Alphathon 2024 Submission: Training a Time-Series LLM for Portfolio Optimization with Parameter-Efficient Online Learning

Ryan Engel, Mario Xerri, and Michael Gannon

Stony Brook University

ryan.m.engel@stonybrook.edu
mario.xerri@stonybrook.edu
michael.c.gannon@stonybrook.edu

Abstract

In this submission for the SQA 2024 Alphathon, we chose to work on the LLM problem sponsored by AllianceBernstein. We address this problem by formulating it as a long-term portfolio optimization task with monthly rebalancing. Our approach was to develop an automated portfolio manager trained to predict optimal stock weights based on technical analysis indicators data. We leverage the Chronos-T5 Large Language Model (LLM), originally designed for time-series forecasting, and adapt it for portfolio generation. By exploiting its pre-trained transformer architecture and self-attention mechanisms, we trained the model to process diverse technical analysis indicators data and make optimized investment decisions. Fine-tuning with Low-Rank Adaptation (LoRA), and following a similar approach to the Parameter-Efficient Reinforcement Learning (PERL) framework, we implement an online learning algorithm to train the model with feedback from its actions each month. We evaluate our strategy in the QuantConnect backtesting platform, where we train the model as if it were a portfolio manager in real time. Over a 10-year backtest, our model demonstrates strong performance relative to the S&P 500 benchmark, generating alpha with effective risk management. An additional out-of-sample backtest, spanning over 15 years, further validated these results.

1 Introduction

Large Language Models (LLMs) have gathered significant attention for their ability to interpret vast amounts of data and generate meaningful predictions. However, their applications in finance remain an emerging field [Yang et al., 2023, Yu et al., 2023, Nie et al., 2024, Ding et al., 2024, Zhang et al., 2024].

Recent studies show promising results with domain-specific LLM fine-tuning [Lu et al., 2024, Zheng et al., 2024]. However, LLMs are computationally expensive to train and fine-tune, making them impractical in many real-world applications. To address these limitations, researchers have explored solutions such as Parameter-Efficient Fine-Tuning (PEFT) [Mangrulkar et al., 2022] and Low-Rank Adaptation (LoRA) [Hu et al., 2021]. Furthermore, these frameworks have been extended to reinforcement learning with PERL (Parameter-Efficient Reinforcement Learning) [Sidahmed et al., 2024].

To the best of our knowledge, no prior work has applied PERL to train LLMs for financial tasks, such as portfolio optimization. This is because PERL is a relatively new technique, and was originally developed for Reinforcement Learning from Human Feedback (RLHF) in LLM policy alignment tasks.

The main contribution of this work is leveraging the Chronos-T5 LLM [Ansari et al., 2024] and training it to optimize a portfolio of stocks based on market feedback rather than human feedback. This approach was inspired by the PERL framework, and we build a similar algorithm for our trading strategy. We evaluate the performance of this method against the 'SPY' index over the same period, demonstrating the competitive edge of our algorithm in comparison to the broader market.

2 Model Architecture

2.1 Original LLM

The Chronos-T5 LLM is a time-series model pre-trained to handle sequential data. Built on the T5 (Text-to-Text Transfer Transformer) architecture, it includes both an encoder and a decoder. Chronos-T5 was further refined for time-series data, utilizing self-attention mechanisms that excel at capturing long-term dependencies in temporal sequences. This architecture enables Chronos-T5 to highlight the most relevant parts of the input data.

2.2 Modified LLM

In our experiments, we utilize the Chronos-T5 Large model (710M parameters). The large parameter space allows Chronos-T5 to capture more complex patterns in the data, improving its ability to generate predictive insights from financial indicators. To adapt this model for portfolio optimization, we added an input projection layer and a final layer, which applies a softmax function to generate portfolio weights. Additionally, we added LoRA adapters to the model, allowing us to fine-tune its parameters with minimal computing power.

