

StockGPT: A GenAI Model for Stock Prediction and Trading*

Dat Mai

April 2024

Abstract

This paper introduces StockGPT, an autoregressive “number” model trained and tested on 70 million daily U.S. stock returns over nearly 100 years. Treating each return series as a sequence of tokens, StockGPT automatically learns the hidden patterns predictive of future returns via its attention mechanism. On a held-out test sample from 2001 to 2023, a daily rebalanced long-short portfolio formed from StockGPT predictions earns an annual return of 119% with a Sharpe ratio of 6.5. The StockGPT-based portfolio completely spans momentum and long-/short-term reversals, eliminating the need for manually crafted price-based strategies, and also encompasses most leading stock market factors. This highlights the immense promise of generative AI in surpassing human in making complex financial investment decisions.

Key words: generative artificial intelligence, transformer, decoder, stock market, investment, trading, return prediction

1 Introduction

Generative artificial intelligence (GenAI)—a set of advanced technologies capable of generating texts, images, videos, programming codes, or arts from instructions via sounds or texts—has taken the society by storm and exerted wide-range influences on many aspects of the world economy (Baldassarre et al. 2023; Mannuru et al. 2023; Sætra 2023). Although it had been around for years, GenAI came to public prominence since the introduction of ChatGPT in November 2022, a chatbox able to generate answers, reasoning, and conversations at human level.

Since its introduction, ChatGPT and similar large language models have quickly made their ways into the investment industry. One common use of ChatGPT for investment is to give trading recommendations directly from news about a company (such as news articles or corporate communications) (Lopez-Lira and Tang 2023). A less direct approach is to rely on similar pretrained language models such as BERT (Devlin et al. 2018) and OPT (Zhang et al. 2022) to generate a sentiment score for each company which is then used to make trading decisions. For example,

*I would like to thank Andrej Karpathy for publicly sharing his lecture and code on the GPT architecture. Dat Mai (datmai@mail.missouri.edu) is affiliated with the University of Missouri-Columbia and MKT MediaStats, LLC (www.mktmediastats.com). The views expressed are solely of the author.

Jiang, Kelly, and Xiu (2022) and Kirtac and Germano (2024) find that stock portfolios formed on sentiment scores generated by BERT and OPT have impressive performance.

This paper contributes to this fast-evolving field first by training a new Generative Pretrained Transformer (GPT) model (Brown et al. 2020) from scratch on numeric stock data (StockGPT) and then by showing that StockGPT produces strong investment performance.¹ Unlike previous finance domain-specific language models that are pretrained on financial *texts* such as FinBERT (Yang, UY, and Huang 2020) and BloombergGPT (Wu et al. 2023), to the best of my knowledge, StockGPT is the first of its kind to be pretrained directly on *numeric* stock return data.² For the trading purpose, using a model trained directly on stock data has three important advantages over models trained on texts: (i) the model learns price patterns directly from price data rather than from news about prices, and (ii) the model predictions are available for each stock at each time point rather than dependent on the availability of news data about stocks, and (iii) the model predicts the whole distribution of future returns rather than a point estimate.

Language models such as GPT operates by predicting the next most likely token given the previous ones, $p(x_{t+1}|x_t, \dots x_1)$. This nature bears a strong resemblance to numeric time series data such as stock returns where data points come in order and the next value is conditional on what comes before it. Hence the natural question is whether the architecture of language models can be applied to numeric time series data. To do so, one fundamental difference between texts and numbers needs to be addressed: texts are a collection of (vast but) discrete tokens while numeric time series are generally continuous. Therefore, to train a generative model for stock returns, I first discretize stock return data into intervals (or “tokens”) then apply the language model architecture.

To build the StockGPT model, I adapt a light-weight version of the GPT architecture, which consists of four attention blocks having about one million parameters. Input into the model is a sequence of 256 consecutive daily returns on each stock (i.e., the block size in language models) which approximates the number of trading days in a year.³ The training objective is to predict the next return value given its previous returns using the transformer architecture, which receives indexes (or positions) of the tokens in a sequence, retrieving their vector representations, and modelling their dependencies via a mechanism called “attention” (Vaswani et al. 2017). The training sample consists of around 50 million daily U.S. stock returns from 1926 to 2000, which covers almost all stocks that have ever been listed on the U.S. stock market during the 20th century. The model is tested on a hold-out sample of around 20 million daily U.S. stock returns from 2001 to 2023.

Notably, the model is trained only once using the training sample and applied off-the-shelf to the out-of-sample period. This study design serves two purposes: (i) it is the cleanest setup to test the effectiveness of the model and (ii) it reduces the computational costs. Despite this simple setup, the model still delivers strong performance up to 23 years after the period it is trained on. In practice, the model should be continually retrained with the arrival of new financial data to uphold its relevance and performance. This is especially needed in a dynamic environment like the stock

¹ChatGPT is GPT finetuned for the conversational purpose.

²Return on day t is computed as $return_t = \frac{price_t + dividend_t}{price_{t-1}} - 1$

³It is a convention in machine learning to specify model parameters in powers of 2.

market featured by a low signal to noise ratio and constantly distributional shifts (Kelly, Xiu, et al. 2023).

During the testing phase, for each stock on each trading day t , StockGPT uses 256 daily returns from $t - 255$ to t to make a return forecast for $t + 1$. The evaluation of the forecasts consists of two steps. First, I examine the accuracy of the forecasts by running cross sectional regressions of realized stock returns on day $t + 1$ onto return predictions for $t + 1$.⁴ The results indicate that StockGPT makes fairly accurate predictions.

