IKT450-G 23H Deep Neural Networks

Oskar Markussen (206302), Lars-Erik Moi (194151)

TEACHER
Morten Goodwin, Aditya Gupta

University of Agder, 2023
Artificial Intelligence Masters,

Abstract

This project, set against the backdrop of high electricity prices in Norway, aims to improve the prediction of electricity prices into the future. Utilizing the Informer model, a variant of the Transformer model, it diverges from the conventional Recurrent Neural Network approach for forecasting. The project is structured to handle the data as a complex, multivariate multi-step time series. To address the challenge of missing data, imputational techniques such as linear interpolation and forward filling are employed. The experimental results indicate the model's possible effectiveness, as evidenced by achieving a root mean squared error (RMSE) of approximately 16.97. This performance is satisfactory in predicting the average electricity price in EUR/MWh six hour into the future, and the observation shows a possible improvement to that result. when improved upon, this project has implications for better informing consumers about future price trends, potentially leading to more efficient energy usage and cost savings, particularly during times of high price volatility. It underscores the potential of advanced neural network models in accurately forecasting economic indicators like electricity prices, offering valuable insights for both consumers and policymakers in the energy sector.

Contents

Chapter 1

Background

1.1 Transformer

In time series forecasting, the encoder-decoder framework is a key method that leverages deep learning techniques. This approach divides the model into two segments: the encoder and the decoder. The encoder's role is to transform historical time series data into a more concise format. The decoder, on the other hand, utilizes this refined data to predict future events.

In scenarios where multi-step forecasting is required, this encoder-decoder architecture is often enhanced with an attention mechanism. This addition aids in integrating time-related data into the model. Within the encoder, there are multiple similar sub-layers, each comprising a self-attention mechanism, multi-head attention mechanism and a feedforward neural network. The decoder is similarly structured, with each sub-layer featuring a multi-head attention mechanism, a self-attention mechanism, and a feedforward neural network. The multi-head attention in the decoder is particularly important as it focuses on the encoder's outputs to aid in future predictions. Meanwhile, the self-attention mechanism in the decoder is essential for understanding the interrelationships among future time points.

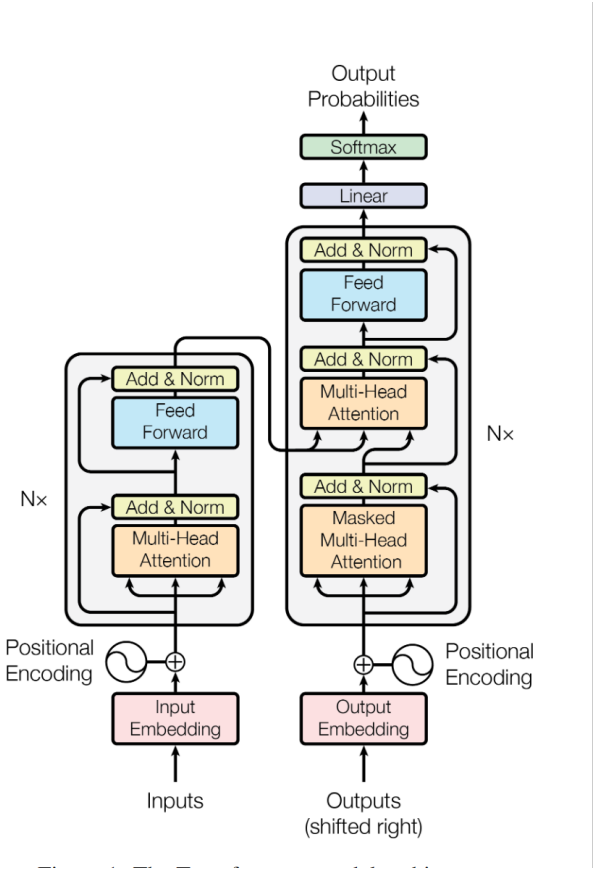

Figure 1.1: high-level view of the Transformer model's architecture, as adopted from source [16]

1.2 Self-Attention and multi-head attention

Self-attention is a technique in neural networks that assigns varying levels of importance to different elements within an input sequence when predicting a specific component. It works by representing each element of the sequence as a vector and computing a weighted sum of these vectors, with the weights determined by the relationships of the elements to the current one[16]. This dynamic approach allows the network to focus on different parts of the sequence based on their relevance, rather than just their position in the sequence. This concept is a key differentiator of transformers from traditional Recurrent Neural Networks (RNNs).

Building upon self-attention, multi-head attention involves multiple parallel attention mechanisms, each with its own linear projections. These mechanisms work simultaneously to compute representations in different subspaces. The final output for each element is formed by concatenating these individual outputs and then applying an additional linear transformation. This structure enables the network to simultaneously focus on various aspects of the input sequence, enhancing performance compared to a single self-attention mechanism[16].

Multi-head attention is particularly advantageous because it captures a range of potentially complementary perspectives of the input sequence, rather than a single generalized view. This ability to discern complex patterns and relationships within sequential data has greatly contributed to the success of the Transformer model in numerous Natural Language Processing (NLP) tasks. In summary, self-attention and multi-head attention are powerful mechanisms that enable neural networks to detect intricate patterns and long-range dependencies in sequential data, playing a pivotal role in the progress of NLP and establishing the current state-of-the-art.

1.3 Informer

The Informer model, as described by [20] is an advancement of the Transformer model, particularly in handling long sequences in time-series data. Its architecture is fundamentally divided into two main parts: an encoder and a decoder. These components work in together to manage large sequential inputs and produce corresponding outputs. The encoder initially processes the raw input, condensing it into a more digestible format. This processed information is then relayed to the decoder, which includes a multi-head attention mechanism and a fully connected layer, making the final output sequence. This method is more efficient compared to the traditional, more laborious dynamic decoding. The model's effectiveness is gauged using a loss function based on the Mean Squared Error (MSE) between the predicted and actual target sequences.

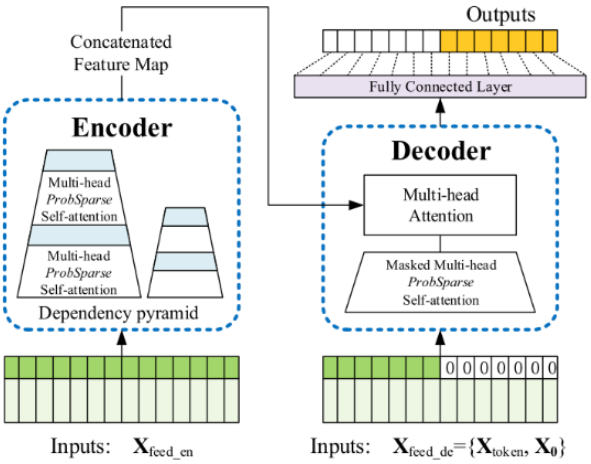

Figure 1.2: high-level view of the Informer model's architecture, as adopted from source [20]

In processing long sequential inputs, the encoder employs a technique called ProbSparse self-attention, which selectively focuses on crucial data features and manages memory use effectively. This process involves sequentially reducing the time dimension of the input across different layers, creating a pyramid-like structure as seen in fig. 1.2, and creating a condensed self-attention feature map. The outputs from all layers are combined to form the encoder's final hidden representation. 1.3 illustrates a single stack in the encoder, the principal stack handling the entire input sequence. It comprises red layers named Attention Block 1, 2, and 3, each progressively halving in size and optimizing the handling of extended input sequences.

