Transparent Networks for Multivariate Time Series

Minkyu Kim[1], **Suan Lee**[2*], **Jinho Kim**[3,4*]
[1]Ziovision Co., Ltd.
[2]Semyung University
[3]Kangwon National University
[4]Korea Information Systems Consulting & Audit Co., Ltd.
minkyu.kim@ziovision.co.kr, suanlee@semyung.ac.kr, jhkim@kangwon.ac.kr

Abstract

Transparent models, which provide inherently interpretable predictions, are receiving significant attention in high-stakes domains. However, despite much real-world data being collected as time series, there is a lack of studies on transparent time series models. To address this gap, we propose a novel transparent neural network model for time series called Generalized Additive Time Series Model (GATSM). GATSM consists of two parts: 1) independent feature networks to learn feature representations, and 2) a transparent temporal module to learn temporal patterns across different time steps using the feature representations. This structure allows GATSM to effectively capture temporal patterns and handle varying-length time series while preserving transparency. Empirical experiments show that GATSM significantly outperforms existing generalized additive models and achieves comparable performance to black-box time series models, such as recurrent neural networks and Transformer. In addition, we demonstrate that GATSM finds interesting patterns in time series.

Code — https://github.com/gim4855744/GATSM
Extended version — https://arxiv.org/abs/2410.10535

1 Introduction

Artificial neural networks excel at learning complex representations and demonstrate remarkable predictive performance across various fields. However, their complexity makes interpreting the decision-making processes of neural network models challenging. Consequently, researchers have widely studied post-hoc explainable artificial intelligence (XAI) methods, which explain the predictions of trained black-box models, in recent years (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Selvaraju et al. 2017; Mothilal, Sharma, and Tan 2020). XAI methods are generally effective at providing humans with understandable explanations of model predictions. However, they may produce incorrect and unfaithful explanations of the underlying black-box model and cannot provide actual contributions of input features to model predictions (Rudin 2018, 2019; Rahnama and Boström 2019; Liu et al. 2021; Akhavan Rahnama 2023; Rahnama et al. 2024). Therefore, their applicability to high-stakes domains – such as healthcare and fraud detection, where faithfulness to the underlying model and actual contributions of features are important – is limited.

Due to these limitations, transparent (i.e., inherently interpretable) models are attracting attention as alternatives to XAI in high-stakes domains (Agarwal et al. 2021; Chang, Caruana, and Goldenberg 2022; Radenovic, Dubey, and Mahajan 2022). Modern transparent models typically adhere to the *generalized additive model* (GAM) framework (Hastie and Tibshirani 1986). A GAM consists of independent functions, each corresponding to an input feature, and makes predictions as a linear combination of these functions (e.g., the sum of all functions). Therefore, each function reflects the contribution of its respective feature. For this reason, interpreting GAMs is straightforward, making them widely used in various fields, such as healthcare (Caruana et al. 2015; Chang et al. 2021), survival analysis (Utkin, Satyukov, and Konstantinov 2022), and model bias discovery (Agarwal et al. 2021; Tan et al. 2018; Kim, Choi, and Kim 2022). However, despite much real-world data being collected as time series, research on GAMs for time series remains scarce. Consequently, the applicability of GAMs in real-world scenarios is still limited.

To overcome this limitation, we propose a novel transparent model for multivariate time series called Generalized Additive Time Series Model (GATSM). GATSM consists of independent feature networks to learn feature representations and a transparent temporal module to learn temporal patterns. Since employing distinct networks across different time steps requires a massive amount of learnable parameters, the feature networks in GATSM share the weights across all time steps, while the temporal module learns temporal patterns. GATSM then generates final predictions by integrating the feature representations with the temporal information from the temporal module. This strategy allows GATSM to effectively capture temporal patterns and handle dynamic-length time series while preserving transparency. Additionally, this approach facilitates the separate extraction of time-independent feature contributions, the importance of individual time steps, and time-dependent feature contributions through the feature functions, temporal module, and final prediction. To demonstrate the effectiveness of GATSM, we conducted empirical experiments on various real-world and synthetic time series datasets. The ex-

*Corresponding author.

perimental results show that GATSM significantly outperforms existing GAMs and achieves comparable performance to black-box time series models, such as recurrent neural networks and Transformer (Vaswani et al. 2017). In addition, we provide visualizations of GATSM's predictions to demonstrate that GATSM finds interesting patterns in time series.

A detailed discussion of related works is provided in Appendix C and we summarize the contributions of this paper as follows: 1) We propose a novel transparent neural network model for multivariate time series, called GATSM. To the best of our knowledge, GATSM is the first transparent model capable of capturing temporal patterns while preserving transparency. 2) We conduct extensive experiments to validate the performance of GATSM. The experimental results demonstrate that GATSM significantly outperforms existing transparent models and achieves performance comparable to black-box models like Transformer. 3) We provide visual interpretations of GATSM's predictions. These visualizations illustrate that GATSM successfully captures complex temporal patterns and derive multiple forms of interpretations, including time-step importance, global feature contributions, local time-independent feature contributions, and local time-dependent feature contributions. Such interpretations significantly enhance the understanding of both model behaviors and underlying dataset characteristics. 4) We thoroughly discuss the potential applications and limitations of GATSM, identifying directions for future works.

