DSA5204 Project Final Report: Can Time Series Imaging Improve Modeling of Temporal Dependencies?

Project Group 8

Yeo Ming Jie, Jonathan Jonathan Lim Zhi Wei Lee Jian Ming Maximillian Tan Zheng Xun

***Abstract*—In this paper, we explore the potential of combining time-series imaging with deep learning architectures to enhance the modeling of temporal dependencies in time-series data. We utilize the Long and Short Term Time Series Network (LSTNet) as our primary deep learning architecture, specifically designed for time series forecasting applications. Furthermore, we investigate the impact of Gramian Angular Fields (GAFs), an innovative method for encoding time series data as images while preserving temporal dependencies, which allows us to harness the power of Convolutional Neural Networks (CNNs) for feature recognition in time-series tasks. We extend this combined approach to time series forecasting, comparing the performance of LSTNet with GAFs against the primary LSTNet model for both 1-step and recursive forecasting methodologies.**

I. Introduction

Time-series data are prevalent in numerous real-world applications, such as speech recognition, natural language processing, and financial forecasting. Recurrent Neural Networks (RNNs) are commonly employed to model time-series data due to their ability to capture the sequential nature and model temporal dependencies. However, RNNs have limitations, such as vanishing gradients and memory constraints, which can hinder modeling long-term dependencies. Time-series imaging has emerged as a novel approach to address these limitations by converting time series data into images and employing Convolutional Neural Networks (CNNs) for feature recognition in time series forecasting.

In this paper, we investigate the effectiveness of time-series imaging in improving the modeling of temporal dependencies for time-series data. We utilize a Deep Neural Network — Long and Short Term Time Series Network (LSTNet) — as our primary architecture [1]. We also explore the impact of Gramian Angular Fields (GAFs) [2], an innovative method for encoding time series data as images while preserving temporal dependencies. This technique allows us to harness the power of CNNs for feature recognition in time-series classification tasks. We will extend this approach to time series forecasting and compare it with our primary LSTN model for both 1-step forecasting and a recursive forecasting methodology.

II. LSTNet

Lai et al. (2017) [1] introduced LSTNet as a novel deep learning framework specifically designed for multivariate time series forecasting. LSTNet was developed to overcome the shortcomings of traditional methods in time series forecasting, such as Autoregressive models and Gaussian processes, which often fail to capture long- and short-term patterns in time series data. By utilizing a combination of CNNs and RNNs, Lai et al. demonstrated the effectiveness of LSTNet in capturing the temporal dependencies in multivariate time series data, achieving significant performance improvements over several state-of-the-art benchmarks.

A. LSTNet Model Architecture

The LSTNet model consists of several key components: a 1D convolutional layer, a recurrent layer (GRU or LSTM), a skip-GRU layer, an optional temporal attention layer, and a final autoregressive layer.

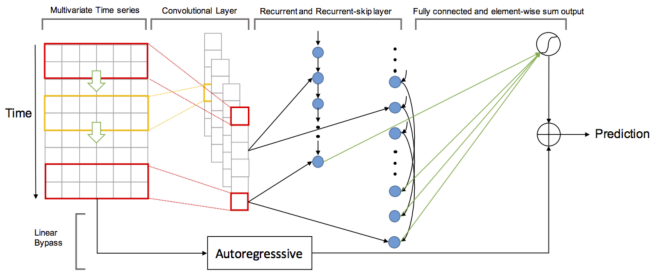

Fig. 1. Model architecture of LSTNet for Multivariate Time Series Forecasting

The 1D convolutional layer is responsible for extracting local patterns and dependencies within a window of observations. This layer helps to reduce the input sequence length and capture short-term patterns in the data. The recurrent layer, typically a GRU or LSTM, is designed to model long-term dependencies across the entire input sequence. The skip-GRU layer is an additional recurrent layer that is connected to the main recurrent layer with skip connections [4], enabling the model to capture patterns at different time scales. The optional temporal attention layer allows the model to selectively focus on different time steps of the input sequence, which can be particularly useful when dealing with noisy or irregularly sampled time series data [3]. Lastly, the final autoregressive layer combines the output of the previous layers and directly

models the dependencies between the target variable and its lagged values to produce the final forecast.