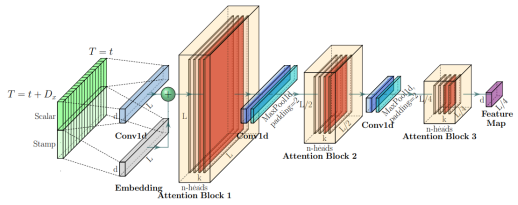

Figure 1.3: One stack of encoder, as adopted from source [20]

The decoder, shown in 1.2, receives the encoder's bundled output and employs generative inference for efficient production of long sequential outputs, surpassing traditional dynamic decoding. It starts with a token from a previous input segment and a zero-initialized placeholder. The decoder then utilizes masked multi-head attention, preventing each position from attending to future positions, thereby avoiding auto-regression. The sequence's final output is generated by a fully connected layer.

1.4 Fourier Transform

The Fourier Transform is a mathematical technique used to transform data for analyzing sequential or time-series data. It transforms data from the time domain to the frequency domain, revealing the frequency components of a signal. This transformation aids in feature extraction, making it easier to identify patterns, trends, and anomalies in data-sets like audio, financial time series, or sensor data [14]. In AI, the Fourier Transform is used for tasks such as signal processing, data compression, and enhancing machine learning models, especially in areas like anomaly detection, pattern recognition, and integrating with deep learning architectures. This approach is key in efficiently handling and interpreting large and complex data-sets in various AI applications.

Another Sequence based transformation is the usage of sinus and cosinus to represent a circulation. It can be explained like a sinus and cosinus transformation, where the data is represented in a circular way, often connecting time-series. One can think of this as a way to divide a recurring data sequence to improve pattern recognition. An example of how this could be used, is to "tell" the data model that the end of a month is not far away in sequence from the start of the next month, or too introduce the concept of weeks to a time date column. [11]

1.5 Lagged Features

Lagged features refer to the inclusion of data from previous time steps into the current analysis frame or input sequence. This means that for a given timestamp, say 01.01.2023 01:00:00, the model not only considers data from this specific moment but also integrates data from preceding time slots, such as 01.01.2023 00:00:00. The time step could vary depending on the desired result. Lagged features is often used as a feature extraction method to increase pattern data in a data-set and can be used further in correlation–based models [18].

1.6 Linear Interpolation, Forward and Backward Filling

Linear Interpolation, Forward Filling, and Backward Filling are techniques used in data processing to handle missing values, often valuable in sequential data. Linear Interpolation estimates missing values by creating a straight line between two known values in a data-set, ideal for time-series data where this linear approach can approximate the missing data points semi realistically. Forward Filling, carries forward the last known value to fill subsequent missing values, while Backward Filling, works similarly but fills the missing values with the next known value. These methods are sometimes good to use in preparing data-sets for analysis, ensuring continuity and reducing the impact of data gaps. Another tool often used with linear interpolation is the LightGBM framework made by Microsoft.[10]

1.7 Rolling Mean and Standard deviation

Rolling mean and standard deviation are measures used to analyze time series data, and a way to smooth out short-term fluctuations and highlight longer-term trends or cycles. The rolling mean (or moving average) calculates the average of a subset of data points within a specified window that 'rolls' through the data set. For each position of the window, it averages the data points within it, and this process is repeated as the window moves across the data set step by step. Similarly, the rolling standard deviation measures the variability or dispersion of the data points within the moving window. It calculates the standard deviation of the values within each window. Which gives insight into the consistency of the data over time. [21]

1.8 Zero Mean Normalization

Zero mean normalization, is a possible scaling transformation that can be used as a preprocessing step in time series forecasting . This process adjusts each feature in the data-set to have a mean of zero and a standard deviation of one. It ensures that all features contribute equally to the model's learning process, regardless of their original scale.

1.9 Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is a widely used metric in both statistics and machine learning for measuring the accuracy of a model in predicting quantitative data. It calculates the square root of the average of the squared differences between predicted and actual values, providing a measure of the magnitude of prediction errors. The squaring of errors in RMSE gives greater weight to larger errors, making it particularly sensitive to outliers. A lower RMSE value indicates a model with better predictive accuracy. [7]

1.10 Mean Squared Error (MSE)

Mean Squared Error (MSE) calculates the average of the squares of the errors, where the error is the difference between the predicted values and the actual values. By squaring the errors, MSE emphasizes larger errors more than smaller ones, making it particularly sensitive to outliers in the data. The result is a non-negative number where a lower MSE indicates a model with a closer fit to the actual data. [7]

Chapter 2

Introduction

We where given a choice by our professors to implement a Deep Neural Network (DNN) based model to solve a problem of our choice. We where approached by two of our seniors, these two seniors was in connection with a up and coming electricity based company. This company was very interested in the usage of AI and DNN to solve one of their problems. Which was the prediction of electricity prices over time. This involved trying to predict the prices in a sequenced matter. This problem of sequential predictions intrigued our curiosity and we wanted to research into this problem they had provided. By researching a little about sequential prediction models, and some previous knowledge from class, we stumbled upon a current state of the art model called a transformer.

In which we learned that with the introduction of the transformers, a type of sequence prediction model, a new and more efficient way of handling sequence predictions was put in to place [13]. By utilising transformers, newer models that could be used to predict more sequence related problems started to appear. One such model was the informer, created by a team consisting of Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, Wancai Zhang [20]. This invention has been argued to be more efficient at long time sequence predictions by ”The Chinese University of Hong Kong" [17]. This is is why when we where tasked with creating a Norwegian electricity prediction based model we decided to approach the informer as a solution to our problem.

The goal of the model is to input some parameters such as previous electricity, previous weather reports, and other possible effecting parameters such as import and export of Norwegian electricity, to train the model on finding underlying patterns to predict the future sequence of electricity prices. With quantitative research we aim to provide an overall analyses of how the informer could be used to predict the electricity prices over a sequence of time, as well as a result of how our experiences with the training of the informer became.

