Enhancing Time Series Forecasting: Fulfilling the Potential of Crossformer with Feature Engineering

Chenyuan Zhou

May 18, 2025

Abstract

The Crossformer model, while powerful for multi-variate time series forecasting through its cross-attention mechanism, often struggles to reach its full potential with limited or noisy data. This paper introduces a hybrid approach that combines traditional time series decomposition techniques with the Crossformer architecture to enhance forecasting performance. We employ drift window-based seasonal decomposition to extract trend and seasonality patterns which are then fed as additional features to the Crossformer model. Our experiments on the ETTh1 dataset demonstrate that this approach not only improves prediction accuracy but also leads to more stable training, with an approximate 15% reduction in mean squared error compared to the baseline Crossformer. Furthermore, we show that training the model to predict both original and engineered features yields better performance than focusing solely on original features, suggesting that explicit modeling of time series components guides the learning process effectively.

1 Introduction

Time series forecasting is a fundamental task in many fields, including finance, energy, and transportation. Traditional methods for time series forecasting, such as ARIMA and exponential smoothing, rely on statistical properties of the data but often struggle with complex, non-linear patterns and interactions between multiple time series. In recent years, transformer-based models have shown remarkable success in capturing these complex relationships.

The Crossformer architecture represents a significant advancement in transformer-based time series forecasting. It is specifically designed to capture temporal dependencies and relationships between different time series through a cross-attention mechanism, enabling it to learn interactions and make predictions based on information from multiple sources. This capability gives Crossformer a substantial advantage over traditional time series forecasting methods which typically analyze each series independently.

However, the Crossformer model's effectiveness is contingent on having adequate data quality and quantity. Its strength lies in learning complex relationships between different time series, but this requires sufficient features and data for effective training. When data is limited or noisy, the model may struggle to learn these relationships effectively, leading to suboptimal performance.

This paper proposes a hybrid approach that integrates traditional time series decomposition techniques with the Crossformer model to enhance forecasting accuracy. By extracting meaningful patterns from time series data through decomposition methods and providing these as additional features to the Crossformer, we can help the model better understand the underlying structure of the data and improve its forecasting capabilities.

Our main contributions are:

- A novel approach for enhancing Crossformer with feature engineering based on traditional time series decomposition techniques
- A drift window method for localized time series decomposition that preserves temporal context
- An evaluation framework that prevents future information leakage during testing while maintaining computational efficiency during training
- Empirical evidence showing that training the model to predict both original and engineered features leads to better performance than focusing solely on original features

2 Related Work

2.1 Time Series Decomposition

Time series decomposition has long been a fundamental technique in time series analysis. Classical methods include Seasonal and Trend decomposition using LOESS (STL) [1] and seasonal decomposition of time series by moving averages. These approaches decompose a time series into trend, seasonal, and residual components, which can help in understanding the underlying patterns in the data.

2.2 Transformer-based Time Series Forecasting

The Transformer architecture, initially introduced for natural language processing tasks, has been adapted for time series forecasting in recent years. Models like Informer [2] and Autoformer [3] have shown impressive results by leveraging the self-attention mechanism to capture long-range dependencies in time series data.

The Crossformer model [4] further extends this approach by introducing a cross-attention mechanism that can learn interactions between different time series, allowing it to make predictions based on information from multiple sources.

2.3 Hybrid Approaches

Several studies have explored hybrid approaches that combine traditional time series methods with deep learning. For instance, N-BEATS [5] incorporates domain knowledge about trend and seasonality into a deep neural network architecture. Similarly, LSTM-MSNet [6] combines LSTM networks with multiple seasonal decomposition. Our work follows this hybrid philosophy but specifically focuses on enhancing the Crossformer architecture with features derived from time series decomposition.

3 Methodology

Our approach aims to enhance the Crossformer model by incorporating domain knowledge from traditional time series analysis. We focus on extracting meaningful features from time series data using decomposition techniques and providing these features as additional inputs to the Crossformer model.

3.1 Feature Engineering through Time Series Decomposition

We first analyze the time series data to identify suitable methods for feature extraction. Figure 1 shows an example of the original features in our dataset, which exhibits complex seasonality patterns.

