# TSKANMixer: Kolmogorov–Arnold Networks with MLP-Mixer Model for Time Series Forecasting

**Young-Chae Hong**[1], **Bei Xiao**[1], **Yangho Chen**[1]
[1]Amazon
Seattle, WA 98109
hongych@amazon.com, bxxiao@amazon.com, yanghoc@amazon.com

**Abstract**

Time series forecasting has long been a focus of research across diverse fields, including economics, energy, healthcare, and traffic management. Recent works have introduced innovative architectures for time series models, such as the Time-Series Mixer (TSMixer), which leverages multilayer perceptrons (MLPs) to enhance prediction accuracy by effectively capturing both spatial and temporal dependencies within the data. In this paper, we investigate the capabilities of the Kolmogorov-Arnold Networks (KANs) for time-series forecasting by modifying TSMixer with a KAN layer (TSKANMixer). Experimental results demonstrate that TSKANMixer tends to improve prediction accuracy over the original TSMixer across multiple datasets, ranking among the top-performing models compared to other time series approaches. Our results show that the KANs are promising alternatives to improve the performance of time series forecasting by replacing or extending traditional MLPs.

## Introduction

Time-series analysis is essential across a wide range of domains, including retail (Böse et al. 2017), finance (Taylor 2008), economics (Granger and Newbold 2014), transportation (Chen et al. 2001; Yin et al. 2021), energy (Martín et al. 2010; Qian et al. 2019; Heidrich et al. 2020), healthcare (Bui et al. 2018; Kaushik et al. 2020), and climate (Wu et al. 2023), where understanding and forecasting temporal patterns is crucial for decision-making and planning. In recent years, various deep learning (DL)-based forecasting models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), multi-layer perceptrons (MLPs), and Transformers, have been extensively studied to capture the complexity in real-world time-series datasets that are often multivariate with complex, non-linear dependencies among them (Wang et al. 2024b; Liu and Wang 2024).

However, contrary to the common intuition that DL-based models should be more effective than univariate models, it is shown that Transformer-based models can indeed be significantly worse than simple univariate temporal linear models on many commonly used forecasting benchmarks since they suffer from overfitting (Nie et al. 2022; Zeng et al. 2023). Instead, recent work has demonstrated that simple univariate linear models can outperform such deep learning models on several commonly used academic benchmarks. Recently, Chen et al. (Chen et al. 2023), inspired by the well-known MLP Mixer architecture in computer vision (Tolstikhin et al. 2021), proposed a fully MLP-based architecture for time series forecasting, Time-Series Mixer (TSMixer), by alternatively stacking multiple MLPs to capture temporal information in the time-domain and cross-variate information in the feature-domain. The authors showed that state-of-the-art performance can be achieved without necessarily relying on Transformers by demonstrating TSMixer's superior performance on benchmarks like the M5 dataset.

On the other hand, more recently, Kolmogorov-Arnold Networks (KANs) (Liu et al. 2024) was proposed as a promising alternative to MLPs. Unlike traditional MLPs that have fixed activation functions on nodes, KANs utilize learnable activation functions on edges and perform instead a simple summation on nodes. The authors introduce KANs as a powerful new neural network architecture that can improve performance and interpretability compared to MLPs. This obviously opens opportunities for further improving deep learning models which rely heavily on MLPs (Liu et al. 2024).

Recent research has explored the application of KANs for time-series. Xu et al. (Xu, Chen, and Wang 2024) investigated the use of KANs for time series forecasting and demonstrated that two KAN models significantly outperformed traditional forecasting methods. Similarly, Vaca-Rubio et al. (Vaca-Rubio et al. 2024) showed that KANs outperformed conventional Multi-Layer Perceptrons (MLPs) in a real-world satellite traffic forecasting task, providing more accurate results with considerably fewer learnable parameters. Finally, Genet et al. (Genet and Inzirillo 2024) proposed the adaptation of KANs to temporal sequences by combining recurrent neural networks (RNNs) and KANs. These researches confirm that the idea developed in the original KAN paper works well on real-world use cases and is highly relevant for time series analysis. In this paper, inspired by the KANs, we propose a new neural network architecture, TSKANMixer, by investigating the application of KANs to TSMixer for time series forecasting.

This paper is structured as follows. Section 2 presents the related work, providing fundamental background on KANs

and TSMixer. Section 3 introduces the overall architecture of TSKANMixer, which uses a KAN layer in TSMixer. Computational experiments are presented in Section 4. Finally, conclusions are provided in Section 5.

## Related Work

### Time-Series Mixer (TSMixer)

TSMixer is an MLP-based architecture for time series forecasting (Chen et al. 2023), which analyzes the performance of linear models for time series forecasting rather than RNNs or Transformer-based frameworks and demonstrates its competitive performance on several time series forecasting benchmarks. TSMixer consists of multiple MLP layers across time and feature dimensions (i.e., time-mixing and feature-mixing MLP block) to capture time-domain temporal patterns and feature-domain cross-variate information alternatively with residual connections and batch norm. The residual designs ensure that TSMixer retains the capacity of temporal linear models. In contrast to recent Transformer-based models, the architecture of TSMixer is relatively simple to implement. Despite its simplicity, it demonstrates that TSMixer remains competitive with state-of-the-art models at representative benchmarks (Chen et al. 2023). The detail of TSMixer architecture is shown in Figure 1.

