THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Data Science Program

Capstone Report - Fall 2024

Time Series Model Applications

Liang Gao

Supervised by
Amir Jafari

Abstract

In this study, we compare the performance of classical time series models (AR, MA, ARMA, ARIMA) and modern deep learning techniques (LSTM, Bi-LSTM, Seq2Seq) alongside a state-of-the-art transformer model. These models have been individually studied and applied to specific problems, but the lack of a standardized benchmark for comparing their performance across diverse datasets creates a significant gap in the literature. The goal of this research is to benchmark the performance of representative time-series models across different datasets. We evaluate the models using three publicly available datasets and compare their prediction accuracy using metrics such as MSE, RMSE, and MAE. The results show that modern deep learning techniques outperform classical models, with the simplest architecture, LSTM, achieving the lowest MSE across all datasets. Although the transformer model demonstrated superior performance compared to classical models, it did not outperform the modern techniques. This suggests that for the datasets used in this study, complex transformer architectures may not be necessary to achieve optimal results. We hope our study can provide some insights into time series studies.

Contents

1 Introduction

Time series data refers to a sequence of data points collected or recorded at successive, equally spaced time intervals [1]. Time series modeling is a rapidly advancing field that has garnered significant attention from the research community in recent decades. The primary objective of time series modeling is to systematically collect and analyze historical data to develop a model that accurately represents the underlying structure of the series. This model is then used to predict future values or make forecasts based on the observed patterns in the data.

Time series analysis is a critical area of research with applications spanning diverse fields, such as finance [2], social science [3], climate science [4], energy management [5], and healthcare [6]. Numerous models have been developed to address specific problems in these domains, ranging from classical linear models like AR, MA, and ARIMA to advanced deep learning architectures such as LSTM and Transformers.

The Autoregressive(AR) model has been employed to predict the instantaneous phase and frequency of neural oscillations in EEG data, demonstrating its effectiveness in real-time forward prediction for shorter intervals [7]. Researchers such as Jafari and Jafari (2024) have utilized the Autoregressive Moving Average (ARMA) model in combination with the Levenberg-Marquardt algorithm for speech recognition applications, demonstrating its effectiveness in handling time series data in this domain [8]. As demonstrated in the study by [9], the LSTM network was used to forecast the COVID-19 outbreak in Canada, predicting trends, potential stopping points, and multi-day case counts. As discussed by [10], the authors proposed a method using a Time Series Transformer (TST) to recognize fault modes in rotating machinery. They combined the TST with a time series tokenizer and demonstrated its superior fault identification capability compared to traditional CNN and RNN models.

However, there is a noticeable lack of benchmarking studies comparing these models' performances on standardized datasets. The primary objective of this study is to evaluate and compare the performance of classical time series models(linear models) [11]—Autoregressive(AR), Moving Average (MA), Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA)—with advanced deep learning models (non-linear models), including Long Short-Term Memory (LSTM) [12], Bidirectional LSTM (BiLSTM) [13], sequence-to-sequence architectures [14], and the state-of-the-art Transformer model [15].

In this report, we selected representative models and evaluated their performances on three publicly available datasets. [The analysis yielded insightful conclusions], which we hope will provide valuable guidance to the readers.

2 Problem Statement

Time-series analysis is a critical field with widespread applications, and numerous models have been developed to address its challenges. The field has seen the development of numerous models, from traditional statistical approaches like Auto-Regressive (AR), Moving Average (MA), and Auto-Regressive Integrated Moving Average (ARIMA), to advanced machine learning techniques such as Long Short-Term Memory (LSTM), Sequence-to-Sequence (Seq2Seq), and the state-of-the-art Transformer model. While these models have been individually studied and applied to specific problems, the lack of a standardized benchmark for comparing their performance across diverse datasets creates a significant gap in the literature.

The goal of this research is to benchmark the performance of representative time-series models across different datasets. Each dataset was preprocessed to ensure consistency and alignment with the experimental design. These datasets were chosen to evaluate the performance of models across diverse applications, providing comprehensive insights into their strengths and limitations.

Multiple performance metrics, such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), are employed to ensure a robust and comprehensive evaluation. These metrics

provide a quantitative foundation for assessing the accuracy and reliability of each model across the datasets. Through this analysis, we aim to identify the strengths and limitations of different modeling approaches and offer insights into their applicability for diverse time series forecasting challenges.

3 Data

We utilized three public and realistic datasets in this study to evaluate the performance of various time series models:

- **Weather Station Beutenberg Dataset** [16]: This dataset contains meteorological measurements, such as temperature, humidity, and wind speed, recorded every 10 minutes, capturing fine-grained temporal variations in environmental conditions. The data was accessed from Kaggle, and the target variable is the temperature in Celsius.
- **Power Consumption of Tetouan City** [17]: This dataset comprises the energy consumption data of Tetouan City, recorded at 10-minute intervals, offering detailed observations of energy usage patterns. It offers insights into power usage patterns and facilitates the evaluation of energy forecasting models. The dataset was retrieved from the UCI Machine Learning Repository and used to forecast the power consumption in Zone 1.
- **Air Pollution Forecasting Dataset** [18]: This dataset includes hourly measurements of air pollutants along with meteorological features, providing insights into atmospheric quality over time. It serves as a multivariate time series dataset to assess forecasting models in pollution monitoring. The data was sourced from Kaggle, and the target variable is the pollution called PM2.5 concentration.

4 Models

4.1 Classical models

Autoregressive (AR). In an Autoregressive model, we forecast the variable of interest using a linear combination of past values of the variable. An AR process of order n_a AR(n_a) can be written as:

$$y(t) + a_1 y(t-1) + a_2 y(t-2) + \cdots + a_{n_a} y(t-n_a) = \epsilon(t)$$

Moving Average (MA). Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors. A Moving Average process of order order n_b, (MA(n_b)) can be represented as:

$$y(t) = \epsilon(t) + b_1 \epsilon(t-1) + b_2 \epsilon(t-2) + \cdots + b_{n_b} \epsilon(t-n_b)$$

Autoregressive Moving Average (ARMA). The Autoregressive-Moving Average model is the combination of AR and MA models, which can be represented as:

$$y(t) + a_1 y(t-1) + a_2 y(t-2) + \cdots + a_{n_a} y(t-n_a) = \epsilon(t) + b_1 \epsilon(t-1) + b_2 \epsilon(t-2) + \cdots + b_{n_b} \epsilon(t-n_b)$$

In comparison to pure autoregressive (AR) and moving average (MA) models, the Autoregressive Moving Average (ARMA) models offer an efficient linear approach for modeling stationary time series. This is because ARMA models can capture the underlying process of the data with fewer parameters, making them a more effective choice for such data structures. By combining both autoregressive and moving average components, ARMA models strike a balance between flexibility and simplicity in modeling time series dependencies.