The steps we will take to achieve the intended results is by first trying to predict any data, then following with a sequential prediction of the data, and lastly a experimental pushing of data predictions, with this we mean to increase the amount of data predicted until we reach a less satisfactory RMSE in our model. The last task would be to analyse the statistical data and further explain on our experiences with the model.

This project we hope to further enlighten how the company could use AI and DNN to predict the electricity prices, and if the statistical result are satisfactory we hope it could be used as a base model to solve their problem of prediction.

Chapter 3

Methodology

In our project focused on forecasting electricity price by time series, we required extensive data-sets, including non-standard pattern data that extends beyond typical electricity-related information. Much of the data for this project was sourced from reputable and freely accessible websites and official publications, ensuring reliability. However, the quantity of data obtained was frequently less than ideal for our training needs.

3.1 Data-set

When creating a data-set we had to decide on entries that could correlate with the forecasting of electricity prices, some ideas came to mind but our overall entries that would be included before preprocessing of the data-set, was the electricity price of low, high, last, volume and average from the company Nordpool [1], weather forecasting for the Kristiansand including air pressure at sea level, air temperature at 2m, cloud area fraction, relative humidity at 2m, wind direction at 10m, as well as wind speed at 10m, which was provided by an API to the Norwegian Meteorological Institute [2]. Another entry that was decided to be included was the electricity (EL) Production, EL Consumption, EL Import, EL Export, and EL Flow from Norway which was provided by statnett [3]. We had decided to add the price of oil and gas, but was stopped by doing so because of the fee for that information, as well as the quality of free data, such as price per hour. The entry we will be predicting is the Average electricity prices from Nordpool, which is set to be in euro per megawatt/hour, and the time steps is each hour.

3.2 Preprocessing of Data-set before training

After gathering the different data entries for the data-set, we went forward with deciding on how to preprocess the data. There was multiple of different approaches to data- and pattern extraction that could improve the data-set, but some key features was needed to be addressed, which was the correlation of sequenced based patterns, as well as how to handle missing data.

To apprehend the correlation between the sequenced data and the time-series, we decided on a few approaches, such as adding a Fourier entry to the "Avg" column. This addition was implemented to introduce a frequency domain perspective into our data-set. By applying the Fast Fourier Transform (FFT) to the "Avg" data, we transformed this time-series data from the time domain to the frequency domain. This transformation allows us to capture and analyze periodic patterns and trends within the data that may not be very apparent in the raw time-series.

Another approach was the usage of sinus and cosinus transformation columns to apply a circular pattern to three main time features, we deemed important, which was days, weeks and months. Applying this was used to find patterns within days, weeks and months and would help in adding

possible correlations between dates in our data-set.

To adjust for the missing data points in our data-set we decided on two different techniques that we wanted to try which was linear interpolation and forward fill. These two are used as a way to replace missing data and reduce data leaks. There were alternative methods available to address our dataset's shortcomings, such as developing a prediction model to fill in the gaps. However, we decided that these approaches were not within the scope of our current project. More importantly, they would have required a significant investment of time and resources. Therefore we chose the more time-efficient solution that aligns with our project's immediate objectives and constraints.

After this preprocessing we decided to add three additional features to our data-set which was lagged features, binary correlation (modified One-hot encoding) [8] , rolling mean and standard deviation. The lagged features was used to try to create a better correlation between the previous data, and and the current. This feature was used to try to extract more patterns. The binary correlation was used between the production column and consumption column. Where if the production was lower than the consumption then the correlation column would be set to 1 or else to would be set to 0. This was used to try to add a connection between the two columns, consider they are closely related to how the energy prices increase when there is a need for import.

Rolling mean and standard deviation was last added to the electricity avg column cumulative over a 7-day periods to add a layer of data smoothing, effectively minimizing daily fluctuations and highlighting underlying weekly trends. First standard deviation is zero. This approach is said to be particularly useful in identifying more stable, long-term patterns, making it less influenced by short-term daily variations [9]. Another technique which we considered was the usage of AutoRegressive Integrated Moving Average (ARMIA) as a way to extract patterns, but we opted to use the Rolling mean and standard deviation instead. [15]

To combat the varying scales of values in our data-set, we initially considered implementing a scaling transformation. However we decided against it. This decision was influenced by the fact that our existing informer model already incorporates a zero-mean normalization process. This normalization is applied both during data loading and reversed upon completion of the model run. As a result, adding an additional scaling step seemed redundant and potentially disruptive to the established workflow of our model. [6]

3.3 Testing four primary times series

When discussing the training of the model it was opted to try with four different prediction lengths which was decided to be 1 hour, 6 hours, 12 hours and 24 hours, or 24 prediction steps. The the predictions sequences in an informer is done in chunks by taking a large segment of past data and generating a forecast for a future period in one go. Despite this it will often get a worse result the more prediction steps that it has to do, because of the increase complexity. This is the reason for not increasing the prediction length any further, because if the result of the previous amount of sequences is not satisfactory then going further would not have any positive effect on the model.

3.4 Testing of Parameters with Sweep

As the four different prediction lengths will have different optimized parameters it was decided to test and optimize each parameter dependent on each prediction length. This is where we decided to use an AI tool/website called Weights and Biases [4]. Weights and Biases is an AI tool used to keep track of each run that was performed and give an estimate of which parameters where important for that run/sweep. With Weights and Biases we used their method called Sweeping which is a method of configuring which parameters that is to be tested and changed over a period of runs, where the

tool will try to get a better result on the targeted loss function after each sweep, with the usage of Bayesian optimization.

In our prediction model, we decided to use Mean Squared Error (MSE) as our loss function and Root Mean Squared Error (RMSE) as a visual indication of our loss. We chose MSE because it heavily penalizes larger errors, which is important for our model as we aim to minimize significant inconsistencies between predictions and actual values. The squaring of errors in MSE magnifies the impact of these larger deviations, making our model more sensitive to them and therefore more accurate in minimizing them. This was important considering a wrong prediction in electricity price could create a greater financial loss. On the other hand, RMSE being the square root of MSE, provides us with a more understandable scale. It expresses error in the same units as our target variable, making it easier to understand and communicate the model's performance. While RMSE still maintains the emphasis on larger errors, its linear scale offers a more nuanced view of our model's accuracy.

3.4.1 Fine tuning and analysing of model

After a batch of runs with our sweeping, we would look at each run with its parameters and further analyse which parameter ranges we would test next. This was then done iterative until we decided that it was no longer favorable to continue optimizing the parameters, or the time frame for this project was coming close.

