Deep Learning Coursework Assignment

Low-Rank Adaptation of Large Language Models for Time Series Forecasting

Laura Just Fung (lj441)

April 7, 2025

Word count: 2196

1 Introduction

Time series forecasting is a challenging problem in machine learning when compared to other tasks such as image classification or natural language processing. This is because time series datasets are often comprised of incomplete data from different sources. Additionally, such problems often involve extrapolation from very small amounts of information [1].

This report demonstrates that by encoding time series data as a string of digits, time series forecasting can be directly translated to next-token prediction in text. Thus, larga language models (LLMs) can be repurposed to this task fairly naturally, as shown by Gruver et al., who showed that LLMs are rather capable zero-shot time series forecasters. In this report, the Qwen2.5-0.5B-Instruct model [2] along with Low-Rank Adaptation (LoRA) of the q and v projection matrices are used to explore this concept further and demonstrate performance of a LLM that has been fine-tuned towards this task.

2 Compute constraints

A budget of 10^{17} floating point operations (FLOPS) was applied to this coursework, including all reported experiments as well as the final training run. The FLOPS accounting for primitives are defined in Table 1. The extrapolated FLOPS for other operations are shown in Table 2.

Operations	FLOPS
Addition/Subtraction/Negation	1
Multiplication/Division/Inverse	1
ReLU/Absolute value	1
Exponentiation/Logarithm	10
Sine/Cosine/Square root	10

Table 1: Standardised FLOPS for common primitive operations as defined for the rest of this report.

While Qwen2.5-0.5B-Instruct uses SwiGLU activations as well as Grouped Query Attention, the FLOPS for SiLU activations as well as standard multi-headed attention have

Operation	FLOPS
$M_{m\times n} \times M_{n\times p}$ matrix multiplication	$mp(2n-1)$
SiLU($M_{m\times n}$)	$13mn$
RMSNorm($M_{m\times n}$)	$m(5n+10)$
Softmax($M_{m\times m}$)	$m(12m-1)$
Embedding RoPE($M_{m\times n}$)	mn
Rank r LoRA($M_{m\times m}$)	$2rm^2$

Table 2: Extrapolated FLOPS for operations done whilst training and validating the LLM with LoRA.

been used to simplify the calculations. For similar reasons, it has been assumed that the FLOPS of backpropagation are exactly twice that of the forward pass. Additionally, for all reported FLOPS, any operations performed outside of the model are not included. Finally, the cost of computing the rotary positional encodings have been ignored.

Using Tables 1 and 2, a Python script to calculate the FLOPS of the model's hyperparameters and input size was created. The script is located in `src/flops.py` and is used to calculate the FLOPS for the model's training and inference

3 Preprocessing

A text-based numeric encoding method adapted from the time series data preprocesing scheme described by Gruver et al. was implemented for all data given to Qwen2.5-0.5B-Instruct. This preprocessing ensures that the numeric sequences are appropriately formatted for the best performance of Qwen2.5-0.5B-Instruct's tokenizer and forecasting.

First, as the numeric values in the dataset may vary, to standardise the numeric range of the data and control the token length, a simple scaling was applied, as shown in Eq. 1

$$x_t' = \frac{x_t}{\alpha}, \tag{1}$$

where α is chosen based on the distribution of the dataset.

Additionally, the scaled numeric values were rounded to a fixed number of decimal places to ensure uniformity and consistent representation when tokenized.

The data used for this coursework involves multivariate Lotka-Volterra time series data. Thus, the following encoding is used, as shown in Eq. 2:

```python
def scale_and_encode(prey, predator, alpha, decimals):
    """
    Scale and encode the prey and predator data.

    Args:
        prey (np.ndarray): The prey values.
        predator (np.ndarray): The predator values.
        alpha (float): Scaling factor.
        decimals (int): Number of decimal places for scaling.
    Returns:
        encoded (str): The encoded data string.
    """
    prey = np.array(prey)
    predator = np.array(predator)
    data = np.stack([prey, predator], axis=-1)
    rescaled = data/alpha * 10
    rescaled = np.round(rescaled, decimals=decimals)
    series = np.column_stack((rescaled[:, 0], rescaled[:, 1]))
    encoded = ';'.join([','.join(map(str, row)) for row in series])
    return encoded
```

Listing 1: Function to preprocess the time series data. The function takes in the prey and predator values, the scaling factor α, and the number of decimal places to round to. It returns the encoded string representation of the time series data.

$$\begin{pmatrix} P_0 & Q_0 \\ P_1 & Q_1 \\ \vdots & \vdots \\ P_t & Q_t \end{pmatrix} \rightarrow P_0, Q_0; P_1, Q_1; \ldots; P_t, Q_t, \tag{2}$$

where P_t refers to the prey value at time t and Q_t refers to the predator value at time t. The comma is used to separate the predator and prey values, while the semicolon is used to separate the time steps.

