DATA SCIENCE: TIME SERIES FORCASTING

Noureddine BERTRAND, Dorian GROUTEAU, Olivier JIANG, Lukas KAIBEL, Hui ZHAO

UFR Mathématiques et Informatique, Université of Paris cité

Mathematik und Informatik, Freie Universität Berlin

E–mail: {*noureddine-kamil.bertrand — dorian.grouteau — olivier.jang — hui.zao*}*@etu.u-paris.fr, lukas.kaibel@fu-berlin.de*

SUMMARY: The purpose of this report is to present our process to forecast uni variate and multivariate data using basic statistical technique and advanced technique. We will showcase the performance of RNN's technique and compare them with CNN and statistical method. Time Series Forecasting have many field of application ranging from market prediction to flood forecasting.

Key words. Time Series Forcasting – Data science – RNN – LSTM – ARIMA – CNN

1. INTRODUCTION

Time series forecasting constitutes the task of leveraging historical data to predict future trends within a time series. This domain finds application across diverse fields such as economics, energy consumption analysis, weather forecasting, and climate change prediction. With the emergence of Transformer models, significant advancements have been made in time series forecasting, making them the predominant architecture in recent studies (Nie et al., 2023). Additionally, Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) have shown promising results in this domain.

In this report, we outline our implementation and perform a comparative analysis of the effectiveness of different forecasting models. We aim to predict the next 100 values for both the univariate ETTh1 dataset and a multivariate exchange rate dataset. The remainder of the report is structured as follows: Section 2 provides an overview of related work in time series forecasting. Section 3 delves into the architectural details of the different models implemented in our study. Section 4 outlines our experimental setup, describes the datasets utilized and compare our result. Finally, Section 5 presents our conclusions and suggests avenues for further improvement.

2. RELATED WORK

The time series forcasting task could be approach with different tool. We list them as follow.

2.1. XGBOOST

XGBoost, which stands for "Extreme Gradient Boosting," is a highly efficient machine learning algorithm designed for structured or tabular data. It has gained significant popularity in the data science community due to its speed, accuracy, and flexibility. XGBoost is a type of gradient boosting framework that creates an ensemble of decision trees to improve prediction accuracy through a process known as gradient boosting.

2.2. ARIMA

Autoregressive Integrated Moving Average (ARIMA), is a statistical method. It combines three key components: autoregression (AR), differencing (I), and moving average (MA). The autoregressive component models the relationship between an observation and a certain number of lagged observations. The differencing component transforms the data to achieve stationarity, while the moving average component models the relationship between an observation and a residual error from a moving average model applied to lagged observations.

2.3. Transformers

Initially used for Natural Language Processing task, transformers and the attention mechanism showed particular promising result in time series forecasting. Transformers, renowned for their ability to capture complex patterns and long-range dependencies in sequential data, have been adapted to process time series data by treating each time step as a sequence element. The attention mechanism allows these models to weigh the relevance of past observations dynamically, enabling them to make more accurate predictions based on historical trends.

2.4. CNNs

Convolutional Neural Networks (CNNs) have demonstrated promising results in time series forecasting tasks. Recent advancements, such as the PatchMixer architecture introduced by Gong et al., have shown remarkable performance, surpassing state-of-the-art models. CNNs leverage convolutional operations to extract spatial and temporal patterns from time series data, enabling them to effectively capture local and global dependencies. The PatchMixer architecture, in particular, has gained attention for its ability to efficiently process time series data by leveraging patch-based representations and mixer layers. These advancements highlight the potential of CNNs in improving forecasting accuracy and handling complex time series data across various domains.

2.5. RNNs

Recurent Neural Network has been the primary choice for time series forcasting. LSTM and GRU cell are particularly good for grasping intersect information but tend to decrease their performance as the forecast horizon increases.

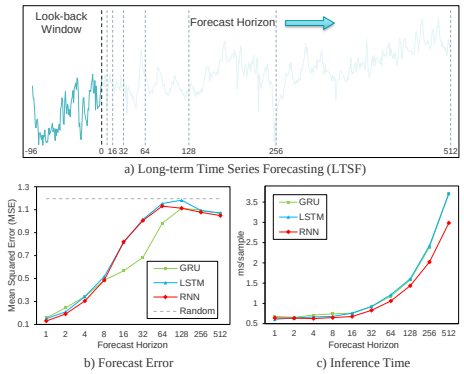

Fig. 1: Challenges faced by vanilla RNN and its variants in LTSF. Data is obtained from the ETTh1 dataset.