B. Recurrent-Skip GRU layer

The Recurrent-Skip GRU layer is a novel recurrent structure proposed in the LSTNet model to address the challenge of capturing very long-term dependencies in time series data. It does this by introducing skip connections between the current hidden cell and the hidden cells in the same phase in adjacent periods. This is particularly beneficial for datasets with clear periodic patterns, such as electricity consumption and traffic usage, where traditional GRUs and LSTMs may struggle due to gradient vanishing issues.

The Recurrent-Skip GRU layer differs from the usual GRU in its update gate mechanism, which incorporates the hidden cell state from a specific number of hidden cells skipped through, denoted by p. The updating process for the Recurrent-Skip GRU layer can be formulated as:

$$\begin{aligned} r_t &= \sigma\left(x_t W_{xr} + h_{t-\rho} W_{hr} + b_r\right) \\ u_t &= \sigma\left(x_t W_{xu} + h_{t-p} W_{hu} + b_u\right) \\ c_t &= \mathrm{RELU}\left(x_t W_{xc} + r_t \odot \left(h_{t-p} W_{hc}\right) + b_c\right) \\ h_t &= \left(1 - u_t\right) \odot h_{t-\rho} + u_t \odot c_t \end{aligned}$$

In this formulation, the input of the Recurrent-Skip GRU layer is the output of the convolutional layer. The value of p can be easily determined for datasets with clear periodic patterns (e.g., $p = 24$ for hourly electricity consumption and traffic usage datasets) and must be tuned otherwise. Empirically, it has been found that a well-tuned p can considerably boost the model performance even for datasets without clear periodic patterns. Furthermore, the LSTNet could be easily extended to contain variants of the skip length p. Lastly, a dense layer is used to combine the outputs of the Recurrent and Recurrent-skip components.

C. Successes of LSTNet

Since its introduction, LSTNet has been successfully applied in various time series forecasting tasks across different domains. For instance, LSTNet has been employed in electricity load forecasting [9], traffic flow prediction [10], and stock market forecasting [11], demonstrating its versatility and effectiveness in capturing complex temporal dependencies. In these studies, LSTNet has been shown to outperform traditional time series forecasting models, such as ARIMA and exponential smoothing, as well as other deep learning models, including vanilla LSTMs and CNNs. The success of LSTNet in these applications highlights its potential for improving forecasting accuracy and providing valuable insights in a wide range of time series analysis tasks.

III. Gramian Angular Fields

Gramian Angular Fields (GAFs) [2] are a powerful tool for representing time-series data as images, which enables the application of computer vision techniques and deep learning algorithms in time-series analysis. GAFs effectively capture temporal dependencies within time series data by projecting data points into a polar coordinate system, where the data value is encoded as the angular cosine and the radius corresponds to the time index of each point.

A. Encoding Procedure

Given a time-series $X = \{x_1, x_2, \ldots, x_n\}$, the encoding process involves the following procedure:

1) **Rescaling:** We first normalize the time-series to the interval $[-1, 1]$ by applying mean normalization:

$$\tilde{x}_i = \frac{(x_i - max(X)) + (x_i - min(X))}{max(X) - min(X)}. \tag{1}$$

2) **Representing in polar coordinates:** Next, we compute the polar coordinates of the scaled time series by computing angular cosine:

$$\begin{cases} \phi = \arccos\left(\tilde{x}_i\right), & -1 \le \tilde{x}_i \le 1,\ \tilde{x}_i \in \tilde{X} \\ r = \frac{t_i}{N}, & t_i \in \mathbb{N} \end{cases} \tag{2}$$

3) **Calculate Gramian Angular Matrices:** Consider the pairwise trigonometric sum/difference of each point in the time series data to identify temporal correlations within different time intervals.

$$\begin{aligned} GASF &= [\cos(\phi_i + \phi_j)] \\ &= \begin{bmatrix} \cos(\phi_1 + \phi_1) & \cdots & \cos(\phi_1 + \phi_n) \\ \vdots & \ddots & \vdots \\ \cos(\phi_n + \phi_1) & \cdots & \cos(\phi_n + \phi_n) \end{bmatrix}, \end{aligned} \tag{3}$$

$$GADF = [\sin(\phi_i - \phi_j)], \tag{4}$$

where GASF and GADF denote the Gramian Angular Summation and Difference Field respectively. Figure 2 provides a visual intuition on the encoding process.