2 Problem Statement

The simple linear model is designed to fit the conditional expectation $g\left(\mathbb{E}\left(y \mid \mathbf{x}\right)\right) = \sum_{i=1}^{M} x_i w_i$, where $g(\cdot)$ is a link function, M indicates the number of input features, y is the target value for the given input features $\mathbf{x} \in \mathbb{R}^M$, and $w_i \in \mathbb{R}$ is the learnable weight for x_i. This model captures only linear relationships between the target y and the inputs $\mathbf{x}$. To address this limitation, GAM (Hastie and Tibshirani 1986) extends the simple linear model to the generalized form as follows:

$$g\left(\mathbb{E}\left(y \mid \mathbf{x}\right)\right) = \sum_{i=1}^{M} f_i\left(x_i\right), \tag{1}$$

where each $f_i(\cdot)$ is a function that models the effect of a single feature, referred as a feature function. Typically, $f_i\left(\cdot\right)$ becomes a non-linear function such as a decision tree or neural network to capture non-linear relationships. However, this form of GAM cannot handle time series data. One straightforward method to extend GAM to time series, adopted in NATM (Jo and Kim 2023), is applying distinct feature functions to each time step and summing them to produce predictions:

$$g\left(\mathbb{E}\left(y_t \mid \mathbf{X}_{:t}\right)\right) = \sum_{i=1}^{t} \sum_{j=1}^{M} f_{i,j}\left(x_{i,j}\right), \tag{2}$$

where $\mathbf{X} \in \mathbb{R}^{T \times M}$ is a time series with T time steps and M features, and t is the current time step. This method can handle time series data as input but fails to capture temporal patterns because $f_{i,j}\left(\cdot\right)$ still does not interact with previous time steps. To overcome this problem, we suggest a new form of GAM for time series.

	Time series	Temporal pattern	Dynamic length
Tabular GAMs	✗	✗	✗
NATM	✓	✗	✗
GATSM (our)	✓	✓	✓

Table 1: Advantages of GATSM. Unlike existing tabular GAMs that cannot handle time series data, and NATM which only supports fixed-length time series without capturing temporal dynamics, GATSM effectively handles varying-length time series and captures complex temporal patterns.

Definition 3.1 *GAMs for time series, which capture temporal patterns hold the following form*:

$$g\left(\mathbb{E}\left(y_t \mid \mathbf{X}_{:t}\right)\right) = \sum_{i=1}^{t} \sum_{j=1}^{M} f_{i,j}\left(x_{i,j}, \mathbf{X}_{:t}\right). \tag{3}$$

In Equation (3), $f_{i,j}\left(\cdot,\cdot\right)$ can capture interactions between current and previous time steps. Therefore, GAMs adhering to Definition 3.1 can capture temporal patterns. However, implementing such a model while maintaining transparency poses challenges. In the following section, we will describe our approach to implementing a GAM that satisfies Definition 3.1. To the best of our knowledge, no existing literature addresses Definition 3.1. Table 1 shows the advantages of our GATSM compared to existing GAMs.

3 Generalized Additive Time Series Model

Figure 1 shows the overall architecture of GATSM. Our model has two modules: 1) feature networks, called time-sharing neural basis model (NBM), for learning feature representations, and 2) masked multi-head attention (MHA) for learning temporal patterns.

Time-Sharing NBM

Assume a time series $\mathbf{X} \in \mathbb{R}^{T \times M}$ with T time steps and M features. Applying existing GAMs defined in Equation 1 to this time series require $T \times M$ feature functions, which becomes problematic when dealing with large T or M due to increased model size. This limits the applicability of GAMs to real-world datasets. To overcome this problem, we extend NBM (Radenovic, Dubey, and Mahajan 2022) to time series as:

$$\tilde{x}_{i,j} = f_j\left(x_{i,j}\right) = \sum_{k=1}^{B} h_k\left(x_{i,j}\right) w_{j,k}^{nbm}. \tag{4}$$

We refer to this extended version of NBM as time-sharing NBM. Time-sharing NBM has B basis functions, with each basis function $h_k(\cdot)$ taking a feature $x_{i,j}$ as input. The feature-specific weight $w_{j,k}^{nbm}$ then projects the basis to the transformed feature $\tilde{x}_{i,j}$. As Equation (4) depicts, the basis functions are shared across all features and time steps, drastically reducing the number of required feature functions

