09.07.2025

Justus Purat & Alexander Kammeyer
Software Project Distributed Systems

# Consumption Data Forecast for HPC Systems

## Wrap-up

M. Ch
A. Er
A. Huth
M. Karn
Y. Kaya
M. Zent

Institute for Computer Science, FU Berlin

# Overview

- Introduction
- Summary of the Sprints
- Presentation of the employed model
- Application overview and experimental observations
- Comment on Lag Llama
- Outlook

# Project background

- Goal: Optimization of HPC clusters by making them more adaptive

  → rescheduling tasks to times where energy prices are low

  → choosing alternative energy sources
- Some datasources have a limited forecast

  → forecast energy prices and co2 emission using a ML model
- Train ML using time-series data from InfluxDB

# Summary of the Sprints

## Sprint 1

- Database connection
- Queried the database to get familiar with the structure and shape
- Analysing the data with python and pandas library
- Creating Charts
- Analysed the results from the Group of the last year

# Summary of the Sprints

## Sprint 2

- Looked for correlations
- Feature Engineering
- Made thoughts about the Model we will use
- worked on
  - Random Forest Regression
  - Prophet
  - Lag Lama

# Summary of the Sprints

## Sprint 3

- Short Sprint
- Worked further on the models, esp. LSTM and Lag Llama

# Summary of the Sprints

## Sprint 4

- Finished the preprocessing part
- Included Max Pooling Layers in LSTM
- Where able to predict 2h into the future
- Started with the Report

# Data Handling

**Feature Engineering:**

- Finalized the Univariate approach.
  - because of the uncertainty of availability of correlated features at the time of forecasting.
  - all other state-of-the-art forecasting models use similar univariate approach.

- Found lagged values of price in the past which correlated highly with current price values.
  - best lagged features were within the 1 week period in the past.

- Created rolling means over the windows of upto 1 week from the past; same as lagged features.
- Also used sine and cosine transformations for time based features to cater for their cyclic nature.

**Preprocessing:**

- After many experimentations, found that 48 hours sequence in the past to predict 1 hour in the future give the best results.
- Training and testing splits (90% training & 10% testing).
- Applied MinMax scalers on features and labels to normalize their values.

# Loss Function

**Drawbacks of mse:**

- Quadratic penalties for large errors.
- Makes the model more conservative by finding a middle ground that minimizes squared error across all data points.
- Symmetry; all errors are treated the same; no domain knowledge can be embedded.

**Custom Loss function:**

- Value of delta controls the behavior of the loss function.
- Quadratic for errors less than delta and linear for errors greater than delta.
- Asymmetric nature - we can customize it according to our domain needs e.g. It is better to underpredict than over predict. Over predictions incur 30% more penalty than under predictions.
- It also gives us more control through adjustment of penalties for peaks and dips.
- Penalty increases for under prediction as y_true (normalized price) gets higher for peaks. Similarly, penalty increase for over prediction if y_true gets lower for downward spikes.

# Model Architecture

**1D Convolution and Max pooling:**

- 1D convolution applies filters over 1 dimensional temporal sequences.
- The filters learn local patterns and features across different time steps within a sequence.
- Filters are sliding windows that detect specific "shapes" in data (e.g., a sudden spike, a dip, a short-term trend reversal).
- Max pooling reduces the size of sequences while retains all the important information.