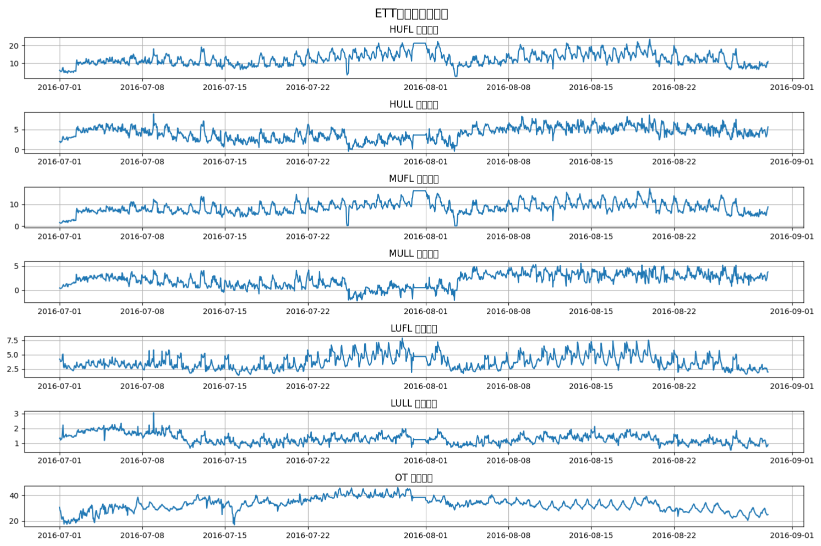

Figure 1: Original features of the time series data showing complex seasonal patterns.

Traditional time series decomposition methods like STL decomposition can extract trend and seasonality components. However, applying these methods to the entire time series would overlook local periodicity features. Therefore, we propose a drift window approach combined with STL decomposition to capture both global and local patterns.

3.2 Drift Window Method for Local Decomposition

Our drift window method involves sampling windows from the time series and applying STL decomposition to each window separately. The results from overlapping windows are then averaged to generate the final seasonal and trend prediction features. This approach preserves local context while maintaining computational efficiency.

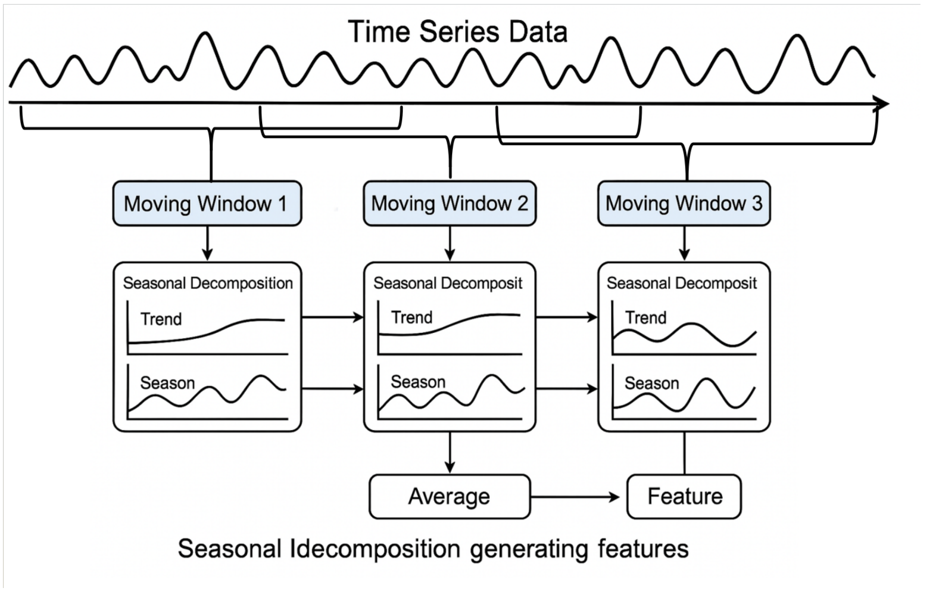

Figure 2: Illustration of the drift window method for local decomposition.

Through extensive experimentation, we determined that a daily periodicity (24 hours) is more stable for our high-frequency dataset compared to weekly periodicity (168 hours), which leads to the overfitted result. Figures 3 and 4 compare the decomposition results with different periodicity choices.

