university of groningen / faculty of science and engineering / bernoulli institute

Forecasting Carbon Intensity and Solar Generation in the Building Sector

Bachelor's Project Thesis

Matej Priesol, s4340094, m.priesol@student.rug.nl
Supervisors: J.D. Cardenas Cartagena & R.F. Cunha

Abstract: The building sector is one of the biggest contributors to climate change and greenhouse gas emissions. However, there are numerous ways in which it can provide benefits to the electrical grid, such as using solar photovoltaic generation for self-sufficient energy production. This study investigates the effectiveness of different forecasting neural network models in order to optimize carbon emissions and solar energy generation. Three different multi-input multi-output models are considered: Multilayer Perceptron, Long Short-Term Memory, and Gated Recurrent Unit. Each model is evaluated using four different criteria: RMSE score for carbon intensity forecasts, RMSE score for solar generation forecasts, training time, and prediction time. The results indicate minimal differences between the models, but the Gated Recurrent Unit achieves slightly lower RMSE scores. However, Multilayer Perceptron is less complex and the training time is a lot faster, with only a small increase in RMSE scores. Therefore, both models are viable options for forecasting carbon intensity and solar generation.

1 Introduction

1.1 Motivation

Climate change and greenhouse gas emissions are common and important topics of discussion, with the building sector being a major contributor. Despite a 30% reduction in building-related emissions in Europe since 2005, they still accounted for 35% of energy-related emissions in 2021 (EEA, 2023).

This decrease was mainly caused by various improvements such as better insulation or decarbonized heating and cooling systems. However, while these measures reduce the emissions directly caused by fossil fuel use, they often result in increased electricity consumption. Therefore, unless the energy is generated from renewable sources, the emissions simply shift from the building sector to the electricity sector.

Solar photovoltaic (PV) generation is an increasingly popular method of reducing electricity loads by providing a self-sufficient generation during sunny days and storing the energy for other days. In their analysis of the environmental impact of PV power systems, Bošnjaković, Santa, Crnac, and Bošnjaković (2023) showed that the emissions from PV systems are significantly lower than emissions from fossil fuel power plants. It is worth noting that most of the CO_2 emissions come from the production phase of PV systems. Depending on the country, the energy payback in Europe is between 1 and 2.5 years, which is minimal compared to the average 30-year lifespan of these systems.

To further highlight the importance of the topic of climate change and carbon emissions, we might consider the 2030 Agenda for Sustainable Development, created and adopted by all United Nations Member States in 2015 (UN, 2015). It provides a blueprint for peace and prosperity for people and the planet. The core of the agenda are 17 Sustainable Development Goals, three of which are directly related to climate change and sustainable energy: Affordable and Clean Energy, Sustainable Cities and Communities, and Climate Action.

The importance of minimizing CO_2 emissions can't be underestimated, and the gradual shift toward renewable energy is one of the means to achieve that. However, these systems need to be carefully managed to ensure high efficiency. Therefore, understanding the factors that influence both carbon emissions and solar generation is essential. Furthermore, the ability to predict future measurements can lead to fully optimizing these systems, as being able to estimate carbon emissions or solar generation would help with correctly managing them.

1.2 State of Art

Forecasting carbon dioxide emissions is a relatively well-studied topic, both in general and within the

The code of this project can be found in the repository: `https://github.com/MatejPriesol/Forecasting-Carbon-Intensity-and-Solar-Generation`

building sector. Oladokun and Odesola (2015) compared various econometric, building physics, and statistical methods for modeling household energy consumption and carbon emissions. They noted that these methods often struggle with capturing the complex interdependencies and dynamic nature of energy consumption and carbon emissions. Therefore, the authors suggest using more robust modeling approaches.

One such approach is the use of Neural Networks (NN). According to Swan and Ugursal (2009), NN methods are not commonly used to model energy consumption due to their high computational and data requirements. However, some studies did successfully develop NN models. For instance, Aydinalp, Ismet Ugursal, and Fung (2002) accurately modeled and predicted appliance, lighting, and space-cooling energy consumptions using NNs. They found these models were also capable of accurate predictions in households with unusually high or low energy consumption. In a different study, Mihalakakou, Santamouris, and Tsangrassoulis (2002) effectively used NNs to estimate the energy consumption of residential buildings in Athens.

Conversely, deep learning approaches, including NNs, are a lot more common and successful in solar generation forecasting. A review of different forecasting methods for solar PV generation showed that these methods outperform more traditional approaches (Sobri, Koohi-Kamali, and Rahim, 2018). Different Artificial Intelligence (AI) methods were considered, such as NNs or Support Vector Machines. Among more traditional approaches, sky imagery or satellite imaging were analyzed. The research showed that the AI models outperform the traditional approaches, by reducing the prediction error.

While previous research shows various techniques for forecasting both carbon emissions in the housing sector as well as solar generation using PV generation, they are often treated separately. However, to properly study the effect of PV solar generation on carbon emission and climate change, a unified approach is necessary to better understand the relationship between them.

This research aims to bridge the gap between these two closely related topics by exploring the forecasting techniques for both carbon emissions and solar generation. This study is inspired by an AIcrowd competition NeurIPS 2023 CityLearn Challenge (AIcrowd, 2023). For the competition, a dataset is provided with the goal of carbon emission and solar generation forecasting. Since the competition had already finished, it is possible to compare different modeling techniques utilized by the winning participants.

Three different models were observed: Multilayer Perceptron (MLP), linear regression, and autoregressive models (NeurIPS, 2023). Interestingly, Recurrent Neural Networks (RNN) like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) were not used in any of the best-performing models, despite being more suited for processing sequential data (Goodfellow, Bengio, and Courville, 2016). Additionally, these models were built and trained separately for solar generation and carbon emissions.

In this study, an MLP model will be trained and evaluated as a baseline model since it outperformed the rest in the competition. Additionally, LSTM and GRU models will be trained and compared against the baseline. Each model will be trained to forecast both solar generation and carbon intensity at the same time to capture the relationship between the data and its' effect on both variables. All models will be compared with each other to determine which technique is best suited for carbon emission and solar generation forecasting.

2 Theoretical Framework

As mentioned previously, simple statistical modeling and traditional methods have struggled to capture the more complex and dynamic nature of energy consumption modeling, and more robust approaches might be needed (Oladokun and Odesola, 2015). One such approach is deep learning. Deep learning models try to mimic the human brain through the use of interconnected artificial neurons organized in layers. These layers process the data and enable the models to learn patterns within the data. The typical structure includes an input layer, hidden layers, and lastly an output layer (Goodfellow et al., 2016).

2.1 MLP

The first implemented deep learning model is an MLP, which is a type of feedforward neural network, meaning that the information flows only in one direction. In particular, starting with an input layer, through one or more hidden layers, to the output layer. There are no feedback connections, therefore, the network does not retain any information about the previous inputs (Goodfellow et al., 2016). A simple diagram for an MLP architecture is shown in Figure 2.1.

During the training process, input data is fed into the network, and propagated through the layers. Each neuron receives an input from a previous layer, computes the weighted sum of these inputs plus a bias, and the result is passed through an activation function to the next layer. Once an output is reached, the loss is calculated. Through the process of backpropagation, the weights and biases are adjusted using gradient descent. This

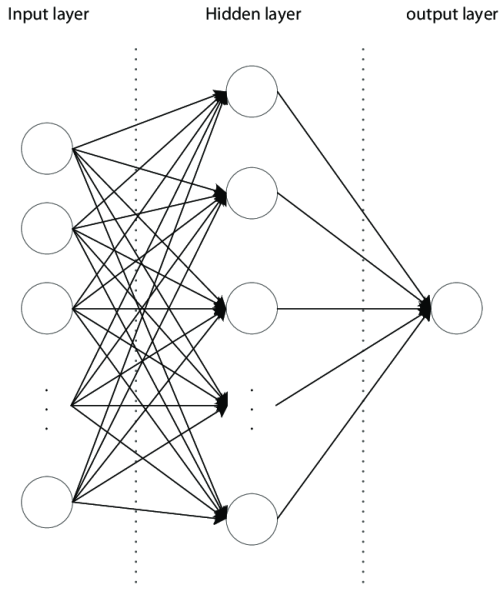

Figure 2.1: Example structure of an MLP consisting of an input layer, a single hidden layer, and an output layer. The figure was taken from Zhao et al. (2015).

iterative process allows the model to learn patterns in the data.