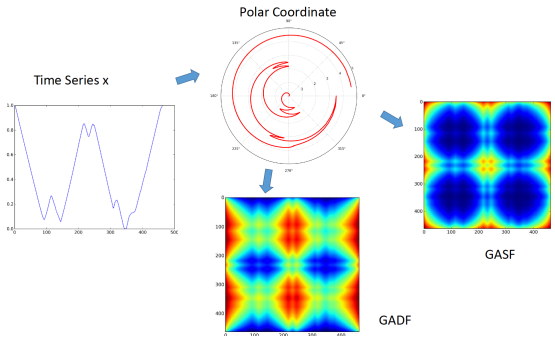

Fig. 2. Illustration of proposed encoding for Gramian Angular Fields

The time dimensionality of the data is encoded into the geometry of the Gram Matrix as the position moves from top-left to bottom-right. Additionally, the transformation is bijective, as the angular cosine is monotonic when $\phi \in [0, \pi]$. This explains the uniqueness of the obtained image and allows

for the reconstruction of the original time series from the main diagonal entries of the GAFs.

B. Successes

In their works, Wang and Oates employed Tiled CNNs on the GAF-transformed images for time series classification. Their proposed framework demonstrated significant performance improvements over state-of-the-art methods, further highlighting the potential of GAFs in combination with advanced deep learning techniques for time series analysis.

C. Related Works

Numerous studies have explored the applications of GAFs in time series classification tasks, which GAFs were originally designed to address [13]. Researchers have successfully combined GAFs with various deep learning architectures, such as CNNs and LSTMs, to effectively classify time series data. These studies have yielded promising results, highlighting the potential of GAFs when used in tandem with deep learning techniques for accurate time series classification.

In recent years, the focus has shifted towards the application of GAFs in time series forecasting tasks [12]. However, in general, there have been few studies of forecasting using deep learning architectures with time series imaging, as it is still an emerging research field. For example, Hong et al. [6] investigated the use of GAFs for day-ahead solar irradiation forecasting. In their work, they integrated GAF transforms with ConvLSTM models, which excel at capturing both spatial and temporal dependencies in time series data. The authors demonstrated that their proposed approach significantly outperformed traditional time series forecasting methods, emphasizing the effectiveness of GAFs in capturing temporal patterns and enhancing prediction accuracy. Such studies have demonstrated the potential of combining time series imaging methods, like GAFs, with advanced deep learning architectures to improve their performance in time series analysis tasks.

IV. Problem Statement

Inspired by the successes of both LSTNet and GAF transforms in diverse applications, our project aims to investigate the possibility of integrating image-based encoding using GAF transforms into the LSTNet model architecture for time-series forecasting. The goal is to simplify the LSTNet model while maintaining or improving its performance. As an exploratory project, our primary objective is to deepen our understanding of GAFs as an image encoding technique and forecasting models such as LSTNet to identify opportunities for enhanced performance and efficiency in time-series analysis. This project falls under the domain of algorithm development.

V. Experimental Set-up

A. Dataset Description

The dataset used in this project is the Electricity Load Diagrams dataset from the UCI Machine Learning Repository [8]. This multivariate time-series dataset contains hourly electricity consumption data, measured in kilowatt-hours (kWh), for 321 clients between 2011 to 2014. However, for the purposes of this project, we focus on the univariate case by sampling the time-series data for one client from the dataset.