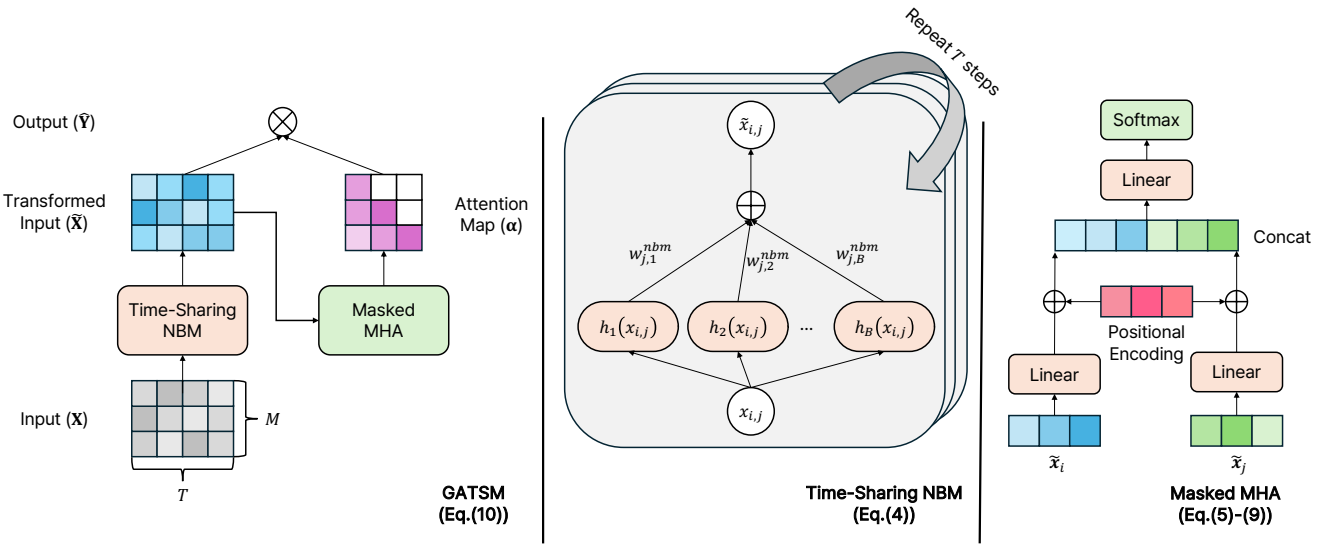


Figure 1: Architecture of GATSM. GATSM comprises two components: Time-Sharing NBM and Masked MHA. Time-Sharing NBM learns nonlinear representations of input features without considering temporal patterns, using parameters shared across time steps. Masked MHA then captures temporal patterns based on these feature representations.

from $T \times M$ to B. We use $B = 100$ and implement $h_k(\cdot)$ using multi-layer perceptron (MLP).

Masked MHA

GATSM employs MHA to learn temporal patterns. Although the dot product attention (Vaswani et al. 2017) is popular, the simple dot product attention has low expressive power (Veličković et al. 2018). Therefore, we adopt the 2-layer attention mechanism proposed by (Veličković et al. 2018) to GATSM. We first transform $\tilde{\mathbf{x}}_i = [\tilde{x}_{i,1}, \tilde{x}_{i,2}, \cdots, \tilde{x}_{i,M}] \in \mathbb{R}^M$ produced by Equation (4) as follows:

$$\mathbf{v}_i = \tilde{\mathbf{x}}_i^\intercal \mathbf{Z} + \mathbf{pe}_i, \tag{5}$$

where $\mathbf{Z} \in \mathbb{R}^{M \times D}$ is a learnable weight, $\mathbf{pe}_i = [pe_{i,1}, pe_{i,2}, \cdots, pe_{i,D}] \in \mathbb{R}^D$ is the positional encoding for the i-th step, and D indicates the hidden size. The positional encoding is defined as follows:

$$pe_{i,j} = \begin{cases} \sin\left(\frac{i}{10000^{2j/D}}\right) & \text{if } j \bmod 2 = 1, \\ \cos\left(\frac{i}{10000^{2j/D}}\right) & \text{otherwise.} \end{cases} \tag{6}$$

The positional encoding helps GATSM effectively capture temporal patterns. While learnable position embedding also works in GATSM, we recommend the sinusoidal positional encoding because learnable position embedding requires prior knowledge of the maximum length of a time series, which is often unknown in real-world settings. For example, in real-world hospital settings, a patient's length of stay continuously increases, and the discharge date is unpredictable. After computing $\mathbf{v}_i$, we calculate the attention scores as follows:

$$e_{k,i,j} = \sigma\left(\left[\mathbf{v}_i \mid \mathbf{v}_j\right]^\intercal \mathbf{w}_k^{attn}\right) m_{i,j}, \tag{7}$$

$$a_{k,i,j} = \frac{\exp\left(e_{k,i,j}\right)}{\sum_{t=1}^{T} \exp\left(e_{k,i,t}\right)}, \tag{8}$$

where k is the index of the attention head, $\sigma(\cdot)$ is a nonlinear activation function, $\mathbf{w}_k^{attn} \in \mathbb{R}^{2D}$ is a learnable weight, and $m_{i,j} \in \mathbb{R}$ is the mask value used to block future information. The time mask is defined as follows:

$$m_{i,j} = \begin{cases} 1 & \text{if } i \geq j, \\ -\infty & \text{otherwise.} \end{cases} \tag{9}$$

Inference

GATSM produces its prediction by combining the transformed features from the time-sharing NBM with the attention scores from the masked MHA.

$$\hat{y}_t = \sum_{k=1}^{K} \mathbf{a}_{k,t}^\intercal \tilde{\mathbf{X}} \mathbf{w}_k^{out}, \tag{10}$$

where K is the number of attention heads, $\mathbf{a}_{k,t} = [a_{k,i,1}, a_{k,i,2}, \cdots, a_{k,i,T}] \in \mathbb{R}^T$ is the attention map defined in Equation (8), $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \cdots, \tilde{\mathbf{x}}_T] \in \mathbb{R}^{T \times M}$ are the transformed features defined in Equation (4), and $\mathbf{w}_k^{out} \in \mathbb{R}^M$ is the learnable output weight.

