Windpower Forecasting with Machine Learning

-

Felix Bötte - 4107268

1 Forecasting Wind Power with Neural Networks

In this report, I present my results of a wind power forecasting task, focusing on the utilization of LSTM (Long Short-Term Memory) encoder-decoder networks for capturing the underlying temporal dynamics of time series. The goal is to accurately predict wind power output of at least one turbine per wind farm output for the next step, next hour, and next day. We were provided with data from a British and Braizilian windfarm. To achieve this, I employ feature selection techniques to identify the most important variables and explore various methods of data preparation to enhance the forecasting performance.
Leveraging the power of LSTM-based neural networks, my aim was focus on their ability to capture temporal dependencies and patterns in time series data.
To optimize the performance of my forecasting model, I started with various data preparation techniques in order to make the data suitable for training, including normalization, handling missing values, and managing outliers. Next, I assess the importance of different variables by automated approaches such as correlation analysis, xgboost, and L1 regularization to identify the most influential factors.
For the remaining time of the project I am aiming to explore feature engineering techniques to extract relevant information from the raw time series data, such as deriving wind speed variations or incorporating seasonal factors.

1.1 Code

The following will be referring to my Code on GitHub. In the root of the repository datainspection is used to get an overview of the individual variables of the dataset and their shape. The file **requirements.txt** contains all the packages that need to be installed to run the code.
The Jupyter Notebook **windfarmbritaindataprocessing.ipynb** reads the raw data of one turbine from the British wind farm dataset into a pandas dataframe and imports the **datapreprocessing.py** file, which contains multiple functions for processing the data. Then, the data is split into target data, which contains only the target variable, **Power (kW)** for britain and **windspeed** for brazil, and input data, which includes all variables. Subsequently, both datasets are min-max normalized, and NaN values are removed.
In the next step one can choose the feature selection meachnism ("pearson", ßpearman", "xgboost", "time-lag-corr") and the number of most important features to be selected. Time-lag-corr refers to the correlation between variables when the target dataset is shifted by a certain amount of timesteps.
Finally the notebook saves a new csv files to the folder "preprocessed-data"with only the data of the number of columns that have been identified the most important by the feature selection algorithm. The corresponding file **windfarmbrazildataprocessing.ipynb** does the same as above except of reading the data from an nc file instead of csv.
In the **lstm-training.py** file the newly created csv file with the variables with the highest

correlation for predicting the target variable is read as a pandas dataframe again and further normalized to have a standard derivation of 1. Next, the data is split into train and test set and the input window and ouptut window for the time series prediction is specified. One timesteps in the data is representing 10 minutes, i.e. if the input and output-window is set to 6 the model predicts the target variable one hour into the future with taking 1 hour of previous data into consideration. A function now shapes the data into corresponding time series splits that are used for training the neural network model. In the following, the hyperparameters for the model are set and the model is initialized accordingly. Based on the hyperparameters, the training loop is started. I added the teacher-forcing method to the training loop to help the model learn temporal dependencies by sometimes being assisted with target data during longer prediction horizons. After training the hyperparameters, model state dictionary, optimizer state dictionary and training/test-loss are saved to file for later evaluation and plots.
The file **LSTM-enc-dec.py** contains the neural networks architecture with the respective training loop. Additionally it has a function for preparing the input data to be compatible with time series prediction.
The file **utility-funcions.py** contains two minor functions for loading text as json and appending two pandas dataframes.
The file **plot-saved-model.py** loads a saved model and preprocessed data from file and plots the time series of the first variable, representing the target variable, of training and test data. Additionally it creates two plots showing the convergence of the training and test loss over the specified rang eof epochs. Hereby the file imports **plots.py** which contains the individual functions for the respective plots.
The file **testing-britain.py** also loads a saved model and a preprocessed dataset which is not the dataset the model was trained on and evaluates the already trained model on a not before seen dataset.
The file **transfer-learning.ipynb** again loads a saved model and preprocessed dataset and trains the model further on a not before seen dataset to increase the models capability to generalize using the concept of transfer learning.
The file **global-model-approach.ipynb** just appends three already preprocessed datasets to one large dataset to increase the variability in the data and maybe increase performance on not before seen datasets.

2 Results

In the following I will give an overview over the results for my chosen models for predicting wind power of the next 10 minutes, 1 hour and 1 day. In order to to prevent this part getting too big and lack of training times for comparing multiple variation of hyperparameters I will present my main results at this point in time with the following hyperparameters:

```
# Setting hyperparameters for training
hidden_size = 256
num_layers = 3
learning_rate = 0.001
num_epochs = 100
input_window = input_window
output_window = output_window
batch_size = 64
training_prediction = "mixed_teacher_forcing"
teacher_forcing_ratio = 0.5
dynamic_tf = True
shuffle = True
loss_type = "L1"
wind_farm = "brazil_time_lag_corr_"
```

Abbildung 1: Hyperparameters of neural network model.

