LLM4TS

Continuation of last semesters PE

by
Amruth Gadepalli (IMT2022065) and Rajanala Sai Dheeraj (IMT2022093)

Introduction:

This project is a continuation of our previous semester's work on the **LLM4TS** model, inspired by the research paper *LLM4TS: Aligning Pre-Trained LLMs as Data-Efficient Time-Series Forecasters*. The core idea of the paper is to leverage pre-trained large language models (LLMs), such as GPT-2, for time series forecasting tasks by introducing a novel alignment mechanism that integrates token, positional, and temporal encodings. This approach allows LLMs, originally designed for natural language, to generalize effectively across diverse time series data.

In the previous phase of the project, we implemented a basic version of the LLM4TS framework and applied it to **Microsoft stock price data**. While this demonstrated the feasibility of using language models for forecasting, our implementation lacked proper hyperparameter tuning and architectural optimizations. As a result, the performance was sub-optimal, particularly for longer prediction horizons.

Objective:

The primary objective of this semester's project is to extend the previously developed LLM4TS framework by applying it to a more complex and domain-relevant dataset — the **NASA POWER weather dataset**, specifically focusing on the **T2M (Temperature at 2 Meters)** variable for **Bangalore Rural**. This dataset was also used in the STDA course, providing a familiar and realistic use case for time series forecasting.

The goals for this phase of the project are:

- **Dataset Transition**: Replace the earlier financial time series (Microsoft stock data) with the NASA weather dataset to evaluate the model's adaptability to climate data.
- **Model Fine-Tuning**: Improve the forecasting performance by fine-tuning the GPT-2-based LLM4TS model with better hyperparameters, learning rate schedules, and training routines.
- **Pipeline Development**: Build a modular and reusable pipeline that supports data preprocessing, model training, evaluation, and prediction, streamlining future experiments.
- **Web Interface**: Integrate the final model with a simple **website interface** to allow users to interact with the model, input custom date ranges, and visualize forecasts for the T2M variable.

This structured upgrade aims to bridge model performance, usability, and accessibility, making the LLM4TS framework more robust and application-ready.

Model and Training Details

For this phase, we continued to use the **LLM4TS** architecture based on the GPT-2 model, as proposed in the original paper. The model incorporates **token, positional, and temporal encodings** to adapt the transformer architecture for time series forecasting tasks. No architectural changes were made to the model itself; however, significant effort was invested in optimizing its performance through hyperparameter tuning and improved training strategies.

Dataset:

The model was trained and evaluated on the **NASA POWER weather dataset**, focusing on the **T2M (Temperature at 2 Meters)** variable for **Bangalore Rural**. We used a **5-year time period (2019–2024)** to ensure a diverse mix of seasonal patterns and weather fluctuations. The dataset was normalized and split into training, validation, and test sets using a standard temporal split.

Training Configuration

After extensive experimentation, the following hyperparameters were selected as optimal for this dataset:

- **Patch Length**: 10
- **Embedding Dimension (D)**: 768
- **Batch Size**: 64
- **Input Sequence Length (T_in)**: 60-time steps
- **Forecast Horizon (T_out)**: 10-time steps
- **Epochs**: 25
- **Learning Rate**: 2.1e-4

Model Pipeline:

- **Initial Plan**: The model was initially designed to train on the last 5 years of weather data for each input request to ensure dynamic learning.
- **Challenges**: Due to limited GPU resources, this training process was taking 2–3 hours per request, making it impractical for real-time use.

- **Checkpoint Optimization**: To overcome this, we introduced a **checkpointing mechanism**. A checkpoint file was generated using the latest 5-year weather dataset, which was created locally due to download restrictions from Kaggle. This process took approximately 4 hours to prepare because of data processing constraints.
- **Pretrained Weights Usage**: The checkpoint stores pretrained weights, enabling the model to skip training and directly perform predictions, significantly improving response time.

Results

Initially, the model was trained on a fixed dataset — Bangalore Rural T2M values from **2019 to 2024** — which allowed for controlled experimentation and parameter tuning. Through multiple iterations of fine-tuning, we were able to **bring down the Mean Squared Error (MSE) from 0.82 to 0.27**, showcasing a strong improvement in the model's ability to learn temporal patterns in the data.