Interpretability

We can rewrite Equation (10) as the following scalar form:

$$\begin{aligned} &\sum_{k=1}^{K} \mathbf{a}_{k,t}^\intercal \tilde{\mathbf{X}} \mathbf{w}_k^{out} \\ &\quad = \sum_{u=1}^{t} \sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{b=1}^{B} a_{k,t,u} h_b\left(x_{u,m}\right) w_{m,b}^{nbm} w_{k,m}^{out} \\ &\quad = \sum_{u=1}^{t} \sum_{m=1}^{M} f_{u,m}\left(x_{u,m}, \mathbf{X}_{:t}\right) \end{aligned} \tag{11}$$

Equation (11) shows that GATSM satisfies Definition 3.1 by encapsulating the attention scores (i.e., $\alpha_{k,t,u}$) and time-sharing NBM (i.e., h_b) into a function $f_{u,m}$. We can derive three types of interpretations from GATSM: 1) $a_{k,t,u}$ indicates the importance of time step u at time step t, 2) $h_b(x_{u,m})\, w^{nbm}_{m,b} w^{out}_{k,m}$ represents the time-independent contribution of feature m, and 3) $a_{k,t,u} h_b(x_{u,m})\, w^{nbm}_{m,b} w^{out}_{k,m}$ represents the time-dependent contribution of feature m at time step t.

4 Experiments

Datasets

We conducted experiments using eight publicly available real-world time series datasets. From the Monash repository (Tan et al. 2020), we sourced three datasets: Energy, Rainfall, and AirQuality. We downloaded another three datasets, Heartbeat, LSST, and NATOPS, from the UCR repository (Bagnall et al. 2018). We downloaded the remaining two datasets, Mortality and Sepsis, from the PhysioNet (Goldberger et al. 2000). We performed ordinal encoding for categorical features and standardize features to have zero-mean and unit-variance. For regression task, we also standardized the target value y to have a zero-mean and unit-variance. If the dataset contains missing values, we impute categorical features with their modes and numerical features with their means. We split the dataset into a 60%/20%/20% ratio for training, validation, and testing, respectively. Appendix D provides further details about the experimental datasets, including their statistics and descriptions.

Baselines

We compare our GATSM with 15 baselines, which can be categorized into five groups: 1) Black-box tabular models including extreme gradient boosting (XGBoost) (Chen and Guestrin 2016) and MLP; 2) Black-box time series models including simple recurrent neural network (RNN), gated recurrent unit (GRU), long short-term memory (LSTM), and Transformer (Vaswani et al. 2017); 3) Transparent tabular models including simple linear model (Linear), explainable boosting machine (EBM) (Nori et al. 2019), NAM (Agarwal et al. 2021), NodeGAM (Chang, Caruana, and Goldenberg 2022), and NBM (Radenovic, Dubey, and Mahajan 2022); 4) A transparent time series model, NATM (Jo and Kim 2023); 5) Multi-step forecasting specialized models including DLinear (Zeng et al. 2023), PatchTST (Nie et al. 2023), and iTransformer (Liu et al. 2024).

Implementation

We implemented XGBoost and EBM models using the `xgboost` (Chen and Guestrin 2016) and `interpretml` (Nori et al. 2019) libraries, respectively. For NodeGAM, we employ the official implementation provided by its authors (Chang, Caruana, and Goldenberg 2022). We developed the remaining models using `torch` (Paszke et al. 2019). All models undergo hyperparameter tuning via `optuna` (Akiba et al. 2019). The pytorch-based models are optimized with the Adam with decoupled weight decay (AdamW) (Loshchilov and Hutter 2019) optimizer on an NVIDIA A100 GPU. Model training is halted if the validation loss does not decrease over 20 epochs. We used mean squared error for the regression task, binary cross-entropy for the binary classification task, and cross-entropy for the multi-class classification task. Appendix E provides further details of the model implementations and hyper-parameters.

Comparison with baselines

Table 2 shows the predictive performances of the experimental models. We report mean scores and standard deviations over five different random seeds. For the regression datasets, we evaluate R^2 scores. For the binary classification datasets, we assess the area under the receiver operating characteristic curve (AUROC). For the multi-class classification datasets, we measure accuracy. We highlight the best-performing model in **bold** and underlined the second-best model. Since the tabular models cannot handle time series, they only take $\mathbf{x}_T$ to produce y_T.

On the Energy and Heartbeat datasets, which have small sample sizes, our GATSM demonstrates the best performance, indicating strong generalization ability. EBM, XGBoost, and Transformer struggle with overfitting on the Energy dataset. For the Mortality and Sepsis datasets, there is no significant performance difference between tabular and time series models, nor between black-box and transparent models. This suggests that these two datasets lack significant temporal patterns and feature interactions. It is likely that seasonal patterns are hard to detect in medical data, and the patients' current condition already encapsulates previous conditions, making historical data less crucial. Since these datasets contain variable-length time series, the performances of NATM, which can only handle fixed-length time series, is not available. On the Rainfall, AirQuality, LSST, and NATOPS datasets, the time series models significantly outperform the tabular models, indicating that these datasets contain important temporal patterns that tabular models cannot capture. Additionally, the black-box models outperform the transparent models, suggesting that these datasets have higher-order feature interactions that transparent models cannot capture. Nevertheless, GATSM is the best model within the transparent model group and performs comparably to Transformer. This result implies that GATSM effectively captures the effects of input features and temporal patterns while maintaining transparency. Overall, GATSM achieved the best average rank in the experiments, followed by the Transformer, highlighting GATSM's robustness.