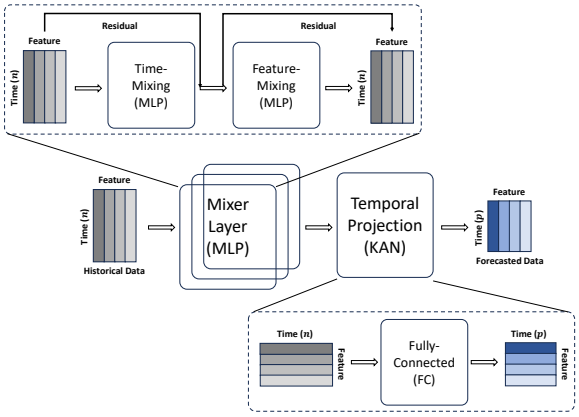



Figure 1: TSMixer for multivariate time series forecasting (Chen et al. 2023)

### Kolmogorov-Arnold Network (KAN)

As MLPs are based on the universal approximation theorem (Cybenko 1989), which states that neural networks with a single hidden layer can approximate any continuous function with finite support, KANs rely on the Kolmogorov-Arnold representation theorem (Arnold 2009a,b). The theorem states that any multivariate continuous function $f(x)$ on a bounded domain, where $x = (x_1, \ldots, x_n)$, can be written as a finite composition of continuous functions of a single variable and the binary operation of addition. Formally, a multivariate continuous function $f(x) : [0, 1]^n \to \mathbb{R}$ can be represented by the finite composition of univariate functions (Liu et al. 2024):

$$f(x) = f(x_1, \ldots, x_n) = \sum_{j=1}^{2n+1} \Phi_j \left( \sum_{i=1}^{n} \phi_{j,i}(x_i) \right) \quad (1)$$

where an outer function is $\Phi_j : \mathbb{R} \to \mathbb{R}$ and an inner function is $\phi_{j,i} : [0, 1] \to \mathbb{R}$.

As a MLP consists of layers where each layer performs a linear transformation followed by a non-linear activation function, a KAN layer can be defined as a matrix $\mathbf{\Phi}$ of univariate functions:

$$\mathbf{\Phi}(\mathbf{x}) = \{\phi_{j,i}\}, \qquad i = \{1, \ldots, n_{in}\},\ j = \{1, \ldots, n_{out}\} \quad (2)$$

where the univariate functions $\phi_{j,i}$ have trainable parameters and $n_{in}$ is the number of inputs and $n_{out}$ is the number of outputs.

Generally, KANs can be expressed by a composition of multiple KAN layers, $y = \mathbf{KAN}(x) = (\mathbf{\Phi_L} \circ \cdots \circ \mathbf{\Phi_1})(x)$ where $L$ is the number of layers. Then, the equation 1 for the Kolmogorov-Arnold representation theorem can be represented by a two-depth KAN layer of shape $[n, 2n + 1, 1]$, consisting of an inner layer with $n_{in} = n$ and $n_{out} = 2n+1$, and an outer layer with $n_{in} = 2n+1$ and $n_{out} = 1$ (Liu et al. 2024).

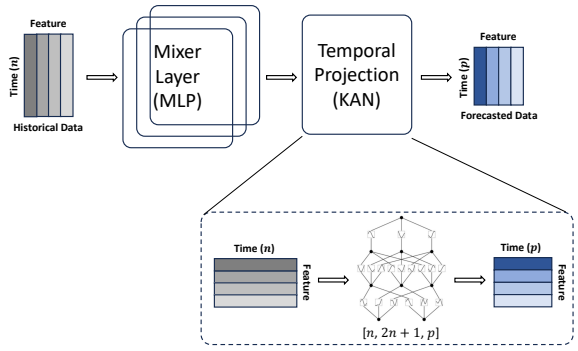
While MLPs employ fixed activation functions on nodes, KANs employ learnable activation functions on edges (Liu et al. 2024). Specifically, KANs learn activation patterns dynamically by replacing traditional linear weights on MLPs with univariate functions parameterized as splines, where a spline is defined by the order $k$ (the degree of the polynomial functions used to interpolate the curve between control points), and the number of intervals $G$ (the number of segments between adjacent control points). During spline interpolation, the control points separated by $G$ intervals are connected to form a smooth curve (Vaca-Rubio et al. 2024). Through learnable activation functions, KANs improve accuracy and interpretability while maintaining comparable or superior performance with more compact architectures across various tasks.

Vaca-Rubio et al. (Vaca-Rubio et al. 2024) demonstrate that KANs consistently outperform MLPs with lower error metrics while achieving better results with reduced computational resources in time series forecasting. However, due to their intrinsic architecture, KANs have $(k + G)$ times more learnable parameters compared to MLPs (Yu, Yu, and Wang 2024). To enhance computational efficiency, several regularization techniques have proven effective in optimizing KAN training (Cheon 2024). Specifically, the incorporation of dropout, weight decay, and batch normalization not only accelerates convergence but also significantly improves the model's generalization capabilities. Additionally, Bayesian optimization can be leveraged to reduce the parameter search space for more efficient training (Snoek, Larochelle, and Adams 2012).