The code for this preprocessing is shown in Listing 1, and this function can be found in the `preprocessor.py` file in the `src` directory.

As an example for the results for this preprocessing implementation with $\alpha = 5$ and 3 decimal places, the time series is encoded as shown in Table. 3, with trailing zeros dropped.

A second example is shown in Table 4.

Stage	Value
Raw data	$\begin{pmatrix} 1.1335121 & 1.1031258 \\ 0.55542254 & 1.2579137 \end{pmatrix}$
Scaled data	$\begin{pmatrix} 2.267 & 2.206 \\ 1.111 & 2.516 \end{pmatrix}$
Encoded data	2.267,2.206;1.111,2.516
Tokenized data	[17,13,17,21,22,11,17,13,17,15,21,26,16,13,16,16,16,11,17,13,20,16]

Table 3: Example of the preprocessing steps for two steps in the time series data. The raw data is scaled, encoded, and then tokenized.

Stage	Value
Raw data	$\begin{pmatrix} 0.8521567 & 0.9479834 \\ 0.6482769 & 0.94272685 \end{pmatrix}$
Scaled data	$\begin{pmatrix} 2.267 & 2.206 \\ 1.111 & 2.516 \end{pmatrix}$
Encoded data	1.704,1.896;1.297,1.885
Tokenized data	[16,13,22,15,19,11,16,13,23,24,21,26,16,13,17,24,22,11,16,13,23,23,20]

Table 4: Another example of the preprocessing steps for two steps in the time series data. The raw data is scaled, encoded, and then tokenized.

4 Baseline

Using the preprocessing method described in Section 3, the untrained Qwen2.5-0.5B-Instruct model was evaluated on the tokenized Lotka-Volterra time series data. The model was given the first 80 points in the time-series data and asked to predict the next 20 points, as shown in Fig. 1. The model was evaluated using mean absolute error (MAE), mean squared error (MSE), the R2 score, and running mean squared error (RMSE) for both the predator and prey time series data. The metrics for system id 870 are shown in Table 5 and Fig. 2.

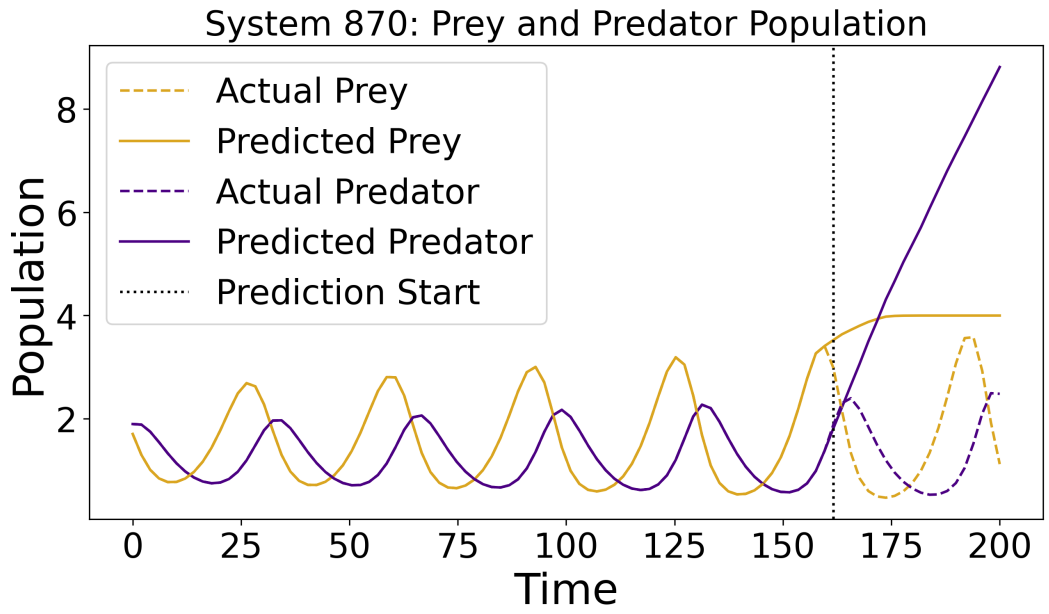

Figure 1: Predictions of the untrained Qwen2.5-0.5B-Instruct model on system 870 of the Lotka-Volterra time series data. The model was given the first 80 points in the time-series data and asked to predict the next 20 points for both the predator (purple) and prey (gold) time series. The model's predictions are shown in the solid lines, while the true values are shown in the dashed lines.