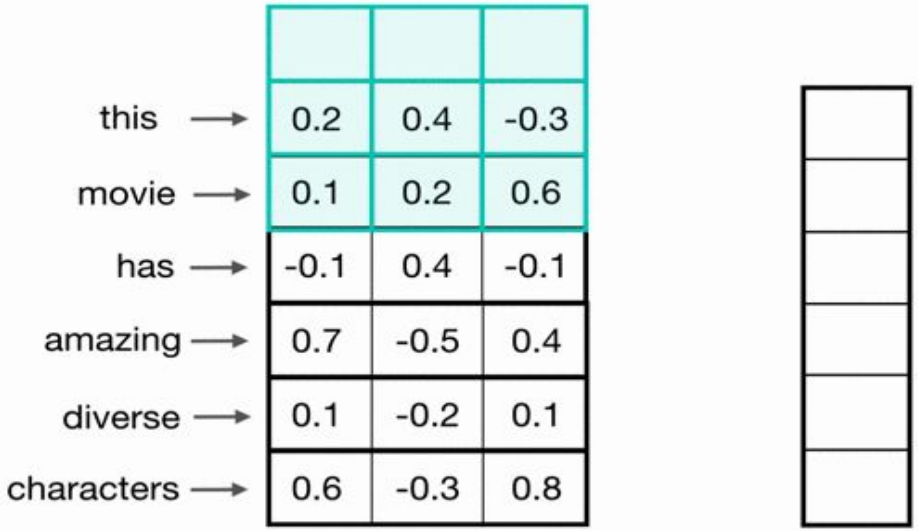
**LSTM:**

- The CNN extracts robust, high-level local features, and then the LSTM processes these learned features sequentially to understand long-term dependencies.
- The combinations allows the LSTM to receive a richer, more abstract representation of the sequence, rather than just raw time series values.