The second evaluation step entails building real time trading portfolios based on StockGPT forecasts. At the market close of each trading day t , I build zero cost portfolios by going long/short the top/bottom decile of stocks having the highest/lowest return forecasts for day $t+1$ and rebalance the portfolio at the $t + 1$ market close. To avoid trading only micro stocks since these stocks are illiquid and incur high transaction and market impact costs, before forming the portfolio, I remove stocks below the 10th percentile market value at the market close.⁵

Under the equal-weighting scheme where each stock receives an equal weight in the portfolio, this daily rebalanced long-short portfolio earns an average annualized returns of 119% from 2001 to 2023, achieving a Sharpe ratio of 6.5. This performance is much higher than the best performing portfolio based on language model predictions in Jiang, Kelly, and Xiu (2022), which has an annual return of 50% and Sharpe ratio of 4.8 from 2004 to 2019. It is noteworthy that while the prediction model in Jiang, Kelly, and Xiu (2022) is retrained every year,⁶ StockGPT is trained only once using data up to 2000. Under the value weighting scheme where each stock weights in the portfolios are proportional to its market value, the StockGPT-based portfolio achieves an average annualized return of 27% and a Sharpe ratio of 1. Since value weighting gives more weight to stocks having higher market values, this result is consistent with the consensus view in asset pricing that small stocks are more predictable due to higher arbitrage costs and illiquidity.

While the annual return of 119% under equal weighting is before transaction costs, under the hypothetical worst-case scenario that the portfolio replaces all of its constituents every day (i.e., a turnover of 400% in a long-short portfolio) and each trade costs 5 basis points, the StockGPT-based strategy still realizes an annual return of 69% net of transaction costs. Under further restrictions by removing stocks having prices under \$1, \$3, and \$5, the strategy delivers gross annual returns of 110%, 86%, and 74% with Sharpe ratios of 6.3, 5.2, and 4.7, respectively. If one day is skipped between return forecasts and portfolio formation, the StockGPT-based portfolio will earn 26% annually with a Sharpe ratio of 1.7. Overall, StockGPT showcases its ability in crafting highly profitable stock trading strategies.

Since StockGPT makes its return forecasts using historical price data, I examine how it relates to common price-based strategies such as momentum and long-/short-term reversals. I find that the StockGPT-based portfolio completely spans these strategies via the spanning test. The StockGPT-

⁴Here, “cross section” refers to the sample of different stocks (or companies) available on each day.

⁵Including micro-cap stocks yields better performance.

⁶Specifically, they retrieve contextual word embeddings from pretrained OPT and BERT and use these embeddings to retrain the return prediction model every year.

based portfolio also spans most factors of the Fama and French (2015) five factor model and the Hou et al. (2021) q-factor model. That a strategy based solely on historical price data delivers such a strong future performance poses a strong challenge to the market efficiency hypothesis of Fama (1970) who argues that the stock market is efficient so historical prices cannot be used to consistently predict future returns.

This paper make contributions to several fields of research. First, it contributes to the fast growing field of training large language models for specific domains such as finance (Wu et al. 2023) and medical (Wang et al. 2023). Unlike these models which are trained on texts, StockGPT is trained directly on numeric stock data for the trading purpose. Second, this paper contributes to the emerging research area of adapting language models to numeric time series analysis (see, for example, Ansari et al. (2024) and the references therein). While these paper attempt to build pretrained foundational models which can be applied to unseen datasets without finetuning, StockGPT is custom built from stock data for stock prediction. Third, this paper makes contribution to the investment literature of applying machine learning techniques to return prediction. For example, Gu, Kelly, and Xiu (2020) apply deep learning methods on firm characteristics to make return prediction while Jiang, Kelly, and Xiu (2023) forecast returns using price graphs via convolutional neural networks. This paper differs by treating stock prices as text tokens and applying a generative AI architecture. Finally, this paper contributes to the asset pricing literature that uses historical price patterns such as momentum (Jegadeesh and Titman 1993), reversals (De Bondt and Thaler 1985; Jegadeesh 1990), and moving averages (Han, Yang, and Zhou 2013) to predict future price movements. Instead of relying on manually crafted patterns, StockGPT automatically extracts hidden patterns predictive of future returns via the state-of-the-art transformer architecture.

2 Model Architecture

2.1 Overview

StockGPT uses a vanilla decoder-only transformer architecture which is the second step of the canonical transformer model developed by Vaswani et al. (2017). The decoder-only transformer is also the architecture of ChatGPT. Figure 1 depicts the sketch of the architecture. Specifically, the decoder receives an input sequence of tokens $x = (x_1, x_2, \dots, x_{t-1}, x_t)$, transforms its via multiple layers of attention, and outputs the probability of each next token, $p(x_2|x_1), p(x_3|x_2, x_1), \dots, p(x_{t+1}|x_t, \dots, x_1)$.