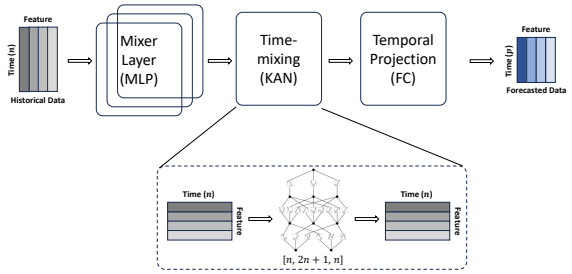
## TSKANMixer Architecture

In this paper, we explore and evaluate the application of a KAN layer to the MLP-based TSMixer architecture. We in-

troduce two architectures of TSKANMixer as illustrated in Figure 2. The proposed models apply the KAN framework to learn complex, non-linear relationships in temporal data. The first proposed architecture, presented in Figure 2a, uses KAN for temporal projection on the time domain as an alternative to a fully-connected layer in TSMixer (Chen et al. 2023). It maps the time series from the input length $L$ to the forecast horizon $H$ by learning the complex relationships between past inputs and future predictions. The second proposed architecture, presented in Figure 2b, extends TSMixer by adding a new KAN-based time mixing layer between mixer layers and temporal projection to intensify the capability to uncover the temporal patterns in time series. All architectures use a two-depth KAN layer.




(a) Version 01: Mixer Layer (=MLP) + Temporal Projection (=KAN)




(b) Version 02: Mixer Layer (=MLP) + Time-mixing Layer (=KAN) + Temporal Projection (=FC)

Figure 2: TSKANMixer Architectures

## Experimental Results

In this section, we evaluate the forecasting performance of the two proposed TSKANMixer architectures, presented in Figures 2a and 2b. We evaluate the performance of our proposed TSKANMixer on commonly used benchmark datasets of multivariate time series that have no missing values and equal lengths across all series: Electricity Transformer Temperature (ETT) long-term forecasting dataset, introduced by Zhou et al. (Zhou et al. 2021), NN5 forecasting competition dataset (Taieb et al. 2012), Computational Intelligence in Forecasting (CIF) 2016 forecasting competition dataset (Štěpnička and Burda 2017), FRED-MD dataset (McCracken and Ng 2016), Exchange dataset(Lai et al. 2018) and Hospital dataset (Hyndman et al. 2008).

**Datasets** The Electricity Transformer Temperature is a crucial indicator in the electric power long-term deployment. This dataset consists of two years of data from two separate counties in China. Each dataset includes the target variable "oil temperature" (OT) and six power load features (Zhou et al. 2021). We use publicly available data that have been pre-processed by Wu et al. (Wu et al. 2021). The NN5 dataset contains 111 time series of daily cash withdrawals from Automated Teller Machines (ATM) in the UK (Godahewa et al. 2021). The Computational Intelligence in Forecasting (CIF) 2016 contains 72 monthly time series. Out of these series, 24 series originate from the banking sector, and the remaining 48 series are artificially generated (Godahewa et al. 2021). In this paper, we use only the 48 series of equal length. The Hospital dataset collects 767 monthly time series showing patient counts related to medical products from January 2000 to December 2006 (Godahewa et al. 2021). The Exchange dataset is the collection of the daily exchange rates of eight foreign countries, including Australia, Britain, Canada, Switzerland, China, Japan, New Zealand and Singapore, ranging from 1990 to 2016 (Lai et al. 2018). The FRED-MD dataset contains 107 monthly time series showing a set of macro-economic indicators from the Federal Reserve Bank (McCracken and Ng 2016). Each dataset is standardized to achieve zero-mean normalization to ensure a fair comparison with TSMixer (Chen et al. 2023). We split the data to ensure that the test set's size closely matches the prediction length, maximizing the amount of data available for training. The statistics of the benchmark datasets and data splits are presented in Table 1.

Table 1: Time Series Forecasting Datasets

| **dataset** | features | time steps | time granularity | data split (train/valid/test) |
|---|---|---|---|---|
| ETTh1/h2 | 7 | 17,420 | 1 hour | 12/4/4 month |
| ETTm1/m2 | 7 | 699,680 | 15 min | 12/4/4 month |
| NN5_daily | 111 | 791 | 1 day | 672/59/59 |
| NN5_weekly | 111 | 113 | 1 month | 96/8/8 |
| CIF_2016 | 48 | 120 | 1 month | 96/12/12 |
| Hospital | 767 | 84 | 1 month | 58/12/12 |
| Exchange | 8 | 7,588 | 1 day | 6829/379/379 |
| FRED_MD | 107 | 728 | 1 month | 698/14/14 |