3 Training Process

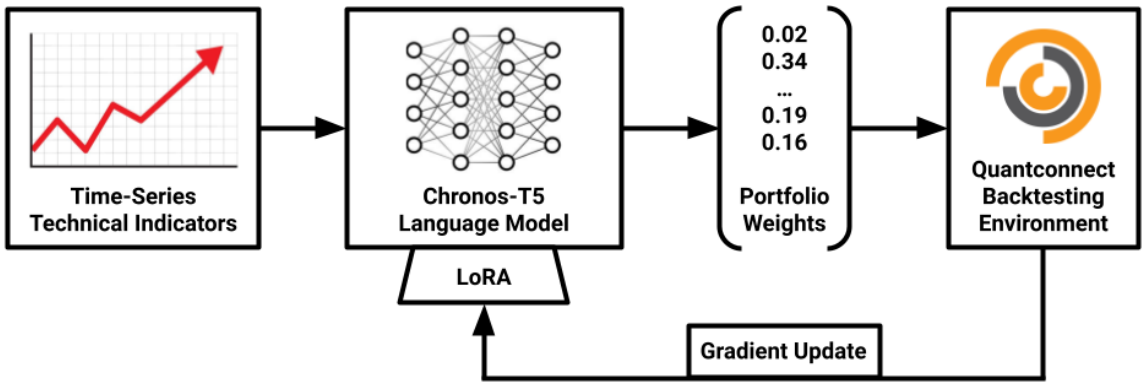

Figure 1: Model Training Process

3.1 Algorithm Implementation

We split the algorithm into two sections, where we perform different actions. This is comprised of the beginning of the month process and the end of month process. In the beginning of the month, we calculate 42 time-series technical analysis indicators, based on each stock's price and volume data. These technical indicators are then used as input to the LLM, which then outputs the portfolio

weights for each stock. The model processes this data and outputs portfolio weights for the 100 selected stocks, which are then allocated in the QuantConnect backtesting environment.

At the end of the month, we assess the performance of the allocated portfolio, and calculate the cumulative returns of each stock. These returns are used in the loss function, which measures the model's ability to maximize returns while balancing a diversified portfolio. The model is updated using Low-Rank Adaptation (LoRA), a parameter-efficient fine-tuning method that adjusts only a small subset of the model's parameters. The loss function is minimized using the AdamW optimizer, which is applied to backpropagate the gradients and update the model's parameters accordingly. This cycle of rebalancing the portfolio and updating the model's parameters, as shown in Figure 1, is repeated monthly, allowing the model to improve its portfolio allocation decisions over time.

3.2 Training with Low-Rank Adaptation

A key limitation of this research was the restricted GPU resources available through QuantConnect. Training an LLM such as the 710M parameter Chronos-T5 with only a single 32GB GPU is computationally inefficient, leading to a focus on parameter efficiency with LoRA. LoRA is a parameter-efficient fine-tuning method that updates only a small portion of a model's parameters, enabling effective training on limited hardware. This approach retains the general knowledge from pre-training while adapting the model to domain-specific tasks.

By optimizing parameters with less computing power, LoRA allows us to efficiently train LLMs on large datasets, making it a practical solution for financial applications without requiring expensive hardware resources. While LoRA mitigated some of the resource limitations, it was not a complete solution. We were constrained to using 16-bit floating point precision (FP16) instead of the more accurate 32-bit (FP32) format, which likely limited the model's performance.

3.3 Online Learning and Portfolio Optimization

To train our model, we employ an online learning approach, where the objective is to directly maximize the portfolio's expected cumulative return each month. The model learns from the outcomes of its portfolio allocations and adjusts its parameters to improve future decisions. This allows the model to adapt to market changes as they happen, instead of relying on historical time-series forecasting.

Rather than using a traditional reinforcement learning algorithm, which would involve exploring different actions stochastically, we adopt a deterministic optimization approach to reduce the computational complexity of our strategy. In this setup, the portfolio weights are generated directly by the model, based on the technical indicators input data computed for the past month.

The loss function is defined to maximize the expected return of the portfolio with an additional entropy regularization term to encourage diversification in the portfolio:

$$\mathcal{L}(w, r) = -w^T r + \eta \cdot (-w^T \cdot \log(w + \epsilon))$$

where w^T represents the generated portfolio weights, r denotes the returns of each stock over the month, η is a regularization strength parameter, and ϵ is a small constant added for numerical stability. The first term in the loss function maximizes the expected portfolio return, and encourages the model to exploit the strategy it learns. The second term penalizes concentrated allocations, promoting diversity in stock selection, and encouraging exploration by regularizing the entropy of the portfolio.

4 Technical Analysis Indicators Data

The technical analysis indicators for this project were calculated using the TA-Lib python library, focusing on various groups including Cycle Indicators, Momentum Indicators, Overlap Studies, Price

Transforms, Statistical Functions, Volatility Indicators, and Volume Indicators. We deliberately excluded non-time-series indicators that produce output not aligned with our approach.