For predicting the next time step I chose an input and output window of one. For predicting the next hour I chose 6 for input and output window. Finally, for predicting the next day I was choosing an input window of 6 and output window of 144 (remembering one timestep representing 10minutes).

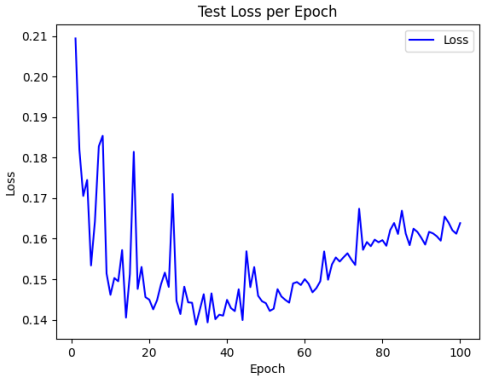

Abbildung 2: Britain: LSTM Test Loss with output window = 1 and input window = 6

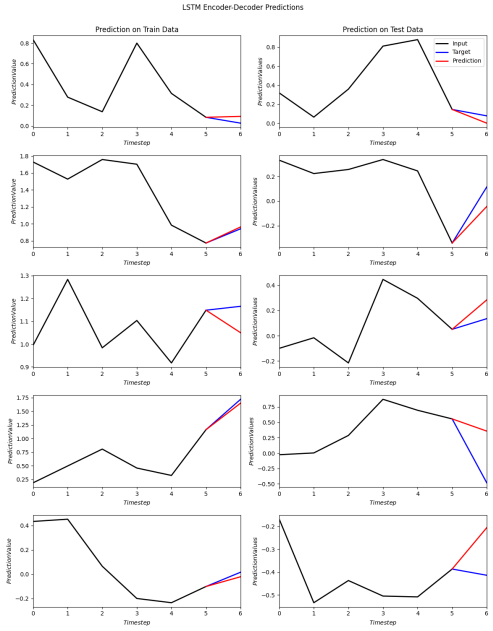

Abbildung 3: Britain: Time Series Prediction with output window = 1 and input window = 6

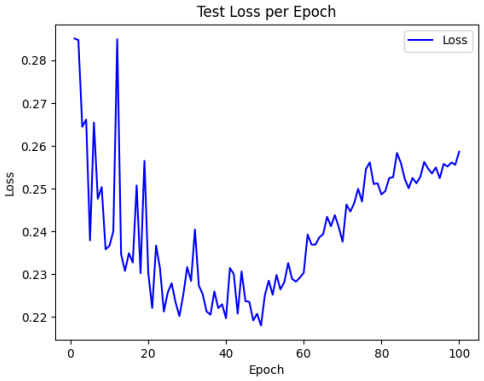

Abbildung 4: Britain: LSTM Test Loss with output/input window = 6

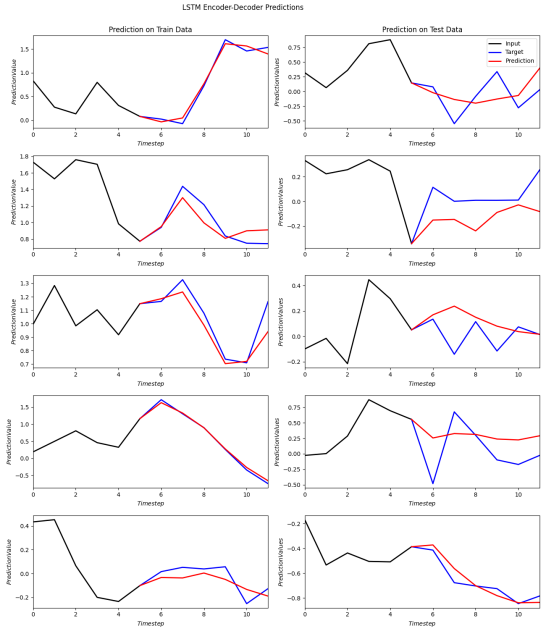

Abbildung 5: Britain: Time Series Prediction with output/input window = 6

The plots for Britain: LSTM Test Loss with output window = 144 and input window = 6, are not available at the moment due to lack of time.