**Experimental Setup** We focus on evaluating the impact of the KAN layer on TSMixer by comparing it to the original architecture. Thus, we follow the experimental settings in the TSMixer research (Chen et al. 2023) for ETT datasets about data split and hyperparameters. We set the input length $L = 512$ as suggested in Chen et al. (Chen et al. 2023) and evaluate the results for a forecast horizon of $H = 96$. For TSKANMixer's hyperparameters on ETT, we employ a shallower architecture with fewer mixer blocks and a larger batch size compared to TSMixer. Specifically, while TSMixer uses 4 or 6 mixer blocks, TSKANMixer employs only 2 blocks. Similarly, the batch size differs signifi-

cantly: 32 for TSMixer and 320 for TSKANMixer. To utilize PyKAN (Liu et al. 2024), which is implemented in PyTorch, we converted TSMixer's TensorFlow code to PyTorch to implement TSKANMixer. We verified the code conversion by comparing the results with those reported in the original TSMixer paper (Chen et al. 2023) using the ETT dataset, as shown in Table 4 in the Appendix.

In addition, we extensively perform experiments on various publicly available datasets that were not included in the original TSMixer paper (Chen et al. 2023). We conduct a grid search for TSMixer on the hyperparameter spaces: batch size = $\{8, 16, 32\}$, mixer blocks = $\{2, 4, 6\}$, dropout = $\{0.3, 0.5, 0.7, 0.9\}$, feature hidden size = $\{8, 16, 32, 64\}$, and learning rate = $\{0.0001, 0.001\}$. The models are trained for 1000 epochs with proper early stopping. We select the best configuration of TSMixer for the results shown in Table 2. For TSKANMixer's hyperparameters, we conducted manual exploration with limited parameter combinations, as an exhaustive grid search was computationally prohibitive due to the larger parameter space introduced by KAN parameters (e.g., B-spline grids, order of B-spline, and KAN hidden size). Training is also limited to 200 epochs with strict early stopping for the extended datasets. Further details on hyperparameters are summarized in Table 5 in the Appendix.

As benchmark comparisons, we select various state-of-the-art time series models including MLP-based Series-core Fused Time Series (SOFTS) (Han et al. 2024), MLP-based TimeMixer (Wang et al. 2024a), GNN-based Spectral Temporal Graph Neural Network (StemGNN) (Cao et al. 2020), Transformer-based Informer (Zhou et al. 2021), and Simple MLP for multivariate forecasting. All of these models use the same prediction length ($H$) and input length ($L$) for each dataset as we do for TSMixer (Chen et al. 2023). We calculate mean squared error (MSE) and mean absolute error (MAE) as the evaluation metrics. We minimize the mean square error (MSE) or the mean absolute error (MAE) as a loss function and evaluate it over a forecast horizon. All models were trained and tested on an ml.g4dn.xlarge GPU instance, powered by a single NVIDIA T4 GPU with 16GB memory.

**Experiments** We evaluate two versions of TSKANMixer proposed in Figure 2 on popular multivariate forecasting benchmark datasets, comparing them against TSMixer and other state-of-the-art time series models. Table 2 summarizes the comprehensive comparison of 8 time series forecasting models across 10 datasets using MSE and MAE metrics. The top three results for each dataset are highlighted in bold, with the best performance underlined.

Overall, the evaluation results in Table 2 show that no time series forecasting model dominantly outperforms others across all datasets. Among benchmark models, TSMixer and SOFTS demonstrate relatively better performance than other models, followed closely by Informer, while TimeMixer shows moderate performance. MLP and StemGNN exhibit lower accuracy. Notably, StemGNN encounters an out-of-memory issue on the ETT dataset. As a result, StemGNN's performance on the ETT dataset is not reported in Table 2.

The performance improvements of TSKANMixer models compared to TSMixer are indicated by percentage changes ($\Delta\%$) under TSKANMixer in Table 2. For instance, on the ETTh2 dataset, TSKANMixer (v02) shows a substantial 18.97% improvement in MSE and 9.41% in MAE over the TSMixer. The performance improvements from TSMixer show that the predictions obtained by one of TSKANMixer are better than the baseline TSMixer by Chen et al. (Chen et al. 2023) in MSE or MAE across eight datasets, except for CIF 2016 and FRED-MD. In particular, TSKANMixer demonstrates the best or second-best performance on ETTh1, ETTh2, ETTm1, ETTm2, NN5 daily, NN5 weekly, Hospital, and FRED-MD. Both versions of TSKANMixer achieved a top-three ranking 7 times each out of 10 datasets. The result implies that the KAN layer improves prediction performance over the original TSMixer architecture. As an exception, in the CFI 2016 case, all models show poor performance on multivariate predictions, showing significantly high MSE on the normalized dataset. Only StemGNN shows the best performance, and it is the only dataset where StemGNN ranks in the top three. This could imply that the dataset has different time series characteristics that are not captured by current variants of TSKANMixer and TSMixer.

On the other hand, TSKANMixer exhibits significantly slower training times compared to TSMixer due to the incorporation of the KAN layer. According to PyKAN (Liu et al. 2024), the primary bottleneck of KAN is its slow training process, as KANs introduce additional complexity and computations. The study reports that KANs are typically 10 times slower than MLPs, given the same number of parameters. The training limitation constrains the testing of TSKANMixer on larger datasets and hinders extensive hyperparameter tuning in this paper. In addition, TSKANMixer sometimes requires more epochs to complete training than TSMixer on ETT datasets. As a result, there is a case that TSKANMixer's training time is approximately up to 50 times slower than that of the original TSMixer as shown in Table 3.

