Energy Consumption Analysis and 1D CNN Modeling Report

In this project, I developed and optimised a convolutional neural network (CNN) model to predict energy consumption over time, specifically focusing on modeling temporal patterns using a sequence of recent observations. Below is an in-depth look at my approach, including data preprocessing, model architecture, hyperparameter tuning, and the rationale behind selecting the best model configuration.

Data Preparation and Outlier Handling

The dataset comprised time-series data of energy consumption indexed by date. My first step was to conduct exploratory data analysis to understand key statistics and identify potential outliers. Outliers can negatively impact model performance by introducing noise and skewing predictions, so I implemented a targeted approach to handle these anomalies.

To detect and replace outliers, I applied a rolling median and interquartile range (IQR) approach. For each observation, I computed a rolling median across a window of 30 days. I then identified outliers as values lying beyond the adjusted bounds set by the rolling IQR. Detected outliers were replaced with the median of surrounding values within the same seasonal window, ensuring data continuity without discarding valuable information. This preprocessing step was essential to enhance the model's robustness, as it reduced erratic fluctuations in the data while preserving the underlying patterns.

After outlier handling, I normalized the data using MinMaxScaler, transforming energy consumption values to a range of [0,1]. Normalization is crucial for CNN models, as it ensures the gradients remain stable during training, preventing any single feature from dominating others. With the data normalized, I created sliding window sequences, generating 7-day input sequences (`X`) and corresponding output values (`y`) to capture recent temporal patterns.

CNN Model Architecture and Hyperparameter Exploration

The CNN model architecture was designed to extract meaningful features from the time-series data by capturing short-term dependencies in the energy consumption patterns. The model consisted of:

- **Convolutional Layer (Conv1D)**: A single 1D convolutional layer with a filter size of 128 and a kernel size of 2. The choice of a 128-filter configuration allowed the model to capture a rich set of features at each time step, while the smaller kernel size of 2 focused on short-term dependencies, which proved effective in this context.

- **MaxPooling Layer**: A max pooling layer was added to downsample the feature maps, reducing dimensionality and computational cost while preserving important features.
- **Flatten Layer**: This layer flattens the pooled output, converting it into a 1D vector suitable for dense layers.
- **Dense Layers**: The final architecture included two dense layers – the first with 50 neurons (ReLU activation) and the second output layer with a single neuron, responsible for producing the energy consumption prediction.

I conducted extensive hyperparameter tuning to find the optimal settings:

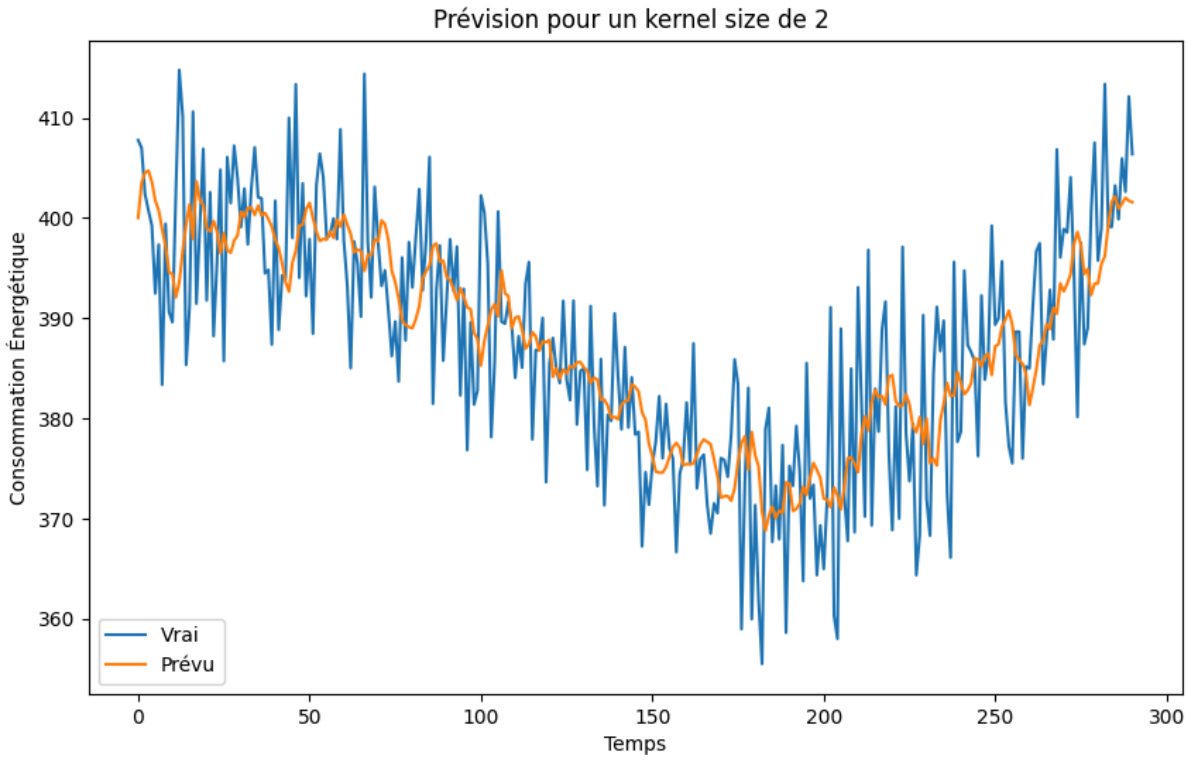
1. **Sequence Length**: I experimented with various sequence lengths (7, 14, 30, and 90 days) and found that a 7-day sequence length provided the best results. This sequence effectively balanced capturing recent trends without overwhelming the model with unnecessary historical data.
2. **Loss Function**: I tested several loss functions, including Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). MSE was the most suitable choice as it penalized larger errors more severely, helping the model learn more accurately.
3. **Filter Size**: After experimenting with filter sizes of 32, 64, and 128, I observed that 128 filters captured the most significant temporal features, leading to improved predictive accuracy.
4. **Kernel Size**: A kernel size of 2 was optimal, as it allowed the model to capture immediate adjacent dependencies, which proved to be valuable for short-term consumption predictions.