Autoregressive Integrated Moving Average (ARIMA). Autoregressive integrated moving average (ARIMA) model is a generalization of the autoregressive moving average model with differencing. In the general form, the ARIMA(n_a, d, n_b) model can be written as:

$$\left(1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}\right) \left(1 - q^{-1}\right)^d y(t) = \left(1 + b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b}\right) \epsilon(t)$$

Notation:

- $y(t)$ is the value of the time series at time t,
- $a_1, a_2, \ldots, a_{n_a}$ are the coefficients of the AR model,
- n_a is the order of the autoregressive model,
- $b_1, b_2, \ldots, b_{n_b}$ are the coefficients of the MA model,
- n_b is the order of the moving average model,
- $\epsilon(t)$ is a white noise (WN $\sim (0, \sigma_\epsilon^2)$),
- d is the number of non-seasonal order differencing.

4.2 Modern techniques

Long Short-Term Memory (LSTM). Long Short-Term Memory (LSTM) networks, introduced by Hochreiter and Schmidhuber in 1997, are a specialized type of Recurrent Neural Network (RNN) [19] designed to effectively capture long-term dependencies in sequential data [12]. Unlike traditional RNNs, which suffer from the vanishing gradient problem, LSTMs utilize a unique architecture comprising a cell state and three gates—input, forget, and output gates. These gates regulate the flow of information, enabling the network to selectively store, update, and retrieve information over extended sequences, making LSTMs particularly suitable for time series forecasting and natural language processing tasks. The architecture of the LSTM model can be represented in Figure 1:

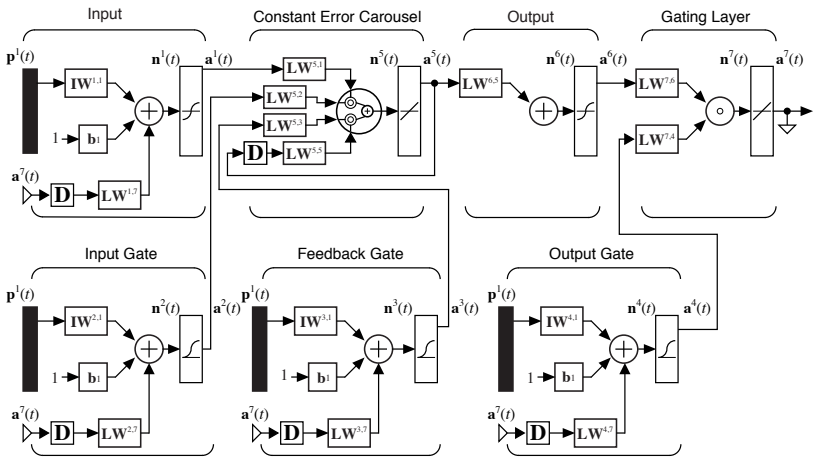

Figure 1: The LSTM - Model Architecture [20]

Bidirectional Long Short-Term Memory (BiLSTM). Bidirectional Long Short-Term Memory (BiLSTM) networks are an advanced extension of LSTM models that process sequential data in both forward and backward directions, leveraging information from past and future contexts simultaneously. This architecture employs two separate LSTMs: one analyzing the input

sequence in its original order and the other in reverse. By combining their outputs, BiLSTM networks provide a richer representation of the data, making them particularly effective for tasks like time series forecasting, natural language processing, and speech recognition [13].

Sequence-to-sequence (Seq2Seq). Sequence-to-sequence (Seq2Seq) models are widely used for tasks requiring the transformation of input sequences into output sequences of varying lengths, such as language translation or time series forecasting. In the implemented architecture, the model follows an encoder-decoder framework enhanced by an attention mechanism:

- **Encoder:** An LSTM processes the input sequence, summarizing it into a fixed-length context vector.
- **Attention Mechanism:** A linear layer computes attention weights, enabling the decoder to focus on relevant encoder outputs dynamically.
- **Decoder:** A second LSTM generates the target sequence, step-by-step, using the context vector and attention-adjusted encoder outputs.
- **Output Layer:** A final linear layer maps the decoder outputs to the target prediction space.

The attention mechanism addresses the limitations of fixed-length representations, improving performance on tasks involving long sequences. This architecture is inspired by the attention-based Seq2Seq models introduced by [21].

4.3 State of Art

Transformer. The Transformer model, introduced by [15], has significantly advanced deep learning with its attention mechanism and parallelized architecture. Unlike traditional recurrent models that process sequences sequentially, Transformers handle entire sequences simultaneously, making them more efficient in capturing long-term dependencies. The model consists of two main components: the encoder and the decoder. The encoder processes input sequences using self-attention and feed-forward layers, while the decoder generates outputs by attending to both the encoder's representation and previous outputs. Key features include the scaled dot-product attention, which calculates relationships between tokens using queries, keys, and values, and positional encoding, which incorporates sequence order information. Due to their capacity to model non-linear and long-range dependencies, Transformers have outperformed recurrent models in various tasks, including time series forecasting. The architecture of the Transformer model can be represented in Figure 8:

