LLMs for Time Series: an Application for Single Stocks and Statistical Arbitrage

Sebastien Valeyre *† Sofiane Aboura ‡

December 12, 2024

Abstract

Recently, LLMs (Large Language Models) have been adapted for time series prediction with significant success in pattern recognition. However, the common belief is that these models are not suitable for predicting financial market returns, which are known to be almost random. We aim to challenge this misconception through a counterexample. Specifically, we utilized the Chronos model from Ansari et al. (2024) and tested both pretrained configurations and fine-tuned supervised forecasts on the largest American single stocks using data from Guijarro-Ordonnez et al. (2022). We constructed a long/short portfolio, and the performance simulation indicates that LLMs can in reality handle time series that are nearly indistinguishable from noise, demonstrating an ability to identify inefficiencies amidst randomness and generate alpha. Finally, we compared these results with those of specialized models and smaller deep learning models, highlighting significant room for improvement in LLM performance to further enhance their predictive capabilities.

*Machina Capital, Paris, France
†Valeyre Research, Cannes, France
‡University of Paris XIII – Sorbonne Paris Nord, Villetaneuse, France

1 Introduction

LLMs (Large Language Models) gained widespread popularity when ChatGPT convinced many that machines were intelligent enough to reason like humans, even though the underlying techniques merely determine the most likely sequences of words that could respond to a prompt. The introduction of the transformer architecture by Vaswani et al. (2017) was a key development, as it enabled fast training on large datasets. LLMs are composed of many layers of transformers and have around a billion parameters. The T5 (Text-To-Text Transfer Transformer) architecture was introduced by Raffel et al. (2023), and further advanced the field from T5-small size with 60 million of parameters to T6-11B with 11 billion of parameters). Amatrianin (2023) describes the catalog of LLMs models from Albert to ChatGPT and classifies T5 among others.

Garza et al. (2023) introduce TimeGPT, the first foundation model for time series, capable of generating accurate predictions for diverse datasets not seen during training. They evaluate their pre-trained model against established statistical, machine learning, and deep learning methods, demonstrating that TimeGPT zero-shot inference excels in performance. Ansari et al. (2024) also sought to adapt these highly efficient LLMs to time series forecasting. Their approach involved representing real numbers in different bins (using a vocabulary of 4096 tokens) and training a T5 architecture on a wide range of time series data (around 90 billion observations). They produced several pretrained models of varying sizes, ranging from tiny (11 millions of parameters) to large. TimesFM (Time Series Foundation Model) is another pretrained time-series foundation model developed by Das et al. (2024). Rasul et al. (2024) uses a lagged features for tokenization. Nie et al. (2023); Ekambaram et al. (2023) are also applying LLM to multi time series. Motivated by recent advances in large language models for Natural Language Processing, Ansari et al. (2024); Das et al. (2024) designed a time-series foundation model for forecasting whose out-of-the-box zero-shot performance on a variety of public datasets comes close to the accuracy of state-of-the-art supervised forecasting models for each individual dataset. Their models are based on pretraining a patched-decoder style attention model on a large time-series corpus, and can work well across different forecasting history lengths, prediction lengths and temporal granularities.

Brugiere and Turinici (2024) tested transformers for financial time series and showed that the algorithm cannot predict returns, but can only predict squared returns. Konigstein (2024) stated that LLMs present challenges, but also opportunities, particularly for long-term financial time series forecasting.

Deep learning has become very popular among researchers in finance, both for asset pricing and systematic strategies: Guijarro-Ordonnez et al. (2022) implemented transformers (with only 769 parameters) but coupled them with convolutional layers and tested them on quantitative trading strategies applied to single stocks. Their results, without accounting for trading costs, were encouraging. Guijarro-Ordonnez et al. (2022) applied deep learning algorithms to residual daily returns after removing common factors using techniques such as Principal Component Analysis (PCA), Fama-French factors, or Instrumental Principal Component Analysis (IPCA), implementing the methodology from Kelly et al. (2019), who used financial data to constrain the eigenvectors. This approach was also developed and justified by Valeyre (2019). Transformers have also been used by Jiang et al. (2023) to identify patterns in images for time series forecasting. Wood et al. (2022) applied deep learning techniques with only a few parameters to identify the most effective trend-following indicators enhancing and timing trend-following strategies. Transformers have also been used to extract common returns, as demonstrated by Gu et al. (2021). Qyrana (2024) applied a simple STR factor to residual returns using an autoencoder-based factor model, subsequently generating a highly profitable trading strategy. Chen et al. (2021) used a deep learning technique to identify anomalies in asset pricing,