B. Exploratory Data Analysis

To gain a deeper understanding of the time-series data, we conducted exploratory data analysis on the selected univariate time-series. Figure 3 presents the autocorrelation and partial autocorrelation plots of the data, which reveal strong seasonality trends, as evidenced by the spikes at multiples of 24 (hours) in both the ACF and PACF plots.

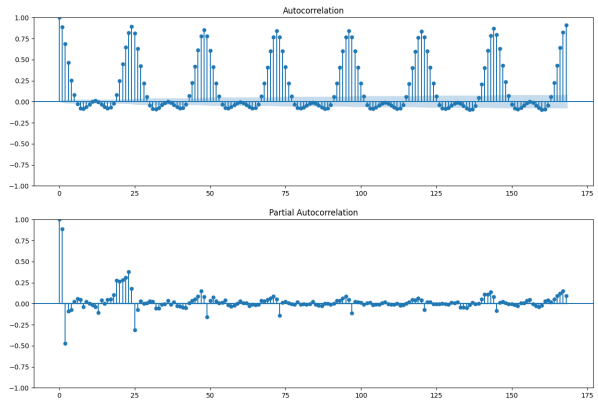

Fig. 3. Autocorrelation and Partial Autocorrelation Plots

We further examined the seasonality trends by decomposing the time series into its trend, seasonality, and residual components using the Seasonal-Trend Decomposition by Loess (STL) method. Figure 4 displays the decomposition plot, which confirms the presence of seasonality in the data while the trend component demonstrates a clear upward trajectory. The residual component appears random, with no discernible patterns.

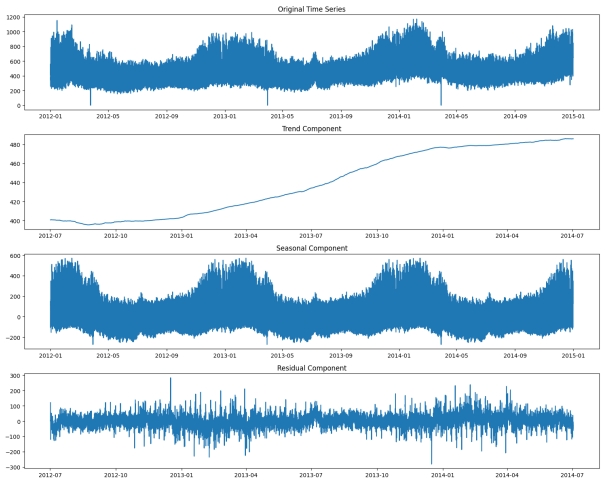

Fig. 4. Time Series Decomposition

The pronounced seasonality and increasing trend observed in the time series data suggest that traditional statistical models

may struggle to effectively model this dataset. Consequently, our project aims to investigate the potential of GAF transforms and LSTNet for time series forecasting within this context.

C. Training Split and Data Pre-processing

We split the dataset into training and validation sets using an 80-20 ratio. To generate our feature-label pairs for the training dataset, we use a windowed approach. We create input features for our model by sliding a window of size 168 (corresponding to 24 hours * 7 days) across the time-series data. This means that the model uses the past week's worth of data to make predictions. The window slides across the dataset with a stride of 1, ensuring that each hour is considered as input for model training. In the subsequent experimental section, we will discuss the extensions to the data pre-processing specific to incorporating GAF transformations.

D. Evaluation Metrics

To evaluate the performance of our models in forecasting, we adopt the standard performance metrics of Mean Squared Error (MSE) and Mean Absolute Error (MAE), given by the following expressions:

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(\hat{y}_t - y_t)^2, \quad \text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|\hat{y}_t - y_t| \qquad (5)$$

Here, $\hat{y}_t$ is the predicted value at time t, y_t is the actual value at time t, and n is the total number of samples. Since the electricity dataset exhibits high variability, we will adopt MAE as our primary metric for training, as it does not penalize large errors as much as MSE does. This choice allows for a more robust evaluation of the model's performance in forecasting the highly stochastic electricity consumption data.