2.6. Hybrid

The hybrid approach uses the best of two worlds. The Segment Recurrent Neural Network (Lin et al., 2023) uses the transformer approach embedded in a recurrent neural network using GRU cells. This model (Fig.2) aims to reduce the RNN iteration to facilitate the convergence. The first phase consist of encoding segment wise iteration and is followed by a decoding phase. SegRNN showed stat of the art result and outperform traditional SOTA approach.

3. MODEL ARCHITECTURE

Given the intricate nature of models like transformers or SegRNN, we opted to prioritize simplicity and gradually escalate complexity. Our research centers on implementing basic models, namely ARIMA, XGBoost, RNNs, and CNN, with the intention of systematically enhancing their sophistication. We performed various test regarding the configuration of the model.

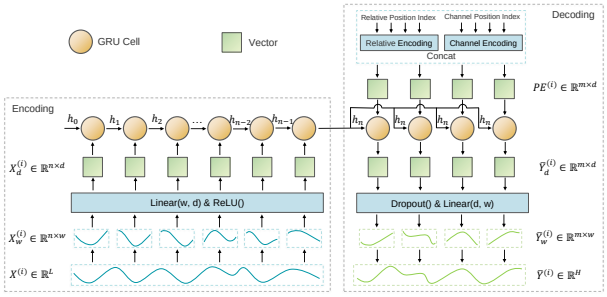

Fig. 2: The model architecture of SegRNN.

3.1. XGBoost

The XGBoost model employs an ensemble of 100 decision trees, each contributing to predictions through weighted combinations. A learning rate of 0.1 controls step size shrinkage. To limit overfitting, the maximum depth of each tree is set up to 3 levels. The objective function 'reg:squarederror' guides optimization, minimizing mean squared error between predicted and actual values.

3.2. ARIMA

To construct an ARIMA model, several steps are followed. First, the time series data is prepared by cleaning and ensuring a consistent time interval. Next, the parameters of the ARIMA model are identified: p (autoregressive order), d (integration order), and q (moving average order). These parameters are determined by analyzing the autocorrelation and partial autocorrelation plots of the time series data. The autoregressive order was set to 4, the integration order to 1 and the moving average to 4. By selecting a lower value for d, the model captures more effectively the autocorrelation and seasonality of the data, leading to better forecast results. A large value of d brings a worse MAE. For p and q values, if they are set higher the results are worse and if they are too low then it is irrelevant. Having d to a low value and either p or q to a low value too and the other one to a greater value leads to approximately the same results than the values presented above. Once the parameters are identified, the model is trained using historical data. Finally, the model's performance is evaluated using validation data, and adjustments may be made to improve forecasting accuracy.

3.3. RNNs

The first model (Fig.3) consist of a Long Short-Term Memory (LSTM) layer configured with 256 units. Preceding the LSTM layer is an input layer specified with the shape (1000, 1) for the ETTh1 dataset and (500,1) for the exchange rate dataset. Additionally, a dropout layer with a dropout rate of 0.2 is incorporated to mitigate overfitting, promoting better generalization. Following the LSTM layer, a flatten layer reshapes the output into a one-dimensional vector, enabling seamless connectivity with the subsequent dense layer. The final layer in the architecture is a dense layer comprising 150 units for ETTh1 and 100 for the exchange rate dataset, responsible for performing a linear transformation of the input data.

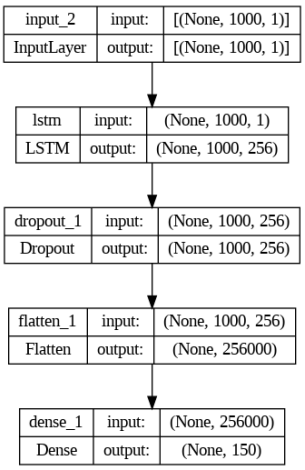

Fig. 3: One layer LSTM

The second model replace the LSTM layer of the first model by a GRU layer.

The third model (Fig.4) is composed of a first layer of LSTM followed by a second layer of GRU.

3.4. CNNs

The CNN architecture (Fig.5) goes as follow:

(i) The convolutional layer applies a one-dimensional convolution operation with 64 filters and a kernel size of 10. This layer is responsible for extracting spatial features from the input sequence using the specified filter configuration and activation function (ReLU).

(ii) The subsequent dense layer consists of 64 units and utilizes the ReLU activation function. This fully connected layer further processes the extracted features, allowing the model to capture complex patterns in the data.