Our goal, in this study, is to evaluate deep learning algorithms with more than 11 million parameters for forecasting financial returns. In contrast, studies such as Chen et al. (2021), Jiang et al. (2023), Wood et al. (2022), Guijarro-Ordonnez et al. (2022), and Brugiere and Turinici (2024) utilized models with at most a few hundred parameters so their models were not really "deep learning" models. This limitation was due to their inability to pre-train models with millions of parameters using large datasets from outside the financial industry.

So we first conducted a zero-shot evaluation of the predictions from pre-trained and fine-tuned supervised time series foundation LLMs Chronos by Ansari et al. (2024), which were pretrained on 13 datasets that do not include single stock data or stock indices, using the datasets of the residual returns of American single stocks published by Guijarro-Ordonnez et al. (2022). The interest in using only zero-shot evaluation is that it provides more convincing results, as overfitting is less likely in this case. Indeed overfitting is a major concern in machine learning when applied to trading, which often makes honest results appear suspect. Another interesting aspect is demonstrating how the algorithm can adapt and display 'intelligence' in trading without being specifically trained for that purpose. We aim to simulate a portfolio that goes long on positive predictions and short on negative predictions.

Secondly, we seek to compare these results with well-known standard Short Term Reversal (STR) or trend-following approaches, as described in Jegadeesh (1990); Jegadeesh and Titman (1993).

2 The methodology of our empirical backtest

2.1 Chronos as the LLM model

The model used was "amazon/chronos-t5-tiny" version of the chronos with 11 million of parameters which was pretrained by Ansari et al. (2024) on 14 datasets (Brazilian cities temperatures, Mexico city bikes, Solar, Spanish energy and weather, Taxi, USHCN, weatherbench, wiki daily, wind farms) but not on financial time series. The model has 11 millions of parameters.

We used a 'context' period of 100 days so that Chronos guess the next day, knowing only the previous 100 days. We focused only on the next day return forecast. We used 100 days as a compromise to avoid running out of memory while giving Chronos a chance to capture some patterns. Additionally, we decided to limit the study to predicting the next daily returns. Although we could have considered predicting the next weekly or monthly returns, we believed it would be easier for Chronos to capture patterns over a very short-term horizon.

We adjusted the 11 million weights of the "amazon/chronos-t5-tiny" model through training (fine-tuning) using our datasets, setting τ, the maximum training steps, to 5, 15, or 40. Training was conducted daily during the backtest, using the data available on each respective date and starting from the weights of the previous day.

The details of the parameters for both the pretrained case and the fine tuning are described in the appendix B.1 and B.2.

2.2 Financial market time series as data

We used 3 different datasets of the residual daily returns released by Guijarro-Ordonnez et al. (2022), derived from their analysis of securities in the CRSP dataset from January 1978 to end 2016. Their focus was on the most liquid stocks to mitigate trading and friction issues. Specifically, they considered stocks whose market capitalization in the prior month exceeded 0.01% of the total market capitalization of that month, resulting in a selection of approximately the largest 550 stocks on average. Guijarro-Ordonnez et al. (2022) released their datasets of residual returns on GitHub. We utilized their three datasets, each corresponding to their standard parameters with $K = 5$ (i.e. with 5 factors):

- IPCA factors with a rolling windows of 240 months with the details described in Kelly et al. (2019)
- PCA factors with a rolling windows of 252 days
- FF factors (Fama-French 3 factor model+ investment and profitability factors) with a rolling windows of 60 days

$K = 5$ appears to be the optimal according to Guijarro-Ordonnez et al. (2022) when applying their convolution and transformer process with a gross sharpe ratio of 3.21 for Fama-French, 3.36 for the PCA and 4.16 for the IPCA. Nevertheless the sharpe in net appears to be significant only before 2006.

Using residual returns allows for a less correlated dataset, which is crucial for deep learning. We can note that Gu et al. (2021) for example also used the same three different datasets to test their model.

2.3 Description of the different simulated strategies

Our experiment was organized in three parts.