There are two possible issues with MLPs in the context of time-series forecasting. First, MLPs can suffer from a vanishing gradient problem. As gradients of the loss function propagate backward through the network, they can become extremely small due to the use of sigmoid or hyperbolic activation functions, effectively vanishing in deep neural networks. Second, the lack of recurrent connections makes the MLPs less suited for sequential data, as the model cannot learn long-term dependencies between the data points. Despite these possible issues, the simple architecture of MLPs allows for fast training times and makes the model less complex compared to RNNs.

2.2 LSTM

To address the vanishing gradient problem and allow the model to learn long-term dependencies between the data, RNNs are considered. A diagram for an RNN architecture is visible in Figure 2.2.

The first RNN implemented is the LSTM. The biggest difference between an MLP and an LSTM is that LSTMs contain memory cells. These memory cells allow LSTMs to maintain information over long sequences of data. The internal state of a memory cell can be updated over time, enabling the network to retain past information and carry it forward (Goodfellow et al., 2016).

LSTMs use a gating mechanism to control the flow of information into and out of the memory cells (Figure 2.3). There are three gates responsible for the flow of information: the input gate, the forget gate, and the output gate. The input gate is responsible for updating the cell state, it regulates the flow of information into the memory cell. The forget gate controls whether information in the memory cell is retained or discarded. Lastly, the output gate determines what information is passed on to the next time step.

2.3 GRU

The final type of NN considered in the project is a GRU. Like LSTMs, GRUs are a type of RNN that can learn and maintain long-term dependencies in the data. However, the gating system of a memory cell is different from an LSTM. Rather than three different gates, it only contains two: the update gate and the reset gate. The update gate controls the amount of old information that is retained, combining the functions of input and forget gates in an LSTM. The reset gate then controls how much of the past information is forgotten (Goodfellow et al., 2016).

GRUs are considered to be less complex than LSTMs due to the simpler architecture consisting of only two gates and no separate cell state. Despite this simpler architecture, GRUs are capable of capturing the long-term dependencies of the sequential data. Depending on the dataset, they can sometimes outperform LSTMs with faster training times and lower complexity.

3 Methods

3.1 Task

To determine the most appropriate model for forecasting carbon intensity and solar generation, three different models are considered: MLP, LSTM, and GRU. Due to the potential limitations of MLP when handling sequential data, the MLP is considered as the baseline model against which the LSTM and GRU are compared.

All models are trained on the same dataset and evaluated using the same evaluation data. The performance of each model is assessed using the following 4 metrics: the RMSE score of carbon intensity forecast, the RMSE score of solar generation forecast, the training time, and lastly the prediction time. The formula for calculating the RMSE score can be seen in Equation 3.1,

$$RMSE(y, \hat{y}) = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \qquad (3.1)$$

where n is the number of data points, $\hat{y}$ is the predicted observation, and y is the actual observation.

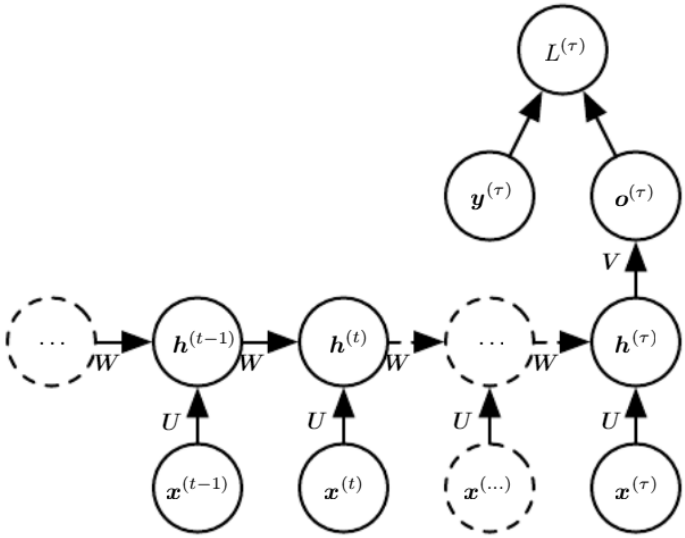

Figure 2.2: And RNN architecture with a single output at the end of the sequence. At each timestep t, x^t represents the input, h^t is the hidden layer activation, o^t represents the output, y^t is the target, and the loss is L^t. U, V, and W, are the weight matrices for input-to-hidden, hidden-to-output. and hidden-to-hidden connections, respectively. The figure was taken from Goodfellow et al. (2016).

The RMSE score is an important indicator of the model's accuracy since it measures the average distance between the predicted and actual values. When dealing with large deep-learning models, the training and prediction times are important indicators of the model complexity, and therefore, are also considered.

All models are using the same window approach for the forecast. A period of 48 hours is used as an input to the model, while the next 24 hours are used as an output.

3.2 Data

The data for this project was taken from the AICrowd CityLearn Challenge 2023 (AIcrowd, 2023).

The available data comprise multiple datasets, each consisting of 720 rows representing a measurement from a specific hour. In total, there are 30 days of data, spanning the whole month of June.

The first three datasets provide information about three different buildings in the same neighborhood. The building data consists of 15 different features such as time of the day, indoor temperature and humidity, and solar generation among other things.

Additionally, a dataset containing weather measurements from the corresponding area is provided, comprising 16 different features, including in-depth information about humidity, temperature, or solar radiation.

Lastly, datasets providing information about electricity pricing and carbon intensity are also included.

3.3 Data Preprocessing

3.3.1 Dependent variables

The models forecast two separate features: Carbon Intensity (CI) and Solar Generation (SG). CI is already provided in one of the datasets and the SG is included for each building separately. Therefore, a new feature is created where the mean SG over the three buildings is calculated, and the separate measurements are removed.

3.3.2 Data cleaning

The data was explored for potential outliers and the necessary steps for data cleaning. The dataset had no missing or null values, and all data points were in consecutive order without any hours skipped. While some variables showed minor fluctuations, there weren't any unreasonable values to suggest potential recording errors. Therefore, all values were kept. Features with constant values were removed, and all datasets were combined into a single one.

3.3.3 Training, Validation, and Evaluation

Because no separate evaluation data was provided in the competition, the evaluation dataset was created from the existing data. Additionally, a validation dataset was also created to evaluate different model architectures.

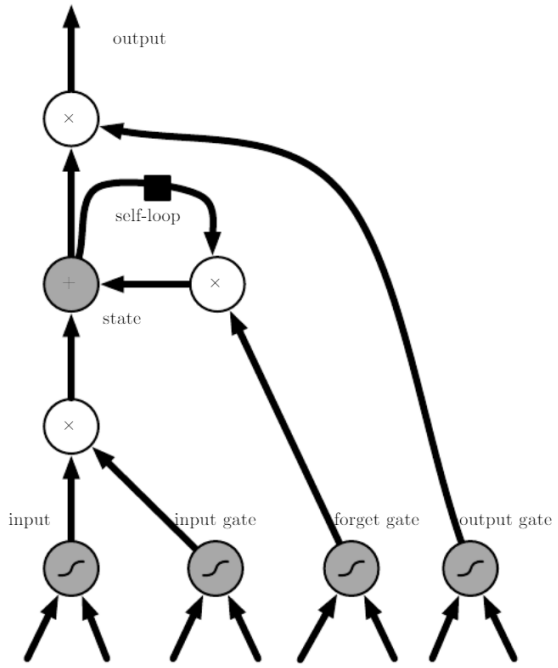

Figure 2.3: Block diagram of the LSTM gating mechanism. An input feature is computed with an artificial neuron unit. The state unit has a self-loop which is controlled by the forget gate, and the black square represents a delay of one time step. The figure was taken from Goodfellow et al. (2016).

Considering the whole dataset consists of only 720 data points, it was important to maximize the number of data points in each set. As previously mentioned, 48 hours are used to forecast the following 24 hours. This allows for an overlap between the sets without introducing a bias. The last 24 data points from the training set were also included in the validation set, serving only as an output in the training set, and only as an input in the validation set. The same procedure was repeated for the last 24 data points of the validation set, which were added to the evaluation set.

As a result, three datasets were obtained. The training set consisted of 576 data points and was used to train the model. The validation set consisted of 96 data points and was used to evaluate different model architectures to determine the best-performing one. Lastly, the evaluation set, also consisting of 96 data points, was used to assess the most accurate model architecture.

3.3.4 Input-Output batches

The final step in the training, validation, and evaluation split, was to organize the data into input-output batches that could be used during the separate phases. Each batch consisted of 48 input data points and the immediately following 24 output data points. This resulted in 505 batches for training, 25 batches for validation, and 25 batches for evaluation.