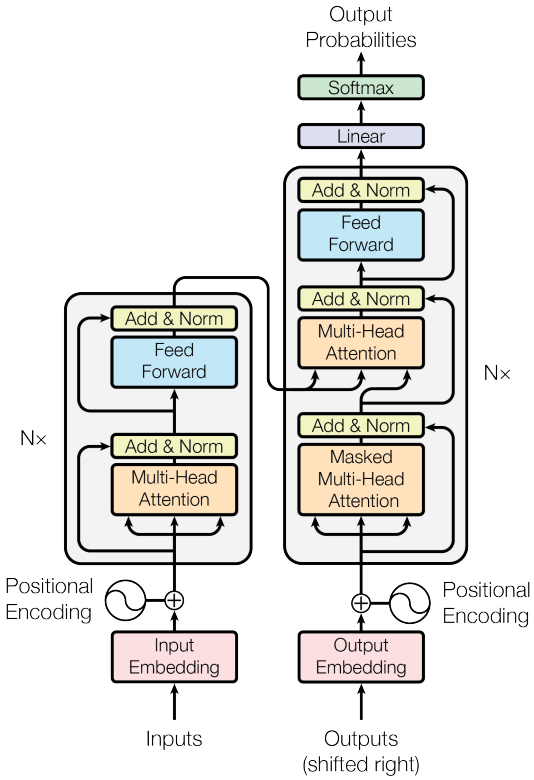

Figure 2: The Transformer - Model Architecture [20]

5 Analysis

In this study, we used time series domain knowledge, including ACF/PACF plot and generalized partial autocorrelation (GPAC), to help determine the order for classical time series linear models. Besides domain knowledge, we also used a hyperparameter optimization framework - Optuna, in the experiment. These tools are introduced below.

ACF/PACF. The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) are fundamental tools in time series analysis [11, 22, 23]. The ACF measures the correlation between a time series and its own lagged values. The formula for the autocorrelation at lag k is:

$$\rho_k = \frac{\sum_{t=k+1}^{n}(X_t - \bar{X})(X_{t-k} - \bar{X})}{\sum_{t=1}^{n}(X_t - \bar{X})^2}$$

It shows how a time series is related to its previous values, with autocorrelations computed at different lag values. The ACF can help identify the presence of trends or seasonality, and it is useful for detecting the order of the Moving Average (MA) model. A significant autocorrelation at a specific lag indicates that past values are predictive or future values at that lag.

The PACF is similar to the ACF but measures the correlation between a time series and its lagged values after removing the effect of intermediate lags. The formula for the Partial Autocorrelation at lag k is:

$$\phi_{kk} = \frac{\mathrm{Cov}(X_t, X_{t-k} \mid X_{t-1}, X_{t-2}, \ldots, X_{t-k+1})}{\sqrt{\mathrm{Var}(X_t) \cdot \mathrm{Var}(X_{t-k})}}$$

The PACF shows the direct correlation between a value and its lagged version without the interference of other intervening time lags. The PACF is particularly useful in identifying the order of the Autoregressive (AR) model.

In time series modeling, the comparison of ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) plots helps determine the underlying pattern of the data, aiding in the selection of an appropriate model. When the ACF plot gradually tails off while the PACF plot cuts off after a certain lag, it suggests an Autoregressive (AR) model. On the other hand, if the ACF plot cuts off and the PACF plot tails off, it indicates a Moving Average (MA) model. This behavior reflects the distinct structures of AR and MA processes, where AR models depend on past values, while MA models are based on past error terms.

GPAC. The Generalized Partial Autocorrelation (GPAC) is a tool used in time series analysis, particularly in the context of Auto-Regressive Moving-Average (ARMA) modeling. It is used to estimate the order of the ARMA model when the auto-regressive (AR) and moving-average (MA) orders (n_a and n_b, respectively) are not known prior.

The formula to calculate the generalized partial autocorrelation can be written as:

$$\phi_{kk}^{j} = \frac{\begin{vmatrix} \hat{R}_y(j) & \hat{R}_y(j-1) & \cdots & \hat{R}_y(j+1) \\ \hat{R}_y(j+1) & \hat{R}_y(j) & \cdots & \hat{R}_y(j+2) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{R}_y(j+k-1) & \hat{R}_y(j+k-2) & \cdots & \hat{R}_y(j+k) \end{vmatrix}}{\begin{vmatrix} \hat{R}_y(j) & \hat{R}_y(j-1) & \cdots & \hat{R}_y(j-k+1) \\ \hat{R}_y(j+1) & \hat{R}_y(j) & \cdots & \hat{R}_y(j-k+2) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{R}_y(j+k-1) & \hat{R}_y(j+k-2) & \cdots & \hat{R}_y(j) \end{vmatrix}}$$

The GPAC array table provides a visual representation of the GPAC coefficients:

GPAC array table

j	$k=1$	$k=2$	$\ldots$	$k=n_a$
0	ϕ_{11}	ϕ_{22}	$\ldots$	
1	ϕ_{11}	ϕ_{22}	$\ldots$	
$\vdots$	$\vdots$	$\vdots$	$\ddots$	
n_b				$\phi_{n_a n_a}$

The rows represent the lag j, starting from 0 up to n_b. Second, the columns represent the lag k, starting from 1 up to n_a. The cells contain the GPAC coefficients ϕ_{kk}.

The first row of the GPAC array table, in fact, the partial autocorrelation (PAC) function, which is used to estimate the order of the auto-regressive (AR) component of the ARMA model when $n_b = 0$.

The pattern observed in the GPAC table, where a column has constant values and a row consists entirely of zeros, identifies the orders of n_a for the AR component and n_b for the MA component. Specifically, the row number represents the MA order, while the column number indicates the AR order. This structure provides a systematic approach to determining the appropriate parameters for ARMA modeling.

Optuna. In addition to using the time series domain knowledge (ACF/PACF plots and GPAC table), we also use Optuna to determine the order of classical linear time series models. Optuna is an open-source hyperparameter optimization framework designed for efficiency and flexibility in machine learning and deep learning workflows [24]. It employs an automated search algorithm, primarily based on Bayesian optimization, to find optimal hyperparameter configurations. Optuna uses a trial-and-error approach within a user-defined search space and supports pruning techniques to eliminate unpromising trials early, significantly reducing computation time. Its integration with major machine learning libraries and visualization tools makes it a versatile choice for both researchers and practitioners.

6 Experimentation setting

Our experiments were conducted using AWS cloud computing resources tailored to the computational needs of the models. Classical time series models—AR, MA, ARMA, and ARIMA—were executed on AWS CPU instances, as these models are computationally lightweight. On the other hand, the deep learning models, including LSTM, BiLSTM, Seq2Seq, and Transformer, required substantial computational power and were, therefore, trained on AWS GPU instances to ensure efficiency and faster convergence.