3.5 Architecture of the Model Utilized

The architecture of the model utilized as pre-metioned is the informer model, which is a modified transformer. In our project we utilized ReLU (Rectified Linear Unit) and GeLU (Gaussian Error Linear Unit) as activation functions, we tried to utilize two separate to test which gave a better result for our data-sets. ReLU (Rectified Linear Unit) is a simple, linear activation function used in neural networks, outputting zero for negative inputs and the input itself for positive inputs. GELU (Gaussian Error Linear Unit), on the other hand, is a non-linear function that allows some negative inputs to pass, based on a probabilistic model. The key difference lies in GELU's ability to handle negative inputs more smoothly, reducing the likelihood of dead neurons that can occur with ReLU. [5]

The parameters that was specified in our training outside of the standard pre-determent parameters in the informer was; sequence length, label length, prediction length, encoder in, decoder in, ProbSparse attention factor, dimension of model, number of heads, encoder layers, decoder layers, dimension of the fully connected layer, dropout, batch size, learning rate, epochs, and early stop patience.

Some of those parameters was constant throughout the training runs of the model, such as the parameters shown below in the table 3.1.

encoder layers	decoder layers	epochs	patience	...
4	2	10000	3 or 5	

Table 3.1: Constant Parameters

The other parameters/configurations was optimized with the usage of sweep configurations. This means that it would change from run to run and try to optimize the values of the configured parameters talked about earlier.

The data-sets was split in to training set, validation set and test set, Where the training set was equivalent to 70% , the validation was equal to 10% and the test set was equal to 20% of the data-set at the start of our training, however it was changed to 90% training, 5% validation and 5% testing after observed improvement. We discussed the usage of cross validation, and k-fold with our training but we decided against it based on that our data is sequenced based, and it could lead to inconsistent data.

Chapter 4

Result

4.1 Data-set

Our project utilized various data-sets, one spanning the entire year of 2023 and another covering the period from 2021 to 2023. Initially, our data range extended from 2020 to November 2023. However, we identified a significant issue with the 2020 to 2021 portion, where almost half of the data entries contained zeros. We determined that this would adversely affect our analysis's accuracy and, therefore, decided to exclude this period. Example of how some of the data look like in 2020, in figure 4.1

Product	High	Low	Last	Avg	Volume
PH-20200101-01 (X)	40,00	34,12	39,51	39,06	11,10
PH-20200101-01 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-02 (X)	38,12	34,62	38,12	35,25	9,60
PH-20200101-02 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-03 (X)	34,30	32,28	32,28	33,75	12,50
PH-20200101-03 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-04 (X)	33,80	30,30	33,80	31,70	6,00
PH-20200101-04 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-05 (X)	33,40	33,39	33,40	33,39	3,70
PH-20200101-05 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-06 (X)	35,40	32,30	32,90	33,99	72,90
PH-20200101-06 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-07 (X)	34,08	29,00	34,08	32,07	78,10
PH-20200101-07 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-08 (X)	29,72	29,39	29,72	29,64	0,90
PH-20200101-08 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-09 (X)	0,00	0,00	0,00	0,00	0,00
PH-20200101-09 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-10 (X)	30,00	30,00	30,00	30,00	20,00
PH-20200101-10 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-11 (X)	29,98	28,44	29,00	29,40	48,70
PH-20200101-11 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-12 (X)	30,49	29,00	29,40	30,01	73,90
PH-20200101-12 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-13 (X)	35,59	28,50	28,50	31,29	220,60
PH-20200101-13 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-14 (X)	32,33	23,50	23,50	29,26	187,90
PH-20200101-14 (I)	0,00	0,00	0,00	0,00	0,00
PH-20200101-15 (X)	32,69	25,40	26,60	29,18	204,10
PH-20200101-15 (I)	0,00	0,00	0,00	0,00	0,00

Figure 4.1: example of nordpool missing data [1]

The data-set comprises 34 columns, structured as shown in table 4.1. For a detailed view of all columns, refer to the appendix.

Date	High	Low	Last	Avg	Volume	...
2021-01-01 00:00:00	24.0	23.0	23.0	23.57	8.7	...
2021-01-01 01:00:00	24.1	23.0	23.0	23.63	8.7	...
...	...	...	...	...	...	...
2023-11-19 22:00:00	94.1	80.78	94.1	88.08	240.2	...
2023-11-19 23:00:00	87.88	78.49	87.62	81.92	204.1	...

Table 4.1: Partial Data-set Overview

To address gaps in the data-set see figure 4.2, we applied two methods of interpolation: forward filling see figure 4.3 and figure 4.4 for linear interpolation.

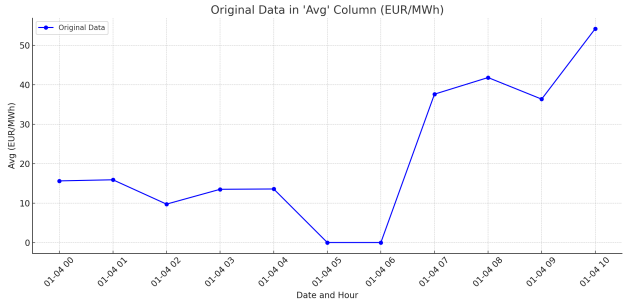

Figure 4.2: example of original data

After applying forward fill:

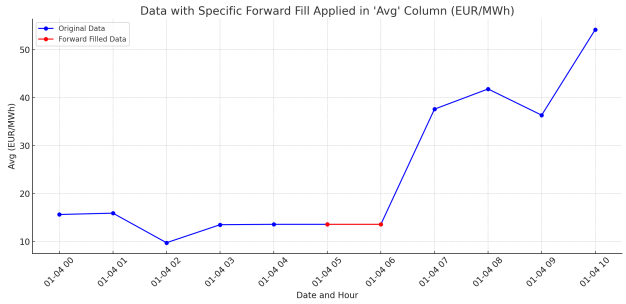

Figure 4.3: example of original data forward filled

after the linear interpolated:

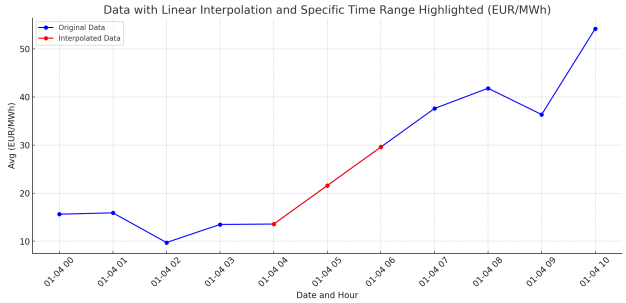

Figure 4.4: example of original data linear interpolated

In our analysis, Fast Fourier Transformation (FFT) showed better performance compared to using only sinusoidal and cosinusoidal transformations. As FFT inherently includes these patterns, we chose to drop the individual sinusoidal and cosinusoidal transformations to eliminate redundancy.