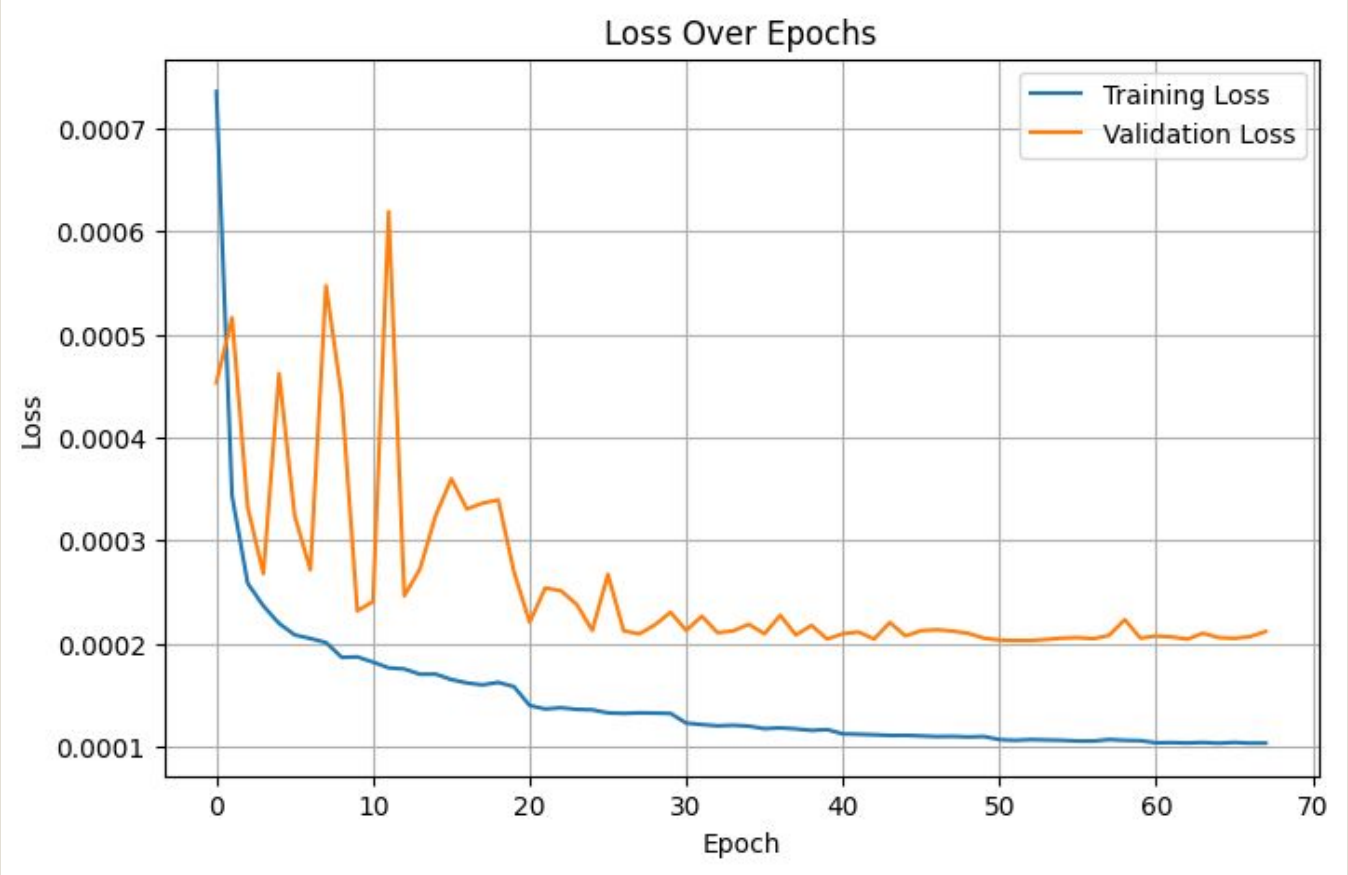
**ANN:**

- The tail of the model is a simple neural networks which gives us our forecast for 1 hour in the future.