There were two main reasons for creating these batches. First, it ensured the correct input-output pairs. Since the models use a 48-hour window to forecast the following 24 hours, each batch had to follow the specified shape. Second, it transformed the data into a correct shape accepted by all three models.

3.3.5 Data Normalization

Some features had largely varying scales as visible in Figure 3.1. The features in the plot were selected to highlight this trend, but many other features showed similar or even bigger ranges. Therefore, it was important to normalize the data to ensure a similar scale for each feature. Otherwise, certain features could dominate the learning process.

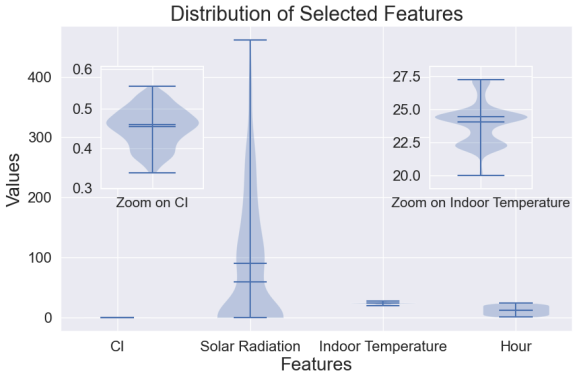

Figure 3.1: Distribution of selected variables. Because the range of CI and Indoor Temperature is very small, the figure also contains zoom on these parts to better visualize the differences.

The data was normalized using the min-max normalization approach. The formula for min-max scaler can be seen in Equation 3.3,

$$v_i' = \frac{v_i - min_{train}(v)}{max_{train}(v) - min_{train}(v)} \tag{3.2}$$

where v_i' is the normalized value, v_i is the original value, $min_{train}(v)$ is the minimum value from the training set for a specific feature v, and $max_{train}(v)$ is the maximum value from the training set for a specific feature v.

To prevent information leakage between the training phase and the validation or evaluation phase, it is important to highlight that the minimum and maximum values for each feature were taken from the training set. The normalization process was then applied to all three sets using the same minimum and maximum measurements.

3.3.6 Feature Selection

To select the features used for the model, the absolute value of the Pearson Correlation Coefficient was used,

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{3.3}$$

where r is the correlation coefficient, x and y are two specific variables from the dataset that are being correlated, x_i is the value of x-variable from the sample, $\overline{x}$ is the mean of the values of the x-variable, y_i is the value of y-variable from the sample, and $\overline{y}$ is the mean of the values of the y-variable.

All variables were correlated with CI and included in case the absolute value was > 0.4. This threshold was determined through empirical testing on the validation set. A correlation heatmap displaying the 10 features most correlated with CI is shown in Figure 3.2.

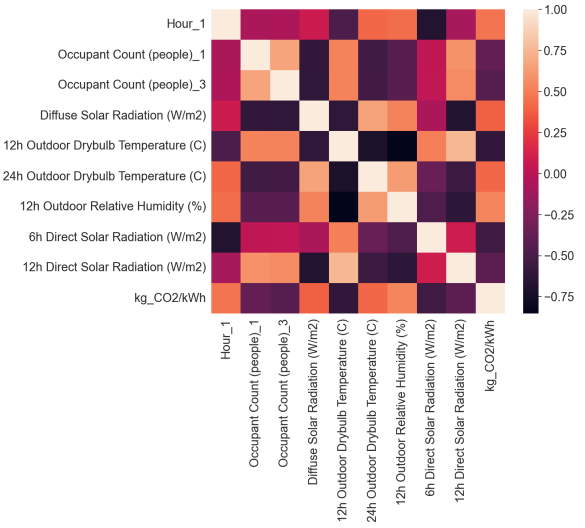

Figure 3.2: Heatmap of Pearson correlation for the 10 most correlated variables with CI

3.4 Models

3.4.1 Model Architecture

All three networks in the project are multi-input, multi-output models. Multiple features are used as input for each network, and two different dependent variables are forecasted. Therefore, a multi-input, multi-output model architecture is appropriate to use. For all three models, there is a shared input layer, as well as other optional shared layers, such as preprocessing layers for the MLP. Each model has two different heads with separate hidden layers, each head corresponding to one of the dependent variables. There are also two separate output layers. A general multi-input, multi-output model architecture is visible in Figure 3.3.

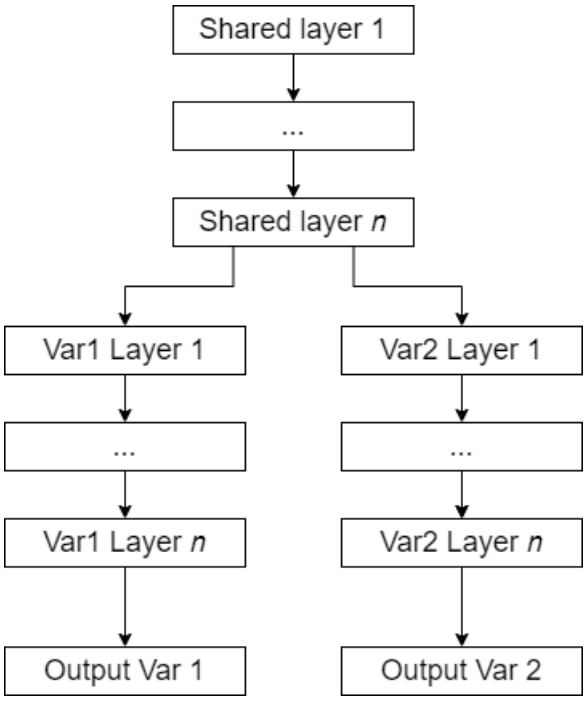

Figure 3.3: Generalised multi-input, multi-output model architecture with separate heads for each output variable

During the training phase, cross-validation was used to help prevent over-fitting of the model. Given the nature of a time-series forecast, a cross-validation on a rolling basis was used. Therefore, the data used for the validation in each fold always follows the data used for training. Additionally, to ensure similarity to the output shape, only 24 data points are used as the validation data in each fold.

Early stopping was used to minimize overfitting. In case the validation error starts increasing, the model is possibly overfitting on the training data, and therefore, the current fold in the training process terminates. Because the model has two separate heads leading to two separate forecasts, two different validation loss values are calculated in each epoch: one for CI, and one for SG. The sum of these two values is then considered for early stopping.

3.4.2 Model Architecture Evaluation

The best model architecture was chosen through empirical testing based on the model performance on the validation data. This was achieved through a grid search for different hyperparameters. The following hyperparameters were considered and tuned:

1. Number of hidden layers: 3, 4, and 5
2. Number of neurons: 192, 96, 48, or 24
3. Number of folds for cross-validation: 3, 4, or 5

4. The patience parameter: 4, 5, 6, 7, or 8
5. Number of epochs: 20, 50, 100
6. Activation functions: ReLU, Tanh, or Leaky ReLu. ReLU scales the output to the range $[0, \infty)$ and Tanh to the range of $(-1, 1)$. Leaky ReLU is also considered since it may help with diminishing the vanishing gradient problem (Yilmaz and Poli, 2022). It scales the output to the range $(-\infty, \infty)$.

All the different models were compared based on the RMSE score for both CI and SG on the validation data. As previously mentioned, there were 25 forecasts in total for each model architecture. Therefore, the average RMSE score over all 25 forecasts was calculated,

$$Avg\_RMSE = \frac{\sum_{i=1}^{n=25} RMSE_i}{n} \tag{3.4}$$

where $RMSE_i$ is the RMSE score for i-th prediction calculated using the formula in Equation 3.1, and n is the number of forecasts, in this case, 25.

The average RMSE was calculated for CI and SG and then summed. The model architecture with the lowest combined RMSE value was then selected as the final model. After the model architectures were determined, all models were evaluated using the evaluation data. Similar to the validation process, the average RMSE score for both CI and SG was calculated. Additionally, the total training time, as well as prediction time (the time it takes to predict a single forecast) were considered.

4 Results

4.1 Model architecture selection

To determine the best-performing model, different architectures were evaluated using the RMSE score metric for both CI and SG. Different hyperparameter combinations and the whole process are described in Section 3. Most hyperparameters were shared between all three models, in particular, all models had 3 hidden layers, 4 folds were used for cross-validation, the patience parameter was set to 4, the number of epochs was set to 50, and the activation function was Leaky ReLU. The models differed in the number of neurons in each layer. The MLP performed best with the number of neurons set to 192, 96, and 48 in the 3 hidden layers, respectively, while LSTM and GRU had the number of neurons set to 48, 48, and 24. The results for all three models are shown in Table 4.1.