We also validated three state-of-the-art forecasting models, DLinear, PatchTST, and iTransformer. Experimental results indicate that these forecasting models do not consistently outperform the other comparative models. This is likely because their architectures are primarily optimized for forecasting tasks rather than regression or classification, and their large model sizes make them susceptible to overfitting. The performances for the three forecasting models on the Mortality and Sepsis datasets are unavailable because these models cannot handle variable-length many-to-many prediction tasks as same as NATM. We provide additional experiments on multi-step forecasting and synthetic tumor size

Model Type	Model	Energy (R^2↑)	Rainfall (R^2↑)	AirQuality (R^2↑)	Heartbeat (AUROC↑)	Mortality (AUROC↑)	Sepsis (AUROC↑)	LSST (Acc↑)	NATOPS (Acc↑)	Avg. Rank
Black-box Tabular Model	XGBoost	0.094 (±0.137)	0.002 (±0.002)	0.532 (±0.019)	0.679 (±0.094)	0.707 (±0.015)	**0.816** (±0.007)	0.424 (±0.012)	0.200 (±0.049)	9.75 (±4.773)
	MLP	0.459 (±0.101)	0.011 (±0.004)	0.423 (±0.031)	0.654 (±0.082)	0.842 (±0.014)	0.786 (±0.007)	0.417 (±0.008)	0.211 (±0.065)	8.875 (±3.044)
Black-box Time Series Model	RNN	0.320 (±0.122)	0.068 (±0.020)	0.644 (±0.032)	0.661 (±0.078)	0.581 (±0.040)	0.782 (±0.009)	0.422 (±0.029)	0.592 (±0.110)	8.500 (±2.673)
	GRU	0.435 (±0.107)	0.089 (±0.034)	0.701 (±0.018)	0.694 (±0.052)	0.818 (±0.014)	0.785 (±0.010)	0.629 (±0.013)	0.931 (±0.045)	4.500 (±2.619)
	LSTM	0.359 (±0.112)	0.090 (±0.031)	0.683 (±0.026)	0.648 (±0.042)	0.790 (±0.020)	0.779 (±0.008)	0.491 (±0.082)	0.908 (±0.035)	6.875 (±3.603)
	Transformer	0.263 (±0.263)	**0.098** (±0.035)	**0.711** (±0.027)	0.690 (±0.040)	0.844 (±0.019)	0.789 (±0.010)	**0.679** (±0.019)	**0.967** (±0.029)	4.125 (±3.980)
Forecasting Model	DLinear	0.058 (±0.247)	0.033 (±0.012)	0.473 (±0.023)	0.672 (±0.074)	N/A	N/A	0.315 (±0.010)	0.933 (±0.018)	9.833 (±4.355)
	PatchTST	0.312 (±0.172)	0.056 (±0.016)	0.488 (±0.032)	0.578 (±0.082)	N/A	N/A	0.589 (±0.027)	0.769 (±0.058)	8.666 (±4.367)
	iTransformer	-0.080 (±0.309)	0.055 (±0.012)	0.627 (±0.049)	0.642 (±0.047)	N/A	N/A	0.538 (±0.040)	0.797 (±0.041)	8.833 (±4.491)
Transparent Tabular Model	Linear	0.482 (±0.112)	0.004 (±0.001)	0.241 (±0.019)	0.637 (±0.070)	0.838 (±0.017)	0.723 (±0.011)	0.311 (±0.010)	0.206 (±0.045)	11.875 (±4.941)
	EBM	-0.200 (±0.409)	0.004 (±0.001)	0.324 (±0.014)	0.666 (±0.056)	0.729 (±0.017)	0.802 (±0.011)	0.408 (±0.016)	0.164 (±0.053)	11.625 (±4.406)
	NAM	0.363 (±0.218)	0.006 (±0.002)	0.300 (±0.013)	0.645 (±0.026)	0.853 (±0.014)	0.800 (±0.006)	0.400 (±0.011)	0.242 (±0.040)	9.375 (±4.719)
	NodeGAM	0.398 (±0.195)	0.006 (±0.002)	0.380 (±0.032)	0.681 (±0.046)	**0.854** (±0.013)	0.802 (±0.007)	0.400 (±0.028)	0.247 (±0.012)	7.750 (±4.892)
	NBM	0.330 (±0.251)	0.007 (±0.003)	0.301 (±0.012)	0.716 (±0.039)	0.852 (±0.014)	0.799 (±0.006)	0.388 (±0.014)	0.189 (±0.029)	9.250 (±4.892)
Transparent Time Series Model	NATM	0.304 (±0.122)	0.038 (±0.011)	0.548 (±0.028)	0.724 (±0.043)	N/A	N/A	0.452 (±0.010)	0.878 (±0.058)	6.833 (±2.927)
	GATSM (ours)	**0.493** (±0.173)	0.073 (±0.027)	0.583 (±0.026)	**0.843** (±0.025)	0.853 (±0.015)	0.797 (±0.007)	0.570 (±0.024)	0.956 (±0.027)	**3.375** (±1.996)

Table 2: Predictive performance comparison of various models on single output tasks.

datasets in Appendix F.

Ablation study

Choice of feature function We evaluate the performance of GATSM by changing the feature functions to three models: Linear, NAM, and NBM. Table 3 presents the results of this experiment. The simple linear function performs poorly because it lacks the capability to capture non-linear relationships. In contrast, NAM, which can capture non-linearity, shows improved performance over the linear function. However, NBM stands out by achieving the best performance in six out of eight datasets. This indicates that the basis strategy of NBM is highly effective for time series data.