VI. Experiments

A. Baseline LSTNet Model Implementation

We first begin our experiments by implementing the baseline LSTNet model as described in the original LSTNet paper. The model is trained to perform a one-step ahead forecast, which involves predicting the value at the next time step using only the available information up to the current time step. Figure 5 illustrates the actual and predicted one-step ahead forecasts of the time series data using the baseline LSTNet model. Additionally, a zoomed-in view of the last 5% of data points is provided to better visualize the performance of the model.

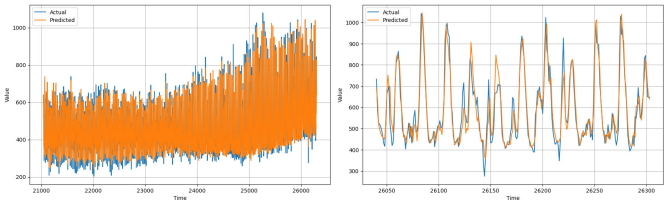

Fig. 5. Forecasting results on Baseline LSTNet model. Blue lines represent the original time series data, while orange lines are the forecast values.

The results appear quite promising on visual inspection, as the model accurately predicts the amplitude and does not exhibit any significant phase lag. To put these results into context, we also compare the performance of the LSTNet model with a naive forecast, where the predicted value for each time step is simply the value observed at the previous time step.

TABLE I
Comparison of Naive Forecast and Baseline LSTNet Model

	Naive Forecast	Baseline LSTNet
MSE	4951.53	1919.96
MAE	51.25	32.69

As shown in Table I, the baseline LSTNet model achieves a significantly lower MSE and MAE compared to the naive forecast on the test set, indicating that the model is effectively capturing the underlying patterns in the data and beating persistence in the one-step ahead forecasting task.

B. LSTNet with GAF Input Data

To test the LSTNet model with GAF transformed time-series input data, we modify the data pre-processing pipeline. We generate feature-label pairs using a sliding-window approach, perform GAF transformation on each window, and normalize the target labels relative to each window of time series data. We also adjust the LSTNet model's architecture to accommodate GAF input data. We modify the Convolutional Component to use a Conv2D layer with MaxPooling for image processing, followed by a reshaping operation to prepare the sequential input for the Recurrent Components of the model.

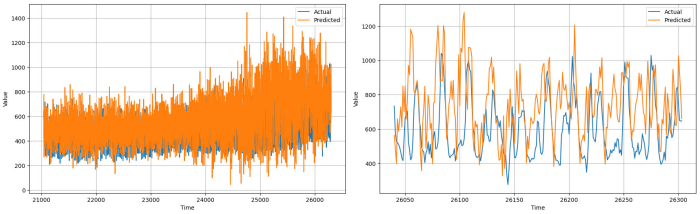

Fig. 6. Forecasting results on GAF transformed inputs and modified LSTNet

Through our experiments, incorporating GAF transformed images into LSTNet did not yield positive results. As shown in Figure 6, the forecast seemed relatively stable in the first half of the validation set; however, the amplitude prediction significantly deviated from the actual results. Furthermore, we also observed a phase lag, with peaks and troughs occurring opposite to the actual data. In fact, we were unable to outperform the persistence model, as the mean absolute error (MAE) was 156.08 for the forecast, while that for persistence was at 51.25.

C. Test with 2DConvLSTM

Considering the unsatisfactory results with the modified LSTNet using 2D Conv and reshaping for the recurrent

components, we hypothesized that there might be a loss of information in the original GAF encoding arising from the reshaping operations of our proposed modified LSTNet framework.

To explore this possibility, we decided to test a different model architecture: 2DConvLSTM. The 2DConvLSTM model combines the convolutional and recurrent layers in a single layer, allowing the model to learn both spatial and temporal features directly from the GAF encoded images. By using 2DConvLSTM, we aimed to preserve the information from GAF encoding more effectively and potentially improve the forecasting performance. This approach was also motivated by the work of Hong et al. [6] as previously discussed in III-C.