An observation we had under our training of the models was that despite the similarity in loss and RMSE between the forward fill and the linear interpolated data-sets, the optimized results was entirely forward fill. However this does not mean we observed the linear interpolated to be performing significantly worse but the following data will all be from the forward fill data-set.

An example of the similarities in loss and RMSE is shown in the graphs, 4.5 and 4.6.

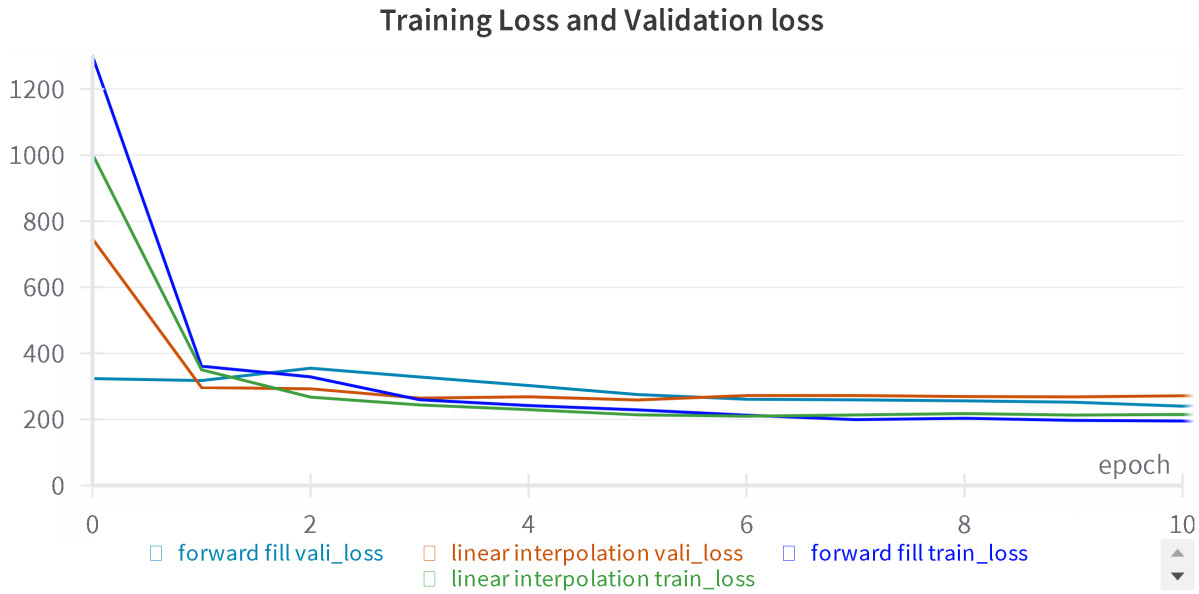

Figure 4.5: Train Loss Vali Loss for 2023 dataset

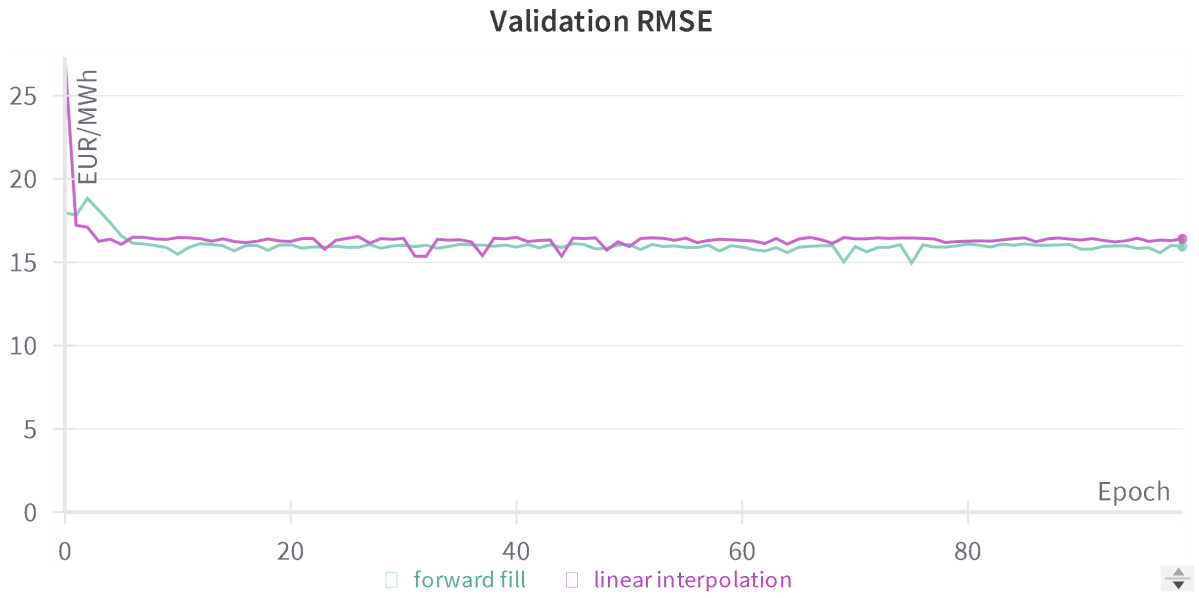

Figure 4.6: Validation RMSE for 2023

4.2 Optimizing results

We employed the sweep method to optimize our model configurations. This process was guided by Bayesian optimization, with a specific focus on minimizing the 'RMSE' (Root Mean Square Error).

It's important to note that the parameters such as 'sequence length' (seq_length), 'label length' (label_length), and 'prediction length' (pred) were not manually selected for each run. Instead, these parameters were determined through the optimization process itself, but in a specific set range, see figure 4.7. Essentially, the values chosen for each of these parameters represent the optimal configurations that yielded the best performance in terms of RMSE. This approach ensures that our model tuning is both efficient and effective, leveraging Bayesian optimization to systematically explore the parameter space and identify the most promising combinations.

```
sweep_config = {
    'method': 'bayes',  # Bayesian optimization
    'metric': {
        'name': 'RMSE',  # Metric to optimize
        'goal': 'minimize'  # Aim to minimize validation loss
    },
    'parameters': {
        'batch_size': {
            'min': 8,
            'max': 128,
        },
        'factor': {
            'min': 3,
            'max': 9,
        },
        'seq_len': {
            'values': [6*3*3, 7*3*3 , 8*3*3,24*3*3]
        },
        'label_len': {
            'values': [6*3,7*3,8*3,24*3]
        },
        'pred_len': {
            'values': [24]
        },
    }
}
```