Design of temporal module We evaluate the performance of GATSM by modifying the design of the temporal module. Table 4 presents the results. GATSM without the temporal module (Base) fails to learn temporal patterns and shows poor performance in the experiment. GATSM with only positional encoding (Base + PE) also shows performance similar to the Base, indicating that positional encoding alone is insufficient for capturing effective temporal patterns. GATSM with only MHA (Base + MHA) outperforms the previous two methods, demonstrating that the MHA is beneficial for capturing temporal patterns. Finally, our full GATSM (Base + PE + MHA) significantly outperforms the other methods, suggesting that the combination of PE and MHA creates a synergistic effect. Consistent with our previous findings in Section 4, all four methods show similar performances on the Mortality and Sepsis datasets, which lack significant temporal patterns.

Positional Encoding As described in Section 3, we recommend sinusoidal positional encoding over learnable position embeddings because the maximum length of time series is often unknown. In contrast, learnable position embeddings require knowledge of the maximum length, which is typically infeasible in practice. Nevertheless, we conducted an experiment comparing sinusoidal positional encoding and learnable position embedding to determine if significant performance differences exist. Table 5 presents the performance comparison between learnable position embedding and sinusoidal positional encoding on GATSM. The experimental result demonstrating no significant difference between these two methods.

Inference speed

The inference speed of machine learning models is a crucial metric for real-world systems. We evaluate the throughput

Feature Function	Energy	Rainfall	AirQuality	Heartbeat	Mortality	Sepsis	LSST	NATOPS
Linear	0.283(±0.277)	0.071(±0.024)	0.563(±0.019)	0.766(±0.024)	0.832(±0.015)	0.735(±0.012)	0.398(±0.030)	**0.972**(±0.020)
NAM	0.304(±0.229)	0.068(±0.021)	0.564(±0.019)	0.838(±0.032)	0.851(±0.013)	**0.801**(±0.005)	0.553(±0.023)	0.933(±0.039)
NBM	**0.493**(±0.173)	**0.073**(±0.027)	**0.583**(±0.026)	**0.843**(±0.025)	**0.853**(±0.015)	0.797(±0.007)	**0.570**(±0.024)	0.956(±0.027)

Table 3: Ablation study on different feature functions.

Temporal Module	Energy	Rainfall	AirQuality	Heartbeat	Mortality	Sepsis	LSST	NATOPS
Base	0.452(±0.087)	0.007(±0.002)	0.299(±0.012)	0.661(±0.043)	**0.854**(±0.013)	0.798(±0.008)	0.392(±0.006)	0.192(±0.027)
Base + PE	0.397(±0.054)	0.007(±0.003)	0.299(±0.012)	0.681(±0.068)	0.852(±0.013)	**0.799**(±0.007)	0.385(±0.027)	0.228(±0.029)
Base + MHA	0.368(±0.230)	0.048(±0.017)	0.555(±0.020)	0.821(±0.044)	0.847(±0.020)	0.779(±0.033)	**0.595**(±0.013)	0.856(±0.059)
Base + PE + MHA	**0.493**(±0.173)	**0.073**(±0.027)	**0.583**(±0.026)	**0.843**(±0.025)	0.853(±0.015)	0.797(±0.007)	0.570(±0.024)	**0.956**(±0.027)

Table 4: Ablation study on the temporal module.

	Energy	Rainfall	AirQuality	Heartbeat	Mortality	Sepsis	LSST	NATOPS
Learnable PE	0.519(±0.097)	0.072(±0.026)	0.570(±0.023)	0.838(±0.048)	0.855(±0.018)	0.797(±0.006)	0.570(±0.023)	0.950(±0.021)
Sinusoidal PE	0.493(±0.173)	0.073(±0.027)	0.583(±0.026)	0.843(±0.025)	0.853(±0.015)	0.797(±0.007)	0.570(±0.024)	0.956(±0.027)

Table 5: Performances of learnable position embedding and sinusoidal positional encoding on GATSM.

	Energy	Rainfall	AirQuality	Heartbeat	Mortality	Sepsis	LSST	NATOPS
NAM	65.3K	1.8M	5.1M	139.1K	772.2K	23.9K	2.3M	147.9K
NBM	45.5K	1.1M	1.0M	55.9K	375.8K	6.5K	1.6M	85.6K
Transformer	30.9K	240.5K	174.2K	15.7K	161.9K	134.6K	214.4K	68.3K
NATM	5.3K	699.3K	241.3K	1.3K	N/A	N/A	28.6K	19.2K
GATSM	6.1K	350.6K	192.8K	1.2K	4.9K	3.8K	126.5K	12.5K

Table 6: Inference throughputs per second of different models.

	Energy	Rainfall	AirQuality	Heartbeat	Mortality	Sepsis	LSST	NATOPS
NAM	307.2K	39.36K	115.2K	780.8K	524.8K	172.03M	81.79K	314.88K
NBM	5.39M	677.65K	2.03M	13.78M	9.26M	3.04G	1.35M	5.39M
Transformer	5.3M	5.14M	19.4M	32.72M	1.29M	3.79M	1.32M	8M
NATM	44.25M	921.89K	2.77M	316.32M	N/A	N/A	2.84M	15.68M
GATSM	777.19M	16.2M	48.58M	5.59G	1.87G	3.04G	48.62M	276.84M