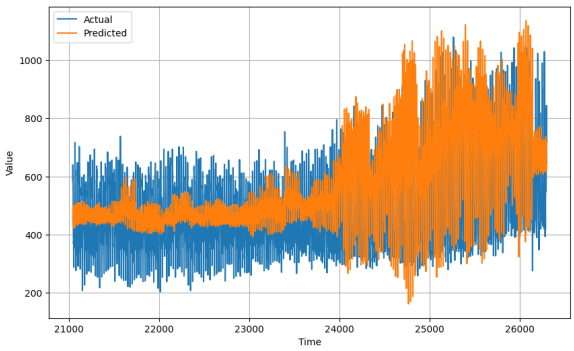

Fig. 7. Forecasting results on GAF inputs with 2DConvLSTM model

Unfortunately, we were still unable to yield any positive results from the experiment, as observed in Figure 7. Moreover, both the model training and evaluation time was exponentially longer. This increase in computation time can be attributed to the more complex nature of 2DConvLSTM layers, which require a higher number of calculations compared to standard LSTM layers, as they integrate both convolutional and recurrent operations.

VII. Extension to LSTNet: Recursive Forecast

The baseline one-step-ahead LSTNet model showed promising results. As an extension, a recursive forecast was conducted using this model (Figure 8).

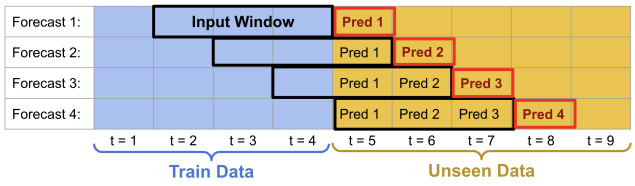

Fig. 8. Recursive forecast using one-step ahead model

A recursive forecast involves using previous forecasts to predict subsequent steps [5]. In this case, the one-step-ahead model was used to recursively predict multiple steps ahead. For the first week of the test set, we observed that the recursive forecast provided a good prediction of amplitude and phase (see Figure 9), with a stable mean absolute error of about 10%. However, the performance declined drastically over a period of 10 weeks (see Figure 10), with amplitude decaying and bias increasing due to accumulated errors.

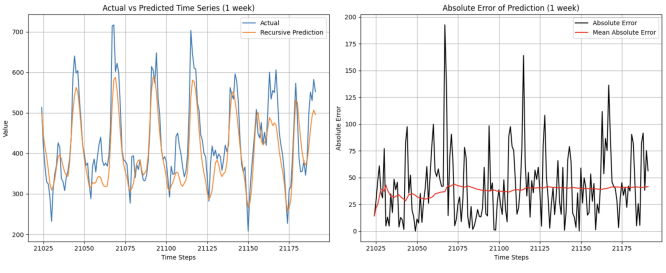

Fig. 9. Recursive results for the first week

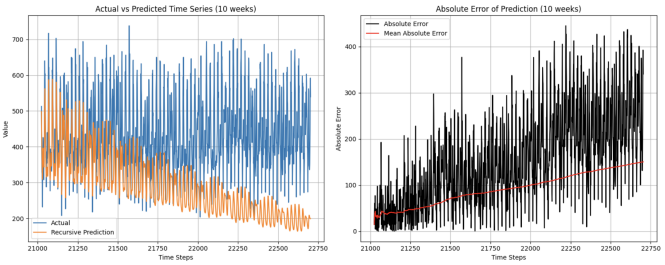

Fig. 10. Recursive results for the first 10 weeks

A. Rectifying the Recursive Forecast

To address this issue, the accumulating errors were modeled and predicted, forming a labeled dataset used to train a second LSTNet model (Figure 11). In the forecasting phase, both the baseline model and the error model were utilized to obtain a baseline prediction and an error prediction, respectively. The rectified prediction for the next time step was obtained by adding both predictions [5].