2.3.1 Strategy based on the forecast of the zero-shot version of Chronos

First, we implemented a "zero-shot evaluation", which means without any fine-tuning (or training). The weights of Chronos were not trained on financial data. Our experiment consists of, for each day, from 2001-12-26 to 2016-12-30, and for each dataset (IPCA, PCA, FF):

- Computing $\hat{\chi}_{d,i}$, the Chronos average prediction of the next daily return, derived in the Eq 2 conditioned to a rolling window of the last 100 days. In Eq 2 we used the average of different "equiweight" scenarii χ computed by Chronos of $\hat{r}_{d+1}$ knowing only $r_{d,i}...r_{d-99,i}$. We used two possible inputs for Chronos:

  - Either the last 100 residual daily returns $r_{d,i}...r_{d-99,i}$ of the single stock i, when $\alpha = 0$ in Eq 1.

  - or the last $\hat{r}_{d,i}...\hat{r}_{d-99,i}$ the exponential moving average of the last 100 residual daily returns derived in Eq 1 with $\alpha > 0$. We tested α values of 0.1, 0.2, 0.3, 0.4, 0.5, and 0.8, all different from zero. This option allows the model to account for the well-known weak negative autocorrelation of daily returns, potentially improving its forecasting ability. However, in this case, the model must outperform the Short-Term Reversal strategy described in Eq. 10.

- Predicting of the next returns with $\tilde{\chi}_{d,i}$ which is derived in Eq 3.

- Calculating the weights of the portfolio $\hat{\omega}$ derived in Eq 6 which ranks through $\Re$ = ArgSort every day d the different $\tilde{\chi}_{d,i}$. In this method the median rank is withdrawn through $\frac{N}{2}$ where N is the number of stocks. A normalization of the weights is derived in Eq 6 to target a gross investment of 1. This process ensures that the portfolio is 50% long and 50% short every day, with weights proportional to the distance in ranking from the median stock according to $\tilde{\chi}$. Valeyre (2019) proved that this approach is the mathematically optimal method and better than just buying the top quintile and short the bottom quintile. A 'resized' version is also tested when the weights are also inversely proportional to the volatility as derived in Eq 5 where σ are the standard deviation of the daily returns on the previous 100 days and $\mathbf{M}$ is the median.

- Simulating, $\mathcal{P}_{d+1}$, the performance of the portfolio for the next day through Eq 8. We then reconstructed the cumulative returns and calculated the gross Sharpe ratio, excluding any trading costs. We also simulate $[\mathcal{P}_{d+1}]$ for the resized version through Eq 9.

$$\hat{r}_{d+1,i} = \alpha \hat{r}_{d,i} + r_{d+1,i} \tag{1}$$

$$\hat{\chi}_{d,i} = \mathbf{E}\left[\chi\left(\hat{r}_{d+1,i} | \hat{r}_{d,i} ... \hat{r}_{d-99,i}\right)\right] \tag{2}$$

$$\tilde{\chi}_{d,i} = \hat{\chi}_{d,i} - \alpha \hat{r}_{d,i} \tag{3}$$

$$\omega_d^{\chi} = \Re\left[\Re\left(\tilde{\chi}_d\right)\right] - \frac{\mathrm{N}}{2} \tag{4}$$

$$\left[\omega_d^{\chi}\right]^r = \left(\Re\left[\Re\left(\tilde{\chi}_d\right)\right] - \frac{\mathrm{N}}{2}\right) \frac{\mathbf{M}\left(\sigma_0 ... \sigma_N\right)}{\max(\sigma, \mathbf{M}\left(\sigma_0 ... \sigma_N\right)} \tag{5}$$

$$\hat{\omega}_{d,i}^{\chi} = \frac{\omega^{\chi}{}_{d,i}}{\sum_i |\omega^{\chi}{}_{d,i}|} \tag{6}$$

$$\left[\hat{\omega}_{d,i}^{\chi}\right]^r = \frac{\left[\omega^{\chi}{}_{d,i}\right]^r}{\sum_i \left|\left[\omega^{\chi}{}_{d,i}\right]^r\right|} \tag{7}$$

$$\mathcal{P}_{d+1} = \sum_i \hat{\omega}_{d,i}^{\chi} \times r_{d+1,i} \tag{8}$$

$$\left[\mathcal{P}_{d+1}\right] = \sum_i \left[\hat{\omega}_{d,i}^{\chi}\right] \times r_{d+1,i} \tag{9}$$

.