To enhance model interpretability and performance, we applied a Variance Inflation Factor (VIF) filter to eliminate indicators with high multicollinearity. Any indicators with high VIF values were removed to prevent the model from being influenced by redundant or highly correlated features, ensuring that each selected indicator contributed unique and valuable information to the model's decision-making process. After filtering, we retained a total of 42 distinct technical indicators.

Extensive data cleaning was performed to ensure the integrity of the dataset. This included removing constant or zero-variance columns, handling missing values, and verifying data consistency. The resulting dataset provided a high-quality input for the model, allowing us to simulate a "technical analysis trader" capable of making optimized portfolio decisions based on real-time market trends.

Additionally, leveraging the LLM for time series analysis helped to mitigate look-ahead bias. LLMs are trained on vast internet text data, including both news and social media, giving them inherent knowledge of historical market movements and company performance, which could introduce bias during a backtest. By focusing exclusively on technical indicators through time-series analysis, we minimize this bias and utilize the pre-trained model's existing architecture to make predictions based on numerical data rather than text data.

5 Experimental Setup

All experiments were conducted using the QuantConnect backtesting engine, utilizing its online cloud platform and GPU nodes. Model training was conducted on a single Tesla V100S-PCIE-32GB GPU provided by QuantConnect. Model training took place during the backtest period, ranging from 6/1/2009 to 12/31/2019. These dates were chosen to align with the start of the ETF constituents dataset from QuantConnect and the end of the Alphathon backtesting period.

After determining the model's architecture and hyperparameters based on the performance during the training period, we evaluated the same strategy on an out-of-sample period, spanning approximately 15 years and 4 months. This period starts on 6/1/2009 and ends 9/30/2024. The data from the out-of-sample period was not used in any way to influence the model's development. For both the in-sample and out-of-sample evaluations, we compared our strategy to a buy and hold strategy of the 'SPY' index for comparison over the same time period, using an initial starting capital of $1,000,000,000 for all experiments.

The stock selection process for our strategy is based closely on that of the S&P 500 stock universe, using the ETF constituents dataset in QuantConnect. Every three months, we selected the top 100 weighted stocks from the 'SPY' index, adding new stocks and removing delisted ones from the portfolio. Every month, the algorithm explained in section 3.1 is conducted to rebalance the portfolio, and update the model. At the conclusion of the experiment, QuantConnect's backtest metrics were analyzed.

Key hyperparameters, such as learning rate=0.01, η=0.01, and LoRA parameters, were experimentally derived. Another notable parameter is the model's initial weights, determined by random seed values. These hyperparameters impact performance of the model, making them important factors to consider. We tested several combinations of hyperparameters and ultimately presented the setup with the most robust risk management, shown in the results section. However, more comprehensive evaluation of hyperparameter tuning could be explored in future work to increase the capabilities of the model.

6 Results

Our experiments demonstrate that training the Chronos-T5 LLM for portfolio optimization yields promising results, particularly in generating alpha and managing risk effectively. We conducted

both in-sample and out-of-sample backtests to evaluate the performance of our algorithm against the 'SPY' index benchmark.

6.1 In-Sample Performance

During the in-sample period, our algorithm demonstrates outperformance of the benchmark across critical risk-adjusted metrics. Our strategy delivered an alpha of 0.007, demonstrating its consistent ability to generate excess returns over the benchmark. Furthermore, our algorithm produced a higher Sharpe Ratio of 0.793 compared to the benchmark's 0.77, and a superior Sortino Ratio of 0.828, surpassing the benchmark's 0.807. These results suggest that our approach delivers more favorable risk-adjusted returns, despite a slightly lower Compounding Annual Return (CAR) of 14.436% compared to the SPY's 14.539%.

Our model also exhibited a beta of 0.951, lower than the benchmark's 0.996, signifying reduced sensitivity to overall market movements and a more controlled exposure to systematic risk. With an annualized volatility of 11.8%—below the SPY's 12.3%—and a maximum drawdown of 16.8% versus the benchmark's 19.3%, the results highlight the algorithm's robust risk management and downside protection. A visual comparison of the algorithm's performance relative to the benchmark is shown in Figure 2.