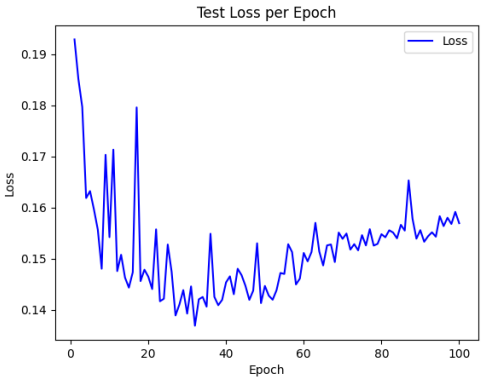

Abbildung 6: Brazil: LSTM Test Loss with output window = 1 and input window = 6

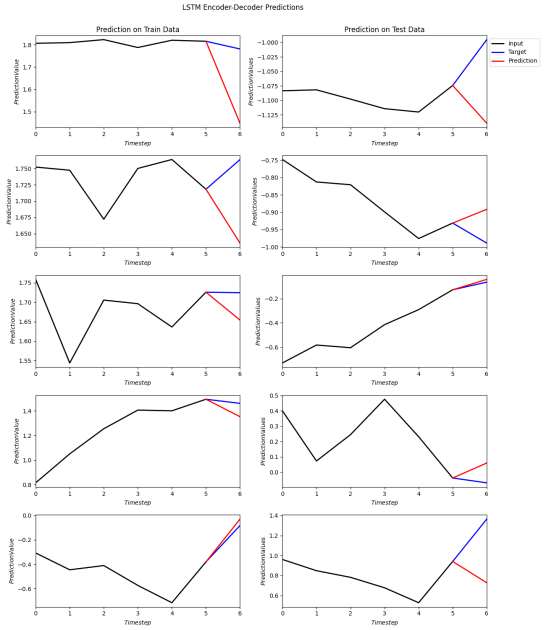

Abbildung 7: Brazil: Time Series Prediction with output window = 1 and input window = 6

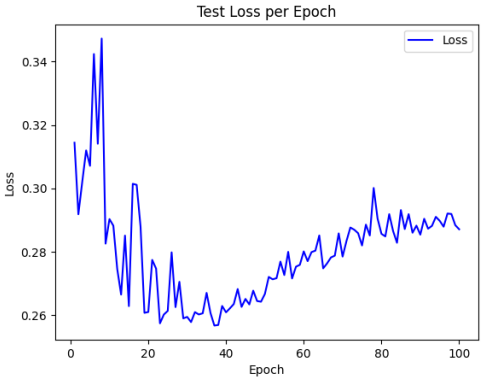

Abbildung 8: Brazil: LSTM Test Loss with output/input window = 6

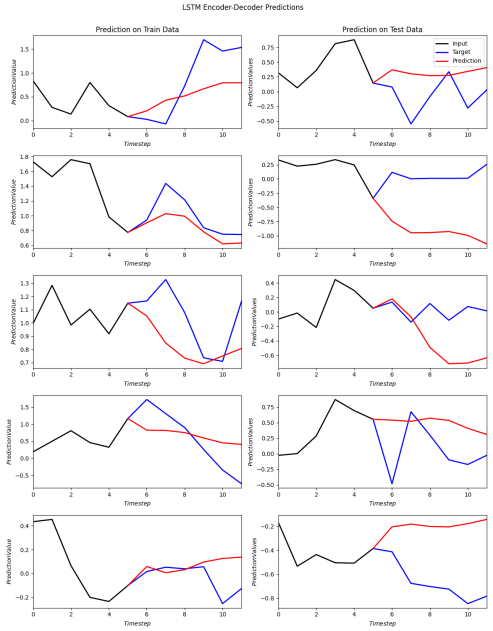

Abbildung 9: Brazil: Time Series Prediction with output/input window = 6

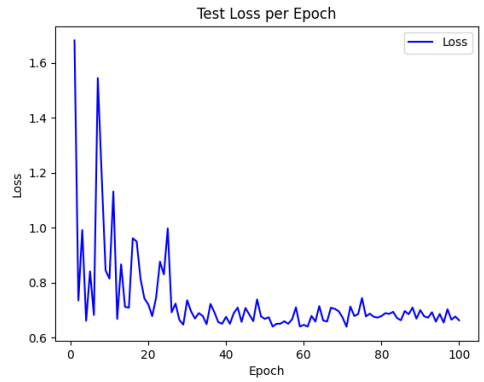

Abbildung 10: Brazil: LSTM Test Loss with output window = 144 and input window = 6

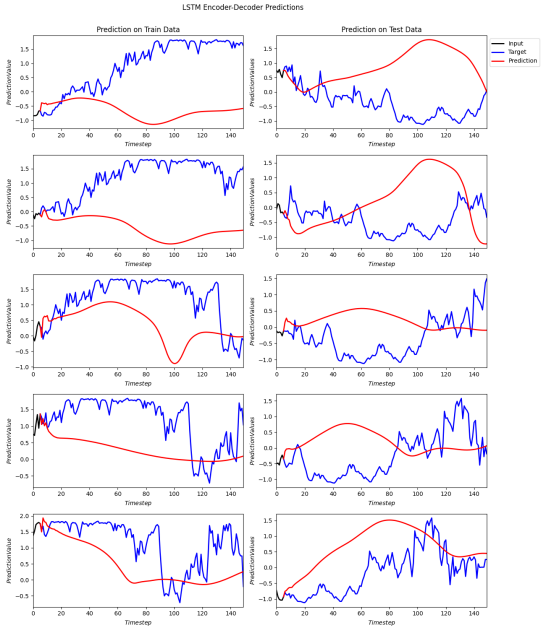

Abbildung 11: Brazil: Time Series Prediction with output window = 144 and input window = 6

For the model below I got better results but I trained it some time before and I feel like I changed something that resulted in worse models in this point in time. Im aiming on correcting this until the final hand in.