(iii) To prevent overfitting, a dropout layer with a dropout rate of 0.2 is introduced after the dense layer. Dropout randomly drops a fraction of the input units during training, reducing the risk of the model relying too heavily on specific features.

(iv) Following dropout regularization, a flatten layer reshapes the output from the previous layer into a one-dimensional vector. This step prepares the data for input into the subsequent dense layer.

(v) The final dense layer comprises 100 units, serving as the output layer of the model. This layer performs a linear transformation of the input data, producing the final predictions or classifications.

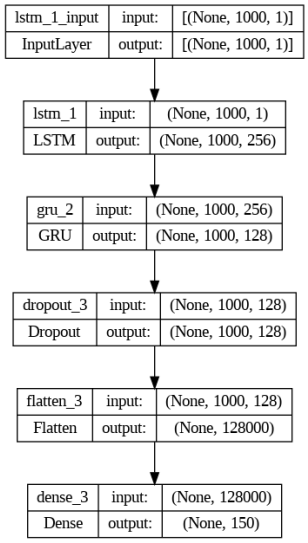

Fig. 4: One layer LSTM followed by one layer GRU

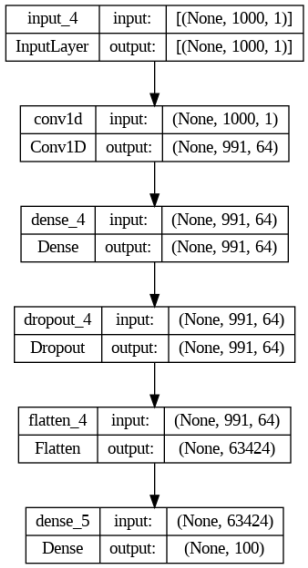

Fig. 5: CNN architecture

3.5. LSTM + CNN

The LSTM+CNN model is a multi-branch neural network architecture (Fig.6) designed to lever-

age the strengths of both Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs). This hybrid approach aims to capture complex patterns and relationships in time series data by utilizing the sequence processing capabilities of LSTM and the feature extraction power of CNN.

The LSTM+CNN model consists of two primary branches: an LSTM branch and a CNN branch. The outputs from these two branches are concatenated and then passed through a dense (fully connected) layer to produce the final prediction.

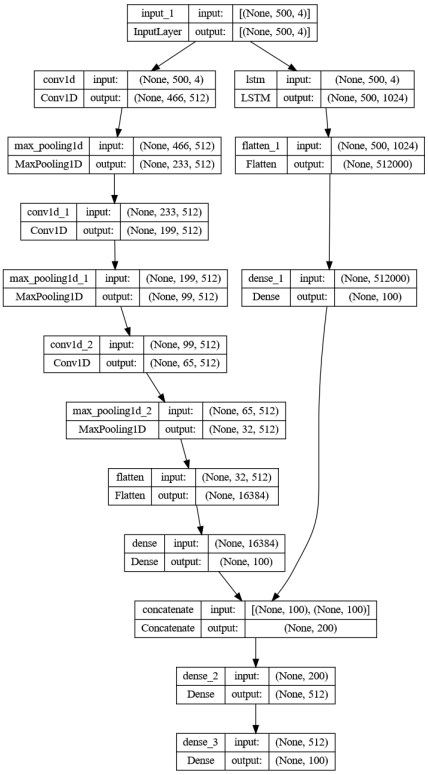

Fig. 6: LSTM+CNN architecture

4. DATA

The project is composed of two dataset. The ETTh1 dataset traditionally used in time series bench-marking and an exchange rate dataset.

4.1. ETTh1

This dataset consists of a single variable comprising 17,320 values, with the prediction target centered on the 'OT' column. Due to the wide spread of the data [-4.07 ; 46.00], we performed Z-score normalization that can be describe as follow:

$$Z = \frac{X - \mu}{\sigma} \tag{1}$$

Where X represent the a data point, μ is the mean and σ is the standard deviation of the dataset.

4.2. Exchange Rate

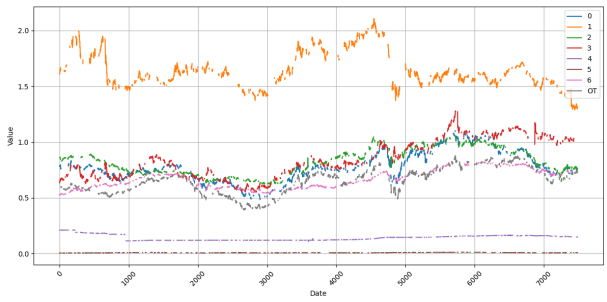

Fig. 7: Plot of all feature with missing values

For this dataset, we are focusing on predicting the next 100 values of the '6' variable. The multivariate exchange rate dataset required preprocessing due to missing values (Fig.7) but didn't need any normalisation. Various methods were considered, including mean imputation, linear interpolation, Last Observation Carried Forward (LOCF), and Next Observation Carried Backward (NOCB).

