Politechnika Wrocławska

Energy Consumption Prediction

June 2025

Tymoteusz Biegański

Contents

1 Objective

This project focuses on predicting total hourly energy consumption based on data from individual household appliances. By analysing the contribution of each appliance, the goal is to accurately estimate overall usage and uncover patterns in energy demand. Such insights can support energy efficiency efforts and future consumption forecasting.

2 Methodology

This section entails the steps taken and requirements to run this project.

2.1 Requirements

This project was developed in **Python**, utilising a range of libraries tailored for **data processing**, **modeling**, and **evaluation**. The key dependencies are as follows:

- **Pandas** and **NumPy** — for efficient **data manipulation** and **numerical operations**
- **Matplotlib** — for **visualizing** training performance and prediction results
- **Scikit-learn** — for **preprocessing**, including encoding, scaling, and computing **evaluation metrics**
- **TensorFlow / Keras** — for building and training the **LSTM-based neural network models**
- **OpenPyXL** — for exporting the results to an **Excel spreadsheet**
- **tqdm** — for displaying **progress bars** during the **grid search**

2.2 Data Processing

The dataset was sourced from Kaggle and contains smart meter readings of household appliances sampled every second, along with weather data from the same period. Since working with second-level data would be inefficient and unnecessary for the task, the readings were averaged into hourly intervals. This reduced computation time and gave a

more meaningful view of overall energy use.

Categorical features like weather summary and icon were one-hot encoded. Time-based features such as hour of the day and day of the week were added. A new column, totalPower, was created to sum the power consumption of selected appliances.

All input features and the target column (House overall [kW]) were scaled using MinMaxScaler. The data was split into training, validation, and test sets, then converted into fixed-length sequences of 48 hours for the LSTM model.

2.3 Base Model

The initial model was a straightforward LSTM architecture. It consisted of a single LSTM layer with 64 units, followed by a dense layer that outputs the final prediction. This structure was chosen to establish a simple baseline. The model was trained using the Adam optimizer and mean squared error as the loss function.

Despite its simplicity, the base model performed decently. On the validation set, it achieved a Mean Absolute Error (MAE) of 0.2553 and an R^2 score of 0.1831. On the test set, the MAE was 0.3523, and R^2 dropped to 0.0903, showing moderate generalisation. The training and validation loss curves are shown in Figure 1.