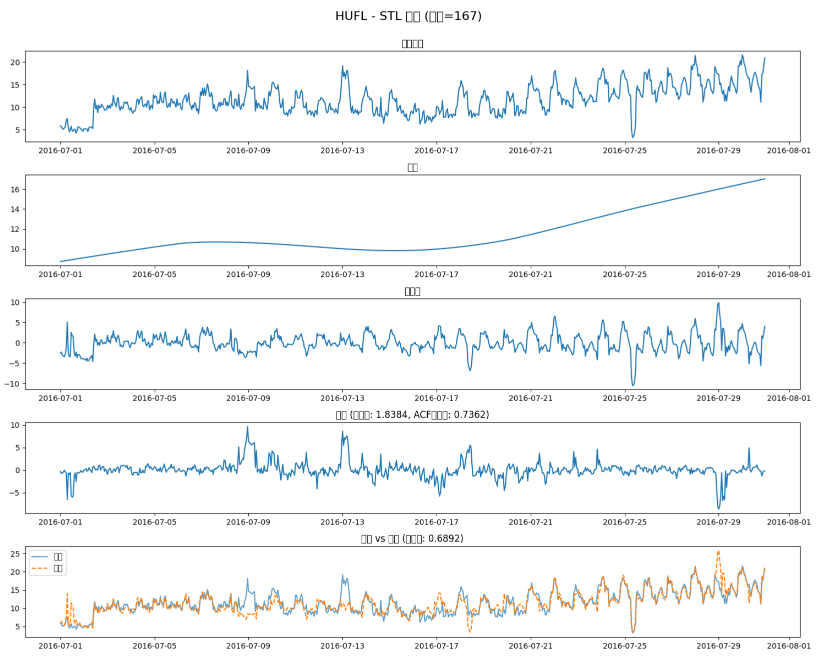

Figure 3: STL decomposition with weekly periodicity (168 hours).

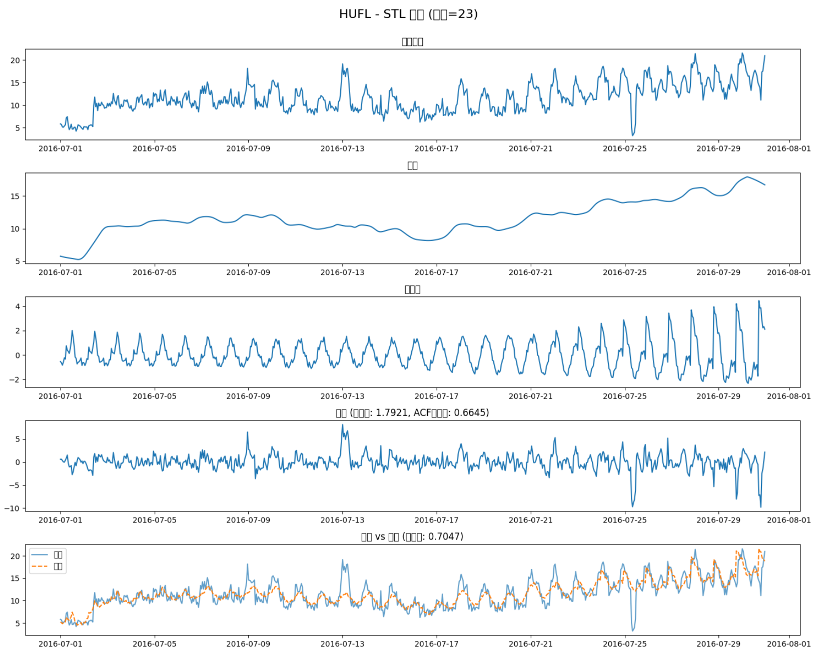

Figure 4: STL decomposition with daily periodicity (24 hours).

We found that a window size of 7 times the period offers a good balance between stability and capturing local periodicity features. For certain features with complex patterns, we implemented a 2-step seasonal decomposition approach, first extracting daily periodicity and then weekly periodicity from the residuals.