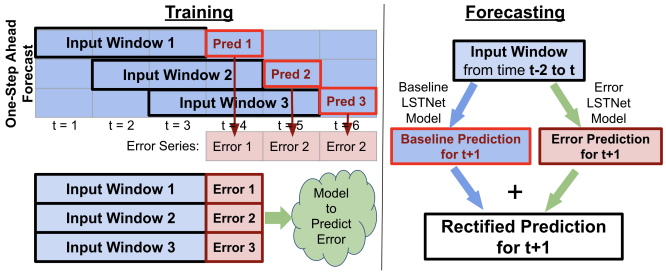

Fig. 11. Methodology for rectifying the recursive forecast

B. Rectified Forecast Results

Modeling the errors helped rectify the issues, resulting in a better amplitude prediction and lower mean average error for the first week (Figure 12). Over 10 weeks, the rectified forecast maintained its amplitude and corrected the bias, with a slower increase in mean absolute error. By employing a second LSTNet model to model the errors, the recursive forecast was rectified, and the mean average error was reduced by half.

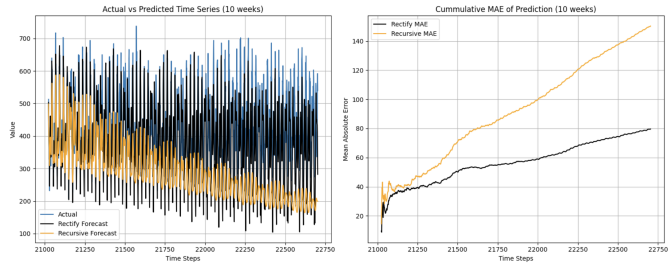

Fig. 12. Rectified forecast results for the first 10 weeks

For the first week, it was shown that the recursive forecast errors tend to occur at the peaks and troughs. This was corrected in the rectified forecast (Figure 13). It is interesting to note that compared to the recursive forecast, the rectified forecast gave a much better prediction of amplitude and phase even on the 10th week (Figure 13).

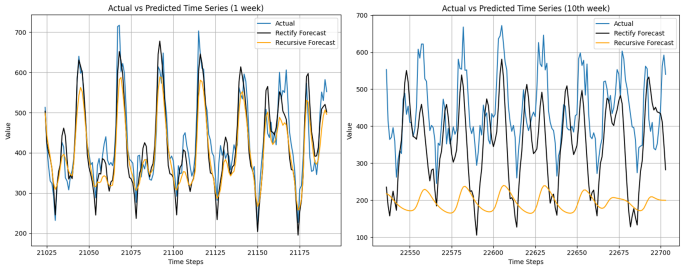

Fig. 13. Rectified forecast results: 1st week (left) and 10th week (right)

VIII. Hyperparameter Tuning

Next, we conducted hyperparameter tuning on the baseline LSTNet Model for the time-series data input and the modified LSTNet model for the GAF data input to improve the test MAE scores.

A. Train-Validation-Test Split and Data Pre-processing

To reduce overfitting, the time-series data was further split into a Train-Validation-Test split using a 60-20-20 ratio, with the validation data taken from the tail end of the training set. Early stopping, using validation loss as the stopping criteria, was implemented during hyperparameter tuning. To allow for more trials, we increased the stride to 10. We investigated the number of trainable parameters for both LSTNet models, which yielded 164,809 and 31.6 million, respectively. Developing an optimized model within a short time frame is therefore challenging.

B. Hyperparameter Tuning Process

We implemented the Keras Tuner for a more effective tuning process. The Hyperband optimizer was utilized, based on the work by Li et al. (2018) [7]. We chose the Hyperband optimizer over Random or Bayesian optimization due to its ability to implement early stopping on trials with unsatisfactory results, allowing for more trials within the same computational time.

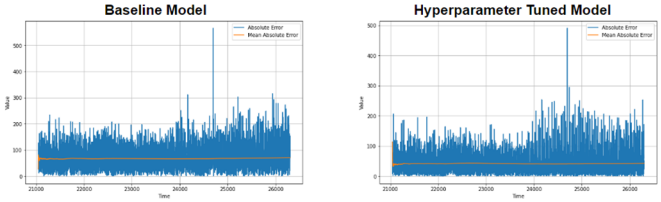

Fig. 14. Comparison of Baseline and Tuned models for time-series input

For both LSTNet models, we tuned the following hyperparameters: 1) number of filters, 2) kernel size, 3) number of GRU units, 4) number of Multihead Attention units, 5) number of skips, 6) number of lags, 7) dropout rate, 8) type of regularizer, 9) regularization factor, and 10) learning rate. We removed the Skip-GRU layer due to its in-built incompatibility. We conducted an ablation study on the baseline model by removing the skip-GRU model, resulting in an increase of test MAE by approximately 0.2.