Initially, we applied a rolling mean imputation with a window size of 20 and a minimum period of 1 (meaning it calculates the mean even if there are fewer than 20 non-NaN values available) and then shifts the result by one position, we than perform a LOCF and NOCB for filling the leading and tailing missing values (Fig.10). In a second approach, we performed a linear interpolation followed by LOCF and NOCB (Fig.11). Ultimately, the latter method was selected, although no significant differences in performance were observed.

Finally, as Figure 7 shows, even with missing value, some variables doesn't seem to be correlated with the target (variable '6'). We tried to analyse the degree of correlation between the feature with the goal to eliminate uncorrelated value (Fig.8 and 9).

A Value close to 1 has a strong correlation while a value close to 0 represent the absence of influence between the two considered variable. We can see from the Figure 8 and Figure 9 that the imputation has negligible influence on the correlations. Surprisingly, we can see a height correlation between or target variable and the variable '5'. We will finally eliminate the variable '1', '4', '5' and 'OT' as better result were obtain without them.

Another challenge associated with this dataset is its limited size (only 7488 data points). This limitation raises concerns, particularly when further dividing the dataset into training, validation, and testing sets, potentially resulting in insufficient data for ef-

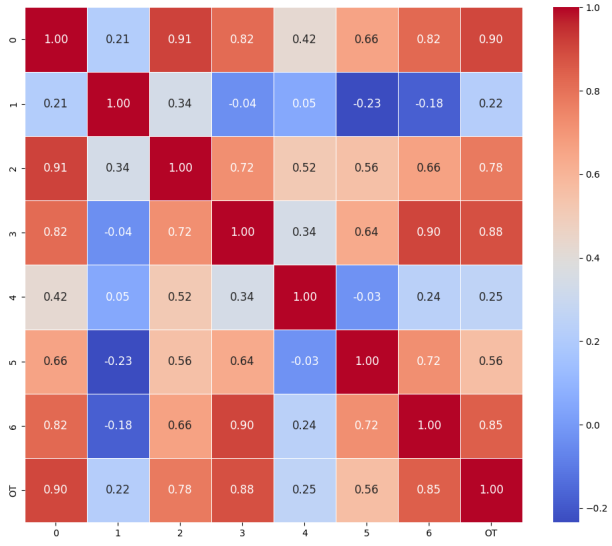

Fig. 8: Correlation after imputation

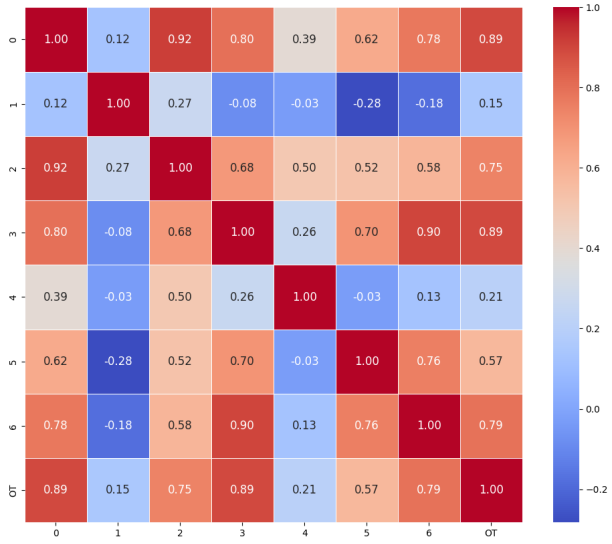

Fig. 9: Correlation before missing value imputation

fective learning. We will address this concerne in the next section.

5. EXPERIMENT

In this section, we describe the experimental setup and results obtained using various forecasting models. Our goal is to evaluate the performance of different approaches on two distinct datasets: the univariate ETTh1 dataset and the multivariate exchange rate dataset. For both datasat, we performed a data split to obtain a training, validation and test set (70% / 20% / 10%).