To ensure a fair comparison between different models, we standardized the hyperparameter settings across the modern techniques (LSTM, BiLSTM, Seq2Seq) and the state-of-the-art Transformer model. The following hyperparameters in Table 1 were kept consistent across all models:

Hyperparameter	**Value**
Sequence Length	6
Batch Size	128
Optimizer	Adam
Training Epochs	100
Learning Rate	0.001

Table 1: Hyperparameter Settings for Model Comparison

Additionally, for the modern techniques (LSTM, BiLSTM, Seq2Seq), we used the same number of layers, which is 1, and the same number of hidden sizes, which is 2. This uniformity in hyperparameters was designed to ensure the comparability of results across different models and to isolate the impact of model architecture on performance.

For the Optuna setting, we set the max order for the AR process to 20, the MA process to 10, and the number of trials to 30. The datasets were divided into training and validation subsets, with 80% of the data used for training and 20% for validation. This split ensured sufficient data for model learning while reserving a robust validation set to assess performance. To ensure the comparability of different models across various datasets, we normalized the data before calculating the MSE, RMSE, and MAE using the min-max normalization equation 1.

$$X_{\text{normalized}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{1}$$

7 Discussion and Conclusion

In this study, we analyzed the characteristics of three datasets and systematically investigated the performance of various models on these datasets. In the following sections, we provide a comprehensive discussion of our findings and insights.

7.1 Exploratory Data Analysis

In the temperature dataset, I used the drift method to fill the 9 missing values. There are no missing values in the other two datasets. We performed Exploratory Data Analysis (EDA) to examine the characteristics of the datasets and identify underlying patterns. Various visualizations were created, including sequence plots to observe trends over time and rolling mean and variance plots to assess stationarity; we also generated autocorrelation function (ACF) and partial autocorrelation function (PACF) plots to identify potential model orders and conducted time series decomposition to analyze seasonal, trend, and residual components. The details of ACF and PACF are presented in Section 5. These analyses provided essential insights for model development. Figure 3 presents all the visualizations generated for our selected datasets.

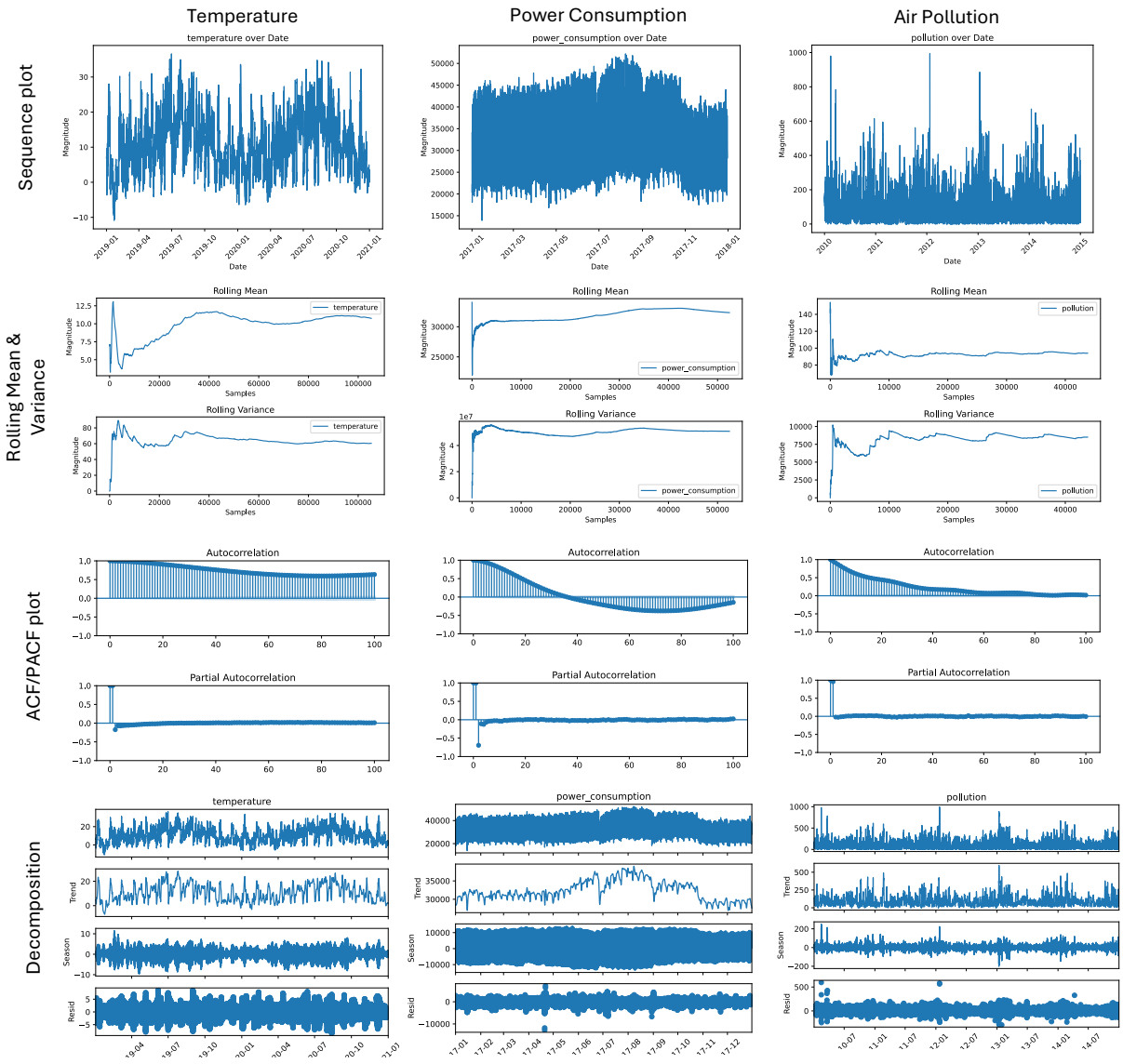

Figure 3: EDA plots

Table 2: Strength of Trend and Seasonality for Each Dataset

	Strength of Trend (%)	**Strength of Seasonality (%)**
Weather	94.33	74.79
Air Pollution	86.01	40.67
Power Consumption	92.41	98.64

All three datasets exhibit stationarity, as indicated by the constancy of rolling mean and variance after considering all samples. The ACF tail-off and PACF cut-off in all three datasets indicate clear AR patterns. Using time series decomposition, we calculated the strengths of trend and seasonality. Table 2 summarizes the strength of trend and seasonality for the three datasets analyzed in this study. For the Weather dataset, the trend strength is remarkably high at 94.33%, while the seasonality is moderate at 74.79%. The Air Pollution dataset shows a significant trend strength of 86.01% but relatively weak seasonality at 40.67%. Finally, the Power Consumption dataset exhibits both a strong trend (92.41%) and an exceptionally high seasonality (98.64%). These metrics underline the varying characteristics of each dataset, guiding appropriate model

selection.