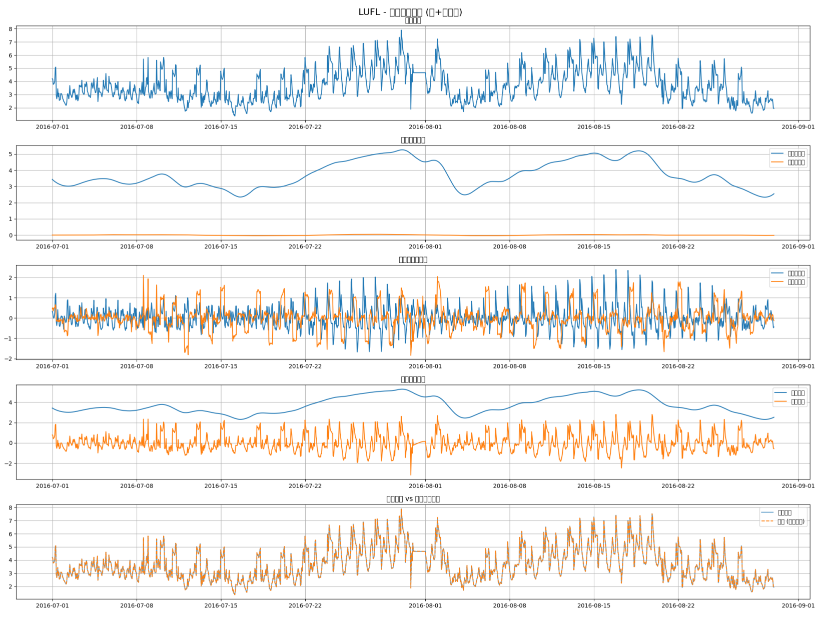
Figure 5: Two-step decomposition for features with complex patterns.

3.3 Trend Smoothing

The drift window method can result in non-smooth trend components, which might lead to overfitting. To address this issue, we run some experiments with different smoothing methods, including moving average, Hodrick-Prescott filter, and Savitzky-Golay filter. We found that the Savitzky-Golay filter provides the best results in terms of preserving the trend while smoothing out noise. Here is the comparison of different smoothing methods.

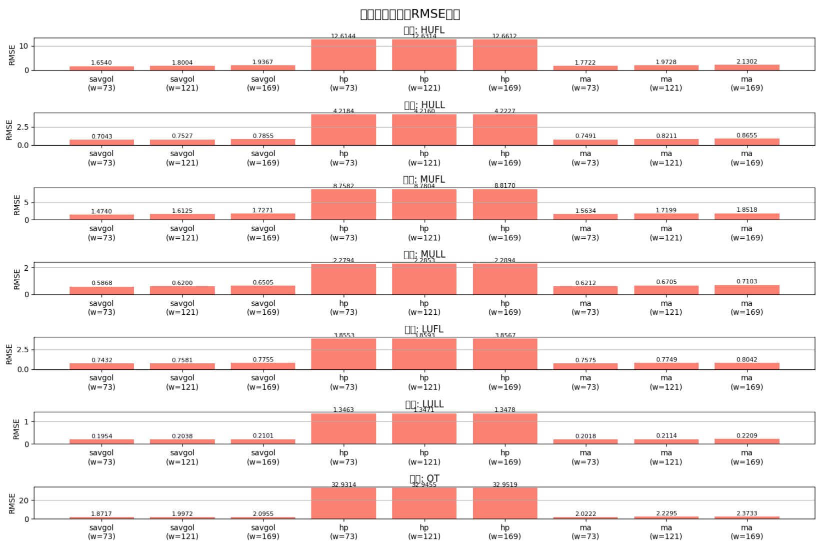

Figure 6: Comparison of different smoothing methods for trend components.

Finally we apply a Savitzky-Golay filter to smooth the extracted trend components, as shown in Figure 7.

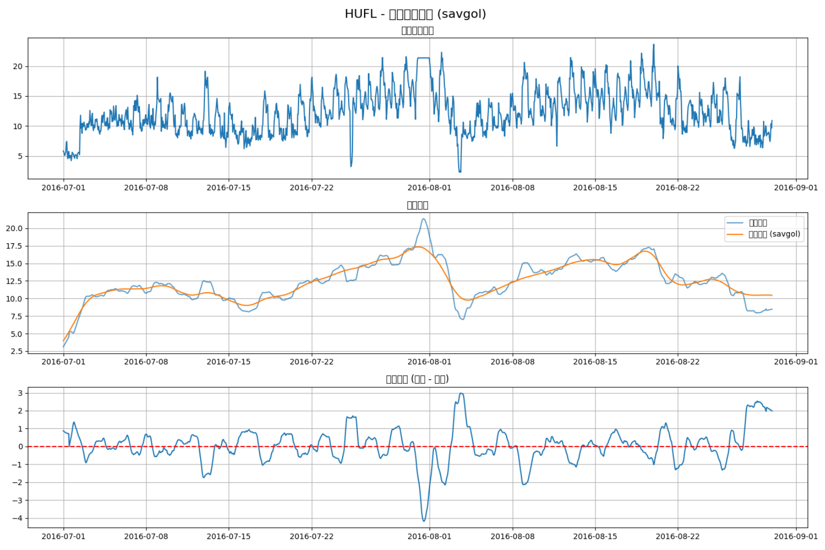

Figure 7: Comparison of original and smoothed trend components using Savitzky-Golay filter.