Figure 4.7: bayes optimalization with sweep

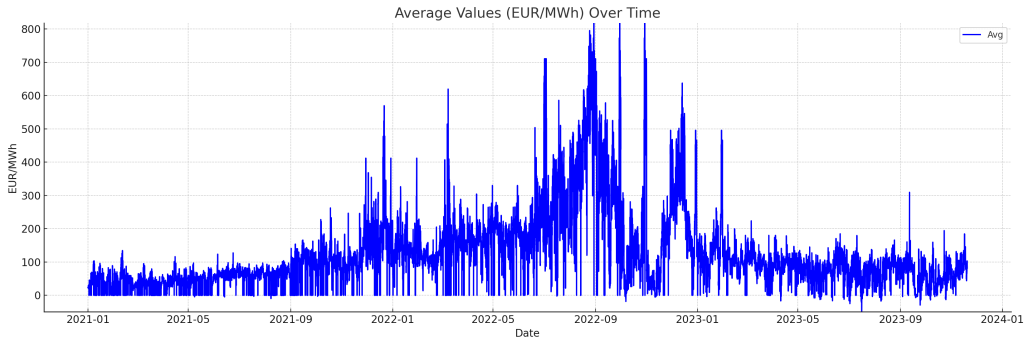

Figure 4.8: 2021-2023 data-set 'Avg'

After looking at the data-set from 2021-2023 we observed that the training data for 2022 was very different from the average price range within the other years, see figure 4.8, which makes sense considering the war in Ukraine was started 24 February 2022. This made things more complicated for our model considering it is a big change in patterns from the norm, and we observed it through our training data that it was indeed effecting our model,see table 4.2. This change in patterns could be somewhat be mitigated if we had the gas and oil price history for our data-set, but considering we did not have this we opted for only using the 2023 data. This would significantly lower the amount of data point we could train our model on but would ultimately lead to a better result.

pred	label_length	seq_length	RMSE	Train Loss	Validation Loss	Test Loss
1	6	72	14.12	861.87	655.94	491.44
6	18	54	19.52	891.88	676.99	392.44
12	54	135	25.71	898.59	951.07	694.99
24	54	135	27.82	1501.81	1131.65	1353.04

Table 4.2: Model Performance Metrics for 2021-2023

After these results and considerations we started to train on the 2023 data-set and as we can see from the table in comparison the 2023 data-set yielded better RMSE in all cases.

pred	label_length	seq_length	RMSE	Train Loss	Validation Loss	Test Loss
1	3	45	9.23	194.53	282.83	94.70
6	18	27	16.97	311.66	1153.66	522.31
12	3	36	21.73	446.51	1273.12	817.29
24	6	9	26.46	682.47	1856.69	1516.93

Table 4.3: Model Performance Metrics for 2023

We visualize the evaluation of our model in Figure 4.9, where the model's predictions are plotted against the actual observed values, referred to as 'ground truth' in the test data set. we can observe that there is a resemblance between the predicted and actual values, indicating the model's capability to capture the underlying patterns in the data in some degree. The values here a inverse transformed from the zero-mean normalization.

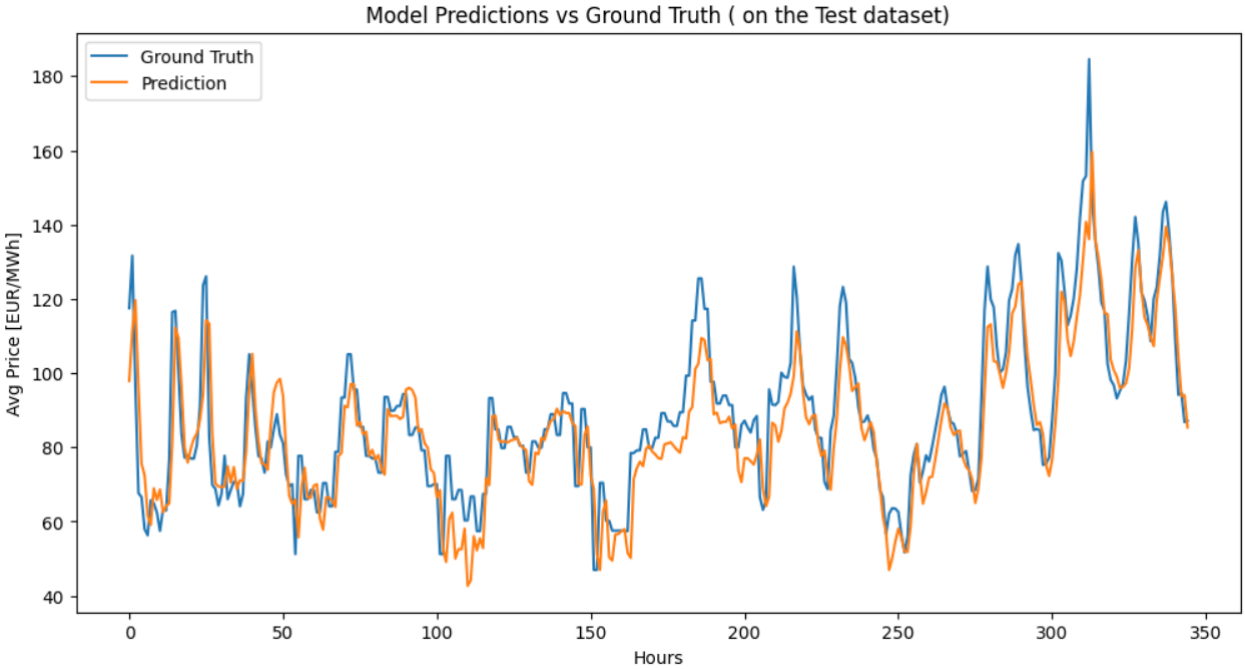

Figure 4.9: prediction vs ground truth(test data-set) on the best model

Following the previous stated optimized result the following in graph 4.10, shows the loss during training of the model and one can observe some over-fitting in all of the loss graphs, we had early stopping on. As we observe in the right bottom of graph 4.10, the graph would even out after more epochs, however with the early stopping we had, one can see that there is a clearly over-fitting at the start of our training.