7.2 Classical models

We implemented four classical linear time series models: AR, MA, ARMA, and ARIMA, each representing different approaches to modeling the temporal dependencies in time series data. Determining the appropriate model order n_a for the Autoregressive (AR) component and n_b for the moving average (MA) component is a critical step in the modeling process. To achieve this, we relied on two fundamental tools from the time series analysis domain. First, the ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) plots were used to examine the relationships between lags. These plots provide visual cues to identify the potential order of the AR and MA components by observing the cutoff or tail-off patterns in the respective plots. Second, the GPAC (Generalized Partial Autocorrelation) table was utilized to systematically assess the lag patterns, offering a structured approach to infer the AR and MA orders from the data.

For the AR model, we determined the order based on the ACF and PACF plots shown in Figure 3. The ACF tails off, while the PACF cuts off after a specific lag, which we used as the order for the AR model. For the MA model, since none of the ACF or PACF plots exhibited a clear MA pattern, we set the maximum MA order to 10 and employed Optuna to identify the optimal order. The GPAC table in Figure 4 was used to determine the order for the ARMA model. Specifically, the column with constant values indicates the order for the AR component, while the row with all values equal to zero indicates the MA component order. The ARIMA model is designed to handle non-stationary datasets, where the number of non-seasonal differencing steps required to make the data stationary defines the integrated component of the model. Since our datasets are already stationary, applying differencing is not strictly necessary. However, from a technical perspective, the ARIMA model remains valid. To ensure consistency and facilitate a fair comparison with other models, we applied first-order differencing to the data and then analyzed the differenced data using the GPAC method. The table in 5 shows the GPAC tables for our differences data. This approach allowed us to identify patterns and determine the appropriate AR and MA orders, enabling us to implement the ARIMA model effectively for comparative purposes.

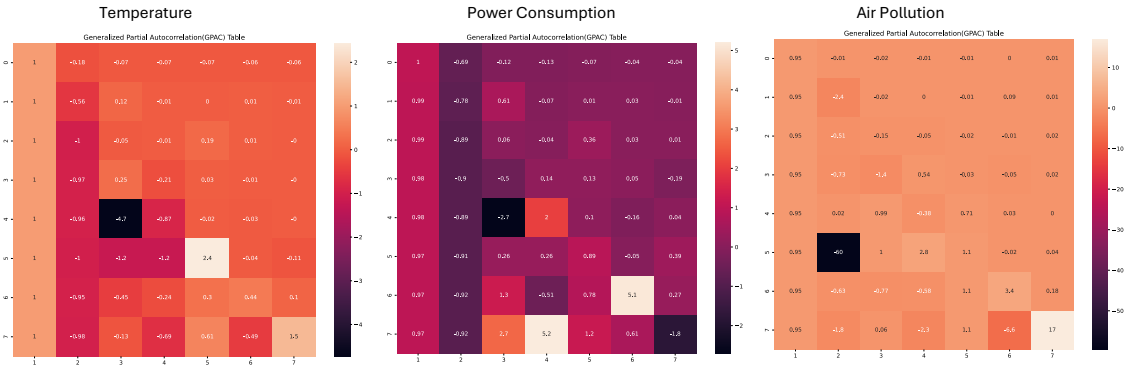

Figure 4: GPAC for Raw Data

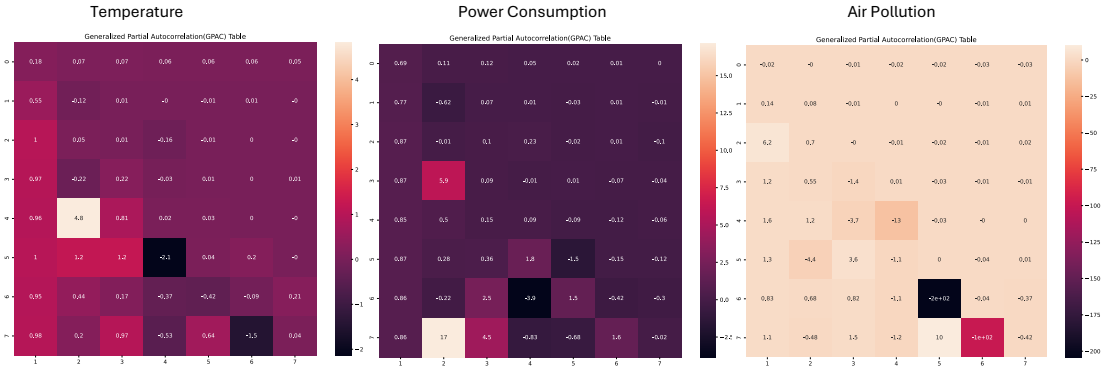

Figure 5: GPAC for Differenced Data

While traditional approaches like domain expertise and manual interpretation of plots were essential, we also employed Optuna, an advanced machine learning optimization algorithm, to automate and enhance the hyperparameter tuning process. By leveraging Optuna's efficient search strategies, we ensured that the selected hyper-parameters minimized errors and improved model performance, complementing the traditional methods with modern computational techniques. For the AR component, we set the max potential order to 20 and 10 for the MA component since there is no MA pattern from ACF and PACF plots. The number of trials for the AR model is 20, MA is 10, and 30 for the ARMA and ARIMA models.