During the training phase, the model learns and updates its parameters via minimizing the cross-entropy loss between a token prediction and its actual value $l(\hat{x}_{t+1}, x_{t+1})$ averaging across all tokens across all sequences in a training batch. During the deployment phase, the decoder generates an output sequence of tokens $(x_{t+1}, x_{t+2}, \dots, x_{t+m})$ one at a time, given the input sequence x . Specifically, it receives an input sequence $(x_1, x_2, \dots, x_{t-1}, x_t)$, converts it into a conditional probability distribution $p(x_{t+1}|x_t \dots x_1)$, and generates the next token from this distribution. The decoder model is “autoregressive” in the sense that its consumes its own generated output at

each time as additional input to generate the next one, i.e., $p(x_{t+2}|\hat{x}_{t+1}, x_t, \dots, x_1)$ where \hat{x}_{t+1} is previously generated.

2.2 Details

Since computer itself does not understand human texts, the transformer first quantifies text tokens via token and positional embedding. Token embedding simply retrieves a unique vector representation for each token in a dictionary containing all available tokens. Positional embedding represents each token *position* by a vector. Without positional embedding, the transformer cannot understand the context and order of tokens. The transformer then sums up token and positional embedding vectors for each token. These embeddings are learnable parameters.

At the heart of the transformer model is the attention mechanism. Accordingly, for each token, the transformer generates three vectors from its embedding vector: key k , query q , and value v . The attention for each token is the weighted sum of its v with all v 's of tokens preceding it, weighted by the product of its q with k 's of those tokens. Intuitively, a token emits a query and the previous tokens that can match its query (i.e., having a high $q \times k$ value) get its attention. k , q , and v are also learnable parameters.⁷ This mechanism constitutes a “self-attention” head and helps the transformer develop a contextual understanding of the tokens. Since each token is only influenced by the tokens before it, this setup is autoregressive.

The transformer concatenates multiple attention heads into a multi-head node which is sequentially followed by multiple linear layers to form an attention block.⁸ Multiple attention blocks are then stacked on top of each other. The last attention block is followed by a layer normalization and a linear layer whose output is converted into a vector of probabilities via a softmax activation. Specifically, at time step t , the transformer outputs $p(x_{t+1}|x_t \dots x_1)$ which is the multinomial distribution over all available tokens in the dictionary. Given this distribution, the model can sample the most likely token at $t + 1$ from its dictionary.

2.3 StockGPT Specifics

Since the transformer can only work with text tokens, to use it on continuous stock return data, the first step is to discretize returns into intervals. Table 1 illustrates the discretization rule. Accordingly, I first convert returns into integer basis points by multiplying them by 10,000 and keeping the integer portion. Next, I cut the basis points into intervals of 50, closed on right. The first interval closes on -10,000. Since stock prices cannot be negative, returns cannot be lower than -10,000 basis points (i.e., -100%); therefore, the first bin contains only -10,000. The last closed

⁷Technically speaking, the model learns the weight matrices that produce these vectors.

⁸Specifically, for StockGPT, in the first step of the attention block, the input goes through a layer normalization, then multi-head concatenation, followed by a linear layer with dropout. The output of this layer is added up to the input via a skip connection. In the second step, the output of the first step goes through a second layer normalization, followed by a linear expanding layer to increase the input dimension by 4 times, a ReLU activation, and a contracting layer to revert to the input dimension with dropout. Again the output of this second step is added up to its input via a second skip connection.

interval is (9,950, 10,000]. Second, for each bin, I use the mid value of each interval to represent its value with the exception that the first bin (-Inf, -10,000] is represented by -10,000 and the last bin (10,000, Inf) by 10,000. In other words, I treat all daily returns greater than 100% as 100%. Values above this threshold are extremely rare since the 1th to 99th percentile of daily returns in the training set is from -9.6% to 11.1%. Finally, the bins are numbered from 0 to 401. Therefore, my return “dictionary” has a total of 402 tokens where each token is a return bin midpoint. As an example of the discretization rule, the following return sequence (-2.4%, 0%, 0%, 5%, 4.8%) is converted into the index sequence (196, 200, 200, 210, 210) which is input into StockGPT.

Besides the vocabulary size of 402, StockGPT has a block size (i.e., length of each input sequence) of 256, token and positional embedding sizes of 128, 4 attention blocks each consisting of 4 self-attention heads, and a dropout probability of 0.2 in various layers. Taken together, StockGPT has 0.93 million parameters. StockGPT is trained in 10,000 training steps with each step consisting of 64 sequences (i.e, batch size) drawn randomly from the training data.⁹ The probability of sampling each stock during training is proportional to the number of daily return observations it has.

As discussed above, to make a return forecast, given a 256-return sequence input (x_{t-255}, \dots, x_t) , StockGPT will output $p(x_{t+1}|x_t, \dots, x_{t-255})$, a multinomial distribution over 402 return bins.¹⁰ The model produces output in terms of bin indexes which are converted to numeric returns using the bin midpoints in Table 1. The expected return for day $t + 1$ will then be the weighted average of return bin midpoints weighted by the corresponding bin probabilities presented by $p(x_{t+1}|x_t, \dots, x_{t-255})$. Alternatively, the expected returns on day $t + 1$ can be computed by sampling many forecasts from $p(x_{t+1}|x_t, \dots, x_{t-255})$ and averaging them. The two approaches will produce the same results if the number of drawn samples is large but the latter approach is more computationally intensive. To make return forecasts over the next m days, we can recursively sample several paths of forecasts $x^j = (x_{t+1}, x_{t+2}, \dots, x_{t+m})$ and average across the paths.