The smoothed trend prevents the Crossformer from overfitting to noisy trend patterns, leading to better generalization.

3.4 Preventing Future Information Leakage

A key challenge in our approach is preventing future information leakage during evaluation. The drift window method might introduce future information into past predictions if not handled carefully. To address this, we implement a masking mechanism during evaluation that ensures the model only uses past and current information for prediction.

We precompute decomposed features during training for efficiency but generate features on-the-fly during evaluation to maintain causal constraints. Our experiments confirm that the model trained with preprocessed features generalizes well to online feature generation during evaluation, with performance dropping by only 0.002 on average.

3.5 Feature Integration

We combine the original features with the engineered features derived from the decomposition process as well as the time features (hour, day of week, month, which is represented as sine and cosine values) to create a comprehensive feature set. The engineered features include:

- Trend component (smoothed)
- Seasonal component (daily periodicity, some combined with weekly periodicity)
- Time features (hour, day of week, month)

We concatenate these features with the original features to form the final input to the Crossformer model.

3.6 Hybrid Training Strategy

We propose a hybrid training strategy where the model is trained to predict both the original features and the engineered features. This approach guides the model to explicitly learn the trend and seasonality patterns present in the data. The loss function is modified as follows:

$$L = \begin{cases} L_{\text{orig}} & \text{if train\_original\_only} = \text{true} \\ L_{\text{orig}} + \lambda L_{\text{eng}} & \text{if train\_original\_only} = \text{false} \end{cases} \tag{1}$$

where L_{orig} is the loss computed on the original features, L_{eng} is the loss computed on the engineered features, and λ is a weighting parameter.

4 Experimental Setup

4.1 Dataset

We evaluate our approach on the ETTh1 (Electricity Transformer Temperature) dataset, which contains hourly readings of various metrics related to electricity transformers. The dataset exhibits complex seasonal patterns and is challenging for forecasting models due to its non-stationarity and noise.

4.2 Implementation Details

We implement our approach using the Crossformer architecture as the base model. The model is trained with the following hyperparameters:

- Input length: 720 time steps
- Output length: 168 or 720 time steps
- Segment length: 24 time steps
- Window size: 2
- Attention factor: 10
- Model dimension: 256
- Number of heads: 4
- Number of encoder layers: 3

The code for preprocessing and feature engineering is implemented as follows:

```
cd Model
python cross_exp/precompute_causal_features.py \
    --data_path datasets/ETTh1.csv --output_dir ./datasets/
```

For model training and evaluation:

```
zsh scripts/ETTh1-f.sh

python eval_crossformer.py --checkpoint_root ./checkpoints \
    --setting_name Crossformer_ETTh1-f_il720_ol168_sl24_win2_fa10_dm256_nh4_el3_itr0
```

5 Results and Discussion

We conduct experiments with different configurations to evaluate the effectiveness of our approach. Table 1 summarizes the main results.

Table 1: Performance comparison of different model configurations

Configuration	**Input Length**	**Output Length**	**MSE**	**MAE**
train_original_only = false (avg 5 itr)	720	168	0.3882	0.4191
train_original_only = true (avg 5 itr)	720	168	0.4016	0.4358
train_original_only = false	720	720	0.4678	0.4860
train_original_only = true	720	720	0.5391	0.5349

5.1 Impact of Hybrid Training

Our results show that the hybrid training approach (train_original_only = false), where the model is trained to predict both original and engineered features, consistently outperforms the baseline approach (train_original_only = true) across different output lengths. This suggests that explicitly guiding the model to learn trend and seasonality patterns improves its ability to forecast the original time series.