2.3.2 Strategy based on the forecast of the fine tuned version of Chronos

Secondly, we used a very naive solution for fine tuning from the pretrained weights which could be a nightmare in practice (Goodfellow et al. (2013)). We trained Chronos which was initiated at the beginning of the backtest with the pretrained weights. The training was realized on a daily basis during the backtest using the available financial market data starting on every day with the weights of the previous day. On every day d, we provided as input to Chronos the updated time series available at day d using the the previous 100 days. We test different parameters for τ the maximal number of steps for the daily training. We also used different values of the parameter α in Eq 1 so that Chronos receives the EMA. Thanks to that feed, Chronos can adapt its weights according to the properties of the financial time series.

The continuous training led to Chronos model to update its weights for each day of the backtest. We then determined the portfolio weights based on the predictions using the fine-tuned weights instead of the pretrained ones. Finnaly we use exactly the same evaluation's receipe than the one described in section 2.3.1 with the only difference that the pretrained weights were replaced every day by the fine tuned weights determined at that day.

We do not claim that our methodology for fine tuning is optimal, as there are likely better methods to empirically determine an improved approach by controling overfitting and the loss of the pretrained weights through the analyze of the statistics of the eigenvalues derived from the millions of weights (Martin (2019, 2024)). However, this falls outside the scope of our current study. For instance, the drawback of our methodology is that the pretraining weights are gradually forgotten over time, which is not an ideal solution.

2.3.3 Evaluation of other strategies for comparaison

Third, we compare the results of Chronos to those obtained by replicating the CNN Transformers model of Guijarro-Ordonnez et al. (2022) whose number of parameters is only 169 which appears quite small compared with the 11 million of the Chronos one. We also include the results achieved using autoARIMA from the statsforecast package (https://pypi.org/project/statsforecast/) as it is a standard benchmark in Machine Learning, as well as the short term reversal STR described in Jegadeesh (1990); Jegadeesh and Titman (1993) which is a well-documented market anomaly that was first noted by Fama (1965).

The short term reversal (STR) strategy was derived in Eq 10 with both $\beta = 1 - \frac{1}{5}$ and $\beta = 1 - \frac{1}{20}$ using a simple exponential moving average on residual returns $r_{d+1,i}$ at day $d+1$ and single stock i when extracting common factors from the IPCA, PCA or FF. The portfolio of the STR is then derived in Eq 11 with $\omega^{\zeta}{}_d$ using the same methodology as above where N is the number of single stocks.

The AutoARIMA was also fit in a continuous way every day during the backtest using the previous 100 days $\times$ N observations and yielded to forecasts $\mathbf{A}_d$ and the portfolio weights $\omega_d^{\mathbf{A}}$ were also derived with the same methodology Eq 12.

$$\tilde{r}_{d+1,i} = \beta \times \tilde{r}_{d,i} + r_{d+1,i} \tag{10}$$

$$\omega_d^{\zeta} = \Re\left[\Re\left(-\tilde{r}_d\right)\right] - \frac{\mathrm{N}}{2} \tag{11}$$

$$\omega_d^{\mathbf{A}} = \Re\left(\Re\left(\mathbf{A}_d\right)\right) - \frac{\mathrm{N}}{2} \tag{12}$$

3 Our empirical results

We observe that the pre-trained Chronos model with $\alpha = 0.3$ effectively identifies opportunities in the financial market, achieving a Sharpe ratio above 3.17 for PCA over a 15-year period, which corresponds to a t-statistic of $3.17\sqrt{15} = 12.27$. However, trading costs are prohibitive, as including a 3 basis point slippage cost per trade results in negative net Sharpe ratios (see Table 1).

α	0	0.1	0.2	0.3	0.4	0.5	0.8
FF	0.07	1.27	1.80	1.84	1.39	1.39	-0.24
PCA	0.04	2.08	2.75	3.17	3.25	2.71	0.07
IPCA	-0.47	0.68	1.19	1.34	1.42	1.18	-0.81

Table 1: Simulation of the Gross sharpe ratio of the strategy based on the zero-shot pretrained prediction of Chronos when using as input the exponential moving average of daily residual returns through using either the IPCA, the PCA or FF. α is the parameter of the EMA from Eq 1. The period is 2002-2016.

Additionally, we note a decline in profitability over time, suggesting that markets may be becoming more efficient or that opportunities are increasingly challenging to capture or that returns used to be more negatively autocorrelated before 2008 (see Figure 1). However, at least until 2007, it was easier for AI to capture inefficiencies.