3 Data

Stock return data comes from Center for Research in Security Prices (CRSP) that collects all historical U.S. stock returns from 1926 to 2023. As standard in asset pricing research, I include only common stocks with a share code of 10 or 11 traded on three main exchanges NYSE, AMEX and NASDAQ. This sample consists of around 70 million stock observations from 1926 to 2023. This sample is then split into two parts: the sample from 1926 to 2000 for training and the sample from 2001 to 2023 for testing. Within the training sample, data from 1926 to 1990 is used for parameter optimization and data from 1991 to 2000 for hyperparameter tuning and evaluation.

During training evaluation, I document that the model using stocks from NYSE alone has lower

⁹During the training phase, the cross-entropy loss stabilizes at around 2.5 after 5,000 training steps. With 402 labels (the number of return bins), the maximum cross-entropy would be $E = -\sum_i \log(1/402) \times (1/402) = 6$. The model is fully trained locally on a MacBook M2 with 64GB RAM and 30 GPU cores.

¹⁰Technically, input of sequence of any length from 1 to 256 (i.e., the block size during training) can be used to make forecasts. However, since StockGPT is trained with block size of 256, I also use input sequence of 256 days in making forecasts to utilize all price patterns the model has discovered during training.

evaluation loss than the one using all three exchanges, 2.55 versus 2.72. This may be because NYSE is the world largest stock exchange that lists high quality large cap stocks while AMEX and NASDAQ list smaller stocks that add noises to the training process. Therefore, the main results focus on the model trained on NYSE data alone while the results using the model trained on all three exchanges are reported in [Table A.1](#). The latter model still produces annual returns of 83% with Sharpe ratio of 5.

During the testing phase, stock returns from all three exchanges are used. As noted in the introduction, the model is trained only once using the training sample and kept unchanged during the testing phase.

4 Results

4.1 Daily Returns

To evaluate the quality of return forecast for day $t + 1$, I first compare it against the actual return on that same day. Specifically, for each trading day t , I run the following cross-sectional regression

$$x_{it+1} = a_t + b_t \times \hat{x}_{it+1} + e_{it+1} \quad (1)$$

where x_{it+1} is the actual realized return of stock i on day $t + 1$ and \hat{x}_{it+1} is its StockGPT return forecast. The slope b_t and regression adjusted R_t^2 are then averaged across all trading days in the test sample. These measure how well StockGPT forecasts track the actual returns. This regression specification is referred to as the Fama-MacBeth regression in the asset pricing literature (Fama and MacBeth 1973).

[Table 2](#) reports the results. Accordingly, the average slope coefficient is 0.5, indicating that a cross-sectional difference of 100 basis points (i.e., 1%) in StockGPT return predictions signals a difference of 50 basis points in realized returns. Moreover, the average cross-sectional R^2 is 1.2% equivalent to a 11% cross-sectional correlation between return predictions and actual returns. For comparison, the average correlation between return forecasts based on language models and actual returns is around 2% in Jiang, Kelly, and Xiu (2022). I also examine the relation between return forecasts for day $t + 1$ and realized returns on day $t + 2$ (i.e., skipping one day). For this test, the slope coefficient is 0.09 and R^2 is 0.4% which translates into a 6% correlation. The slopes in both tests are highly significant with t -statistics over 10. Overall, GPT forecasts track future returns well even after one day.

The main empirical analysis is to examine the trading implications of StockGPT forecasts. To do so, on each trading day t , I buy the top stock decile having the highest return forecasts for $t + 1$ (High portfolio) and sell the bottom decile having the lowest return forecasts for $t + 1$ (Low portfolio). To avoid only trading micro-cap stocks, I remove stocks having market value below the 10th percentile each day. In [Table 3](#), under equal weighting (EW), this long-short portfolio yields an average annual return of 119% with a Sharpe ratio (mean/standard deviation) of 6.5.

If I remove stocks having prices below \$1, \$3, and \$5, the mean returns (and Sharpe ratios) are 110% (6.3), 86% (5.2), and 74% (4.7), respectively. The left graph in Panel A of [Figure 2](#) plots the *log* cumulative returns of these 4 long-short portfolios. These portfolios show a consistent upward trend throughout the 2001-2023 sample with the biggest jump in 2009 after the financial crisis. The right graph in Panel A plots the cumulative returns of each long/short leg of the portfolios. It is clear that StockGPT can symmetrically predict both rising and falling stocks.

Under value weighting (VW), the long-short portfolio without price filter (but still after removing the bottom decile based on market cap) earns an annual returns of 27% and Sharpe ratio of 1. The Sharpe ratio of the portfolios with price filters are at 1, 0.9, and 0.8 for the \$1, \$3, and \$5 price thresholds, respectively. Since value weighting gives more weight to large cap stocks, this result indicates that StockGPT is more effective at forecasting returns of small cap stocks. This is expected since small cap stocks are more likely to be mispriced.

[Table 3](#) also reports the portfolio when return forecasts for $t + 1$ are used to form portfolio for $t + 2$. Under equal weighting, this skipping-one-day portfolio earns 26% annually with a Sharpe ratio of 1.7. Panel B of [Figure 2](#) shows that when one day is skipped, StockGPT forecasts track the returns in the long leg better than the short one.