# Abstract Representation of 1D Convolution



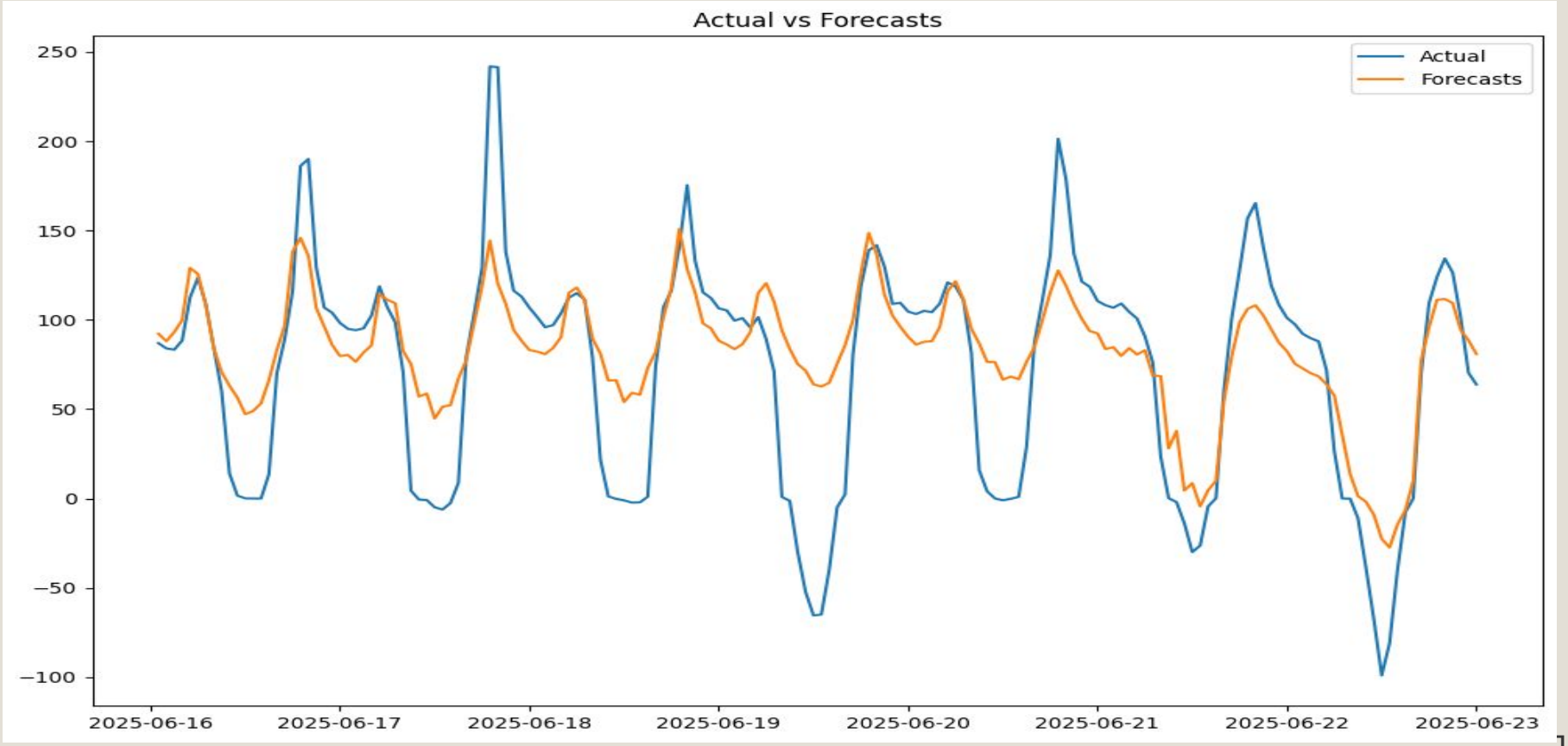
# Training with Custom Loss




**Achieved best results at Epoch 53**

| | |
|---|---|
| loss | 1.0905e-04 |
| mae | 0.0060 |
| mse | 1.0004e-04 |
| val_loss | 2.0292e-04 |
| val_mae | 0.0087 |
| val_mse | 3.1625e-04 |

# Rolling forecasts (Recursively)



# Rolling forecasts (Continued)

**Errors representation:**

- Mean Squared Error: 1767.57
- Mean Absolute Error: 31.02

What this tells is on average a forecasted price value is 31 units away from the actual value.