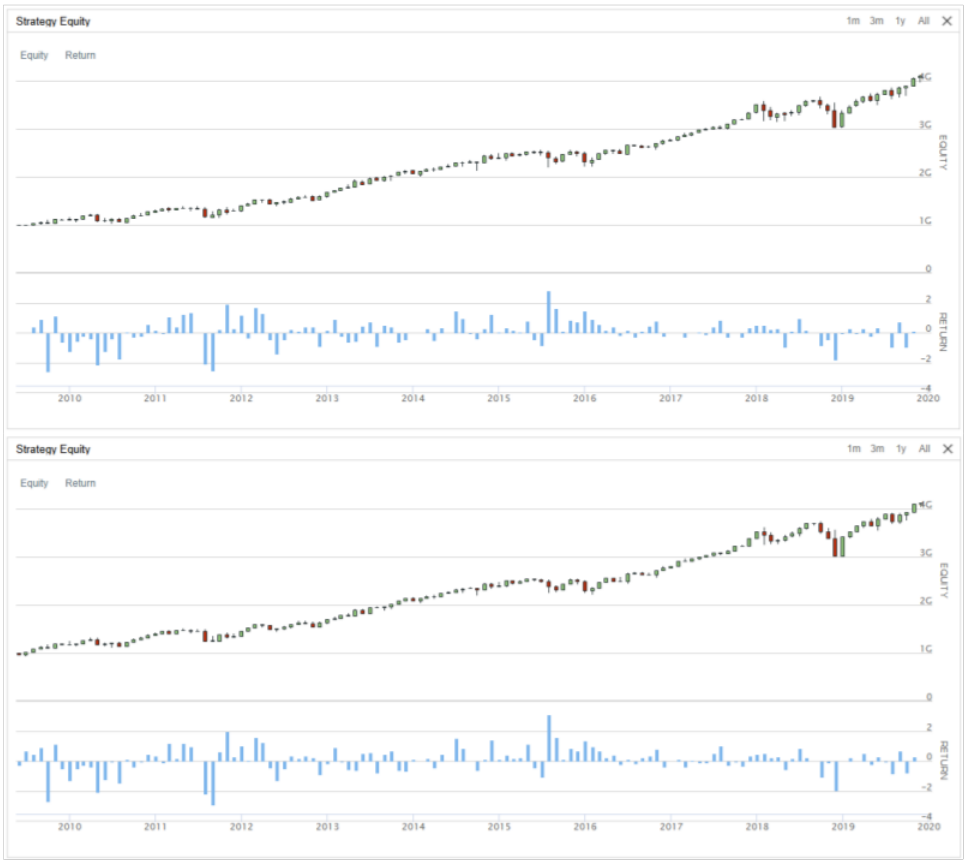

Figure 2: In-Sample Performance Comparison - Top: Our Algorithm, Bottom: SPY Benchmark

6.2 Out-of-Sample Performance

In the out-of-sample period, our algorithm delivered mixed results, with strong risk control but lower returns compared to the market. While the strategy continued to achieve a positive alpha, it was lower than it was in the in-sample backtest, with a value of 0.001, indicating a limited ability to generate excess risk-adjusted returns over the market in out-of-sample data. The Sharpe Ratio of 0.641, slightly below the SPY's 0.657, suggests that our risk-adjusted performance was largely comparable to the benchmark. Similarly, the Sortino Ratio of 0.657, while slightly underperforms the benchmark's 0.672, indicating moderate downside risk management.

Despite a slightly lower Compounding Annual Return (CAR) of 13.856% compared to the benchmark's 14.503%, our strategy demonstrated superior risk control. The model's beta of 0.955, significantly lower than the benchmark's 0.997, reflects reduced market exposure, helping investors navigate volatility more effectively. Furthermore, our strategy exhibited lower annualized volatility (13.8%) compared to the SPY's 14.2%, and a maximum drawdown of 32.5%, which was less severe than the benchmark's 33.7%. These factors highlight our algorithm's ability to manage risk and protect capital, making it a compelling option for investors seeking stable, long-term growth.

Figure 3: Out-of-Sample Performance Comparison - Top: Our Algorithm, Bottom: SPY Benchmark

6.3 Discussion

Overall, our findings highlight the potential of leveraging LLMs for portfolio optimization based on technical analysis, as a robust, data-driven strategy for generating alpha and enhancing risk man-

agement. Both in-sample and out-of-sample evaluations show that our model consistently delivers strong risk-adjusted returns, outperforming traditional benchmarks in key areas, particularly in risk management. While the annual returns are slightly lower than those of the benchmark, the lower volatility and beta compared to the benchmark demonstrate effective risk mitigation, making our approach highly applicable to real-world investment scenarios.