4.2 Relation to Stock Factors

Since StockGPT uses only historical market price data to make return forecasts, it is important to examine how the StockGPT-based portfolio relates to prominent trading strategies based on historical returns. The three most notable patterns are short-term reversal (using returns from month $t - 1$) by Jegadeesh ([1990](#)), momentum (using returns from month $t - 2$ to $t - 12$) by Jegadeesh and Titman ([1993](#)), and long-term reversal (using returns from month $t - 13$ to $t - 60$) by De Bondt and Thaler ([1985](#)). It is also interesting to examine how StockGPT performs relative to leading stock factors such as the five factors of Fama and French ([2015](#)) and the investment-based q5 factors of Hou et al. ([2021](#)).¹¹

As standard in asset pricing research, to examine whether a strategy earns abnormal returns relative to a set of other traded factors, we can perform the following contemporaneous regression:

$$y_t = \alpha + \beta \times x_t + e_t \quad (2)$$

where y_t is the return of the target strategy and x_t is the set of benchmark factors. If α is significant, then y_t earns abnormal returns relative to x_t ; otherwise, y_t is spanned or encompassed by x_t . This test is also referred to as the spanning test.

Panel A of [Table 4](#) reports the results of the spanning tests in which y_t is the daily returns on the StockGPT-based portfolios and x_t is the set of benchmark factors. Accordingly, both the equal-weighted and value-weighted StockGPT portfolios earn sizable and highly significant alphas relative

¹¹The momentum, reversal, and five factors of Fama and French ([2015](#)) are available at https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html while the q5 factors of Hou et al. ([2021](#)) are at <https://global-q.org/factors.html>.

to all 11 benchmark factors. In Panel B, I test whether the StockGPT portfolios span the benchmark factors. The equal-weighted StockGPT portfolio spans momentum, long-term reversal, value, and size while the value-weighted StockGPT portfolio spans 9 out of 11 factors except profitability from Fama-French and earning growth from q5 models. That the value-weighted StockGPT portfolio better spans the other factors than does the equal-weighted StockGPT portfolios is expected since those factors are also value-weighted.

Overall, the spanning tests show that when we let the stock data speak for itself via the attention mechanism in StockGPT, handcrafted price-based strategies such as short-term reversal, momentum, and long-term reversal are no longer needed. Notably, although StockGPT only learns from historical returns over the past 12 months, it completely encompasses the long-term reversal pattern based on returns beyond the past 12 months. Furthermore, StockGPT-based portfolios also encompass most leading stock factors.

4.3 Monthly Returns

While StockGPT delivers superb performance, it requires daily trading and rebalancing. For certain investors, this is not feasible. Hence, the arising question is whether StockGPT can be used to make longer term forecasts to build lower frequency portfolios.

There are two ways to produce long-term return forecasts over the next m days from StockGPT. The first approach is to produce several paths of forecasts $x^j = (x_{t+1}, x_{t+2}, \dots, x_{t+m})$ and average across the paths to compute the expected returns over the next m days. However, this approach is very computationally expensive since there are on average 3,000 to 4,000 stocks traded every day. For each stock on each day, we need to make many m -day forecast paths and average them.

The second approach is to train a new StockGPT model where the training target is to predict return over the next m days (i.e., $p(\bar{x}_{t+1 \rightarrow t+m} | x_t, \dots, x_{t-255})$ where $\bar{x}_{t+1 \rightarrow t+m}$ is mean return over the next m days). I pursue this approach in this section. Specifically, I train a StockGPT model using historical returns to predict mean returns over the next 20 days (i.e., 20 days approximates the number of trading days in a month) with all other specifications kept unchanged from the daily model. During the testing phase, at the end of each month for each stock, an input sequence of 256 previous daily returns for that stock is used by the new StockGPT model to predict the next 20-day returns (i.e., return over the next month).

To evaluate the quality of long-term forecasts by StockGPT, I regress actual realized returns onto 20-day return forecasts via the Fama-McBeth test discussed above. As reported in [Table 5](#), the average slope coefficient is 3 (significant at 5%), indicating that a cross-sectional difference of 100 basis points (i.e., 1%) in StockGPT return forecasts signals a difference of 300 basis points in realized returns. Moreover, the average cross-sectional R^2 is 0.55% equivalent to a 7.4% cross-sectional correlation between return predictions and actual returns. When one month is skipped between 20-day return forecasts and realized returns, the correlation shrinks to zero.

I then form monthly long-short decile portfolios using the 20-day return forecasts. The monthly long-short portfolios after removing the bottom 10th percentile stocks based on market value are

reported in Table 6. The equal-weighted long-short monthly portfolios earn about 13% annually, significant at 1%, with Sharpe ratios around 1. Meanwhile, the value-weighted long-short portfolios earn about 9% to 11% annually, with Sharpe ratios around 0.4-0.5. If one month is skipped between return forecasts and portfolio formation, the returns and Sharpe ratios of the equal-weighted portfolios are halved while those of the value-weighted portfolios remain more stable.

Figure 3 plots the *log* cumulative returns on the equal-weighted portfolios. The long-short portfolios see a stable upward trend from 2001 to 2023. Consistent with the results in Table 5, when one month is skipped, the portfolios lose their performance, mostly going flat in the second half of the test sample. Whether one month is skipped or not, StockGPT does better at predicting stocks in the long leg.

Overall, while the monthly StockGPT model still produces profitable trading strategies, the magnitude of monetary profits is much smaller than that of the daily model. This suggests that autoregressive models such as StockGPT do the best when predicting the next token in the sequence. Moreover, under monthly rebalancing, we have to wait for one month before updating the portfolios, potentially missing out short-term profitable opportunities.