After determining the best-performing architectures, the models were run on the evaluation data. In addition to RMSE scores for CI and SG, the training time and average prediction time were also considered. The results are shown in Figure 4.2.

RMSE SCORE	MLP	LSTM	GRU
Carbon Intensity	0.1067	0.1586	0.1124
Solar Generation	0.056	0.064	0.042

Table 4.1: RMSE scores for the best-performing models ran on the validation data

	MLP	LSTM	GRU
Carbon Intensity	0.1389	0.131	0.1265
Solar Generation	0.057	0.0605	0.0405
Training time (s)	13.662	97.786	78.208
Prediction time (s)	0.103	0.121	0.076

Table 4.2: RMSE scores, training time, and prediction time of the best-performing models ran on the evaluation data

4.2 Qualitative Results

As previously mentioned, there were 25 forecasts in total, each shifted by one hour. 3 example predictions over the whole range are plotted for visualization purposes, one from the beginning, one from the middle, and one from the end. Example CI forecasts are visible in Figure 4.1, and example SG forecasts are visible in Figure 4.2. Additional predictions for both variables can be found in the Appendix (A).

By looking at the visual results, it is hard to spot any significant difference between the models. The patterns for models are relatively similar to each other. There are some small variations between the models, but none of them look a lot worse compared to the others. The models seem to forecast the SG a lot more accurately than the CI.

For CI, all models seem to learn approximately the same pattern. For the first half of predictions, this pattern aligns well with the actual data. However, for the second half of predictions, all models are a bit off compared to the actual data. While the general pattern is maintained, the predictions are shifted back by a few hours.

Regarding SG, there doesn't seem to be any drop in performance for later predictions. The general pattern is maintained, as well as the location of the spikes in generation. All models approximate the actual data very well, to the point where it might be difficult to distinguish one from another.

4.3 Quantitative results

Statistical tests were performed to determine whether there was a difference between the RMSE scores of each model across the 25 forecasts. However, it is important to note that while the RMSE score is a significant indicator of model quality, it is not the only determinant. Both dependent variables, as well as training and prediction times, have to be taken into account. Additionally, model

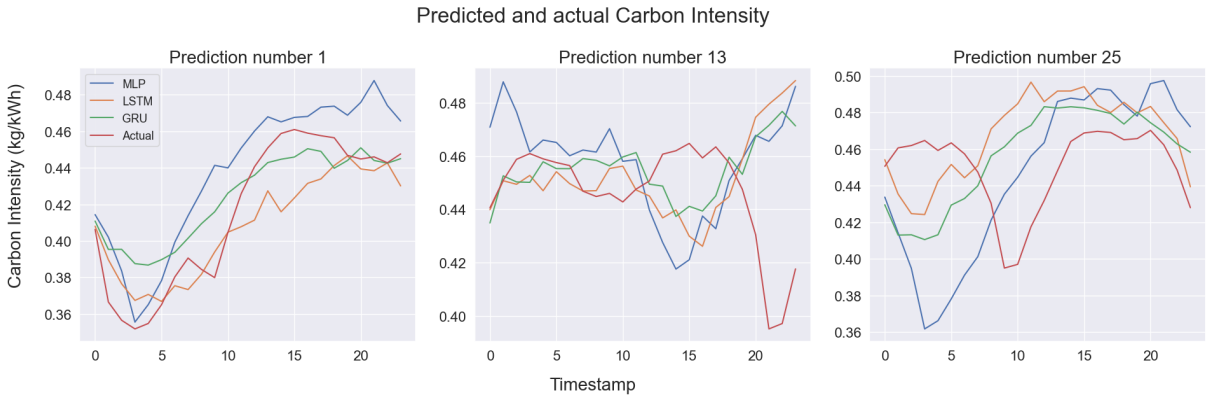

Figure 4.1: Forecast for the CI over the 25 predictions

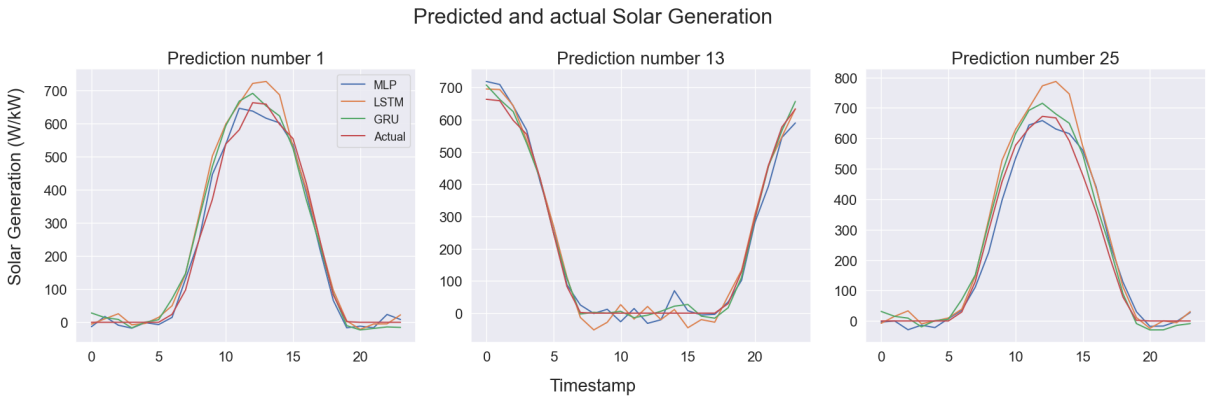

Figure 4.2: Forecast for the SG over the 25 predictions

complexity should also be considered. The purpose of this quantitative analysis is to gain a better understanding of the model performance in terms of RMSE scores; considering the scores are very similar to each other as visible in Table 4.2, it can be difficult to visualize and understand the difference.

To determine the appropriate statistical method, the Shapiro-Wilk normality test was conducted. The results indicated that RMSE scores for CI might not follow a normal distribution, with similar findings for the MLP model for SG. Both LSTM and GRU showed no evidence of non-normality. Table 4.3 summarizes the Shapiro-Wilk test results. Therefore, a non-parametric Wilcoxon signed-rank test was used to compare the RMSE scores for CI, while a parametric two-sided paired t-test was used to compare the RMSE scores for SG.

For CI, a Wilcoxon signed-rank test was applied to all three combinations of models: MLP-LSTM, MLP-GRU, and LSTM-GRU. The results are visible in Table 4.4.

For SG, a two-sided paired t-test was applied to all three combinations of models: MLP-LSTM, MLP-GRU, and LSTM-GRU. The results are visible in Table 4.5.

The RMSE score analysis for CI suggests that both LSTM and GRU perform significantly better

Model and Variable	Test-statistic	P-value
MLP - CI	0.879	0.00753
LSTM - CI	0.831	0.00099
GRU - CI	0.845	0.00179
MLP - SG	0.889	0.01261
LSTM - SG	0.922	0.0645
GRU - SG	0.959	0.4111

Table 4.3: Overview of the Shapiro-Wilk Normality test results

Combination	W	p-value	Result
MLP-LSTM	77	0.03665	Yes
MLP-GRU	74	0.02913	Yes
LSTM-GRU	89	0.08392	No

Table 4.4: Wilcoxon signed-rank test results

Combination	t-statistic	p-value	Result
MLP-LSTM	-0.33801	0.73841	No
MLP-GRU	3.25957	0.00345	Yes
LSTM-GRU	5.6578	< 0.00001	Yes

Table 4.5: Two-sided paired t-test results

than MLP, with no significant difference between LSTM and GRU. The RMSE score analysis for SG suggests that GRU performs significantly better than both MLP and LSTM, while there is no significant difference between MLP and LSTM.

During the training phase, both training and validation losses were observed and plotted. An example training graph from the first fold of MLP is visible in Figure 4.3. A similar pattern was observed also for LSTM and GRU models, with detailed plots for the remaining folds and other models included in the Appendix (B).