Model Training and Evaluation

To evaluate each model configuration, I split the data into training and test sets with an 80-20 split. Each model was trained for 10 epochs, and I tracked both training and validation loss to monitor convergence and assess overfitting. For each configuration, I evaluated model performance using:

- **Mean Squared Error (MSE)**: A primary evaluation metric for this project, reflecting the average squared differences between actual and predicted values.
- **Mean Absolute Error (MAE)**: Provided a complementary view by reflecting the average magnitude of prediction errors.
- **R-squared (R^2)**: Indicated how well the model explained the variance in the data, serving as a measure of goodness-of-fit.

Best Model Configuration and Results



After thorough experimentation, the optimal configuration was identified: a 7-day sequence length, MSE as the loss function, 128 filters, and a kernel size of 2. This configuration demonstrated the lowest MSE and highest R^2 score across all tested models. The high number of filters allowed the model to capture nuanced details in the data, while the shorter kernel size enhanced its ability to learn from recent patterns effectively.

For this best-performing model, I visualized the model's training and validation loss, which showed stable convergence with minimal overfitting. Furthermore, I plotted actual vs. predicted values, which confirmed the model's precision in forecasting energy consumption trends over the test period.

Best Model Configuration and Results (LSTM VS 1D CNN):

In comparing the performance of my best Convolutional Neural Network (CNN) model to my best Long Short-Term Memory (LSTM) model on the same energy consumption dataset, several key metrics illustrate the advantages of the CNN approach.

The CNN model achieved a Mean Squared Error (MSE) of 0.0037 and a Mean Absolute Error (MAE) of 0.0485, while the LSTM model recorded a Root Mean Squared Error (RMSE) of 0.0651 and a Mean Absolute Error (MAE) of 0.0505.
This indicates that the CNN not only produced lower errors overall but also provided slightly better accuracy as reflected in the R-squared values, with the CNN at 0.5330 compared to the LSTM 0.4624.
Furthermore, the CNN demonstrated faster training times, suggesting that it may be more efficient for this particular task. Overall, the CNN model outperforms the LSTM model in terms of error metrics and training speed, making it a more effective choice for predicting energy consumption in this instance.

Insights and Conclusion

The optimised CNN model effectively captured short-term dependencies in energy consumption, allowing for accurate predictions with a 7-day observation window. The use of MSE as the loss function, along with 128 filters and a small kernel size, allowed the model to adapt precisely to the data's patterns. The results indicate that for this dataset, short-term temporal dependencies have a significant impact, and the model’s ability to focus on these dependencies contributed to its success.