Table 3 presents the model orders determined using time series domain knowledge and those obtained via Optuna, along with their respective MSE values. As shown in the table, the differences in MSE between the two methods are minor. This suggests that while domain knowledge can be a valuable tool for determining the appropriate model order, using Optuna is a viable alternative, especially in scenarios where domain knowledge is insufficient or when it is challenging to determine an appropriate order based on the available information. Employing Optuna in such cases provides a systematic and automated approach to optimize model performance without heavily relying on prior knowledge.

We selected the order with the lower MSE from those determined using domain knowledge and Optuna as the final order for each model. To evaluate the performance of the four classical models, we computed MSE, RMSE, and MAE, with the results presented in Table 4. The comparison between predicted and actual values is visualized in Figure 6. It is evident that the performance of the classical models is not very satisfactory, which is reasonable given that these models were designed using the simplest possible approaches based solely on the available information, serving primarily as baseline references.

7.3 Modern techniques

We implemented three modern techniques, including Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), and Sequence-to-Sequence (Seq2Seq).

LSTM, a type of recurrent neural network (RNN), is designed to capture long-term dependencies in sequential data by utilizing specialized memory cells that allow the model to retain important information over long periods. BiLSTM, an extension of LSTM, processes data in both forward and backward directions, which enhances the model's ability to understand context from both past and future time steps. The Seq2Seq model is a more complex architecture that employs two LSTM networks: an encoder to process the input sequence and a decoder to generate the output sequence. This model is particularly useful for tasks like sequence translation and time series forecasting, where there is a need to map input sequences to output sequences.

The three modern techniques were trained with identical hyperparameters, as outlined in Table 1, as well as the same number of layers and hidden sizes. This setup ensures that the results

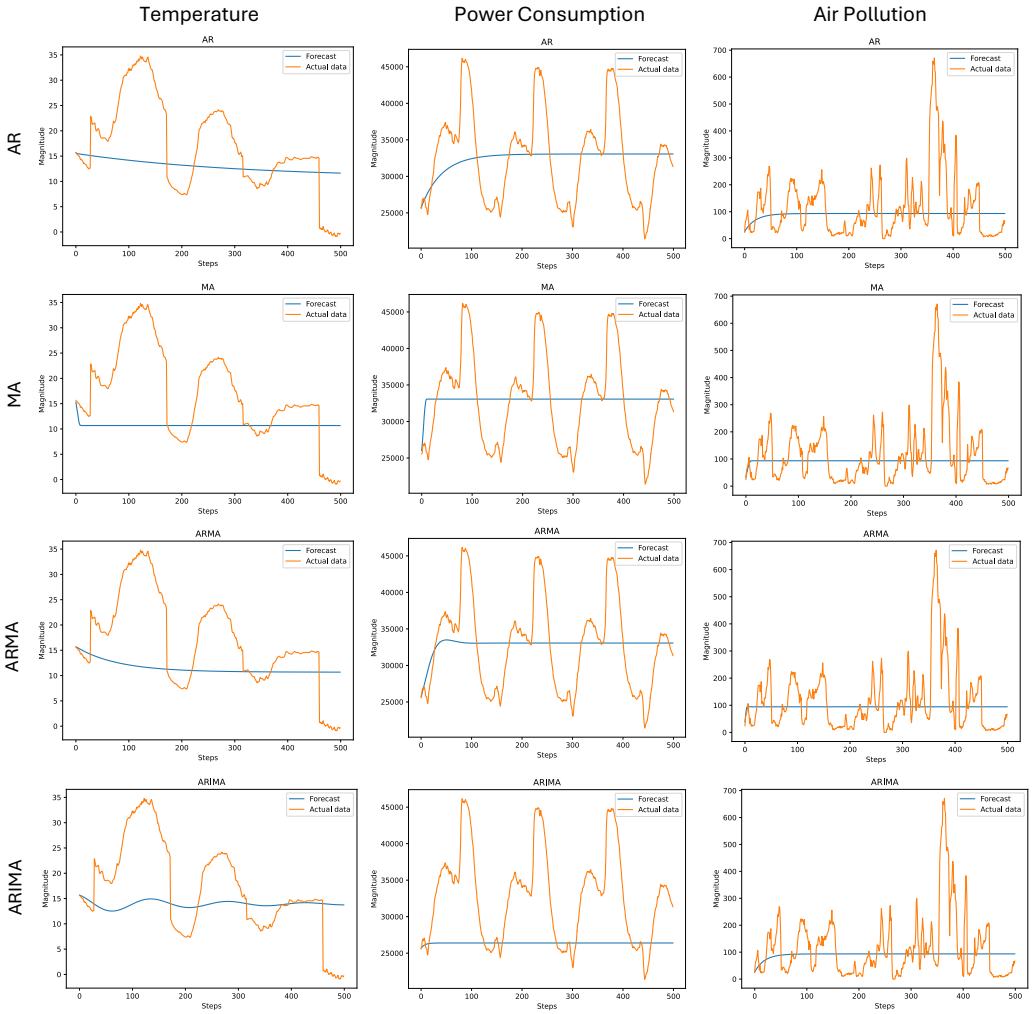

Figure 6: Classical Models: Prediction vs. Ground Truth

	Domain Order	MSE	Optuna Order	MSE
AR				
Temperature	(2)	0.038986	(4)	0.038950
Power Consumption	(2)	0.059108	(10)	0.059213
Air Pollution	(1)	0.019484	(5)	0.019482
MA				
Temperature	–	–	(10)	0.038870
Power Consumption	–	–	(10)	0.059218
Air Pollution	–	–	(10)	0.019472
ARMA				
Temperature	(2,3)	0.038971	(8,2)	0.038892
Power Consumption	(2,2)	0.059206	(4,5)	0.059281
Air Pollution	(1,2)	0.019483	(2,6)	0.019458
ARIMA				
Temperature	(1,1,1)	0.049111	(7,1,6)	0.044493
Power Consumption	(1,1,2)	0.055292	(11,1,6)	0.055068
Air Pollution	(4,1,5)	0.019482	(13,1,10)	0.019479

Table 3: MSE Comparison between Domain Knowledge and Optuna. Both prediction and ground truth are normalized using the equation in 1.