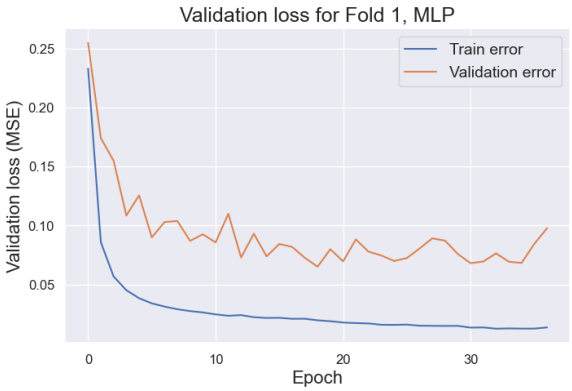

Figure 4.3: An example training and validation error from the first fold of the MLP training.

5 Discussion

5.1 Model Performance

The models were evaluated using four different metrics: RMSE score for CI, RMSE score for SG, training time, and prediction time.

For all three models, there is a noticeable difference between the RMSE score for the CI and SG within the model. This can be attributed to the pattern of both dependent variables. The CI pattern is quite complex and contains a lot of small drops and spikes, while the pattern for SG is a lot smoother and repetitive, as shown in Figure 5.1. As a result, the models are more likely to learn and predict the behavior of SG, resulting in lower RMSE scores.

While looking at the difference in RMSE scores between CI and SG within the models gives us useful insight into the model performance and the nature of the data, comparing these scores between the models is crucial for evaluating their relative performance.

The RMSE scores for CI are fairly similar across all three models, with MLP performing the worst and GRU performing the best. As the quantitative analysis showed, despite GRU having the lowest score, there was no significant difference between

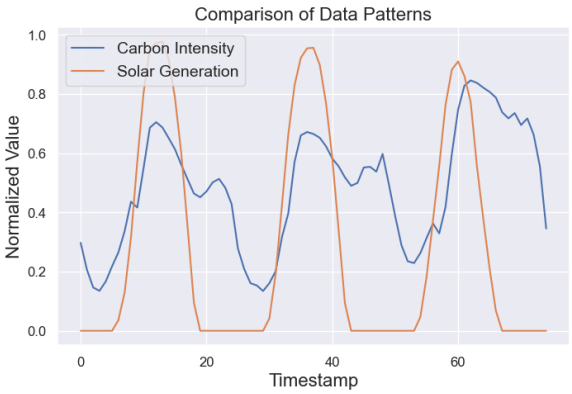

Figure 5.1: First 75 samples for the dependent variables CI and SG. The data is normalized to show the difference in the pattern more clearly.

LSTM and GRU. However, both models performed better than MLP. These results align with the expectations, considering RNNs are better suited for handling sequential data. The more complex pattern of CI was likely better captured by the more complex LSTM and GRU models.

Similarly, the RMSE scores for SG are also very close to each other for all three models. Once again, GRU performed the best, and interestingly, LSTM performed the worst. Based on the quantitative analysis, GRU performed significantly better than both MLP and LSTM, but there was no significant difference between MLP and LSTM. The improved performance of the MLP could be due to the relative simplicity of the SG pattern. It is likely that despite not having any recurrent connections, the MLP architecture was able to learn the underlying pattern relatively well.

Another factor to consider is the dataset used for training. With 720 observations and only 505 input-output training batches from the same month, the range of the data is relatively small and narrow. While this dataset provided a solid foundation for model training, the simplicity of the data may have contributed to the minimal differences observed between the models.

Regarding training time, the results were as expected. The MLP required significantly less training time than LSTM and GRU due to its simpler architecture. Additionally, GRU trained a little bit faster than LSTM.

In terms of prediction times, GRU was the fastest, followed by MLP, and lastly LSTM. However, since these times were all close to 0.1 seconds, the differences are negligible unless a large number of predictions are made simultaneously.

5.2 Feature selection

As mentioned previously, the features were selected using a Pearson Correlation Coefficient. In particu-

lar, all features were correlated against CI using a 0.4 threshold. If the absolute value of the correlation exceeded this threshold, the feature was added to the list of independent variables. Additionally, duplicate features were removed (for example, the feature depicting the current hour was in all three building datasets, but it was kept only once).

The choices of the 0.4 threshold and the exclusion of SG from the correlation comparison were made based on the empirical testing on the training set. Lowering or increasing the threshold led to a decrease in model performance, especially for CI. Increasing the threshold led to a smaller list of independent variables which did not provide enough information for the models. Conversely, decreasing the threshold introduced a lot of noise to the training process, preventing the models from accurately learning the patterns.

Similarly, including SG in the correlation comparison led to the inclusion of a lot more features. Due to the relative simplicity of the SG pattern, more features correlated well with it. This had the same effect as decreasing the threshold for CI: an unnecessary noise was added to the training process and the models struggled with learning the correct patterns for CI.

It is noteworthy that adding these features had a positive effect on the RMSE score for SG. However, due to the relative simplicity of the pattern, the RMSE score for SG was already a lot lower than that of CI, resulting in only a minor decrease. On the other hand, the increase in the RMSE score for CI was more substantial, and therefore, the overall RMSE of the model would not improve.

The weather dataset was the most influential one. Variables like Outdoor Drybulb Temperature, Outdoor Relative Humidity, or Direct Solar Radiation were all included in the independent variables. Apart from the weather dataset, the information about the current hour and the number of occupants in the house were also highly correlated.

The influence of the weather dataset was expected considering its close relation to electricity consumption and, consequently, to both CI and SG. For example, higher outdoor temperatures might indicate sunny days, leading to higher SG. Additionally, it could lead to increased electricity consumption if the buildings have air conditioning. The effect of the current hour was also anticipated as it can be used to determine whether it is a day or night. This likely affects CI, considering that electricity usage is typically lower when people are sleeping compared to the daytime. This effect is even more evident in SG which is also visible from the repetitive pattern. Energy is being generated during the day, but not during the night. The number of occupants was likely connected to CI, as more people in the house at a given time would lead to increased electricity consumption.

5.3 Overfitting

Despite exploring different combinations of the number of epochs, number of folds, and the patience parameter for early stopping, all three models seem to overfit the training data. This trend is particularly evident in the second fold of the MLP training, where small improvements in training error were observed, but the validation error steadily increased, as shown in Figure 5.2. This suggests a possible overfitting as the model was likely learning patterns specific to the training set that were not present in the validation set. With the patience parameter set to 4, the second fold ran for 27 epochs, occasionally improving the validation error, but the overall trend indicated an increase. A similar pattern was observed for the second fold in the LSTM model. The remaining folds and models can be found in Appendix (B).

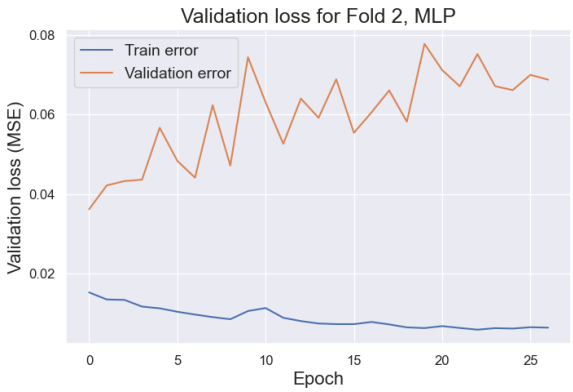

Figure 5.2: Possible overfitting during the MLP training process.

Interestingly, despite the possible overfitting, the models still showed overall improvement. Reducing the patience parameter resulted in the training process terminating quickly and a significant drop in overall performance. A possible explanation could be due to the simplicity of the data. Since there are only 505 input-output training batches and 7 features, the models may be able to learn the underlying patterns fairly accurately and the noise from the training data does not have a significant effect.

This raises the question of the necessity of cross-validation. Given the simplicity of the dataset and the rapid overfitting, could training the models on the entire training dataset for a set number of epochs yield better performance? Empirical testing suggests this might be a viable option. While none of the models without cross-validation outperformed the current models, there was not a large difference between the RMSE scores. Additionally, the removal of cross-validation led to a decreased

training time. For future research, it would be interesting to run a more statistical and thorough comparison to determine if cross-validation is even needed in this particular case.

That being said, it is worth noting that cross-validation is generally an essential part of deep learning. While in this particular case, the models without cross-validation performed comparably with the current models, this was likely due to the simplicity and the small size of the dataset (with only 720 entries and three buildings). In larger applications, cross-validation would likely be necessary to ensure a robust training process.

6 Conclusions

As discussed in previous sections, both LSTM and GRU performed better than MLP when forecasting the more complex CI, and GRU performed better when forecasting SG. However, it is important to note that the differences were relatively small and the RMSE is not the only metric of evaluation. In terms of training times, MLP trained approximately 6 times faster than LSTM and GRU. Although this difference might be negligible with the current dataset, where even the slowest model trained in under two minutes, it could become significant in large applications.