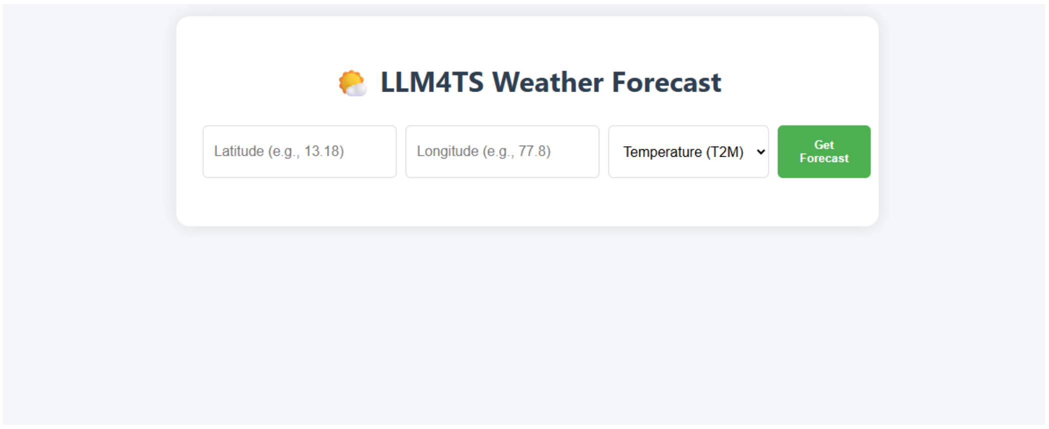
However, in an effort to generalize the approach and **make the pipeline data-agnostic**, we transitioned from static .csv files to a dynamic data pipeline. This pipeline **automatically fetches the last 5 years of data from the current date** via API and processes it into a usable data frame, thereby eliminating the need for manual data downloads or hardcoded files.

After switching to this generalized pipeline, we observed an **increase in the MSE to around 0.52**. Despite extensive further fine-tuning, the model performance consistently plateaued at this value. This led us to conclude that the **MSE of ~0.52 represents a realistic convergence point** given the nature and variability of the incoming data. Interestingly, this value aligns with the MSE range reported in the original LLM4TS paper, reinforcing our understanding that this behaviour may reflect an **inherent property of the dataset and the model's generalization capability**.

Further optimization techniques such as ensemble, better seasonal decomposition, or architectural enhancements remain unexplored and are planned as part of future work.

Web Interface and Model Overview:

Component	Description
Purpose	Web application to forecast 10-day weather data (e.g., Temperature- T2M) for user-specified latitude and longitude using a GPT2-based model.
Frontend	HTML/CSS/JavaScript interface with real-time progress updates and forecast visualization (Chart.js).
Backend	Flask server with endpoints for prediction, progress tracking, and result retrieval.
Model Pipeline	Fetches 5 years of weather data from NASA POWER API, normalizes it, and uses a pretrained GPT2-based model for forecasting.
Model Features	Incorporates token, positional, and temporal encodings to model sequential weather data.
Output	10-day forecast of the selected weather parameter, displayed in a table and chart.



Challenges:

1. **Training Without GPUs**: The initial model training, conducted without GPUs, took at least 20 minutes per session. This was resolved after transitioning to GPU-based training, which significantly reduced the training time.
2. **Model Optimization for Pipelining**: Initially, optimizing the model for dynamic data led to an issue where the error converged to 0.52. Upon further analysis, it was found that pipelining was a key area to improve for better results.
3. **Checkpoint File Generation**: Training the model to generate a checkpoint file, which was a necessary step to store pretrained weights, took several hours to complete due to the dataset's size and processing constraints.
4. **Other Challenges**: Several unforeseen difficulties arose, including managing large datasets locally, balancing memory and processing constraints, and optimizing model performance under dynamic conditions.

Further Scope

1. **Checkpoint Files for Additional Parameters**: Expanding the checkpoint system to include other critical parameters like WHM, T2M_min, T2M_max, and others will improve the model's ability to handle various weather variables.
2. **Improving MSE Convergence**: Further optimization efforts will focus on improving the convergence of the Mean Squared Error (MSE) to achieve better prediction accuracy.
3. **Asynchronous Checkpoint System**: Implementing an asynchronous system that updates checkpoint files at regular intervals will enhance the efficiency of model training and reduce the need for manual intervention.
4. **Generalizing for Other Datasets**: Adapting the current pipeline to work with diverse datasets will increase the model's flexibility and applicability to a broader range of use cases.

5. **Additional Enhancements**: Investigating techniques to reduce memory consumption and increase training speed, as well as experimenting with alternative algorithms, could lead to further improvements in the model's performance.