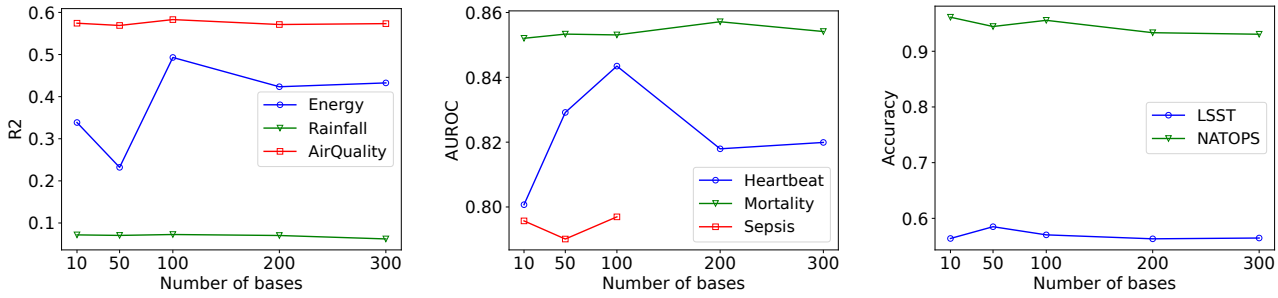
Table 7: FLOPs of different models.

and FLOPs of the models. Table 6 and Table 7 present the results. Since all experimental datasets have fewer features than the number of basis functions in NBM, NAM achieves higher throughput than NBM. Transparent tabular models typically exhibit fast speeds. However, their throughput significantly decreases in datasets with many features, such as Heartbeat, Mortality, and Sepsis, because they require the same number of feature functions as the number of input features. Transformer shows higher throughput than the transparent time series models because it does not require feature functions, which are the main bottleneck of transparent models. Additionally, the PyTorch implementation of Transformer uses the flash attention mechanism (Dao et al. 2022) to enhance its efficiency. NATM has slightly higher throughput than GATSM, as it does not require the attention mechanism and has fewer feature functions compared to the number of basis functions in GATSM.

Inference latency requirements in real-world applications vary significantly based on the context. For instance, real-time clinical decision support systems, such as heart failure prediction models, typically demand latencies below one second per patient, whereas non-real-time applications, such as discharge prediction, have less stringent requirements. As demonstrated in Table 6, GATSM can process at least 1,000 samples per second, making it feasible for real-world scenarios.

Number of basis functions

We evaluate GATSM by varying the number of basis functions in the time-sharing NBM. Figure 2 presents the results for regression, binary classification, and multi-class classification datasets. For the Sepsis dataset, using 200 and 300 basis functions causes an out-of-memory error. For the Energy and Heartbeat datasets, performance improves up to 100 basis functions but shows no further benefit when the number of basis functions exceeds 100. In other datasets, performance changes are not significant with different numbers of basis functions. In addition, there is a trade-off between the number of basis functions and computational speed. Therefore, we recommend generally setting the number of basis



(a) Forecasting (b) Binary (c) Multi-class

Figure 2: Performances of GATSM on the different number of basis functions.

functions to 100. Note that the performance of GATSM with this hyper-parameter depends on the dataset size and complexity. Hence, a larger number of basis functions may be beneficial for more complex datasets.

Interpretation

In this section, we describe four interpretations of GATSM's predictions on the AirQuality dataset. Due to page limits, visualizations for model interpretation are provided in Appendix G. In addition, interpretations for the Rainfall and Mortality datasets can be found in Appendix G.

Time-step importance We plot the average attention scores at the last time step T in Figure 5. The process for extracting the average attention score of time step u at time step t is formalized as $\sum_{k=1}^{K} a_{k,t,u}$. This process is repeated over all data samples, and the results are averaged. Based on Figure 5, it seems that GATSM pays more attention to the initial and last states than to the intermediate states. This indicates that the current concentration of particulate matter depends on the initial and recent states.

Global feature contribution Figure 6 illustrates the global behavior of features in the AirQuality dataset, with red bars indicating the density of training samples. We extract $\sum_{k=1}^{K} h_b\left(x_{t,m}\right) w_{m,b}^{nbm} w_{k,m}^{out}$ from GATSM and repeat this process over the range of minimum to maximum feature values to plot the line. We found that the behavior of *SO2*, *O3*, and *windspeed* is inconsistent with prior human knowledge. Typically, high levels of *SO2* and *O3* are associated with poor air quality. However, GATSM learned that particulate matter concentration starts to decrease when *SO2* exceeds 10 and *O3* exceeds 5. This discrepancy may be due to sparse training samples in these regions, leading to insufficient training, or there may be interactions with other features. Another known fact is that high *windspeed* decreases particulate matter concentration. This is consistent when *windspeed* is below 0.7 in our observation. However, particulate matter concentration drastically increases when *windspeed* exceeds 0.7, likely due to the wind causing yellow dust.

Local time-independent feature contribution To interpret the prediction of a data sample, we plot the local time-independent feature contributions, $\sum_{k=1}^{K} h_b\left(x_{t,m}\right) w_{m,b}^{nbm} w_{k,m}^{out}$, in Figure 7. The main y-axis (blue) represents feature contribution, the sub y-axis (red) represents feature value, and the x-axis represents time steps. We found that *SO2*, *NO2*, *CO*, and *O3* have positive correlations. In contrast, *temperature*, *pressure*, *dew point*, and *windspeed* have negative correlations. These are consistent with the global interpretations shown in Figure 6. Rainfall has the same values across all time steps.

