Introduction

Our code focuses on time series forecasting using PyTorch, specifically employing a Gated Recurrent Unit (GRU) neural network architecture. The dataset used for this task is the Jena Climate dataset, which contains various meteorological parameters recorded at regular intervals. The primary goal is to predict the temperature (T (degC)) based on historical observations.

Data Preprocessing

The initial steps involve loading the dataset, selecting (Date Time and T (degC)) and using the Facebook Prophet library for initial forecasting for feature extraction. The dataset is then enriched with forecasted features, and significant lag features are introduced based on Partial Autocorrelation Function (PACF) analysis and additional features such as wind components and time-related signals are created.

Exploratory Data Analysis

The data is visualized to understand its distribution and relationships. Histograms, 2D histograms, and time-related signals are plotted to provide insights into the characteristics of the meteorological parameters. Notably, wind velocity features are corrected for outliers, and wind components are introduced.

Fourier Transform Analysis

A Fourier transform is applied to analyze the frequency components present in the temperature time series. This provides insights into the dominant periodicities within the data, which can be crucial for time series forecasting.

Normalization and Dataset Splitting

After the preprocessing and feature extraction phase we ended with 24 features and then the dataset is normalized to improve model convergence and stability. It is then split into training and validation sets for model training and evaluation with a ratio of 80% to 20%

Model Architecture:

The GRU model is implemented using PyTorch and comprises a multi-layered architecture for sequence processing. The core of the model consists of two Gated Recurrent Unit (GRU) layers, designed to capture and analyze sequential patterns within the input data. The GRU layers are followed by fully connected layers that enable the model to make predictions based on the learned representations.

- GRU Layers:
    - The first layer processes sequences of input data, where each sequence has a dimensionality of input_size. The hidden state of this layer consists of 64 units, providing the model with the capacity to capture complex dependencies within the sequential data. Stacking two GRU layers allows the model to learn hierarchical representations and abstract features from the input sequences.
    - The batch_first parameter is set to True, indicating that the input data is organized with the batch size as the first dimension.

- Fully Connected Layers:
    - Following the GRU layers, a fully connected layer (fc1) reduces the dimensionality to 8, introducing non-linearity through a Rectified Linear Unit (ReLU) activation function. This step enables the model to extract essential features from the learned representations.
    - A subsequent linear layer (fc2) transforms the 8-dimensional output to the desired output size. This final layer allows the model to generate predictions based on the processed sequential information.

Hyperparameter Tuning:

The flexibility of the codebase allows for extensive experimentation with hyperparameters, crucial for optimizing the performance of the GRU model.

Learning Rate Exploration:

To understand the impact of learning rates on the model's forecasting accuracy, three different values were experimented with: 0.1, 0.01, and 0.001.

- For a learning rate of 0.001, the total Mean Squared Error (MSE) on the validation set was 0.0071. The training process was completed in approximately 10 minutes.
- Using a learning rate of 0.01 resulted in a slightly lower total MSE on the validation set, measuring 0.0069. Notably, the training time was reduced to 9 minutes.
- However, a learning rate of 0.1 led to a significantly higher total MSE on the validation set, reaching 0.0481. while reducing the training time to 6 minutes.

Optimal Learning Rate Selection:

Considering both the MSE values and training times, the learning rate of 0.01 was selected as the optimal choice. The model achieved competitive performance with a reduced training time compared to other rates. Further hyperparameter tuning was deemed unnecessary, as the chosen configuration yielded satisfactory results for forecasting accuracy without compromising efficiency.

Conclusion:

The findings underscore the significance of our meticulous data preprocessing steps, such as leveraging the Facebook Prophet library, introducing lag features based on PACF analysis. These steps were crucial in enhancing the dataset with forecasted features and relevant time-related signals.

The exploratory data analysis provided practical insights into the distribution and relationships within the meteorological parameters. The Fourier transform analysis proved valuable for identifying dominant periodicities in the temperature time series, directly influencing our feature extraction strategy.

The implemented GRU model architecture, with two layers and fully connected layers, was specifically tailored to capture sequential patterns within our dataset.

The informed selection of a learning rate of 0.01 was a key finding, optimizing performance while maintaining efficiency.

In essence, this task has been emphasizing the critical role of thoughtful preprocessing, exploratory analysis, and hyperparameter tuning in building an effective and efficient time series forecasting model