	MSE	RMSE	MAE
AR			
Temperature	0.038950	0.197359	0.161282
Power consumption	0.059108	0.243121	0.204268
Air pollution	0.019482	0.139578	0.101504
MA			
Temperature	0.038870	0.197156	0.160967
Power consumption	0.059218	0.243347	0.204480
Air pollution	0.019472	0.139542	0.101515
ARMA			
Temperature	0.038892	0.197210	0.161051
Power consumption	0.059206	0.243322	0.204456
Air pollution	0.019458	0.139493	0.101845
ARIMA			
Temperature	0.044493	0.210933	0.178464
Power consumption	0.055068	0.234666	0.193346
Air pollution	0.019479	0.139569	0.101566

Table 4: Performance metrics for Classical Models. Both prediction and ground truth are normalized using the equation in 1.

are directly comparable across models, allowing us to isolate and assess the specific impact of each model's architecture on performance. MSE, RMSE, and MAE are also calculated to assess the performance, as summarized in Table 5. Figure 7 illustrates the comparison between the predicted and actual values for LSTM, BiLSTM, and Seq2Seq models. The modern techniques demonstrate significantly better performance than the classical models, as reflected in their lower

MSE values. Additionally, the comparison plots between predictions and actual values further highlight their superior accuracy. Among the three modern techniques, the LSTM model, despite having the simplest architecture, achieves the lowest MSE. This suggests that for the datasets used in this study, the straightforward architecture of the LSTM is sufficient to handle the tasks effectively.

	MSE	**RMSE**	**MAE**
LSTM			
Temperature	0.000127	0.011277	0.004017
Power consumption	0.000175	0.013225	0.008923
Air pollution	0.001260	0.035502	0.018459
Bi-LSTM			
Temperature	0.000131	0.011437	0.004310
Power consumption	0.000179	0.013362	0.009071
Air pollution	0.000261	0.035516	0.019128
Seq2Seq			
Temperature	0.000128	0.011307	0.004099
Power consumption	0.000226	0.015046	0.009872
Air pollution	0.000247	0.035312	0.018662

Table 5: Metrics comparison for Modern Techniques. Both prediction and ground truth are normalized using the equation in 1.

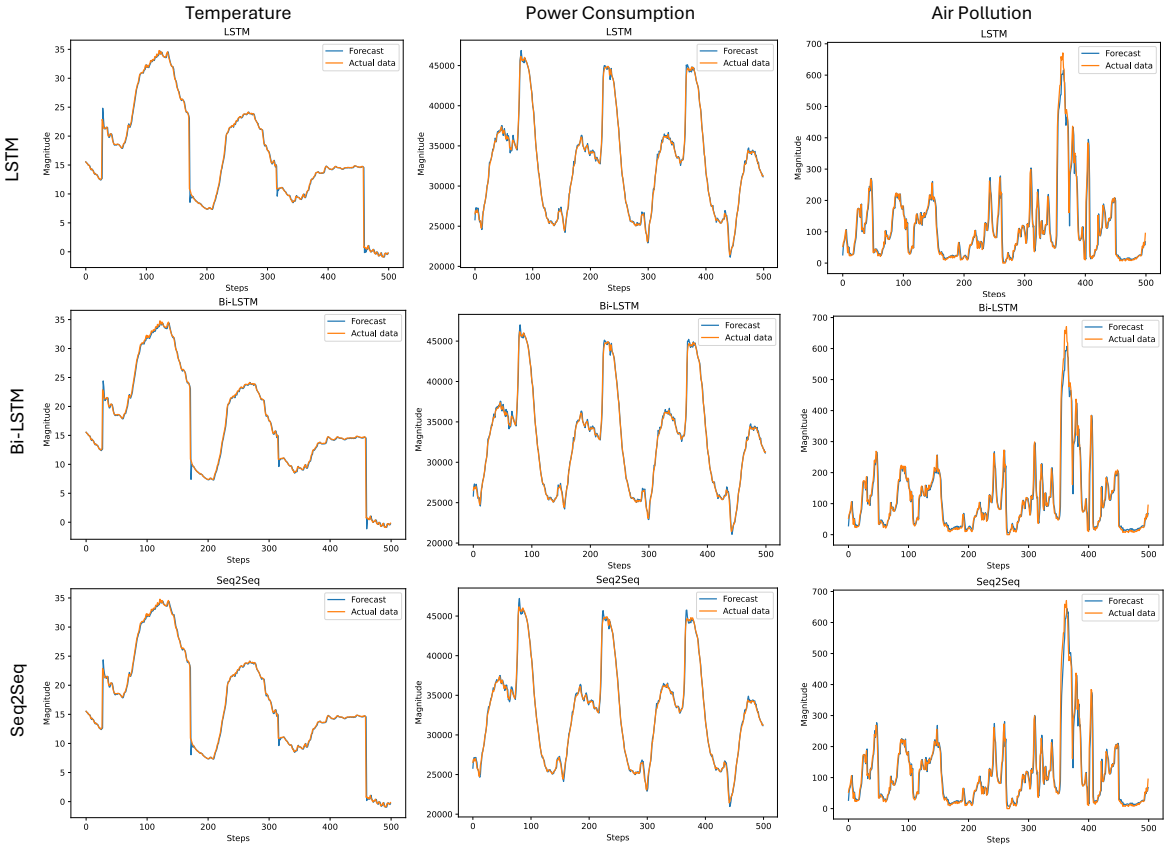

Figure 7: Modern Techniques: Prediction vs. Ground Truth

7.4 State of Art

Finally, we utilized the Transformer model, which has recently become a state-of-the-art architecture due to its self-attention mechanism. The Transformer model allows for more parallelization and better capture of long-range dependencies compared to traditional RNN-based models. The self-attention mechanism enables the model to weigh the importance of different time steps when making predictions, significantly improving forecasting accuracy, especially in complex datasets. To ensure consistency in comparison, we used the same hyperparameters listed in Table 1 as those applied to modern techniques. Based on the prediction vs. actual plots in Figure 8 and the metrics presented in Table 6, the Transformer model outperforms the classical models we implemented. However, its performance does not surpass that of the three modern techniques. This suggests that for these three datasets, a complex Transformer architecture may not be necessary to achieve effective performance.