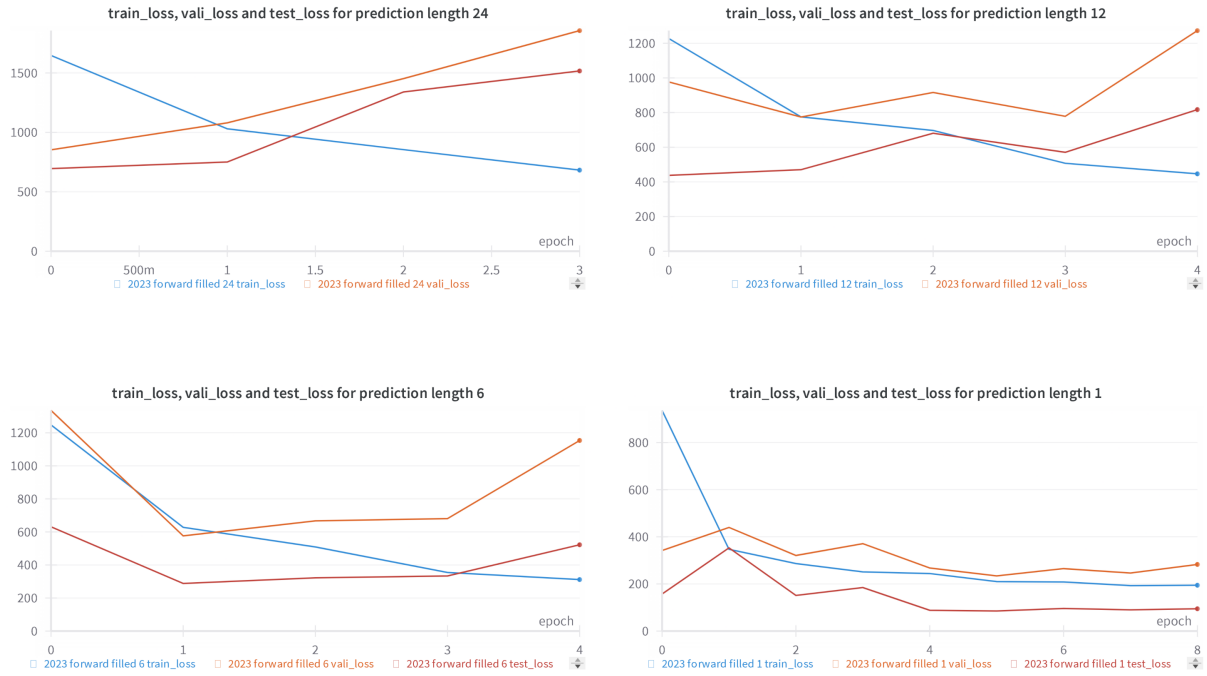

Figure 4.10: four best preforming models in each respective pred length

4.3 Result of configuration data

Utilizing sweep for parameter optimization, we achieved individually tailored parameters for our top-performing models across different prediction lengths. Some of the runs yielding similar outcomes but could involved vastly different parameter combinations, highlighting that optimization can be attained through multiple pathways. The following table 4.4 and table 4.5 details the specific parameter combinations corresponding to the results presented in table 4.3.

pred	label_length	seq_length	batch size	dropout	learning rate	factor
1	3	45	76	0.25	0.0001	4
6	18	27	64	0.25	0.0001	7
12	3	36	14	0.25	0.0001	7
24	6	9	68	0.25	0.0001	6

Table 4.4: Model Parameter Metrics for 2023

heads	fully connected layer	dimension of model	encoder in	decoder in	epochs
11	2048	1078	34	34	8
11	2048	1078	34	34	4
11	2048	1078	34	34	4
11	2048	1078	34	34	3

Table 4.5: Model Parameter Metrics for 2023

Chapter 5

Conclusion and future work

After training our models, and testing out which parameter combination could be optimized, we observed some configurations and conditions that could be improved upon. The following chapter we will be discussing some of the observations we had from the results and methodology, as well as give an explanation for what we could do in the future and lastly an conclusion of our results and experiences.

5.1 Observations and Reflections

We observed several key challenges with the data-set that impacted its suitability for training the Informer model. These challenges presented significant obstacles in optimizing the model's performance, and the most impacting faults laid in the amount and quality of our data points.

5.1.1 Data-set disadvantages

As we only used the 2023 data from our data-set, considering 2022, we expect to get a worse result than anticipated based on the lesser amount of data points the Informer got to train on. This anticipation was confirmed in our observation considering we expected a lower result over all, based on previous knowledge of the informer model. However it was not as significantly decreased as anticipated considering the model still finds considerable patterns such as shown in graph 4.9.

This exclusion of previous yearly data would also impact the models learning on seasonal data which electricity prices is very prone to have, considering the increase of prices during winter months and decrease during summer months.

Another problematic event considering the data-set was the exclusion of the gas and oil price from the data-set, which was one of the key factors intended to extract non-direct patterns, and since this data was exclude we could expect a less generalized model considering it would get less patterns to extract. This lack of data and the inclusion of missing data in already gathered data-set could be one of the reasons for over-fitting some of the models as observed previously in the results. The missing data problem was somewhat solved with the usage of either forward fill or linear interpolation, however its was still observed to be lacking as it could sometimes lead to miss information, this is why as previously state in the methodology that we could have used a different approach such as a Long short-term memory(LSTM) prediction on the missing data, however as stated before we opted for a less time consuming approach.

A last observation with the data-set was that there could have been too many data columns which would in theory lead to multiple difficulties such as the increased risk of multicollinearity where independent variables are highly correlated. This can make it difficult for the model to distinguish between the effects of different features on the target variable. Another difficulty could be the

difficulty in the model to learn relevant patterns, and with the increased complexity, over-fitting is often prone to happen. Which fits well with the results we have gotten.

5.1.2 Data columns changes

Some minor observations we had with our data-set columns was that our usage of "Nordpool" data, which is the buying prices for electricity, might not be as suited to predict relevant data such as the consumer price. Which could be a better prediction target and would be a better alternative to our initial task. Another inconsistency in the data columns might be the exchange rates where most of our data is based on big company numbers rather than consumer numbers. The "Avg" column is measured in euro per MegaWatt/hour, which means it will be a big difference from the average price a consumer would use and could be less efficient to predict on because of its scaling issue.

5.1.3 Parameters and configuration observations and improvements

Another important factor in the results of the predictions, would be the parameters used in the informer. As the prediction length is relatively short and we had a more difficult data-set to work with, it was increasingly more important to have the right parameter combinations, as we observed the minimal change in parameters could lead to a greater change in RMSE, and pattern recognition.

Some ways to combat this parameter sensitivity would be to either increase the prediction length or to increase the parameter optimization range, in which we search for with our sweep functionality. This increase in range, with the added time the sweeping could use to do Bayesian optimization, could improve our loss as it would find a better suite combination of parameters for our informer. Another optional method in which we could use to improve our model parameters, is to use a correlation matrix on the different parameters to optionally find a correlation between them, for future analyses.

5.1.4 Informer Observations and Improvements

As we where training our model the initially usage of a 70% training, 10% validation, and 20% testing split showed to be a disadvantage for our data-set size considering it would train and validate on quite few data points. This could be improved with a bigger data-set, or what we did as explained in the methodology was to change the rate to 90% training, 5% validation, and 5% testing split. However this change in the split rate could be one of the reason why we observed such over-fitting in the observed loss data as talked about in the result chapter. The improved RMSE we got from the change itself could also be a type of fake positive considering we are validating and testing on fewer data points in total. Which makes our results might appear better considering we are using a MSE loss function which heavily punishes deviants, and the more data to test and validate on the more deviants will occur, making it less punishing with less points, in theory.