5 Conclusion

This paper introduces StockGPT, a decoder-only transformer model trained directly on U.S. stock returns. Instead of relying on manually crafted trading patterns using historical stock prices, StockGPT automatically learns the hidden patterns most predictive of future returns via its attention mechanism. Even though trained on daily returns only up to 2000, StockGPT delivers strong trading performance up to 23 years later. The StockGPT-based portfolios encompass common price-based trading strategies such as momentum and long-/short-term reversals and span most leading stock factors such as market, size, value, and investment.

StockGPT can be enhanced in several ways. First, StockGPT should be retrained frequently with the arrival of new stock data to maintain its performance. Second, StockGPT as introduced in this paper is a light-weight model with only around one million parameters. The natural extension is to examine bigger models having more granular return intervals, longer batch size, bigger embedding size, and more layers of attention blocks. Third, examining the long-term forecasts from the daily StockGPT model can be a fruitful direction. Finally, training StockGPT with higher frequency data such as minutely returns can yield interesting results.

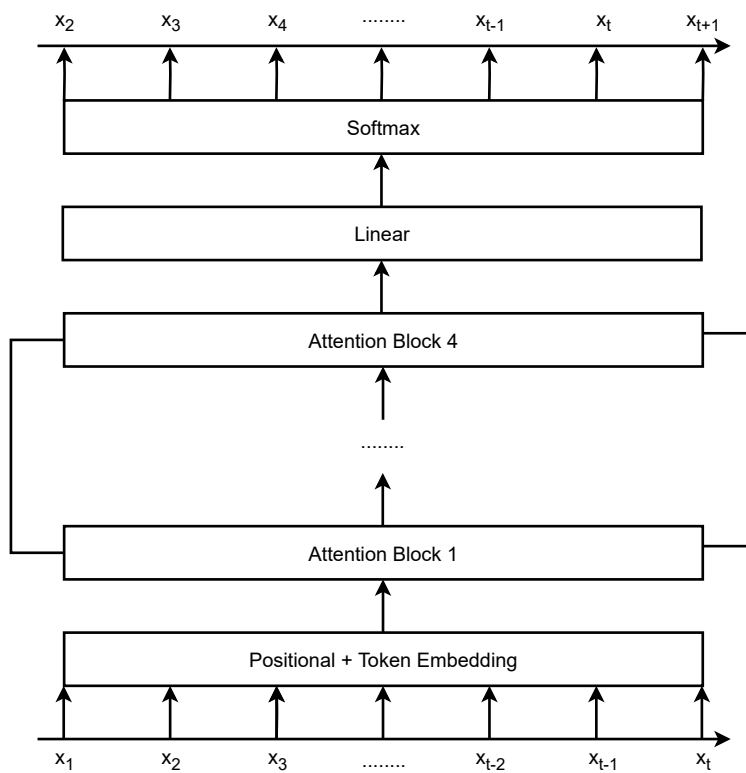
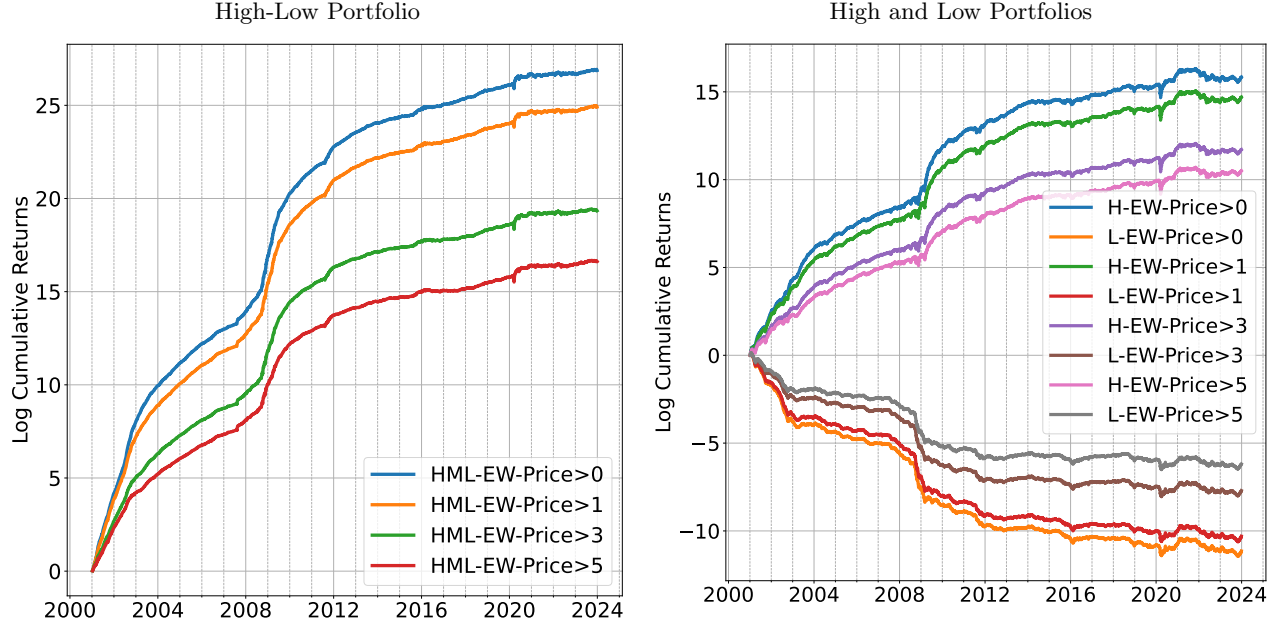