Deep learning is often applied to much bigger datasets, resulting in more complex models. This can significantly increase the time required not only to train the model but also to generate predictions once deployed. For example, if the data were extended with information from other months, more frequent observations, or more buildings, the faster MLP training time would become more significant.

On the other hand, including more data might better highlight the benefits of RNNs in handling sequential data. Therefore, it is difficult to conclude which model performs the best overall. However, with respect to the dataset used, arguments can be made in favor of both GRU and MLP.

GRU had the lowest RMSE score for both CI and SG and also trained faster than LSTM. However, if training time is prioritized, MLP could be a viable option. Despite being outperformed by both LSTM and GRU in the CI forecast, and by GRU in the SG forecast, the differences were relatively small, making MLP a reasonable choice.

For future research, extending the dataset should be the primary focus. While the current data provides an initial understanding of all models and the behavior of each feature, a significantly larger dataset is needed to simulate a real-world scenario. However, there are still many valuable insights to be gained from the current experiment.

First, this study demonstrates the different effects and significance of different data features, particularly highlighting the importance of weather information in carbon emissions and solar generation forecasting. Second, it explored the different model behaviors. It showed the possibility of building models that can accurately forecast both variables, rather than handling them separately. Lastly, the study concluded that NNs are a viable option for this forecasting task. Despite the limited data, all models predicted carbon emissions and solar generation with accuracy.

References

AIcrowd. AIcrowd | NeurIPS 2023 Citylearn Challenge | Challenges, 2023. URL https://www.aicrowd.com/challenges/neurips-2023-citylearn-challenge.

Merih Aydinalp, V. Ismet Ugursal, and Alan S. Fung. Modeling of the appliance, lighting, and space-cooling energy consumptions in the residential sector using neural networks. *Applied Energy*, 71(2):87–110, February 2002. ISSN 03062619. doi:10.1016/S0306-2619(01)00049-6. URL https://linkinghub.elsevier.com/retrieve/pii/S0306261901000496.

Mladen Bošnjaković, Robert Santa, Zoran Crnac, and Tomislav Bošnjaković. Environmental Impact of PV Power Systems. *Sustainability*, 15 (15):11888, January 2023. ISSN 2071-1050. doi:10.3390/su151511888. URL https://www.mdpi.com/2071-1050/15/15/11888. Number: 15 Publisher: Multidisciplinary Digital Publishing Institute.

EEA. Greenhouse gas emissions from energy use in buildings in Europe, October 2023. URL https://www.eea.europa.eu/en/analysis/indicators/greenhouse-gas-emissions-from-energy.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL http://www.deeplearningbook.org.

G Mihalakakou, M Santamouris, and A Tsangrassoulis. On the energy consumption in residential buildings. *Energy and Buildings*, 34(7):727–736, August 2002. ISSN 03787788. doi:10.1016/S0378-7788(01)00137-2. URL https://linkinghub.elsevier.com/retrieve/pii/S0378778801001372.

NeurIPS. The CityLearn Challenge 2023, 2023. URL https://neurips.cc/virtual/2023/competition/66590.

Michael G. Oladokun and Isaac A. Odesola. Household energy consumption and carbon

emissions for sustainable cities – A critical review of modelling approaches. *International Journal of Sustainable Built Environment*, 4(2):231–247, December 2015. ISSN 2212-6090. doi:10.1016/j.ijsbe.2015.07.005. URL https://www.sciencedirect.com/science/article/pii/S2212609015000333.

Sobrina Sobri, Sam Koohi-Kamali, and Nasrudin Abd. Rahim. Solar photovoltaic generation forecasting methods: A review. *Energy Conversion and Management*, 156:459–497, January 2018. ISSN 01968904. doi:10.1016/j.enconman.2017.11.019. URL https://linkinghub.elsevier.com/retrieve/pii/S0196890417310622.

Lukas G. Swan and V. Ismet Ugursal. Modeling of end-use energy consumption in the residential sector: A review of modeling techniques. *Renewable and Sustainable Energy Reviews*, 13(8):1819–1835, October 2009. ISSN 13640321. doi:10.1016/j.rser.2008.09.033. URL https://linkinghub.elsevier.com/retrieve/pii/S1364032108001949.

UN. THE 17 GOALS | Sustainable Development, 2015. URL https://sdgs.un.org/goals.

Ahmet Yilmaz and Riccardo Poli. Successfully and efficiently training deep multi-layer perceptrons with logistic activation function simply requires initializing the weights with an appropriate negative mean. *Neural Networks*, 153:87–103, September 2022. ISSN 08936080. doi:10.1016/j.neunet.2022.05.030. URL https://linkinghub.elsevier.com/retrieve/pii/S0893608022002040.

Zongyuan Zhao, Shuxiang Xu, Byeong Ho Kang, Mir Md Jahangir Kabir, Yunling Liu, and Rainer Wasinger. Investigation and improvement of multi-layer perceptron neural networks for credit scoring. *Expert Systems with Applications*, 42(7):3508–3516, May 2015. ISSN 09574174. doi:10.1016/j.eswa.2014.12.006. URL https://linkinghub.elsevier.com/retrieve/pii/S0957417414007726.

A Appendix

Additional examples of predicted CI values. Due to the large number of predictions, only half were selected.

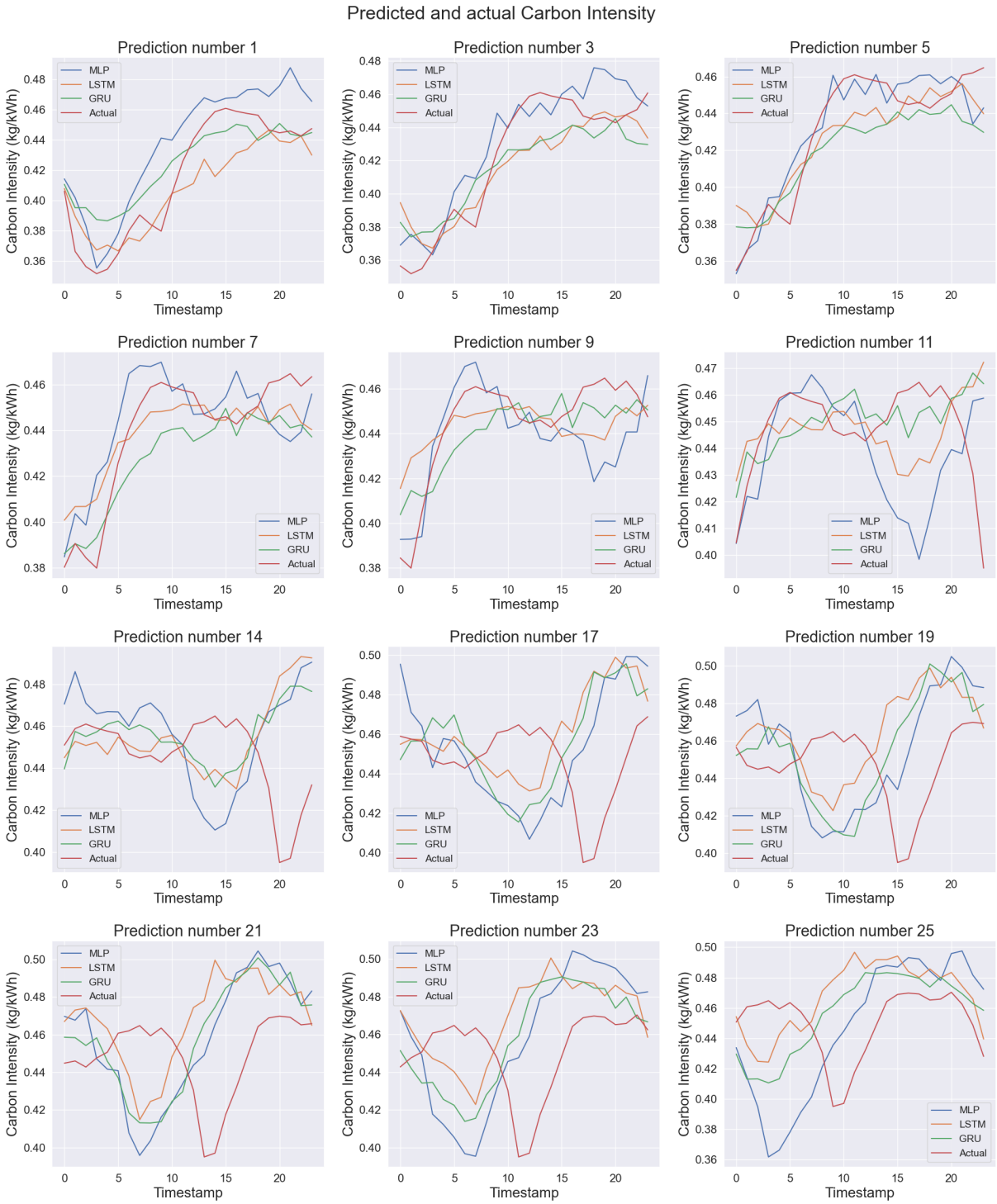

Figure A.1: Overview of the forecasts from the MLP, LSTM, and GRU models compared with actual CI values.