This however, is the error representation if we forecast for 1 week in the future. This error significantly declines for shorter forecast horizons as Michael will discuss later.

# Application Structure

**ConsumptionData.py**

- Data fetching and handling
- Distance correlation analysis
- Transformation to and from gradient values
- Outlier removal
- Adding lagged and rolling mean features

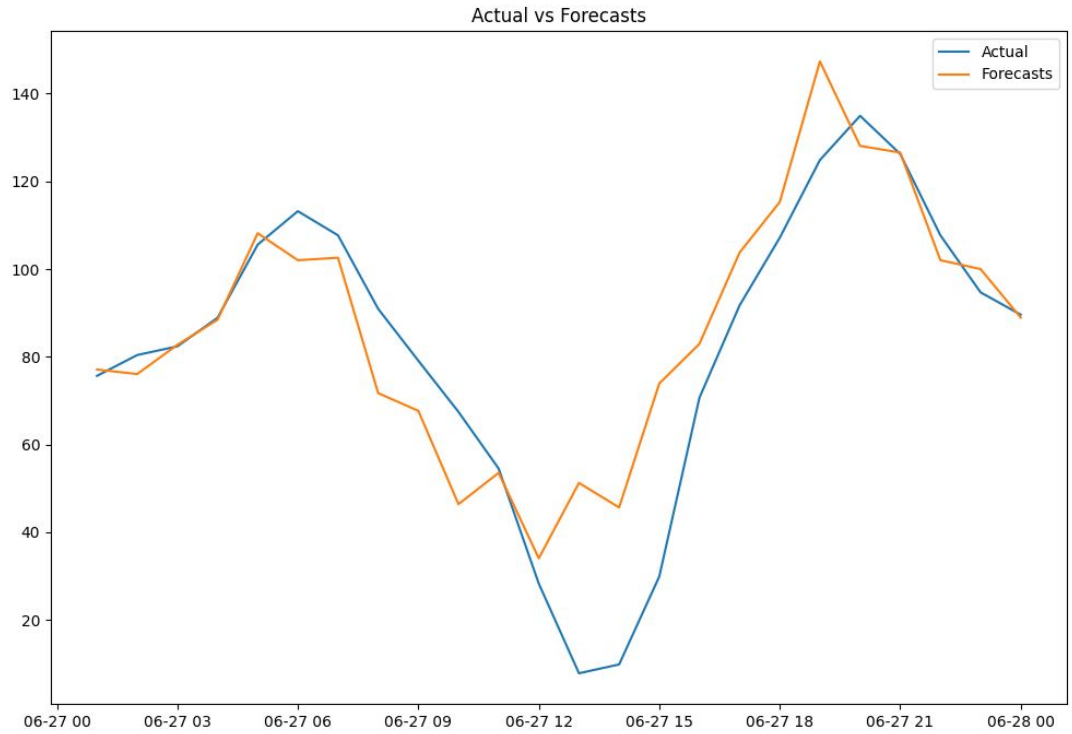
**modelLSTM.py & TwoSidedAsymmetricHuberLoss.py**