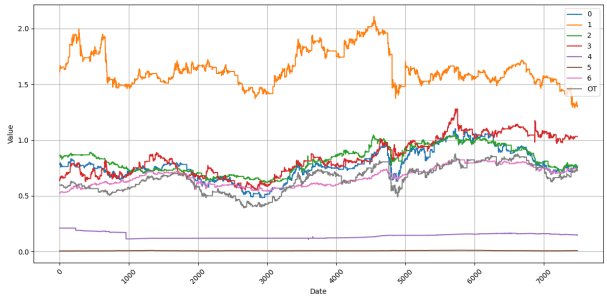

Fig. 10: Mean with LOCF + NOCB

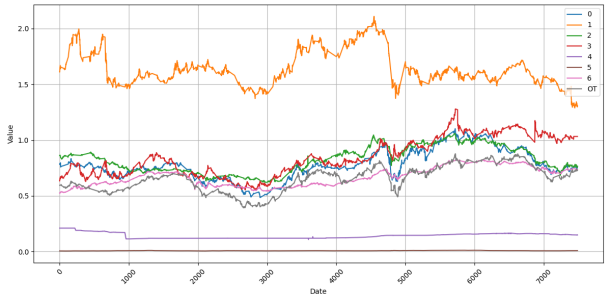

Fig. 11: Linear inputation with LOCF + NOCB

5.1. Step A : ETTh1

For the ETTh1 dataset, we aimed to predict the next 100 values. We used different look-back windows size (500, 720, 1000). We also tried different forecast horizon size (1, 100, 150). Overall, we obtained the best result using a look-back window of 1000 with a forecast horizon of 150 but an offset of -50. This mean that we use value $[1 : 999]$ to predict value $[950 : 1099]$. This offset allow the model to grasp the tendency of the most recent fluctuation.

The performance of the models was evaluated based on standard metrics like Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE). These metrics provide a comprehensive view of the accuracy and robustness of the forecasting models.

The ARIMA model served as the baseline for our experiment. While it was straightforward to implement, it required careful tuning of parameters to achieve optimal results. We managed to achieve a score (MAE) of 1.21 with this model at the beginning of the project, which was encouraging. However, the ARIMA model's performance was generally acceptable, but it struggled with more complex patterns and long-range dependencies. This limitation suggests that it might not be the best choice for datasets with intricate structures or non-linear trends.

For the RNN models, we tested different configurations with LSTM and GRU layers. The best results were obtained with a single-layer LSTM architecture, demonstrating its capability to capture temporal pat-

terns effectively. The LSTM model outperformed the ARIMA model in terms of both MAE and RMSE, indicating its superiority in forecasting. We achieved a best MAE score of 0.44 with this LSTM architecture, reinforcing the model's effectiveness in handling complex time series data.

The CNN model was implemented with one-dimensional convolutional operations to extract spatial features from the time series data. While the CNN model performed reasonably well, its results were generally comparable to those of the ARIMA model, suggesting that this approach might require further optimization for time series forecasting.

The result of the best implementation of each model are shown in Table 1.

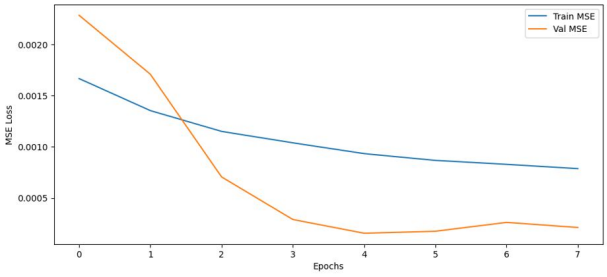

Fig. 12: CNN loss MSE

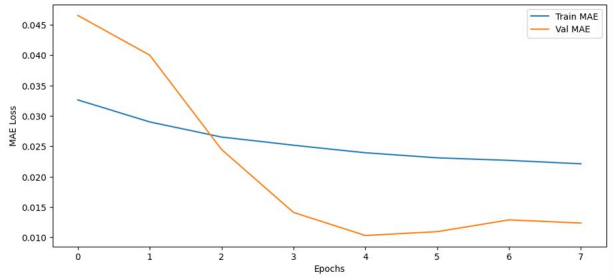

Fig. 13: CNN loss MAE

Table 1: Results Step A

Model	MAE	MSE
ARIMA	1.22	1.17
LSTM	0.44	0.35
CNN	0.97	0.7

5.2. Step B : Exchange rate

For the exchange rate dataset, the challenge was more significant due to the multivariate nature of the data, the presence of missing values and the limited size of the dataset. We employed various imputation techniques to handle the missing data and performed correlation analysis to determine which features were relevant for forecasting. To restrain the effect of a limited size dataset, we used a look back window of 500 and performed a sliding window mechanism where we slide the forecast index by one (ie. [1:499] for the first set than [2:500] for the second,etc..).