Additional examples of predicted SG values. Due to the large number of predictions, only half were selected.

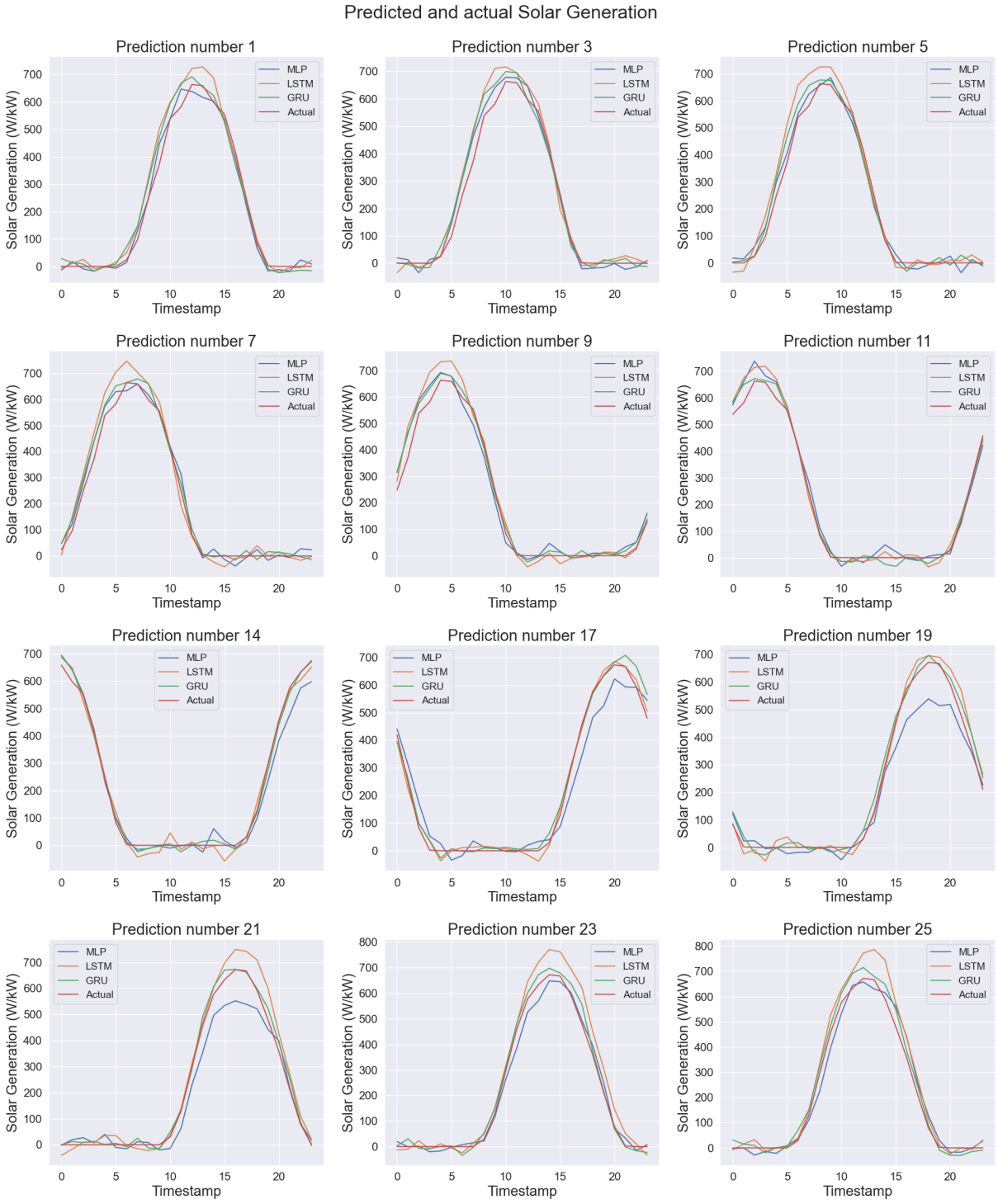

Figure A.2: Overview of the forecasts from the MLP, LSTM, and GRU models compared with actual SG values.

B Appendices

Validation loss plotted over the 4 folds of the training process for the MLP model. The validation and train error are summed for the CI and SG variables.

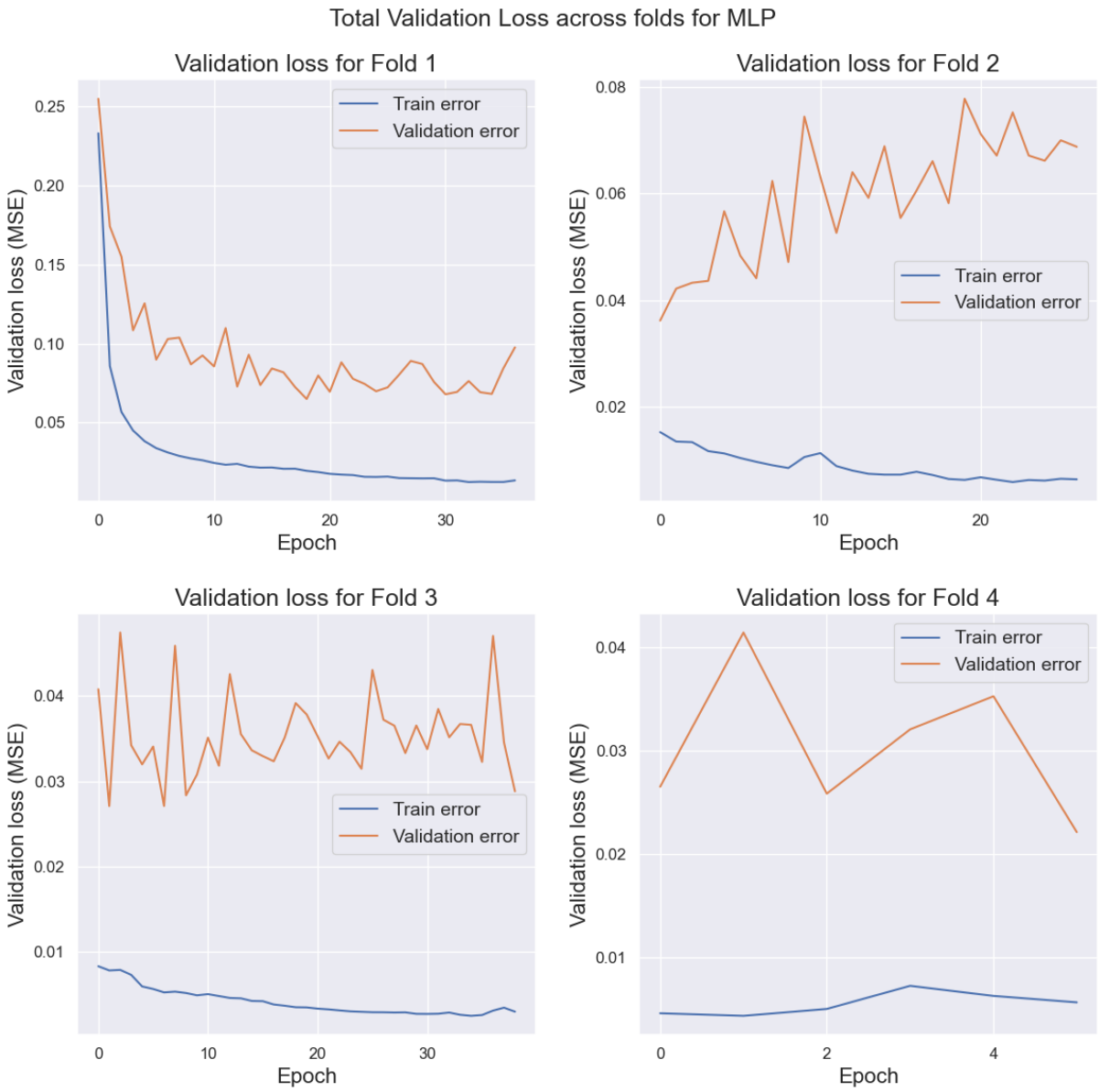

Figure B.1: MLP validation loss during the training

Validation loss plotted over the 4 folds of the training process for the LSTM model. The validation and train error are summed for the CI and SG variables.