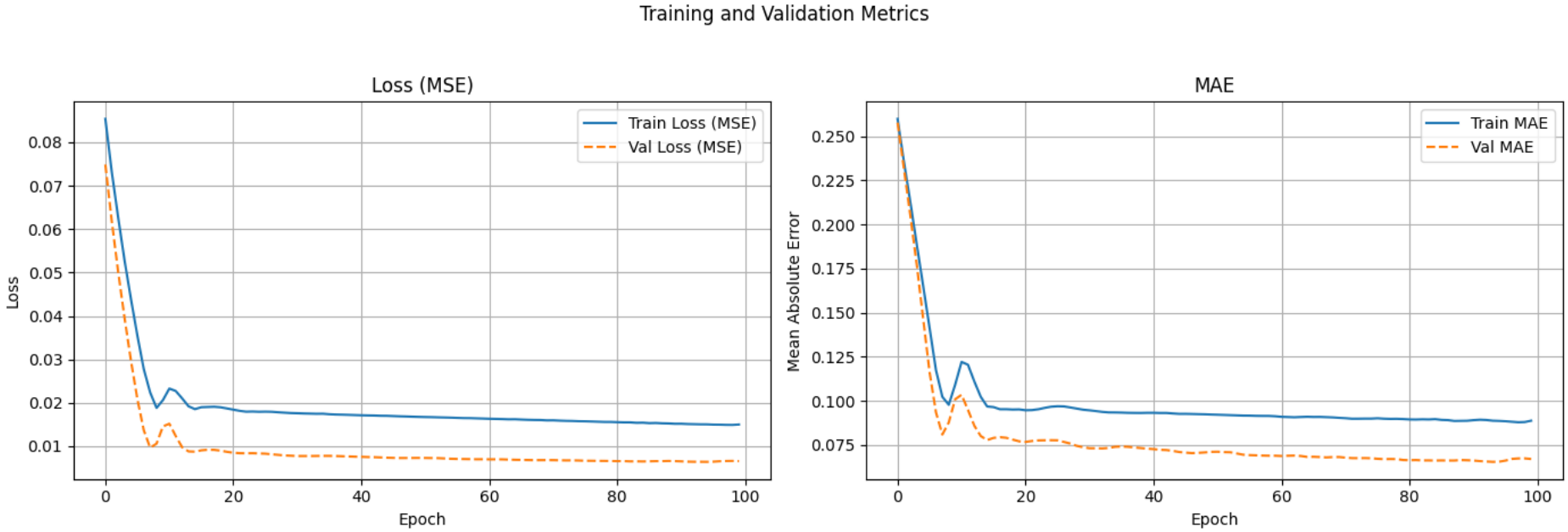

Figure 1: Base LSTM Model: Training and Validation Loss / MAE

2.4 Advanced Model and Grid Search

To improve performance, a more flexible and deeper architecture was explored using grid search. This included testing variations with one or two LSTM layers, optional bidirectional connections, different activation functions (‘tanh‘, ‘relu‘), dropout, and dense layers of different sizes.

The search revealed that deeper and bidirectional models generally improved accuracy, though at the cost of longer training time. Interestingly, some smaller models delivered competitive results with significantly less computational overhead.

Validation and test performance improved across the board compared to the base model. Figure 2 shows actual vs predicted values on the validation set. Figure 3 displays training loss and MAE over epochs for one of the top configurations.