To illustrate the slow training process, we visualize the training and validation losses over the training epochs for TSMixer and TSKANMixer, as shown in Figure 3. On the ETT datasets, TSMixer starts with a relatively low initial loss value compared to TSKANMixer. It reaches the best epoch at an earlier stage (e.g., less than 50 epochs) and starts overfitting afterwards. This is shown by the increasing validation loss and the divergence between its training loss and validation loss as the number of epochs increases in Figure 3a. On the other hand, TSKANMixer shows a poor initial loss value, but it steadily decreases the validation loss as the number of epochs increases without overfitting quickly (e.g., the best epoch happens after 50 epochs), as shown in Figure 3b. TSKANMixer effectively captures the underlying generalized patterns present in the data, rather than falling into local optima.

## Conclusion and Future Work

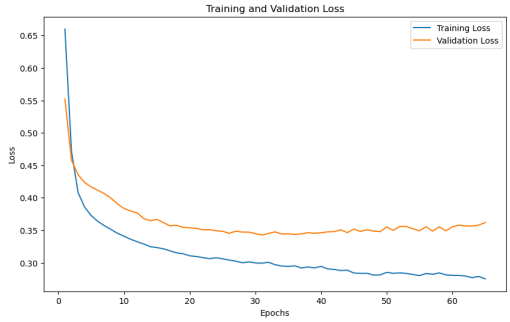
In this paper, we explored the application of KAN to the TSMixer model for time series forecasting and introduced two variants of TSKANMixer. We demonstrate that the

Table 2: Evaluation results on the public time-series datasets

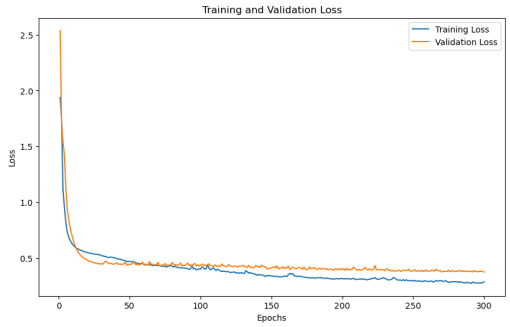
| | | | TSKANMixer (v01) | | TSKANMixer (v02) | | TSMixer | | SOFTS | | TimeMixer | | MLP | | StemGNN | | Informer | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **dataset** | L | H | MSE (Δ%) | MAE (Δ%) | MSE (Δ%) | MAE (Δ%) | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTh1 | 512 | 96 | **0.285** (33.57%) | 0.398 (2.69%) | **0.296** (31.00%) | 0.405 (0.98%) | 0.429 | 0.409 | 0.609 | 0.441 | 0.516 | 0.415 | 0.517 | 0.595 | - | - | **0.337** | 0.403 |
| ETTh2 | 512 | 96 | 0.199 (-2.05%) | 0.334 (1.76%) | **0.158** (18.97%) | 0.308 (9.41%) | **0.195** | 0.340 | **0.135** | 0.277 | 0.593 | 0.552 | 0.418 | 0.529 | - | - | 0.208 | 0.373 |
| ETTm1 | 512 | 96 | **0.190** (34.26%) | 0.296 (13.20%) | 0.281 (2.77%) | 0.348 (-2.05%) | 0.289 | 0.341 | **0.211** | 0.307 | 0.271 | 0.355 | 0.406 | 0.378 | - | - | **0.259** | 0.377 |
| ETTm2 | 512 | 96 | **0.131** (9.66%) | 0.268 (3.60%) | **0.109** (24.83%) | 0.251 (9.71%) | 0.145 | 0.278 | 0.148 | 0.279 | **0.108** | 0.245 | 0.240 | 0.398 | - | - | 0.215 | 0.362 |
| NN5_daily | 56 | 56 | **0.521** (-1.36%) | 0.498 (-1.01%) | **0.506** (1.56%) | 0.485 (1.62%) | **0.514** | 0.493 | 0.545 | 0.506 | 0.627 | 0.582 | 0.641 | 0.582 | 0.561 | 0.515 | 0.544 | 0.521 |
| NN5_weekly | 16 | 8 | **0.878** (2.34%) | 0.731 (1.08%) | **0.897** (0.22%) | 0.736 (0.41%) | 0.899 | 0.739 | 0.938 | 0.771 | **0.901** | 0.736 | 1.195 | 0.883 | 1.758 | 1.014 | 1.177 | 0.859 |
| CIF_2016 | 24 | 12 | 3.631 (-34.58%) | 1.026 (-31.37%) | 2.936 (-8.82%) | 0.895 (-14.59%) | **2.698** | 0.781 | **2.585** | 0.687 | 3.736 | 0.934 | 4.963 | 1.241 | **2.475** | 0.760 | 5.275 | 1.385 |
| Hospital | 24 | 12 | **1.429** (11.08%) | 0.928 (6.64%) | 1.556 (3.17%) | 0.979 (1.51%) | 1.607 | 0.994 | **1.338** | 0.875 | 1.525 | 0.939 | **1.454** | 0.939 | 1.475 | 0.929 | 1.784 | 1.031 |
| Exchange | 60 | 30 | 0.017 (5.56%) | 0.099 (7.47%) | **0.016** (11.11%) | 0.094 (12.15%) | 0.018 | 0.107 | **0.011** | 0.084 | 0.025 | 0.115 | 0.139 | 0.295 | 1.802 | 1.060 | **0.015** | 0.088 |
| FRED_MD | 48 | 12 | **0.037** (-5.71%) | 0.133 (-6.4%) | **0.036** (-2.86%) | 0.125 (0%) | **0.035** | 0.125 | 0.052 | 0.122 | 0.046 | 0.126 | 0.126 | 0.255 | 0.101 | 0.202 | 0.049 | 0.145 |