Figure 1: StockGPT Architecture

Panel A: Next Day



Panel B: Skipping 1 Day

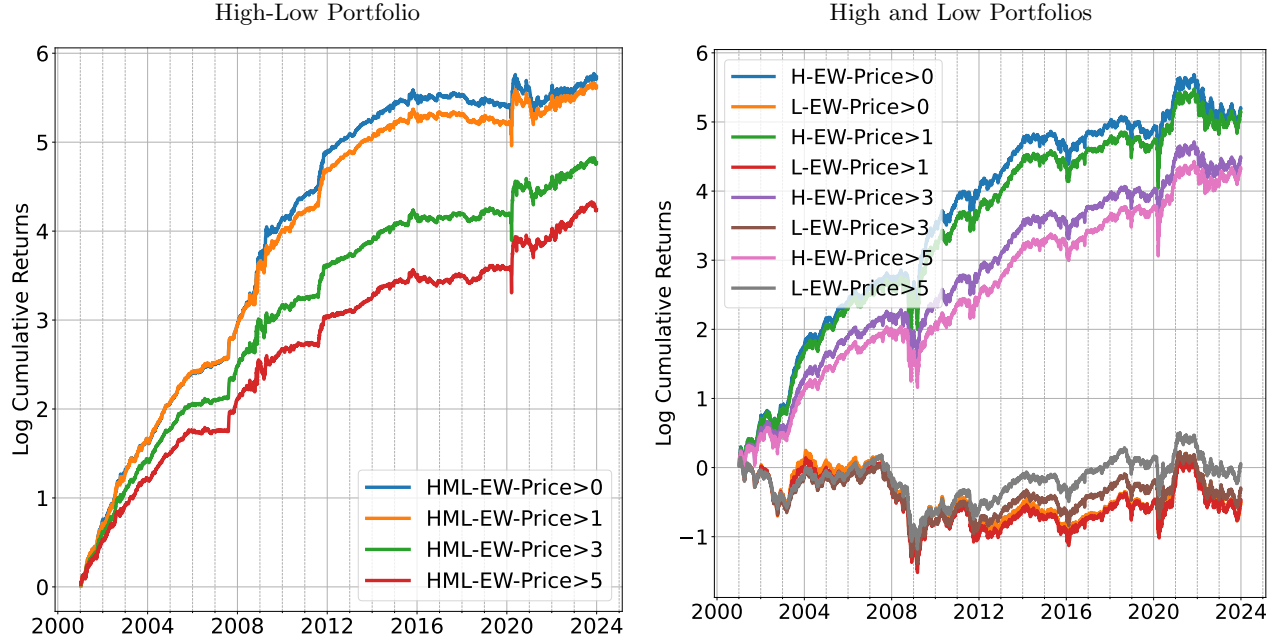
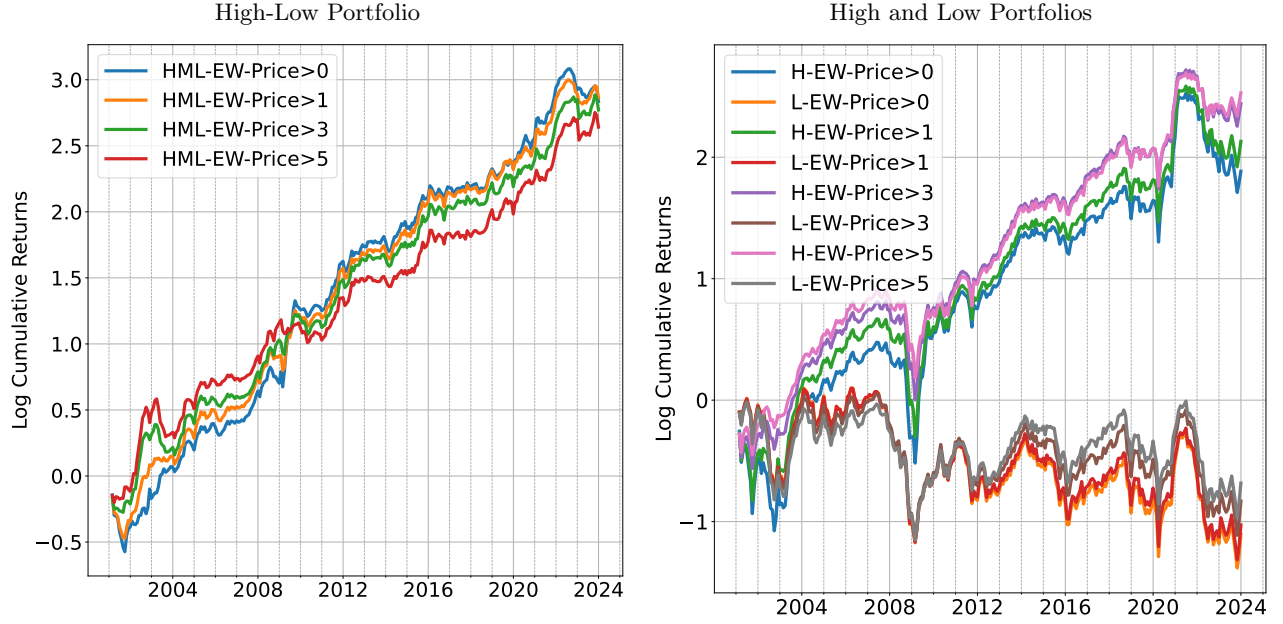


Figure 2: Daily Cumulative Returns

This figure plots the *log* cumulative returns of long-short high-minus-low (HML), high (H), and low (L) portfolios formed from StockGPT return forecasts. Portfolios are equal-weighted. Portfolios are formed after excluding stocks in the bottom decile based on market value. Different portfolios correspond to different market price thresholds under which stocks are excluded. Panel A (B) shows results when return forecasts for day $t+1$ are used to formed portfolios for day $t+1$ ($t+2$). The left (right) panel shows results for the long-short (each leg) portfolios. The sample is daily from January 2001 to December 2023.