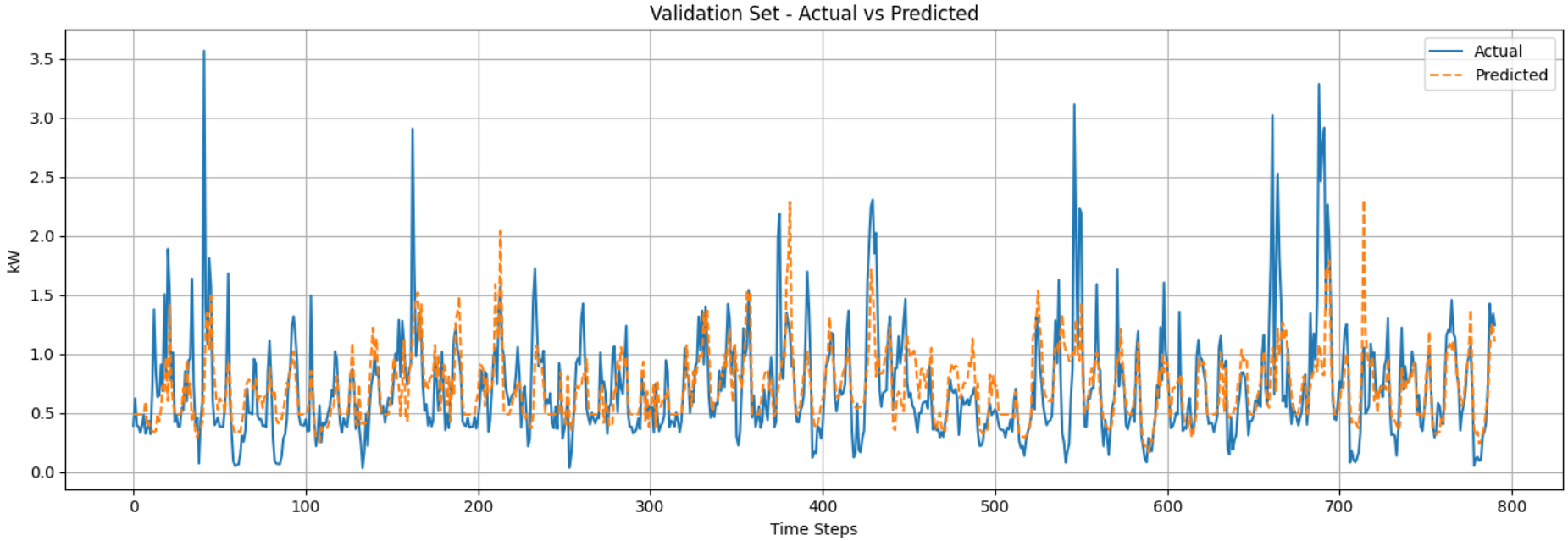

Figure 2: Advanced LSTM: Validation Set - Actual vs Predicted

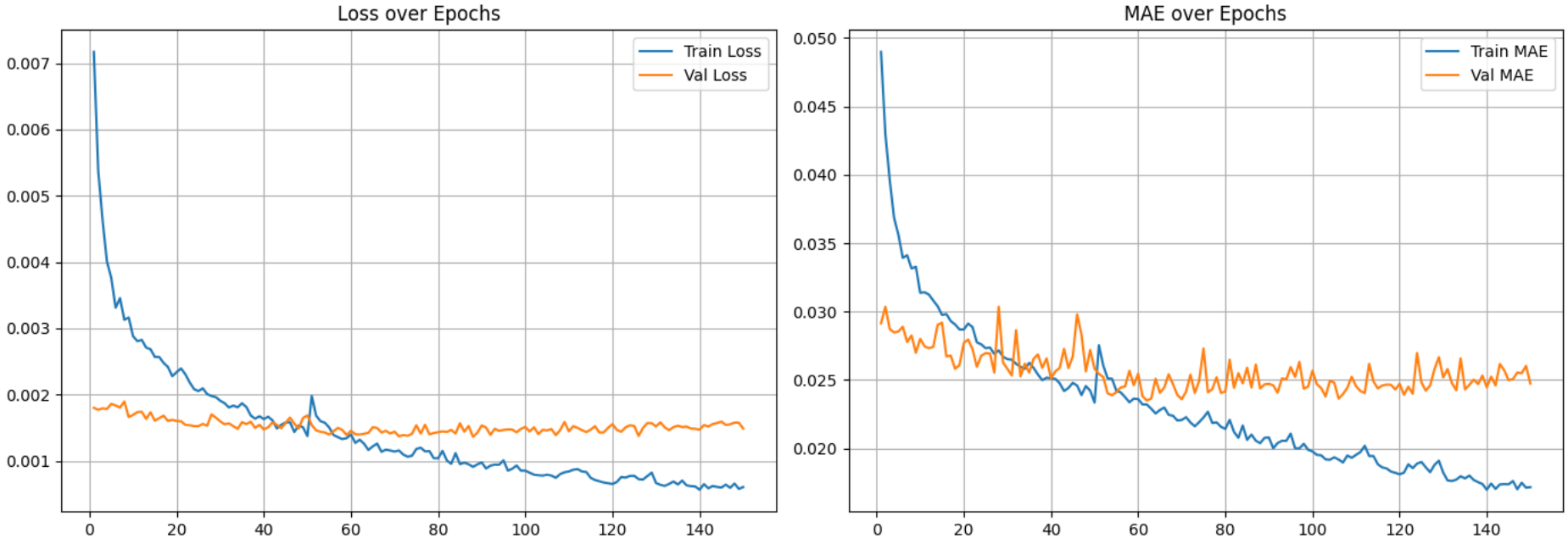

Figure 3: Advanced LSTM: Loss and MAE Over Epochs

The training and validation loss and MAE curves in Figure 1 indicate that the base LSTM model learns effectively, with both metrics decreasing steadily and showing minimal overfitting. In Figure 2, the predicted values from the advanced LSTM model closely track the actual values on the validation set, suggesting good generalization capability and accurate temporal modelling. Finally, Figure 3 shows a smooth decline in both loss and MAE over epochs for the advanced model, confirming that it not only trains stably but also maintains consistent performance improvement across iterations.

The following table presents the results of an extensive grid search conducted to optimise the hyperparameters of the LSTM models. Each row corresponds to a unique configuration, varying in parameters such as the number of LSTM layers, activation functions, optimizer, batch size, dropout rate, and dense layer structures. Evaluation metrics including MAE, MSE, RMSE, R^2, MAPE, and MBE were used to assess performance. The grid search was essential for identifying the most effective model architecture and training configuration, balancing both prediction accuracy and generalisation. It highlights the trade-offs between training depth, complexity, and overfitting risk. Notably, some configurations showed excellent MAE and R^2 scores with low MAPE, while others indicated instability or overfitting—emphasising the importance of thorough hyperparameter tuning.