As we worked with the informer and its scaling alternative, zero mean normalization, we discovered that the previously mentioned scaling might not be the best scaling option for our data-set. According to the article "A Large Comparison of Normalization Methods on Time Series" [12], implies that alternative normalization methods, like maximum absolute scaling, can outperform z-normalization, particularly in classification tasks using Euclidean Distance (ED). Despite the fact that our model is not used to classify, the argument still stand that other normalization techniques can be better suited, depending on the tasks and data-sets. Which is something that could be used to improve the predictions of our model.

5.2 Problems with informer as our model for short term forecasting

According to the creators of the informer, and authors of the extending article ”Expanding the prediction capacity in long sequence time-series forecasting"[19] page 17, which addresses that the informer could easily be over-fitted to short sequence data, and they think that it might be linked to its attention mechanism, since it uses a probability distribution to focus on key elements of the encoder input rather than the whole sequence. Which is something we have observed similarly in our results. As all of our predictions is within what they would call ”short sequence data", and are prone to over-fitting.

5.3 Future experiments and work

In our future work, we aim to address several key aspects identified during our project. Firstly, we plan to expand the data-set by solving the data problems regarding previous years like 2022, to enhance the model's learning. Which might lead to more understanding in seasonal variations in electricity prices. We would also like to add variables such as, gas and oil prices, into the data-set to extract more comprehensive patterns and improve model generalization.

Additionally, we intend to refine our approach to handling missing data. While forward fill and linear interpolation were effective for our relative small project, exploring methods like Long Short-Term Memory (LSTM) predictions for missing data could offer more accurate solutions.

Another area of focus will be on optimizing the number of data columns. Reducing the number of columns could alleviate issues such as multicollinearity and model over-fitting, thereby enhancing the Informer model's ability to learn relevant patterns more efficiently.

We also plan to reevaluate the selection of data columns, considering alternative prediction targets like consumer prices instead of ”Nordpool" data for electricity, which might align better with our forecasting objectives.

Regarding the Informer model's parameters, we aim to experiment with extending the prediction length and expanding the parameter optimization range. This could be achieved through more comprehensive Bayesian optimization using our sweep functionality, potentially leading to better loss minimization and improved model performance. We also could have tried a different optimization method such as gird search.

Finally, we want to explore different normalization methods, as our findings suggest that zero mean normalization might not be the most suitable for our data-set. Alternative methods like maximum absolute scaling could potentially yield more accurate predictions, aligning with recent research in the field.

Through these future efforts, we anticipate advancing our model's predictive accuracy and RMSE, especially for short-term forecasting, while trying to mitigating the risks of over-fitting and other challenges observed in our initial results.

5.4 Conclusion

In conclusion, our exploration of the Informer model for forecasting electricity prices revealed several areas for future improvement. We recognized the limitations of our data-set, particularly in terms of size and scope, and the exclusion of important variables like gas and oil prices. Additionally, we looked at the potential over-fitting issue linked to the model's attention mechanism and the challenges posed by the data-set's multicollinearity and missing data. Moving forward, we plan to expand and diversify/improve our data-set, refine our data handling techniques, and explore

alternative normalization methods. We also aim to adjust our model's parameters and data column selections to better align with our forecasting objectives, ultimately enhancing the model's accuracy in RMSE and reducing over-fitting as that was our main obstacle.

Bibliography

[1] URL: https://www.nordpoolgroup.com/.

[2] URL: https://thredds.met.no/thredds/catalog.html.

[3] URL: https://www.statnett.no/.

[4] URL: https://wandb.ai/site.

[5] Yuhan Bai. "RELU-function and derived function review." In: *SHS Web of Conferences*. Vol. 144. EDP Sciences. 2022, p. 02006.

[6] Nils Bjorck et al. "Understanding batch normalization." In: *Advances in neural information processing systems* 31 (2018).

[7] Tianfeng Chai and Roland R Draxler. "Root mean square error (RMSE) or mean absolute error (MAE)." In: *Geoscientific model development discussions* 7.1 (2014), pp. 1525–1534.

[8] Mwamba Kasongo Dahouda and Inwhee Joe. "A deep-learned embedding technique for categorical features encoding." In: *IEEE Access* 9 (2021), pp. 114381–114391.

[9] Dimitris Dimoudis et al. "Utilizing an adaptive window rolling median methodology for time series anomaly detection." In: *Procedia Computer Science* 217 (2023), pp. 584–593.

[10] Guilin Huang. "Missing data filling method based on linear interpolation and lightgbm." In: *Journal of Physics: Conference Series*. Vol. 1754. 1. IOP Publishing. 2021, p. 012187.

[11] Peter Laurinec et al. "Adaptive time series forecasting of energy consumption using optimized cluster analysis." In: *2016 IEEE 16th international conference on data mining workshops (ICDMW)*. IEEE. 2016, pp. 398–405.

[12] Felipe Tomazelli Lima and Vinicius MA Souza. "A Large Comparison of Normalization Methods on Time Series." In: *Big Data Research* 34 (2023), p. 100407.

[13] Dieuwertje Luitse and Wiebke Denkena. "The great transformer: Examining the role of large language models in the political economy of AI." In: *Big Data & Society* 8.2 (2021), p. 20539517211047734.

[14] Hagit Shatkay. "The Fourier transform-A primer." In: *Brown University* (1995), pp. 1–20.

[15] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. "A comparison of ARIMA and LSTM in forecasting time series." In: *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE. 2018, pp. 1394–1401.

[16] Ashish Vaswani et al. "Attention is all you need." In: *Advances in neural information processing systems* 30 (2017).

[17] Ailing Zeng et al. "Are transformers effective for time series forecasting?" In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 37. 9. 2023, pp. 11121–11128.

[18] Shouli Zhang et al. "A feature extraction method for predictive maintenance with time-lagged correlation–based curve-registration model." In: *International Journal of Network Management* 28.5 (2018), e2025.

[19] Haoyi Zhou et al. "Expanding the prediction capacity in long sequence time-series forecasting." In: *Artificial Intelligence* 318 (2023), p. 103886.

[20] Haoyi Zhou et al. "Informer: Beyond efficient transformer for long sequence time-series forecasting." In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 12. 2021, pp. 11106–11115.

[21] Eric Zivot et al. “Rolling analysis of time series.” In: *Modeling financial time series with S-Plus®* (2003), pp. 299–346.