For the short-term forecasting scenario (output length = 168), the hybrid approach achieves a 3.3% reduction in MSE and a 3.8% reduction in MAE compared to the baseline. The improvement is even more significant for long-term forecasting (output length = 720), with a 13.2% reduction in MSE and a 9.1% reduction in MAE.

5.2 Training Stability

We observe that our hybrid approach leads to more stable training outcomes. The variance across different training iterations is significantly lower compared to the baseline model, indicating that the engineered features provide valuable guidance to the model during training.

Table 2 presents the detailed results from 5 independent training runs for the long-term forecasting scenario.

Table 2: Comparison of model stability across 5 independent runs (Output Length = 720)

Model Configuration	**MSE (mean $\pm$ std)**	**MAE (mean $\pm$ std)**
Hybrid Model (with future info)	0.4677 ± 0.0041	0.4847 ± 0.0026
Hybrid Model (no future info)	0.4674 ± 0.0036	0.4865 ± 0.0022
Original Model	0.5641 ± 0.0431	0.5481 ± 0.0291

The standard deviation of MSE for the hybrid model is an order of magnitude lower than that of the original model, highlighting the improved stability of our approach.

5.3 Generalization to Online Feature Generation

A key concern with our approach is whether the model trained with preprocessed features (which might contain some future information due to the drift window method) can generalize to online feature generation during evaluation (where no future information is available). Our experiments show that the performance drop is minimal, with MSE increasing by only 0.002 on average when switching to online feature generation.

This result confirms that our approach is robust and practical for real-world applications where future information is not available at prediction time.

5.4 Ablation Study: Impact of Different Decomposition Periods

We conducted experiments with different periodicity choices for the decomposition. Our results show that daily periodicity (24 hours) leads to more stable and effective features compared to weekly periodicity (168 hours) for the ETTh1 dataset. This is likely because the daily pattern is more consistent and the decomposition is more precise at the hourly level, which is important for high-frequency data.

5.5 Discussion

The superior performance of our hybrid approach can be attributed to several factors:

- **Enhanced feature space**: The engineered features provide explicit information about trend and seasonality patterns, enriching the feature space available to the Crossformer model.
- **Guided learning**: By training the model to predict both original and engineered features, we guide it to learn the underlying temporal dynamics explicitly, which helps prevent overfitting to noise.
- **Regularization effect**: The additional task of predicting engineered features acts as a form of regularization, encouraging the model to learn more generalizable patterns.

Interestingly, we found that simply adding engineered features as inputs without modifying the training objective (train_original_only = true) does not yield optimal results. This suggests that the cross-attention mechanism in Crossformer might be overwhelmed by the additional input features without explicit guidance on how to use them.

6 Conclusion

In this paper, we presented a hybrid approach that enhances the Crossformer model with features derived from traditional time series decomposition techniques. Our approach combines the strengths of classical time series analysis with modern deep learning methods to achieve improved forecasting performance.

Key innovations include the drift window method for local time series decomposition, trend smoothing with Savitzky-Golay filters, and a hybrid training strategy that guides the model to learn both original and engineered features.

Our experimental results on the ETTh1 dataset demonstrate that our approach not only improves forecasting accuracy but also leads to more stable training and better generalization. The hybrid training strategy consistently outperforms the baseline approach, with up to 13.2% reduction in MSE for long-term forecasting.

Future work could explore adaptive weighting schemes for the hybrid loss function, integration of other types of engineered features, and extension to multivariate forecasting tasks with complex interdependencies.

Acknowledgments

We would like to thank the Professor Dark for providing training resource.

References

[1] Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on Loess. Journal of Official Statistics, 6(1), 3-73.

[2] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, 35(12), 11106-11115.

[3] Wu, H., Xu, J., Wang, J., Long, M. (2021). Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. Advances in Neural Information Processing Systems, 34.

[4] Zhang, C., Zhou, C., Ding, Y., Zhang, Q., Xie, X., Ma, R., Yang, H. (2022). Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. arXiv preprint arXiv:2211.10984.

[5] Oreshkin, B.N., Carpov, D., Chapados, N., Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In International Conference on Learning Representations (ICLR).

[6] Liu, Y., Gong, C., Yang, L., Chen, Y. (2020). LSTM-MSNet: Leveraging forecasts on sets of related time series with multiple seasonal patterns. IEEE Transactions on Neural Networks and Learning Systems, 31(4), 1396-1409.