Table 1: Full Grid Search Results with Metrics and Time per Run

Units	Layers	BiDir	Dropout	Activation	Dense	Opt.	Batch	Epochs	MAE	MSE	RMSE	R^2	MAPE	Time
64	1	True	0.2	tanh	[64, 32]	adam	32	50	0.2568	0.1781	0.4220	0.1564	43.10	4m 29s
64	1	True	0.2	tanh	[64, 32]	adam	32	100	0.2618	0.1566	0.3957	0.2584	50.93	8m 40s
64	1	True	0.2	tanh	[64, 32]	adam	32	150	0.2564	0.1603	0.4004	0.2407	48.26	13m 3s
64	1	True	0.2	tanh	[64, 32]	adam	64	50	0.2459	0.1381	0.3716	0.3460	50.97	2m 15s
64	1	True	0.2	tanh	[64, 32]	adam	64	100	0.2469	0.1461	0.3823	0.3079	44.64	4m 23s
64	1	True	0.2	tanh	[64, 32]	adam	64	150	0.2400	0.1465	0.3827	0.3063	48.90	6m 33s
64	1	True	0.2	relu	[64, 32]	adam	32	50	0.2476	0.1507	0.3882	0.2864	48.01	2m 29s
64	1	True	0.2	relu	[64, 32]	adam	32	100	0.2470	0.1500	0.3873	0.2895	53.52	4m 42s
64	1	True	0.2	relu	[64, 32]	adam	32	150	0.2461	0.1473	0.3837	0.3026	49.83	6m 42s
64	1	True	0.2	relu	[64, 32]	adam	64	50	0.2384	0.1380	0.3715	0.3462	49.86	7m 0s
64	1	True	0.2	relu	[64, 32]	adam	64	100	0.2373	0.1390	0.3728	0.3417	48.14	1m 21s
64	1	True	0.2	relu	[64, 32]	adam	64	150	0.2370	0.1364	0.3694	0.3538	49.47	2m 20s
64	2	True	0.2	tanh	[64, 32]	adam	32	50	0.2494	0.1506	0.3881	0.2867	50.27	3m 23s
64	2	True	0.2	tanh	[64, 32]	adam	32	100	0.2526	0.1461	0.3822	0.3081	50.03	7m 6s
64	2	True	0.2	tanh	[64, 32]	adam	32	150	0.2524	0.1486	0.3855	0.2961	45.20	13m 55s
64	2	True	0.2	tanh	[64, 32]	adam	64	50	0.2780	0.1554	0.3942	0.2640	59.83	3m 38s
64	2	True	0.2	tanh	[64, 32]	adam	64	100	0.2587	0.1593	0.3991	0.2457	54.85	7m 14s
64	2	True	0.2	tanh	[64, 32]	adam	64	150	0.2474	0.1516	0.3893	0.2822	48.67	11m 18s
64	2	True	0.2	relu	[64, 32]	adam	32	50	0.3724	0.2293	0.4788	-0.0859	102.95	10m 38s
64	2	True	0.2	relu	[64, 32]	adam	32	100	0.2791	0.1671	0.4088	0.2083	63.79	4m 46s

Continued on next page

Table 1 – continued from previous page

Units	Layers	BiDir	Dropout	Activation	Dense	Opt.	Batch	Epochs	MAE	MSE	RMSE	R^2	MAPE	Time
64	2	True	0.2	relu	[64, 32]	adam	32	150	1.1182	1.3827	1.1759	-5.5488	298.93	9m 9s
64	2	True	0.2	relu	[64, 32]	adam	64	50	0.2458	0.1503	0.3877	0.2879	46.78	13m 46s
64	2	True	0.2	relu	[64, 32]	adam	64	100	0.2894	0.1859	0.4311	0.1196	60.73	2m 18s
64	2	True	0.2	relu	[64, 32]	adam	64	150	0.4155	0.2619	0.5118	-0.2404	119.02	4m 21s

The table shows clear patterns in how the models perform with different hyperparameter settings. The best results came from models with 64 hidden units, one LSTM layer, ReLU activation, dense layers sized [64, 32], and a batch size of 64. One model in particular, trained for 150 epochs, had the lowest MAE (0.23697), lowest RMSE (0.369), and a strong R^2 score (0.354). This suggests it was the most accurate and generalised well to unseen data.

But accuracy is not the only thing that matters. Models with more layers or more epochs often took much longer to train. For example, a model with two layers, batch size of 32, and 150 epochs took over 46 minutes to train. It also did not perform well, which points to overfitting and wasted resources.

Taking both accuracy and training time into account, the best overall model used one LSTM layer, ReLU activation, batch size of 64, and 100 epochs. It reached an MAE of 0.2373, RMSE of 0.3728, R^2 of 0.3417, and trained in under five minutes. This makes it both efficient and reliable, a good choice for real-world use where speed and performance both matter.

3 Conclusions

This project did a fair job at predicting total energy use based on appliance-level data. The model is not perfect, but that is to be expected. Some patterns, like seasonal changes or weather effects, are hard to capture without more context. For example, a furnace might not run in summer, which the model cannot guess without temperature data.

Still, the model did manage to follow the general trends. It caught the spikes and drops well enough to suggest that the core idea is sound. With more time and data, it could be made more accurate. In the future, I would consider adding extra features, trying better model structures, or using smarter ways to tune the parameters. But for now, this is a strong first step and a good base to build on.