- Implement the aforementioned LSTM model, and the employed loss function

**training.py & forecast.py**

- Call the training and forecasting routines after corresponding pre-processing

# Experiments on Forecasting a whole Day




**Training data**
3 years,
ending 31.12.2024

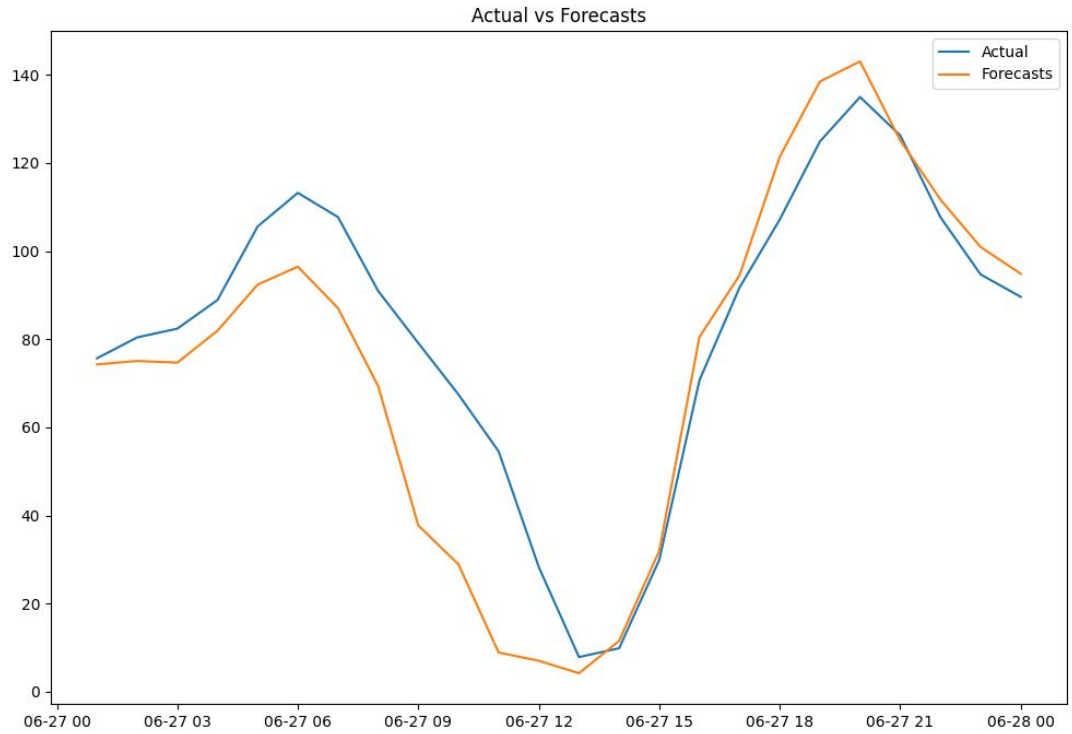
**Absolute values**

**No outlier removal**

**Artificial features**
47 lagged and
rolling-mean features
by dCor > 0.8

**Normalized MAE**
9.2%

# Experiments on Forecasting a whole Day




**Training data**
3 years,
ending 31.05.2025

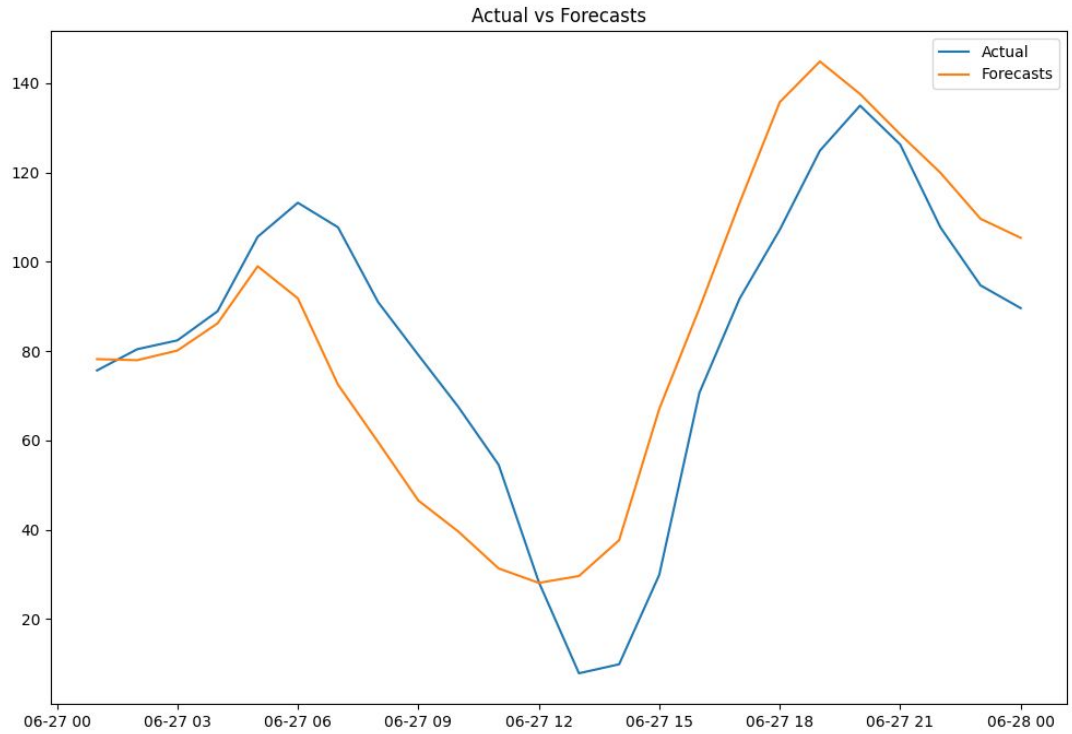
**Absolute values**

**No outlier removal**

**Artificial features**
47 lagged and
rolling-mean features
by dCor > 0.8

**Normalized MAE**
10.3%

# Experiments on Forecasting a whole Day




**Training data**
3 years,
ending 31.12.2024

**Gradient values**

**Outlier removal**
z-score at 3 s

**Artificial features**
39 lagged and
rolling-mean features
by dCor > 0.8

**Normalized MAE**
11.9%

# Observations

- 24h-forecasts achieve a normalized MAE of 9-13% at yearly or monthly retraining
- The closer the training data's end-date to the date for which the forecast has to be done, the worse the normalized MAE is in total, but can become better on certain time sections
- Offset between forecast and actual data of about an hour
- Using gradient data with outlier removal
    - Tends to be more stable, i.e. to preserve the actual curve's form
    - Reduces the number of needed artificial features
    - But performs slightly worse in normalized MAE

# Forecasting with Foundation Model

**Goal:**

- Utilize foundation models in forecasting energy price mix and carbon dioxide emission.

**Fundamental Idea:** A time series foundation model can create forecasts without pretraining, similar to how a large language model (LLM) can output task without being pre-trained on a task.