As can be seen in Fig. 1, the untrained model is unable to predict the next 20 points in the time series data to any degree of accuracy. This is reflected in the metrics shown in Table 5, where the MAE and MSE for both predator and prey time series are very high, and the R2 score is either very low or negative.

	Overall		Out of distribution only	
Metric	Prey	Predator	Prey	Predator
MAE	0.5	0.814	2.249	4.072
MSE	1.245	4.383	6.226	21.917
R2 score	0.145	-0.106	-330.734	-3.919

Table 5: Metrics for the untrained Qwen2.5-0.5B-Instruct model performance on system 870 of the Lotka-Volterra time series data. The model was given the first 80 points in the time-series data and asked to predict the next 20 points. The metrics are calculated for both the overall time series and the out-of-distribution data only.

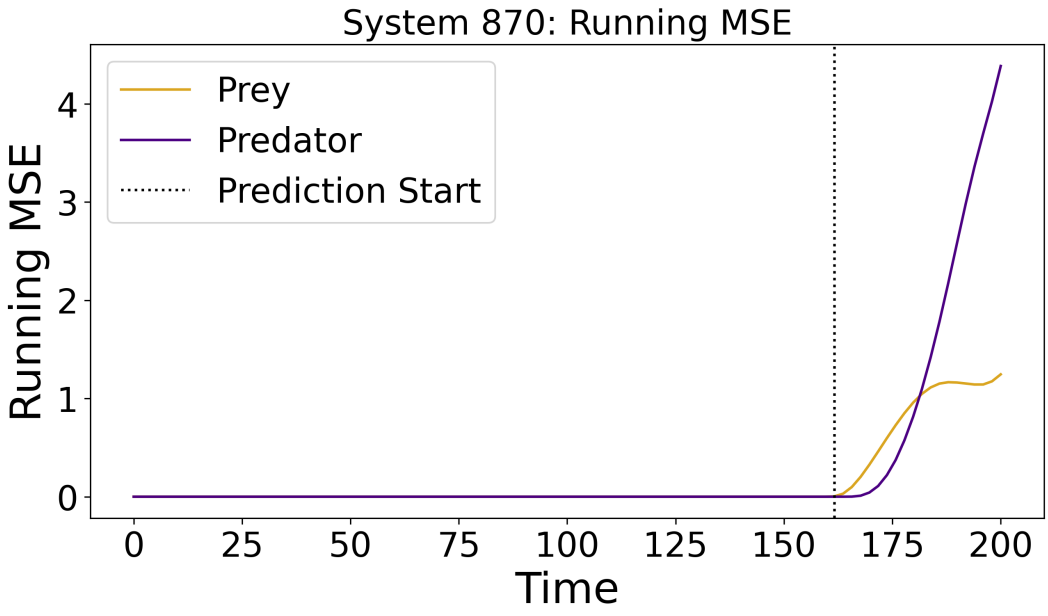

Figure 2: Running mean squared error (RMSE) for the untrained Qwen2.5-0.5B-Instruct model on system 870 of the Lotka-Volterra time series data. The model was given the first 80 points in the time-series data and asked to predict the next 20 points for both the predator (purple) and prey (gold) time series.

References

[1] Gruver N, Finzi M, Qiu S, Wilson AG, Large Language Models Are Zero-Shot Time Series Forecasters; 2024. https://arxiv.org/abs/2310.07820.

[2] Yang A, Yang B, Hui B, Zheng B, Yu B, Zhou C, et al., Qwen2 Technical Report; 2024. https://arxiv.org/abs/2407.10671.

A Use of auto-generation tools

Auto-generation tools were used to help parse error messages throughout the project, and to help format this LaTeX report.

Auto-generation tools were not used elsewhere, for code generation, writing, or otherwise.