It is particularly interesting to observe that the pre-trained version with $\alpha = 0$ is ineffective until 2007 but seems to work after 2008 (see Figure 1). In our interpretation, Chronos is pre-trained on data where 'trend' serves as an efficient indicator, whereas in our dataset, residual returns tend to be negatively autocorrelated in the short term. We believe that $\alpha = 0.3$ is optimal, as it offsets this effect, helping Chronos to overcome biases from its trend-oriented training dataset.

Setting $\alpha = 0.3$ artificially aids Chronos; however, it ultimately gets very correlated to the Short-Term Reverseal (STR) strategy with $\beta = 0.3$ but fails to outperform it . In other words, when Chronos is provided with the Exponential Moving Average (EMA), its performance does not exceed that of a zero forecast. Nevertheless, it avoids being affected by detrimental noise, which is already a positive outcome.

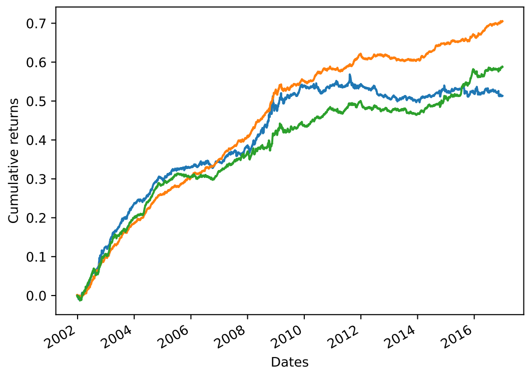
	FF	PCA	IPCA
pretrained Chronos $\alpha = 0.2$	1.80	2.75	1.19
trained Chronos $\alpha = 0$ $\tau = 15$		0.24	
trained Chronos $\alpha = 0.3$ $\tau = 5$	2.12	3.90	2.29
trained Chronos $\alpha = 0.3$ $\tau = 15$		3.97	
resized trained Chronos $\alpha = 0.3$ $\tau = 15$		4.21	
trained Chronos $\alpha = 0.3$ $\tau = 40$		3.80	
CNN Transformer	3.15	5.01	4.29
STR $\beta = 0.2$	2.23	4.16	2.31
STR $\beta = 0.3$	2.16	4.03	2.31
resized STR $\beta = 0.3$	2.31	4.27	2.32
STR $\beta = 0.8$	1.24	2.42	1.76
STR $\beta = 0.95$	0.98	1.38	1.20
autoARIMA	1.43	2.10	1.22

Table 2: Simulation of the Gross sharpe ratio of the strategy α is the parameter of the EMA in Eq 1. β is the parameter of the EMA in Eq 10. τ is 'max steps' input in the fine-tuned version of Chronos. The period is 2002-2016.

Table 2 presents the Sharpe ratios for the fine-tuning case as well as for the benchmarks.

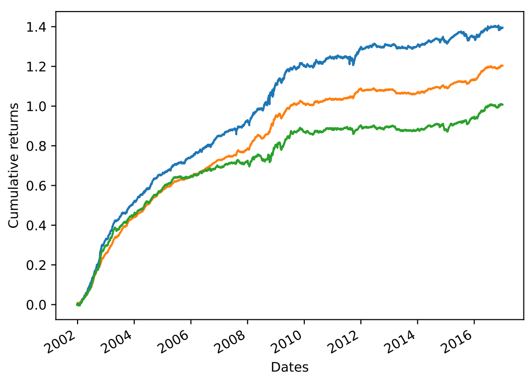
When setting $\alpha = 0$, Chronos requires fine-tuning with $\tau = 15$ to achieve a Sharpe ratio of 0.24, which corresponds to a t-statistic of $0.24\sqrt{15} = 0.92$. It appears to work well until 2008 (see Figure 1), but after that, the pre-trained configuration may have been completely forgotten due to the numerous fine-tuning processes performed since 2002. It might be interesting to test a version where there is a regular reinforcement of the pre-trained configuration to ensure it remains in memory.