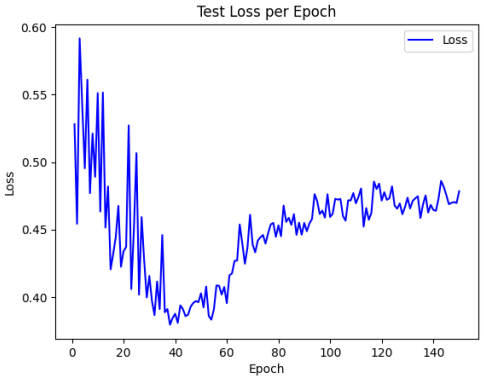

Abbildung 12: Britain: LSTM Test Loss with output/input window = 36

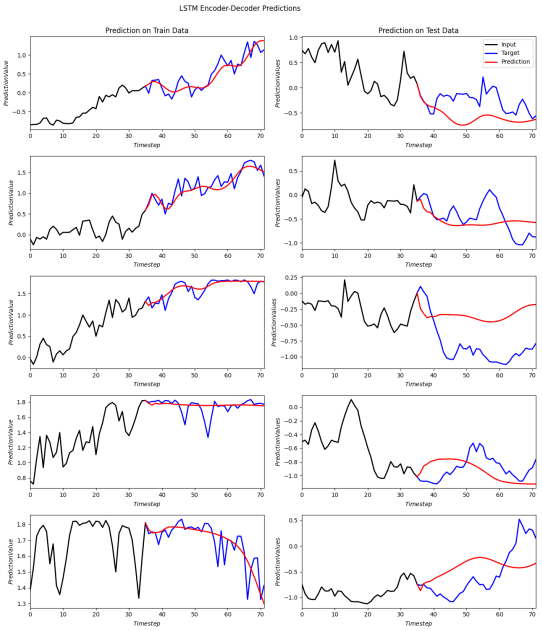

Abbildung 13: Britain: Time Series Prediction with output/input window = 36

Here I trained a model on triple amount of data of the Britain wind farm over multiple years and different turbines to increase variability and training set size:

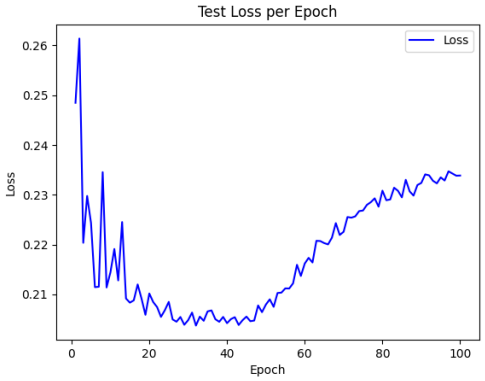

Abbildung 14: Britain: LSTM Test Loss with output/input window = 6 and triple amount of data

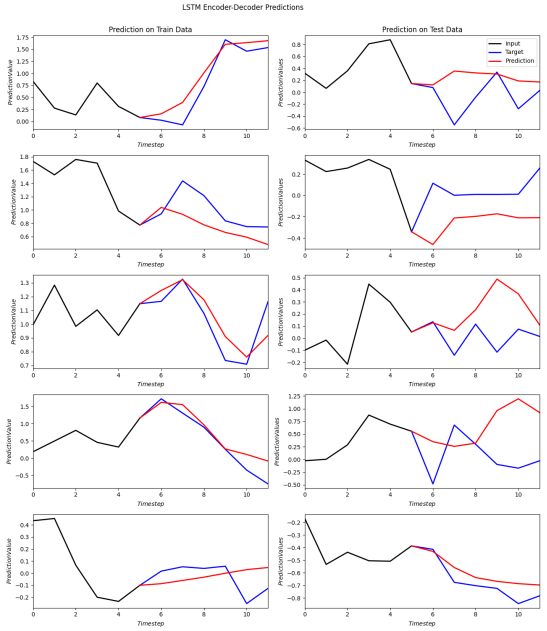

Abbildung 15: Britain: Time Series Prediction with output/input window = 6 and triple amount of data