**Tool: Lag Llama**[1]

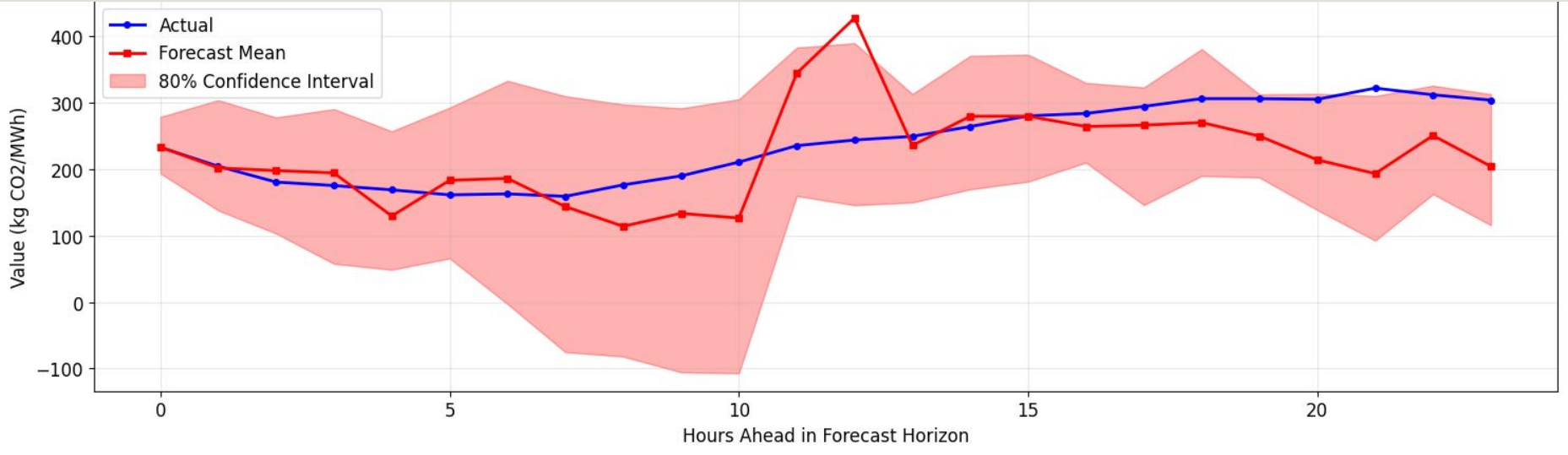
- Trained on 27 time series diverse datasets comprising energy, transportation and economics.
- Training corpus comprised of 7965 univariate time series totalling 352 million tokens - Open Source.

**Why Foundation Model?**

- Represent a new frontier, derived from the success of LLM.
- Powerful generalization capabilities across various tasks and domains, often with minimal or no fine-tuning.
- Achieve scalability and adaptability in time series analysis.