Additionally, the correlation with the STR strategy is not significant when $\alpha = 0$ and $\tau = 15$. In contrast, the CNN-Transformer appears to be primarily a linear combination of STR strategies at different time scales. This suggests that the opportunities captured by Chronos may be more complex than those driven by basic mean reversion. We also observe that the autoARIMA model, which is a classical benchmark in Machine Learning underperforms the STR model, demonstrating that fitting a model is particularly challenging when

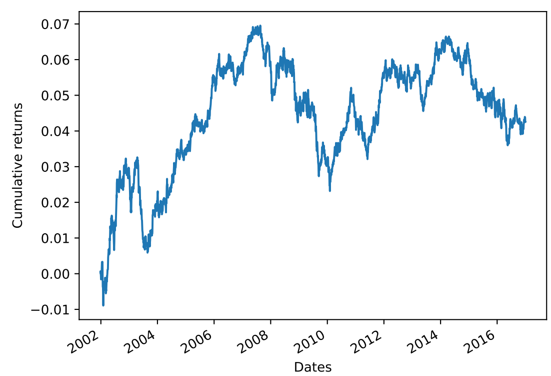


(a) zero-shot pretrained prediction of Chronos. $\alpha = 0.3$. The orange is for pca, the green for FF, the blue for ipca.

(b) zero-shot pretrained prediction of Chronos. $\alpha = 0$. The orange is for pca, the blue for ipca, the green for FF.



(c) STR $\beta = 0.8$. The blue is for ipca, the orange for pca, the green for FF.



(d) fined tuned Chronos. $\alpha = 0$ and $\tau = 15$. the blue for pca.

Figure 1: Simulation of the strategy. α is the parameter of the EMA in Eq 1. β is the parameter of the EMA in Eq 10. τ is 'max steps' input in the fine-tuned version of Chronos.

the data are nearly random and contain significant noise. This results in underperformance compared to more rigid models like STR.

We can also see that resizing the weights inversely proportional to the volatility improves the Sharpe ratio for Chronos, as well as for the benchmarks.

Finally, it is interesting to note that the optimal τ for training appears to be 15. When $\tau = 40$, the Sharpe ratio decreases, suggesting that Chronos may lose some of its pre-trained intelligence.

4 Conclusion

Our results show that AI, specifically LLMs, can be trained on large datasets that exclude financial time series and still exhibit enough intelligence to identify opportunities in the financial market, previously considered too challenging for AI, without the risk of overfitting. Currently, AI lacks the “intelligence” to find opportunities that remain profitable when factoring in trading costs, but we can anticipate that advancements in AI may eventually make this feasible.

Nevertheless, we believe that specialized models, such as those by Valeyre (2024), which theoretically capture well-established opportunities in an optimal way (like trends), will always prove more efficient, while AI could serve as a valuable tool for identifying more complex opportunities.

That belief is justified by the case of the strong outperformance of the STR compared to AutoARIMA, which can capture more complexity but whose noisy fit makes it suboptimal and overly erratic.

References

Xavier Amatriain, 'Transformer models: an introduction and catalog', arXiv:2302.07730v2, 2023

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, Yuyang Wang, 'Chronos: Learning the Language of time series', arXiv:2403.07815, 2024

Pierre Brugiere, Gabriel Turinici , 'Transformer for Times Series: an Application to the S&P500', arXiv2403.02523, 2024

Luyang Chen, Markus Pelger, Jason Zhu, 'deep learning in asset pricing', Management Science 70(2):714-750, 2023

Abhimanyu Das, Weihao Kong, Rajat Sen, Yichen Zhou, 'A decoder-only foundation model for time-series forecasting', Proceedings of the 41st International Conference on Machine Learning, PMLR 235:10148-10167, 2024

Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, Jayant Kalagnanam,l, 'TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting', Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023

Eugene Fama, 'The behavior of stock market prices'. J. Bus., 38:34–105, 1965

Azul Garza, Cristian Challu, Max Mergenthaler-Canseco, 'TimeGPT-1', arXiv:2310.03589, 2023

Ian J. Goodfellow, Mirza, M., Xiao, D., Courville, A., and Bengio, Y., 'An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks', arXiv:1312.6211, 2013

Shihao Gu, B. Kelly, and D. Xiu , 'Autoencoder Asset Pricing Models', Journal of Econometrics, 22, 429–450, 2021

Jorge Guijarro-Ordonez, Markus Pelger, Greg Zanotti, 'Deep Learning Statistical Arbitrage', nber-nsf Time Series Conference, 2021

Zacharia Issa, Blanka Horvath, Maud Lemercier, Cristopher Salvi, 'Non-adversarial training of neuram SDEs with signature kernel scores', 37th Conference on Neural Information Processing Systems (NeurIPS 2023), 2023