To tackle this dataset, we utilized four different models: LSTM, CNN, XGBoost, and a hybrid LSTM+CNN model. After running a series of experiments, we ranked the models based on their Mean Absolute Error (MAE) scores, from best to worst. The order of performance was as follows:

- **LSTM+CNN**: This hybrid model delivered the best MAE, demonstrating the combined strengths of LSTM's sequence-handling capabilities and CNN's feature extraction power.
- **LSTM**: The single-layer LSTM architecture performed well, ranking second in terms of MAE. This result underscores the effectiveness of LSTM in capturing temporal patterns in multivariate time series data.
- **CNN**: Although this model came in third place, it still showed good performance, suggesting that the convolutional approach was effective in recognizing patterns in complex datasets.
- **XGBoost**: This model had the highest MAE, indicating that it was less effective in handling the complexities of the multivariate dataset compared to the other neural network-based models.

The results shown in Table 2 suggest that hybrid models combining convolutional neural network and LSTM offer significant advantages in complex multivariate forecasting tasks. The strong performance of the LSTM+CNN model indicates that further exploration of hybrid approaches could yield even better results.

Table 2: Results Step B

Model	MAE	MSE
LSTM	0.0130	0.0002
CNN	0.0134	0.0003
XGBOOST	0.0654	0.0035
LSTM+CNN	0.0127	0.0002

5.3. Parameters

In machine learning, the performance of models like Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) can vary greatly based on parameter settings. These settings affect not only the accuracy of predictions but also training time and model stability.

5.3.1. *Parameters for LSTM*

- **Number of LSTM Units**: Defines the dimensionality of the output space for each LSTM layer. A larger number of units allows capturing more complex patterns but increases the risk of overfitting.
- **Dropout Rate**: A regularization technique to reduce overfitting.

5.3.2. *Parameters for CNN*

- **Number of Filters**: Determines the number of filters applied by each convolutional layer.
- **Kernel Size**: The size of the convolution kernel. Smaller kernel sizes capture local patterns, while larger ones capture broader patterns.
- **Dense Layer Units**: The number of units in the dense layer to make final predictions.
- **Activation Functions**: Activation functions define how signals are transformed between layers.

To find the optimal settings, various parameter combinations were tried and evaluated to ensure a balance between accuracy and efficiency. This iterative process of parameter tuning involved testing different configurations and observing their impact on model performance.

Careful tuning of these parameters is crucial for optimal performance in time series forecasting. The approach in this report, using balanced parameters, resulted in improved accuracy with reasonable computational requirements.

6. CONCLUSION

In this report, we presented a comparative study of various models for time series forecasting, focusing on univariate and multivariate datasets. The main goal was to evaluate the performance of different models and identify the most effective approaches for forecasting time series data.

For the univariate ETTh1 dataset, the Recurrent Neural Network (RNN) approach, particularly the Long Short-Term Memory (LSTM) model, outperformed the baseline ARIMA model, achieving the best Mean Absolute Error (MAE) score of 0.44. This result highlighted the effectiveness of LSTM in capturing temporal patterns and its superior performance over simpler statistical methods.

For the multivariate exchange rate dataset, the hybrid LSTM+CNN model delivered the best MAE, demonstrating the advantage of combining different neural network architectures for complex time series forecasting. The individual LSTM and CNN models also showed promising results, indicating that neural network-based approaches are generally effective in handling complex data patterns. On the other hand, the XGBoost model, a popular gradient boosting technique, ranked last in terms of MAE, suggesting that it might not be the best choice for this type of forecasting task.

The study underscores the potential benefits of hybrid neural network models, combining diverse architectural strengths. Notably, the LSTM+CNN model exhibits significant performance, indicating promising directions for further exploration in hybrid methodologies. These results advocate for ongoing investigation into advanced hybrid architectures, potentially incorporating transformers, which were not explored in this report, alongside innovative techniques for time series forecasting.

REFERENCES

Gong, Z., Tang, Y., and Liang, J. 2023, PatchMixer: A Patch-Mixing Architecture for Long-Term Time Series Forecasting

Lin, S., Lin, W., Wu, W., et al. 2023, SegRNN: Segment Recurrent Neural Network for Long-Term Time Series Forecasting

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. 2023, A Time Series is Worth 64 Words: Long-term Forecasting with Transformers