The alpha generated in both periods highlights the model's ability to extract insights from vast historical time-series data, offering enhancements to established investment operations with a data-driven strategy. By effectively minimizing risk, and capturing market trends in real time, our approach offers a sophisticated, and unique algorithm that holds significant potential over traditional benchmarks.

7 Future Work

Future research should focus on more extensive hyperparameter optimization. Preliminary results from additional experiments suggest that fine-tuning hyperparameters such as LoRA rank, LoRA alpha, learning rate, and η could significantly increase alpha generation and annual returns, while introducing additional risk as well. These experiments were not included in the final paper to maintain focus on risk management, which was a key strength of the more robust experiments presented.

Further research could also explore scaling computational resources to support the training of LLMs. Given that we trained the 710M Chronos-T5 LLM using a single GPU, expanding to larger models in line with current trends in LLM research could yield substantial improvements in performance.

8 Conclusion

In this submission for the Alphathon 2024 competition, we introduce an innovative approach to portfolio optimization by fine-tuning the Chronos-T5 LLM on time-series technical analysis indicators data. By utilizing Low-Rank Adaptation (LoRA) and developing an algorithm similar to the Parameter-Efficient Reinforcement Learning (PERL) framework, we trained the model to optimize stock portfolio weights based on feedback from its past performance.

Rather than using LLMs on news data or for sentiment analysis, our approach leverages Chronos-T5's deep time-series understanding and self-attention mechanisms to capture market trends and make precise investment decisions. Backtests on QuantConnect demonstrate the capabilities of this approach particularly for risk management and alpha generation, with other metrics comparable to the S&P 500. Despite resource constraints, this method proved both efficient and cost-effective, illustrating the practicality of our approach, and offering a promising direction for future research in quantitative finance.

References

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024. URL https://arxiv.org/abs/2403.07815.

Han Ding, Yinheng Li, Junhao Wang, and Hang Chen. Large language model agent in financial trading: A survey, 2024. URL https://arxiv.org/abs/2408.06361.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Wei Lu, Rachel K. Luu, and Markus J. Buehler. Fine-tuning large language models for domain adaptation: Exploration of training strategies, scaling, model merging and synergistic capabilities, 2024. URL https://arxiv.org/abs/2409.03444.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and B Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. *Younes Belkada and Sayak Paul," PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods*, 2022. URL https://github.com/huggingface/peft.

Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M. Mulvey, H. Vincent Poor, Qingsong Wen, and Stefan Zohren. A survey of large language models for financial applications: Progress, prospects and challenges, 2024. URL https://arxiv.org/abs/2406.11903.

Hakim Sidahmed, Samrat Phatale, Alex Hutcheson, Zhuonan Lin, Zhang Chen, Zac Yu, Jarvis Jin, Roman Komarytsia, Christiane Ahlheim, Yonghao Zhu, Simral Chaudhary, Bowen Li, Saravanan Ganesh, Bill Byrne, Jessica Hoffmann, Hassan Mansoor, Wei Li, Abhinav Rastogi, and Lucas Dixon. Perl: Parameter efficient reinforcement learning from human feedback, 2024. URL https://arxiv.org/abs/2403.10704.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models, 2023. URL https://arxiv.org/abs/2306.06031.

Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W. Suchow, and Khaldoun Khashanah. Finmem: A performance-enhanced llm trading agent with layered memory and character design, 2023. URL https://arxiv.org/abs/2311.13743.

Wentao Zhang, Lingxuan Zhao, Haochong Xia, Shuo Sun, Jiaze Sun, Molei Qin, Xinyi Li, Yuqing Zhao, Yilei Zhao, Xinyu Cai, Longtao Zheng, Xinrun Wang, and Bo An. A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist, 2024. URL https://arxiv.org/abs/2402.18485.

Jiawei Zheng, Hanghai Hong, Xiaoli Wang, Jingsong Su, Yonggui Liang, and Shikai Wu. Fine-tuning large language models for domain-specific machine translation, 2024. URL https://arxiv.org/abs/2402.15061.