Narasimhan Jegadeesh, 'Evidence of predictable behavior of securities returns', Journal of Finance 45 no. 3, 881898, 1990

Narasimhan Jegadeesh and S. Titman, 'Returns to buying winners and selling losers: Implications for stock market effciency', The Journal of Finance 48 no. 1, 6591, 1993

Jingwen Jiang, Bryan Kelly, Dacheng Xiu, '(Re-)Imag(in)ing price trends', Journal of Finance, 2023

Brian Kelly, S. Pruitt, and Y. Su , 'Characteristics Are Covariances: A Unified Model of Risk and Return', Journal of Financial Economics, 134, 501–524, 2019

Nicole Konigstein, 'Time series transformers challenges and opportunities in long-term financial time series forecasting', 20th quant finance conference, cannes, 2024

Charles H. Martin, 'A semi empirical theory of (deep) learning', 2024

Charles H. Martin, 'Traditional and Heavy-Tailed Self Regularization in Neural Network Models', Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, Jayant Kalagnanam, 'A time series is worth 64 words: Long-term forecasting with transformers', ICLR, 2023

Mishel Qyrana, 'Well-diversified arbitrage portfolios through attentional', ssrn, 2024

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu, 'Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer', The Journal of Machine Learning Research, Volume 21, Issue 1, 2020

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, Irina Rish, 'Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting', arXiv:2310.08278, 2024

Sebastien Valeyre, 'Modélisation fine de la matrice de covariance/corrélation des actions', Thèse de doctorat, http://www.theses.fr/2019USPCD057/document, 2019

Sebastien Valeyre, 'Optimal trend following portfolios', Journal of investment strategies, 2024

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, 'Attention Is All You Need', NIPS Conference on Neural Information Processing Systems, 2017

Kieran Wood, Sven Giegrich, Stephen Roberts, Stefan Zohren, 'Trading with the momentum transformer: an intelligent and interpretable architecture', arXiv:2112.08534v3, 2022

A Data

The datasets provided by Guijarro-Ordonnez et al. (2022) are released at `https://github.com/gregzanotti/dlsa-public/tree/main/residuals`

B Parameters of Chronos

We downloaded the python package:

```
!pip install git+https://github.com/amazon-science/
    chronos-forecasting.git
```

B.1 Parameters of pretrained version of Chronos

We used the version "amazon/chronos-t5-tiny" with the following parameters in python:

```
ChronosPipeline.from_pretrained( "amazon/chronos-t5-tiny
    ",device_map="cuda", torch_dtype=torch.bfloat16)
```

```
load_model(
model_id="google/t5-efficient-tiny",
model_type="seq2seq",
vocab_size=4096,
random_init=False,
tie_embeddings=False,
```

```
pad_token_id=0,
eos_token_id=1)
```

```
forecast = pipeline.predict(batch_context, 1)
```

```
predictions = np.mean(forecast.numpy(),axis=1)-
    alpha_chronos*np.reshape(data_train_t[-1,group*
    size_goup_chronos:(group+1)*size_goup_chronos],(np.
    shape(data_train_t[-1,group*size_goup_chronos:(group
    +1)*size_goup_chronos])[0],1)) #-all_timeseries
    [-1,:]#np.quantile(forecast.numpy(), 0.5, axis=1)
eline.predict(batch_context, 1)
```