Table 3: Computational Time

| | | | TSKANMixer (v01) | | TSKANMixer (v02) | | TSMixer | |
|---|---|---|---|---|---|---|---|---|
| **dataset** | L | H | time/epoch (sec) | training time (sec) | time/epoch (sec) | training time (sec) | timee/poch (sec) | training time (sec) |
| ETTh1 | 512 | 96 | 21.29 | 4885.08 | 63.70 | 11869.71 | 4.10 | 263.01 |
| ETTh2 | 512 | 96 | 40.69 | 12276.67 | 63.59 | 19302.69 | 4.74 | 312.52 |
| ETTm1 | 512 | 96 | 88.81 | 11662.25 | 264.63 | 39819.87 | 26.83 | 1282.46 |
| ETTm2 | 512 | 96 | 171.43 | 19242.63 | 263.53 | 63953.10 | 27.26 | 2032.07 |
| NN5_daily | 56 | 56 | 25.82 | 1884.76 | 22.80 | 1869.92 | 0.62 | 135.86 |
| NN5_weekly | 16 | 8 | 6.19 | 1354.61 | 3.21 | 125.01 | 0.08 | 28.89 |
| CIF_2016 | 24 | 12 | 2.97 | 204.98 | 4.03 | 454.99 | 0.07 | 59.96 |
| Hospital | 24 | 12 | 20.32 | 589.23 | 15.09 | 2489.44 | 0.04 | 11.9 |
| Exchange | 60 | 30 | 5.04 | 181.33 | 9.70 | 339.55 | 10.18 | 2421.82 |
| FRED_MD | 48 | 12 | 30.89 | 1730.17 | 19.98 | 3036.34 | 0.33 | 181.19 |




(a) TSMixer




(b) TSKANMixer

Figure 3: Training and validation over epochs on ETTh2

TSKANMixer models generally improve prediction performance over the original TSMixer models. This improvement is achieved by either replacing the fully-connected layer with KAN in temporal projection or by adding a time-mixing layer with KAN. However, we also note that the KAN layer slows down the training process. This work highlights the promising application of KANs in time series analysis. We hope these results provide insights for future research on KAN for time-series forecasting models to improve the capability to capture complex patterns in time series data.

Future work could explore improving the training time for a generalized architecture having wider and deeper KANs beyond the current two-layer model. Additionally, developing a more efficient KAN implementation would facilitate comprehensive hyperparameter tuning on TSKANMixer, potentially unlocking the full potential of KAN-based models. Finally, further exploiting the interpretability and robustness of KAN-based models through symbolic regression could open opportunities to develop more effective and efficient time series models.

## References

Arnold, V. I. 2009a. On functions of three variables. *Collected Works: Representations of Functions, Celestial Mechanics and KAM Theory, 1957–1965*, 5–8.

Arnold, V. I. 2009b. On the representation of functions of several variables as a superposition of functions of a smaller number of variables. *Collected works: Representations of functions, celestial mechanics and KAM theory, 1957–1965*, 25–46.

Böse, J.-H.; Flunkert, V.; Gasthaus, J.; Januschowski, T.; Lange, D.; Salinas, D.; Schelter, S.; Seeger, M.; and Wang, Y. 2017. Probabilistic demand forecasting at scale. *Proceedings of the VLDB Endowment*, 10(12): 1694–1705.

Bui, C.; Pham, N.; Vo, A.; Tran, A.; Nguyen, A.; and Le, T. 2018. Time series forecasting for healthcare diagnosis and prognostics with the focus on cardiovascular diseases. In *6th International Conference on the Development of Biomedical Engineering in Vietnam (BME6) 6*, 809–818. Springer.

Cao, D.; Wang, Y.; Duan, J.; Zhang, C.; Zhu, X.; Huang, C.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; et al. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in neural information processing systems*, 33: 17766–17778.

Chen, C.; Petty, K.; Skabardonis, A.; Varaiya, P.; and Jia, Z. 2001. Freeway performance measurement system: mining loop detector data. *Transportation research record*, 1748(1): 96–102.

Chen, S.-A.; Li, C.-L.; Yoder, N.; Arik, S. O.; and Pfister, T. 2023. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*.

