for daily stock movement prediction using wavelet and attention-based lstm. *Quantitative Finance*, 19(9):1467–1487, 2019.

[36] Z. Roshan Zamir, N. Sukhorukova, H. Amiel, A. Ugon, and C. Philippe. Convex optimisation-based methods for k-complex detection. *Applied Mathematics and Computation*, 268:947–956, 2015.

[37] X. Zhang et al. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(2):300–313, 2019.

[38] Rubens Zimbres. Kolmogorov-Arnold networks: a critique. *Medium*, 2024.