B.2 Parameters of Fine tuned version of Chronos

From the inital version set from the pretrained case at the begining of the period of the backtest, we update the weights of the chronos model at every day of the backtest from the weights obtained at the previous day by executing every day 10 times on 10 subgroups of the universe τ "steps" inside "trainer.train()" (τ successive corrections of the weights using the "adamw torch fused" gradient algo) per day in the backtest using the past 100 days. τ was tested to 5, 15 and 40. τ is the maximum training steps, i.e. the "max steps" parameter in the "TrainingArguments" method. We used the following parameters in python:

```
chronos.ChronosConfig(
tokenizer_class='MeanScaleUniformBins',
tokenizer_kwargs={'low_limit': -15.0, 'high_limit':
    15.0},
n_tokens=4096,
n_special_tokens=2,
pad_token_id=0,
eos_token_id=1,
use_eos_token=True,
model_type="seq2seq",
context_length=length_training_chronos-1,
prediction_length=1,
num_samples=20,
temperature=1,
```

```
top_k=50,
top_p=1,
)
```

```
TrainingArguments(
output_dir=str("./output/"),
per_device_train_batch_size=32,
learning_rate=1e-3,
lr_scheduler_type="linear",
warmup_ratio=0,
optim="adamw_torch_fused",
logging_dir=str("./output/logs"),
logging_strategy="steps",
logging_steps=500,
save_strategy="steps",
save_steps=500,
report_to=["tensorboard"],
max_steps=5,#200000,
gradient_accumulation_steps=2,
dataloader_num_workers=0,#len(loaded_data),
tf32=True,  # remove this if not using Ampere GPUs (e.g
    ., A100)
torch_compile=True,
ddp_find_unused_parameters=False,
remove_unused_columns=False,)
```

```
shuffled_train_dataset = tch.ChronosDataset(
datasets=(tch.create_gluonts_dataset(all_timeseries,
    daily_dates[length_training_chronos+t:
    length_training_chronos+t+1])),  #list(tch.
    create_gluonts_dataset2(loaded_data))
probabilities=[1.0 / len(all_timeseries)] * len(
    all_timeseries),
tokenizer=chronos_config.create_tokenizer(),
context_length=length_training_chronos-1,
prediction_length=1,
min_past=50,
model_type="seq2seq",
imputation_method= None,
mode="training",
```

```
).shuffle(shuffle_buffer_length=100)
```

```
trainer = Trainer(
model=model,
args=training_args,
train_dataset=shuffled_train_dataset,)
```

```
trainer.train()
```

C Parameters of the CNN Transformers strategy

We used the following major parameters provided by Guijarro-Ordonnez et al. (2022) we did not change from `https://github.com/gregzanotti/dlsa-public/tree/main/config`

```
        # Major parameters
        mode: "test"  # can be 'test' or 'estimate'
        results_tag: ""  # optional; try not to use
           underscores in this tag, use dashes instead
        debug: False  # set to True to turn on debug
           logging and file naming
        # Model parameters
        model_name: "CNNTransformer"  # name of a class
           defined in models folder and initialized in
           model folder's __init__.py
        model: {  #  contains parameter settings for
           __init__() function of class with name `
           model_name`
                lookback: 30,  # number of days of
                   preprocessed residual time series to
                   feed into model
                dropout: 0.25,
                filter_numbers: [1,8],
                filter_size: 2,
                attention_heads: 4,
                hidden_units_factor: 2,  # multiplicand
                   of last item in `filter_numbers`;
```

```
            determines number of hidden units (e.g. 2*8 = 16)
        # hidden_units: 16,  # use either hidden_units or hidden_units_factor, but not both
        normalization_conv: True,  # normalize convolutions or not
        use_transformer: True,
        use_convolution: True,
    }
    # Data parameters
    preprocess_func: "preprocess_cumsum"  # name of a function defined in preprocess.py
    use_residual_weights: False  # use residual composition matrix to compute turnover, short proportion, etc.
    cap_proportion: 0.01  # defines asset universe: 0.01 corresponds to a residual data set
    factor_models: {  # number of factors per residual time series to test, for each factor model
        "IPCA": [5],
        "PCA": [5],
        "FamaFrench": [5],
    }
    perturbation: {  # perturbation of residual time series by noise is optional, leave empty or comment out entirely to disable
        # "noise_type" : "gaussian",
        # "noise_mean" : 0.0,
        # "noise_std_pct" : 2,
        # "noise_only" : False,
        # "per_residual" : True,
    }
    # Training parameters
    num_epochs: 100
    optimizer_name: "Adam"  # see PyTorch docs for potential optimizers
    optimizer_opts: {  # see PyTorch docs for optimizer options
```

```
            lr: 0.001
        }
        batch_size: 125
        retrain_freq: 125  # if mode=='estimate', this is the number of obs used to form a test set (chronologically after the training set)
        rolling_retrain: True  # set to False for no rolling retraining (i.e. train once, test for all data past training set)
        force_retrain: True  # force the model to be trained, even if existing weights for the model are saved on disk
        length_training: 1000  # size of rolling training window in trading days
        early_stopping: False  # employ early stopping or not
        objective: "sharpe"  # objective function: 'sharpe' or 'meanvar' or 'sqrtMeanSharpe'
        # Market frictions parameters
        market_frictions: False  # enable or disable
        trans_cost: 0  # cost in bps per txn side per equity, e.g. 0.0005
        hold_cost: 0  # cost in bps for short positions per equity per day, e.g. 0.0001
```