Cheon, J. 2024. Improving Computational Efficiency in Convolutional Kolmogorov-Arnold Networks. *Neural Computing and Applications*, 37(1): 15–30.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4): 303–314.

Genet, R.; and Inzirillo, H. 2024. Tkan: Temporal kolmogorov-arnold networks. *arXiv preprint arXiv:2405.07344*.

Godahewa, R.; Bergmeir, C.; Webb, G. I.; Hyndman, R. J.; and Montero-Manso, P. 2021. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643*.

Granger, C. W. J.; and Newbold, P. 2014. *Forecasting economic time series*. Academic press.

Han, L.; Chen, X.-Y.; Ye, H.-J.; and Zhan, D.-C. 2024. SOFTS: Efficient Multivariate Time Series Forecasting with Series-Core Fusion. *arXiv preprint arXiv:2404.14197*.

Heidrich, B.; Turowski, M.; Ludwig, N.; Mikut, R.; and Hagenmeyer, V. 2020. Forecasting energy time series with profile neural networks. In *Proceedings of the eleventh acm international conference on future energy systems*, 220–230.

Hyndman, R.; Koehler, A. B.; Ord, J. K.; and Snyder, R. D. 2008. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media.

Kaushik, S.; Choudhury, A.; Sheron, P. K.; Dasgupta, N.; Natarajan, S.; Pickett, L. A.; and Dutt, V. 2020. AI in healthcare: time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in big data*, 3: 4.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, 95–104.

Liu, X.; and Wang, W. 2024. Deep Time Series Forecasting Models: A Comprehensive Survey. *Mathematics*, 12(10): 1504.

Liu, Z.; Wang, Y.; Vaidya, S.; Ruehle, F.; Halverson, J.; Soljačić, M.; Hou, T. Y.; and Tegmark, M. 2024. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*.

Martín, L.; Zarzalejo, L. F.; Polo, J.; Navarro, A.; Marchante, R.; and Cony, M. 2010. Prediction of global solar irradiance based on time series analysis: Application to solar thermal power plants energy production planning. *Solar Energy*, 84(10): 1772–1781.

McCracken, M. W.; and Ng, S. 2016. FRED-MD: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, 34(4): 574–589.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*.

Qian, Z.; Pei, Y.; Zareipour, H.; and Chen, N. 2019. A review and discussion of decomposition-based hybrid models for wind energy forecasting applications. *Applied energy*, 235: 939–953.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, 2951–2959.

Štěpnička, M.; and Burda, M. 2017. On the results and observations of the time series forecasting competition CIF 2016. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1–6. IEEE.

Taieb, S. B.; Bontempi, G.; Atiya, A. F.; and Sorjamaa, A. 2012. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert systems with applications*, 39(8): 7067–7083.

Taylor, S. J. 2008. *Modelling financial time series*. world scientific.

Tolstikhin, I. O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. 2021. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34: 24261–24272.

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

Wang, S.; Wu, H.; Shi, X.; Hu, T.; Luo, H.; Ma, L.; Zhang, J. Y.; and Zhou, J. 2024a. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*.

Wang, Y.; Wu, H.; Dong, J.; Liu, Y.; Long, M.; and Wang, J. 2024b. Deep Time Series Models: A Comprehensive Survey and Benchmark. *arXiv preprint arXiv:2407.13278*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34: 22419–22430.

Wu, H.; Zhou, H.; Long, M.; and Wang, J. 2023. Interpretable weather forecasting for worldwide stations with a unified deep model. *Nature Machine Intelligence*, 5(6): 602–611.

Xu, K.; Chen, L.; and Wang, S. 2024. Kolmogorov-Arnold Networks for Time Series: Bridging Predictive Power and Interpretability. *arXiv preprint arXiv:2406.02496*.

Yin, X.; Wu, G.; Wei, J.; Shen, Y.; Qi, H.; and Yin, B. 2021. Deep learning on traffic prediction: Methods, analysis, and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 23(6): 4927–4943.

Yu, R.; Yu, W.; and Wang, X. 2024. KAN or MLP: A Fairer Comparison. *arXiv preprint arXiv:2407.16674*.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 11121–11128.

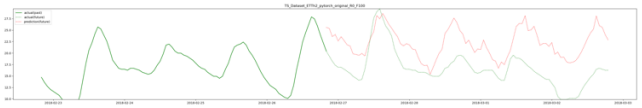
Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11106–11115.