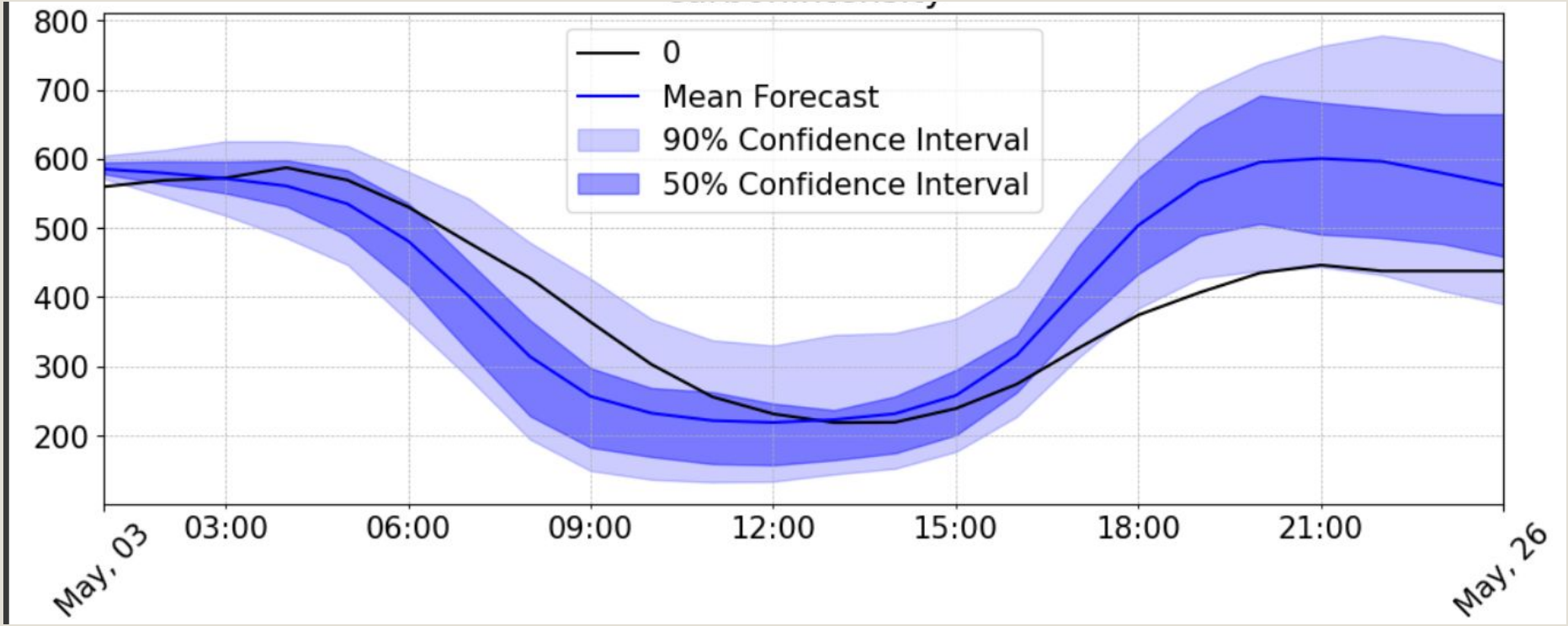
# Zero Shot Forecasting

Forecast for Carbon Intensity (kg CO2/MWh)




| MAE: 49.364 | RMSE: 66.988 | MAPE: 20.4% |
|---|---|---|

# Preliminary Fine Tuning Forecasting




RMSE: 109.520

MAPE: 60.4%

# Observations

**Zero Shot:**

- Model has no idea about the data.
- The model successfully captured the daily pattern (the "shape" of the curve) right away, showing it understands the fundamental drivers of carbon intensity.

**Fine Tuning:**

- Model have been trained for 20 Epochs on the data.
- Performed worse than zero-shot.
- Direction is correct but very high magnitude error.
- Too early for any meaningful prediction.

# Conclusion & Future Remark - Lag Llama

**Conclusion:**

- **The Foundation Model is Powerful**: Lag-Llama's "zero-shot" performance (20% error) was surprisingly accurate and set a very strong baseline. Its prior experience is highly relevant to our data.

- **Initial Fine-Tuning Did Not Beat the Baseline**: Our first attempt at fine-tuning resulted in a higher average error than the zero-shot test. This can happen if the training is too short or if the model struggles to learn multiple, complex patterns at once.

- **Pattern vs. Magnitude**: In both tests, the model did a great job of predicting the *pattern* (the daily ups and downs). The main challenge is improving its accuracy in predicting the *magnitude* (the actual values).

**Future Work**:

- The task could be a challenge for future groups.

# Outlook

- Finalize report

- Make small adjustments to the models, e.g. tweak the selection of the generated artificial features

- Further improving the loss function

- Trying to understand better the behavior of the model at various inputs

# Thank You

# References

[1] K. Rasul et al. (2023), *Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting*, arXiv, arXiv:2310.08278.