Local time-dependent feature contribution We also visualize the local time-dependent feature contributions, $\sum_{k=1}^{K} a_{k,t,u} h_b\left(x_{t,m}\right) w_{m,b}^{nbm} w_{k,m}^{out}$. Figure 8 illustrates the interpretation of the same data sample as in Figure 7. The time-dependent interpretation differs slightly from the time-independent interpretation. We found that there are time lags in *SO2*, *NO2*, *CO*, and *O3*, meaning previous feature values affect current feature contributions. For example, in the case of *SO2*, low feature values around time step 5 lead to low feature contributions around time step 13.

5 Conclusion

In this paper, we proposed a novel transparent model for time series named GATSM. GATSM consists of time-sharing NBM and the temporal module to effectively learn feature representations and temporal patterns while maintaining transparency. The experimental results demonstrated that GATSM has superior generalization ability and is the only transparent model with performance comparable to Transformer. We provided various visual interpretations of GATSM, demonstrated that GATSM capture interesting patterns in time series data. GATSM delivers robust predictive performance along with detailed interpretations of its outputs. This advantage enables accurate and trustworthy predictions, which is crucial in high-stakes domains. We anticipate that GATSM will be widely adopted in various fields and demonstrate strong performance. We discuss the limitations of GATSM and suggest directions for future work in Appendix A. Potential applications and the broader impacts of GATSM across various fields can be found in Appendix B.

References

Agarwal, R.; Melnick, L.; Frosst, N.; Zhang, X.; Lengerich, B.; Caruana, R.; and Hinton, G. E. 2021. Neural Additive Models: Interpretable Machine Learning with Neural Nets. In *Advances in Neural Information Processing Systems*.

Akhavan Rahnama, A. H. 2023. The Blame Problem in Evaluating Local Explanations and How to Tackle It. In *European Conference on Artificial Intelligence*.

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Bagnall, A.; Dau, H. A.; Lines, J.; Flynn, M.; Large, J.; Bostrom, A.; Southam, P.; and Keogh, E. 2018. The UEA multivariate time series classification archive, 2018. arXiv:1811.00075.

Caruana, R.; Lou, Y.; Gehrke, J.; Koch, P.; Sturm, M.; and Elhadad, N. 2015. Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Chang, C.-H.; Caruana, R.; and Goldenberg, A. 2022. NODE-GAM: Neural Generalized Additive Model for Interpretable Deep Learning. In *International Conference on Learning Representations*.

Chang, C.-H.; Tan, S.; Lengerich, B.; Goldenberg, A.; and Caruana, R. 2021. How Interpretable and Trustworthy are GAMs? In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Dao, T.; Fu, D. Y.; Ermon, S.; Rudra, A.; and Ré, C. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems*.

Goldberger, A. L.; Amaral, L. A. N.; Glass, L.; Hausdorff, J. M.; Ivanov, P. C.; Mark, R. G.; Mietus, J. E.; Moody, G. B.; Peng, C.-K.; and Stanley, H. E. 2000. PhysioBank, PhysioToolkit, and PhysioNet. *Circulation*, 101(23): e215–e220.

Hastie, T.; and Tibshirani, R. 1986. Generalized Additive Models. *Statistical Science*, 1(3): 297–318.

Jo, W.; and Kim, D. 2023. Neural additive time-series models: Explainable deep learning for multivariate time-series prediction. *Expert Systems with Applications*, 228: 120307.

Kim, M.; Choi, H.-S.; and Kim, J. 2022. Higher-order Neural Additive Models: An Interpretable Machine Learning Model with Feature Interactions. arXiv:2209.15409.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *International Conference on Learning Regresentations*.

Liu, Y.; Khandagale, S.; White, C.; and Neiswanger, W. 2021. Synthetic Benchmarks for Scientific Research in Explainable Machine Learning. In *Advances in Neural Information Processing Systems*.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Lundberg, S. M.; and Lee, S.-I. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*.

Mothilal, R. K.; Sharma, A.; and Tan, C. 2020. Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.

Nori, H.; Jenkins, S.; Koch, P.; and Caruana, R. 2019. InterpretML: A Unified Framework for Machine Learning Interpretability. arXiv:1909.09223.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*.

Radenovic, F.; Dubey, A.; and Mahajan, D. 2022. Neural Basis Models for Interpretability. In *Advances in Neural Information Processing Systems*.

Rahnama, A. H. A.; and Boström, H. 2019. A study of data and label shift in the LIME framework. In *Workshop on Human-Centric Machine Learning at the 33rd Conference on Neural Information Processing Systems*.

Rahnama, A. H. A.; Bütepage, J.; Geurts, P.; and Boström, H. 2024. Can local explanation techniques explain linear additive models? *Data Mining and Knowledge Discovery*, 38: 237–280.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Rudin, C. 2018. Please Stop Explaining Black Box Models for High Stakes Decisions. In *Advances in Neural Information Processing Systems, Workshop on Critiquing and Correcting Trends in Machine Learning*.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1: 206–215.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization.

Tan, C. W.; Bergmeir, C.; Petitjean, F.; and Webb, G. I. 2020. Monash University, UEA, UCR Time Series Extrinsic Regression Archive. arXiv:2006.10996.

Tan, S.; Caruana, R.; Hooker, G.; and Lou, Y. 2018. Distill-and-Compare: Auditing Black-Box Models Using Transparent Model Distillation. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*.

Utkin, L. V.; Satyukov, E. D.; and Konstantinov, A. V. 2022. SurvNAM: The machine learning survival model explanation. *Neural Networks*, 147: 81–102.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Łukasz Kaiser; and Polosukhin, I. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

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*.