## Appendix A. TSMixer Implemenation

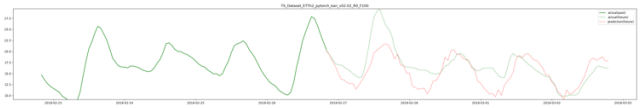
Table 4: TSMixer Comparison on ETT

| | | | TSMixer (TensorFlow) | | TSMixer (PyTorch) | |
|---|---|---|---|---|---|---|
| **dataset** | L | H | MSE | MAE | MSE | MAE |
| ETTh1 | 512 | 96 | 0.361 | 0.392 | 0.429 | 0.409 |
| ETTh2 | 512 | 96 | 0.274 | 0.341 | 0.195 | 0.340 |
| ETTm1 | 512 | 96 | 0.285 | 0.339 | 0.289 | 0.341 |
| ETTm2 | 512 | 96 | 0.163 | 0.252 | 0.145 | 0.278 |

## Appendix B. Forecasted Values Visualization



(a) Forecasted values (H=96) using TSMixer (MAE = 0.340) on ETTh2



(b) Forecasted values (H=96) using TSKANMixer v02 (MAE = 0.327) on ETTh2

Figure 4: Visualization of predictions: true (green) and forecasted (red) values for the target (OT) feature

## Appendix C. Hyperparameters

Table 5: Hyerparamter configurations for TSMixer

| | | | TSMixer | | | | |
|---|---|---|---|---|---|---|---|
| **dataset** | L | H | Batch | Blocks | Dropout | Hidden size | Learing rate |
| ETTh1 | 512 | 96 | 32 | 2 | 0.3 | 64 | 0.0001 |
| ETTh2 | 512 | 96 | 32 | 4 | 0.3 | 64 | 0.0001 |
| ETTm1 | 512 | 96 | 32 | 6 | 0.9 | 16 | 0.0001 |
| ETTm2 | 512 | 96 | 32 | 6 | 0.3 | 16 | 0.0001 |
| NN5_daily | 56 | 56 | 16 | 6 | 0.3 | 64 | 0.001 |
| NN5_weekly | 16 | 8 | 16 | 6 | 0.9 | 64 | 0.001 |
| CIF_2016 | 24 | 12 | 8 | 4 | 0.9 | 8 | 0.001 |
| Hospital | 24 | 12 | 8 | 6 | 0.5 | 16 | 0.001 |
| Exchange | 60 | 30 | 8 | 6 | 0.5 | 64 | 0.001 |
| FRED_MD | 48 | 12 | 32 | 6 | 0.3 | 16 | 0.001 |

Table 6: Hyerparamter configurations for TSKANMixer (v01)

| | | | TSKANMixer (v01) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **dataset** | L | H | Batch | Blocks | Dropout | Hidden size | Learing rate | KAN_dim | KAN_grid | KAN_k |
| ETTh1 | 512 | 96 | 320 | 2 | 0.3 | 64 | 0.0001 | 512 | 5 | 3 |
| ETTh2 | 512 | 96 | 320 | 2 | 0.3 | 64 | 0.0001 | 1025 | 5 | 3 |
| ETTm1 | 512 | 96 | 320 | 2 | 0.3 | 64 | 0.0001 | 512 | 5 | 3 |
| ETTm2 | 512 | 96 | 320 | 4 | 0.3 | 64 | 0.0001 | 1025 | 5 | 3 |
| NN5_daily | 56 | 56 | 16 | 4 | 0.3 | 32 | 0.001 | 56 | 10 | 2 |
| NN5_weekly | 16 | 8 | 8 | 6 | 0.7 | 111 | 0.001 | 33 | 3 | 3 |
| CIF_2016 | 24 | 12 | 16 | 2 | 0.9 | 64 | 0.001 | 12 | 1 | 10 |
| Hospital | 24 | 12 | 8 | 2 | 0.5 | 767 | 0.001 | 24 | 10 | 2 |
| Exchange | 60 | 30 | 128 | 4 | 0.3 | 4 | 0.001 | 15 | 10 | 3 |
| FRED_MD | 48 | 12 | 32 | 4 | 0.3 | 16 | 0.001 | 12 | 10 | 7 |

Table 7: Hyerparamter configurations for TSKANMixer (v02)

| | | | TSKANMixer (v02) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **dataset** | L | H | Batch | Blocks | Dropout | Hidden size | Learing rate | KAN_dim | KAN_grid | KAN_k |
| ETTh1 | 512 | 96 | 320 | 2 | 0.3 | 64 | 0.0001 | 1025 | 5 | 3 |
| ETTh2 | 512 | 96 | 320 | 2 | 0.3 | 64 | 0.0001 | 1025 | 5 | 3 |
| ETTm1 | 512 | 96 | 320 | 2 | 0.3 | 64 | 0.0001 | 1025 | 5 | 3 |
| ETTm2 | 512 | 96 | 320 | 2 | 0.3 | 64 | 0.0001 | 1025 | 5 | 3 |
| NN5_daily | 56 | 56 | 16 | 4 | 0.9 | 32 | 0.001 | 14 | 2 | 3 |
| NN5_weekly | 16 | 8 | 8 | 6 | 0.7 | 32 | 0.001 | 8 | 7 | 3 |
| CIF_2016 | 24 | 12 | 16 | 4 | 0.4 | 24 | 0.001 | 12 | 7 | 3 |
| Hospital | 24 | 12 | 8 | 6 | 0.3 | 16 | 0.001 | 3 | 10 | 2 |
| Exchange | 60 | 30 | 32 | 2 | 0.3 | 16 | 0.001 | 15 | 10 | 2 |
| FRED_MD | 48 | 12 | 16 | 2 | 0.5 | 16 | 0.001 | 5 | 5 | 2 |