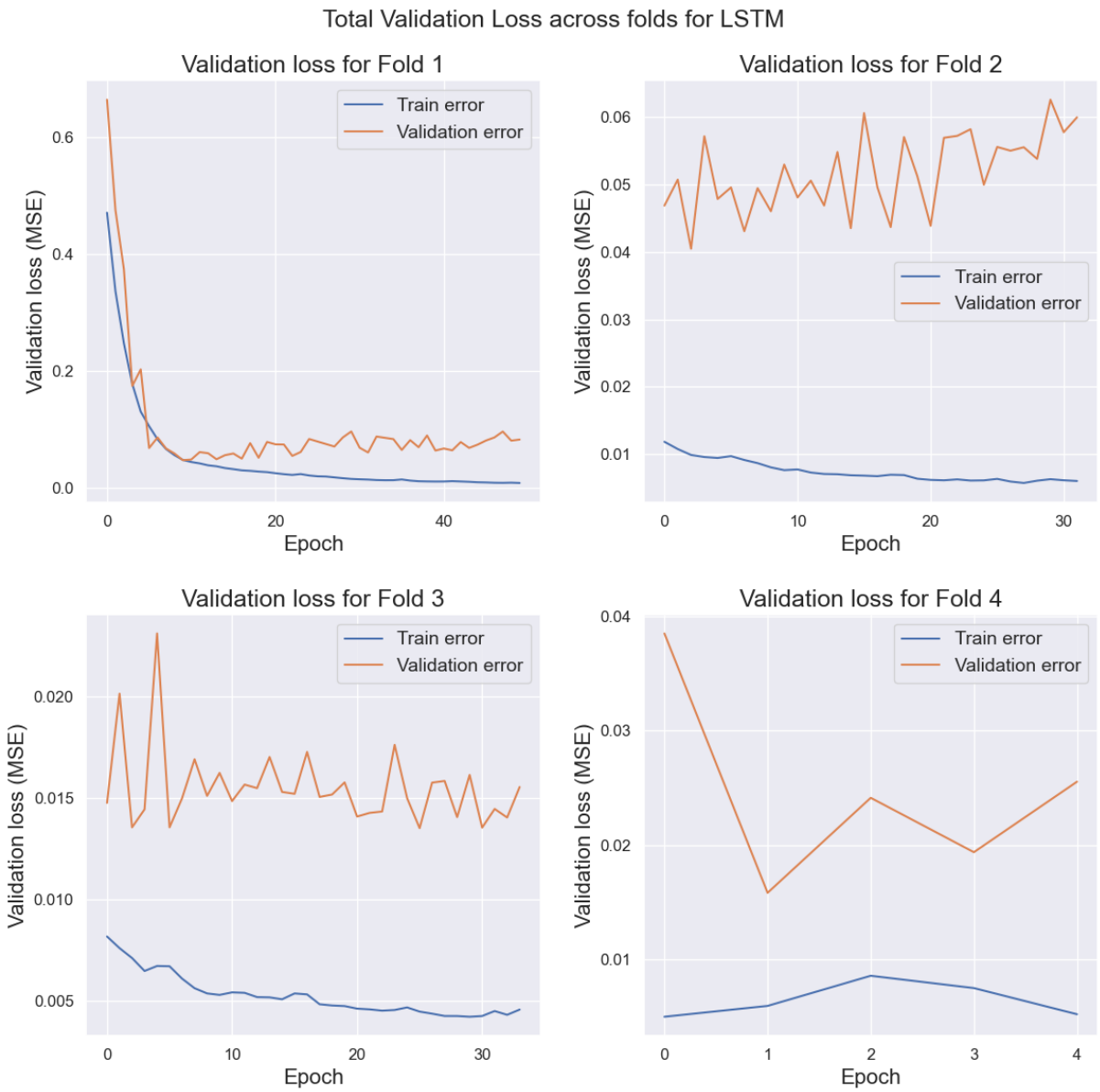

Figure B.2: LSTM validation loss during the training

Validation loss plotted over the 4 folds of the training process for the GRU model. The validation and train error are summed for the CI and SG variables.

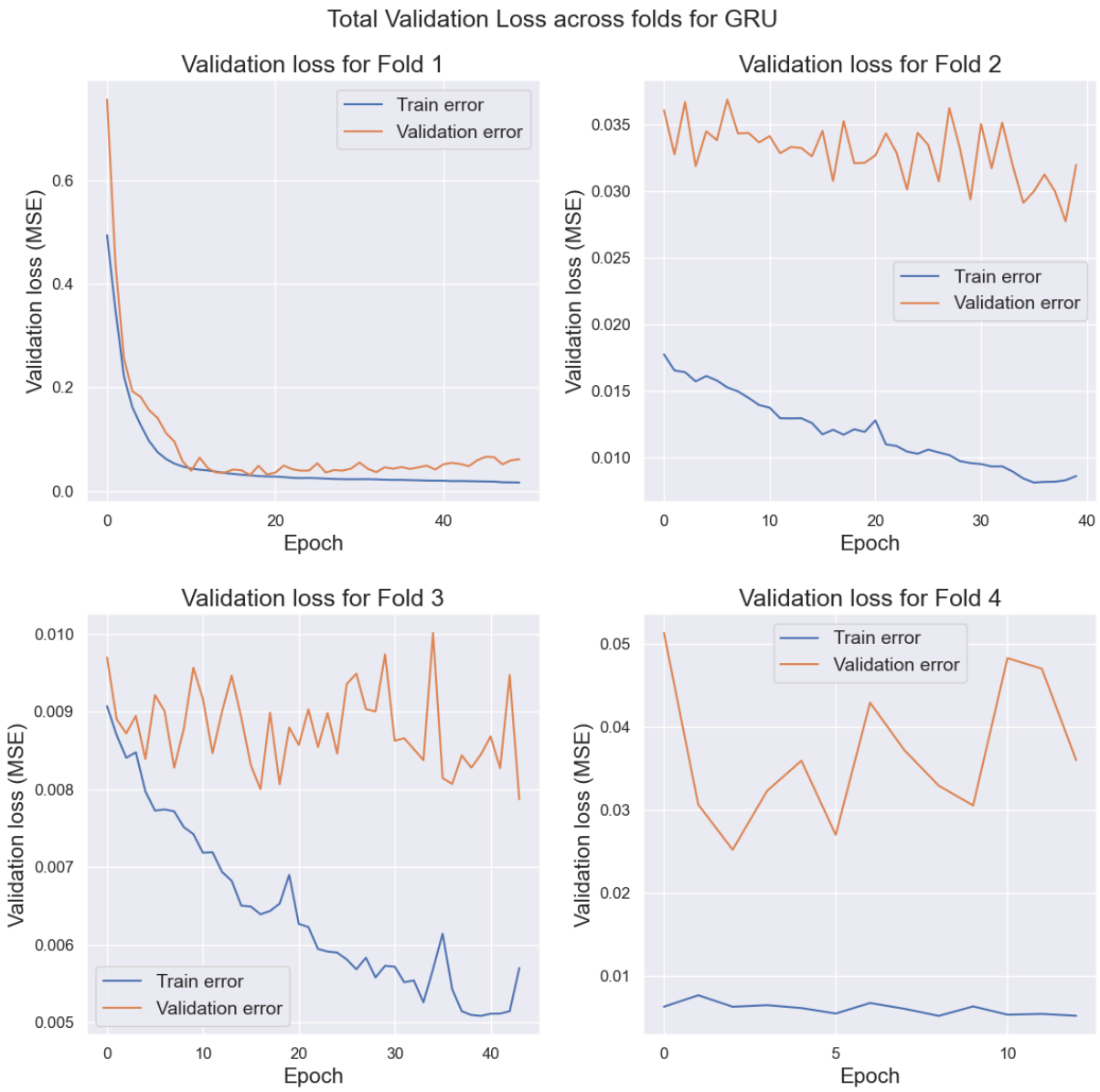

Figure B.3: GRU validation loss during the training