C. Hyperparameter Tuning Results

We conducted a total of 150 trials for each model. Implementing one-step forecast for the baseline and hyperparameter-tuned model for the time-series data resulted in a significant improvement of test MAE from 62.3 to 42.2 (Figure 14). We also applied the same process to the modified LSTNet model for the GAF data input, which improved the test MAE. However, the forecast for the tuned modified LSTNet model was poor, similar to the observations in the GAF cases mentioned earlier.

IX. Conclusion

In this project, we have investigated the potential of combining time series imaging techniques, specifically GAFs, with deep learning architectures for time series forecasting tasks. We have implemented LSTNet and successfully deployed it on a univariate dataset for forecasting purposes. While we attempted to modify the LSTNet architecture to incorporate GAF transformed images, our experiments did not yield positive results. Motivated by the work of Hong et al., we explored the use of GAF inputs with 2DConvLSTM as an alternative approach. However, due to the limited time frame, we were unable to replicate their results. Furthermore, we proposed an extension to the LSTNet model by developing a recursive forecast framework that combines baseline prediction with an error prediction to form a rectified prediction. This approach demonstrated promising results in our experiments.

References

[1] Lai, G., Chang, W., Yang, Y. & Liu, H. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. (arXiv,2017), https://arxiv.org/abs/1703.07015

[2] Wang, Z. & Oates, T. Imaging Time-Series to Improve Classification and Imputation. CoRR. abs/1506.00327 (2015), http://arxiv.org/abs/1506.00327

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. & Polosukhin, I. Attention is All you Need. Advances In Neural Information Processing Systems. 30 (2017), https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[4] Campos, V., Jou, B., Gir´o-Nieto, X., Torres, J. & Chang, S. Skip RNN: Learning to Skip State Updates in Recurrent Neural Networks. International Conference On Learning Representations. (2018), https://openreview.net/forum?id=HkwVAXyCW

[5] Taieb, Souhaib Ben; Hyndman, Rob J (2012): Recursive and direct multi-step forecasting: the best of both worlds. Monash University. Journal contribution.

[6] Hong, Y., Martinez, J. & Fajardo, A. Day-Ahead Solar Irradiation Forecasting Utilizing Gramian Angular Field and Convolutional Long Short-Term Memory. IEEE Access. 8 pp. 18741-18753 (2020)

[7] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," Journal of Machine Learning Research, vol. 18, no. 185, pp. 1–52, 2018.

[8] Artur T. (2015). UCI Machine Learning Repository. Retrieved from https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[9] Chen, L., Wang, Y., & Wang, Y. (2019). Multi-step-ahead electricity price forecasting using a hybrid model based on two-layer decomposition technique and BPNN optimized by fruit fly optimization algorithm. *Applied Energy*, 235, 978-988.

[10] Zhao, Y., Wang, Q., & Wang, D. (2020). LSTNet: Deep Learning Model for Traffic Flow Prediction Considering Spatial-Temporal Correlation. *IEEE Access*, 8, 84044-84053.

[11] Zhang, Y., & Zhao, Q. (2019). Stock market forecasting using a deep learning model based on 1D convolutional and LSTM units. *Neural Computing and Applications*, 31(12), 8719-8728.

[12] Bi, J., Li, H. & Fan, Z. Tourism demand forecasting with time series imaging: A deep learning model. *Annals Of Tourism Research*. **90** pp. 103255 (2021), https://www.sciencedirect.com/science/article/pii/S016073832100133X

[13] Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P. A. (2019). Deep learning for time series classification: a review. Data Mining and Knowledge Discovery, 33(4), 917-963.