	MSE	**RMSE**	**MAE**
Temperature	0.000599	0.024466	0.016530
Power consumption	0.000467	0.021602	0.015685
Air pollution	0.001370	0.037011	0.019526

Table 6: Performance metrics for Transformer model. Both prediction and ground truth are normalized using the equation in 1.

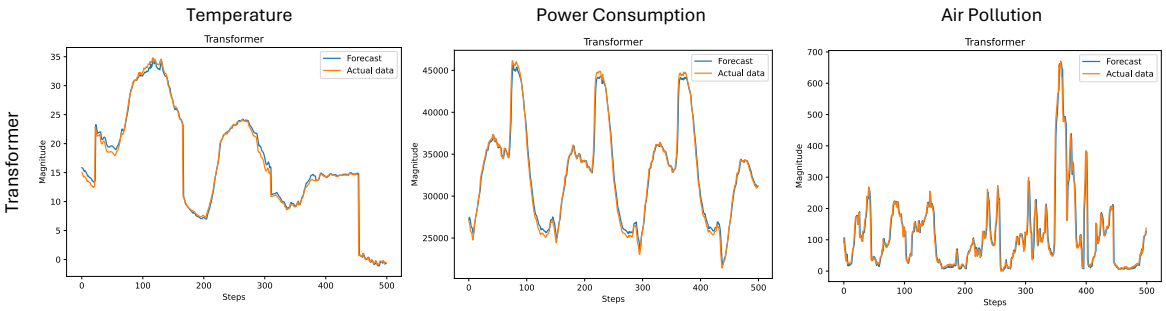

Figure 8: Transformer: Prediction vs. Ground Truth

7.5 Conclusion

In this study, we evaluated the performance of four classical models (AR, MA, ARMA, ARIMA), three modern techniques (LSTM, BiLSTM, Seq2Seq), and a state-of-the-art Transformer model on three publicly available datasets. Overall, our findings indicate that while we implemented relatively simple classical models, both modern techniques and the Transformer model consistently outperformed the classical approaches. Among the deep learning-based models, LSTM, with its simplest architecture, achieved the lowest MSE. This suggests that even the simplest deep learning architecture is sufficient to handle the data effectively for these datasets.

However, there are some limitations to this study. For classical models, we selected relatively simple configurations based on available information, which may not represent their optimal performance. It is possible to identify better-performing classical models with more extensive exploration. Similarly, when using Optuna for hyperparameter tuning, we constrained the maximum potential order and the number of trials. Increasing these limits could lead to the discovery of more suitable models.

For modern techniques and the state-of-the-art Transformer model, we trained all models using a fixed sequence length and a common set of hyperparameters. However, these hyperparameters

may not be the most optimal for each model or dataset.

Future work could include implementing Seasonal Autoregressive Integrated Moving Average (SARIMA) [11] models to better capture the strong seasonality present in the datasets. Additionally, employing advanced techniques like the Box-Jenkins [11] methodology and other model variations could provide a more comprehensive performance comparison. By addressing these limitations and exploring alternative approaches, we aim to further refine our findings and expand the scope of this research.

References

[1] R. Adhikari and R. K. Agrawal, "An introductory study on time series modeling and forecasting," *arXiv preprint arXiv:1302.6613*, 2013.

[2] Y. S. Kim, S. T. Rachev, M. L. Bianchi, I. Mitov, and F. J. Fabozzi, "Time series analysis for financial market meltdowns," *Journal of Banking & Finance*, vol. 35, no. 8, pp. 1879–1891, 2011.

[3] J. M. Box-Steffensmeier, J. R. Freeman, M. P. Hitt, and J. C. Pevehouse, *Time series analysis for the social sciences*. Cambridge University Press, 2014.

[4] M. Mudelsee, "Climate time series analysis," *Atmospheric and*, vol. 397, 2010.

[5] J.-S. Chou and D.-S. Tran, "Forecasting energy consumption time series using machine learning techniques based on usage patterns of residential householders," *Energy*, vol. 165, pp. 709–726, 2018.

[6] R. Gao, Y. Tang, K. Xu, Y. Huo, S. Bao, S. L. Antic, E. S. Epstein, S. Deppen, A. B. Paulson, K. L. Sandler *et al.*, "Time-distanced gates in long short-term memory networks," *Medical image analysis*, vol. 65, p. 101785, 2020.

[7] A. Shakeel, T. Tanaka, and K. Kitajo, "Time-series prediction of the oscillatory phase of eeg signals using the least mean square algorithm-based ar model," *Applied Sciences*, vol. 10, no. 10, p. 3616, 2020.

[8] R. Jafari and A. H. Jafari, "Speech recognition using arma model and levenberg-marquardt algorithm," in *Intelligent Systems Conference*. Springer, 2024, pp. 351–367.

[9] V. K. R. Chimmula and L. Zhang, "Time series forecasting of covid-19 transmission in canada using lstm networks," *Chaos, solitons & fractals*, vol. 135, p. 109864, 2020.

[10] Y. Jin, L. Hou, and Y. Chen, "A time series transformer based method for the rotating machinery fault diagnosis," *Neurocomputing*, vol. 494, pp. 379–395, 2022.

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

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 3104–3112.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[16] B. Mnassri, "Weather station beutenberg dataset," 2020, accessed: 2023-12-11. [Online]. Available: https://www.kaggle.com/datasets/mnassrib/jena-weather-dataset/data

[17] A. Salam and A. El Hibaoui, "Power Consumption of Tetouan City," UCI Machine Learning Repository, 2018, DOI: https://doi.org/10.24432/C5B034.

[18] R. Roy, "Air pollution forecasting - lstm multivariate," 2020, accessed: 2024-12-02. [Online]. Available: https://www.kaggle.com/datasets/rupakroy/lstm-datasets-multivariate-univariate

[19] L. R. Medsker, L. Jain *et al.*, "Recurrent neural networks," *Design and Applications*, vol. 5, no. 64-67, p. 2, 2001.

[20] A. Jafari, "Recurrent neural network (rnn) lstm & gru," 2024, nLP Lecture slides, George Washington University. Screenshot taken by author.

[21] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[22] R. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[23] C. Chatfield and H. Xing, *The analysis of time series: an introduction with R*. Chapman and hall/CRC, 2019.

[24] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.