Panel A: Next Month



Panel B: Skipping 1 Month

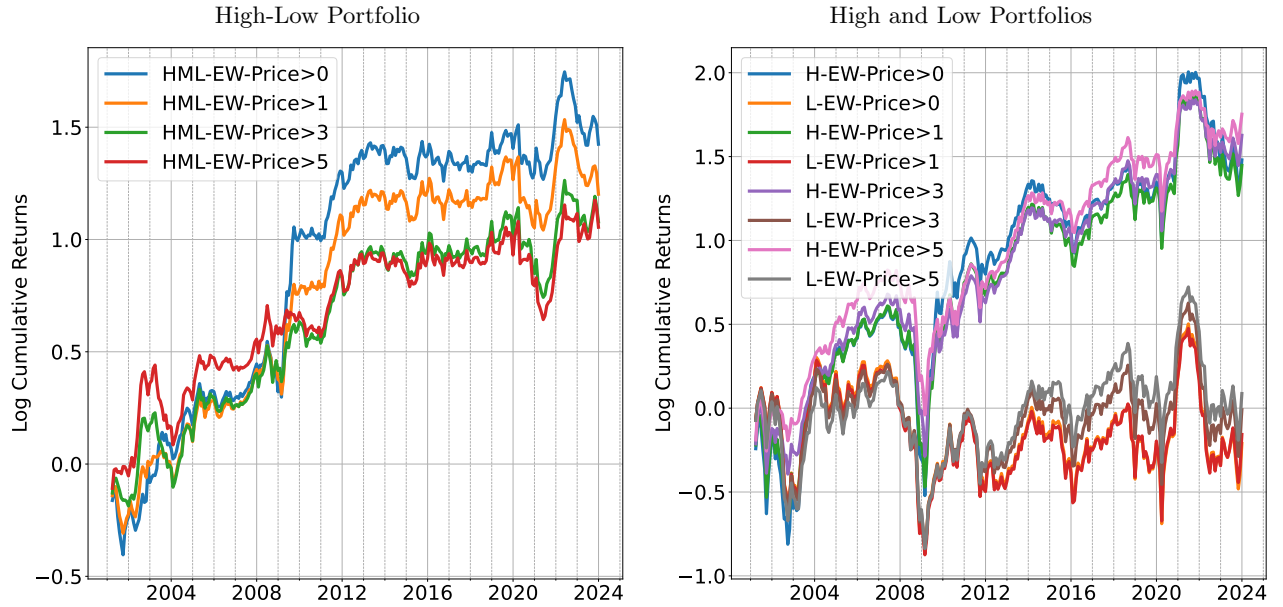


Figure 3: Monthly Cumulative Returns

This figure plots the log cumulative returns of long-short high-minus-low (HML), high (H), and low (L) portfolios formed from StockGPT return forecasts. Portfolios are equal-weighted. Portfolios are formed after excluding stocks in the bottom decile based on market value. Different portfolios correspond to different market price thresholds under which stocks are excluded. Panel A (B) shows results when return forecasts for month $t + 1$ are used to formed portfolios for month $t + 1$ ($t + 2$). The left (right) panel shows results for the long-short (each leg) portfolios. The sample is monthly from February 2001 to December 2023.

Table 1: Return Bins

This table shows the return bins in basis points, bin midpoints, and the corresponding bin indexes.

Return Bin	Bin Midpoint	Bin Index
(-Inf, -10_000]	-10_000	0
(-10_000, -9_950]	-9_975	1
(-9_950, 9_900]	-9_925	2
...
(9_950, 10_000]	9_975	400
(10_000, +Inf)	10_000	401

Table 2: Daily Fama-MacBeth Regression

This table reports the time series averages of slopes and adjusted R^2 's of the following cross-sectional regression

$$x_{it+1} = a_t + b_t \times \hat{x}_{it+1} + e_{it+1}$$

where x_{it+1} is the actual realized return of stock i on day $t + 1$ and \hat{x}_{it+1} is its StockGPT return forecast. Returns are in basis points and R^2 in percentage points. t is the t -statistic of time-series mean of b_t computed using Newey and West (1987) standard error with 20 lags. Horizon 1 (2) means comparing return forecasts for $t + 1$ with actual returns on $t + 1$ ($t + 2$). The sample is daily from January 2001 to December 2023.

Horizon	b	t	R^2
1	0.50	25.18	1.19
2	0.09	10.20	0.41

Table 3: Daily Portfolio Statistics

This table reports the return statistics of the daily long-short StockGPT-based portfolios. Mean and SD (standard deviation) are in annualized percentage points; Mean/SD (Sharpe ratio) is annualized; Min, Max, and MDD (max drawdown) are in percentage points; and t-Mean is t -statistic of the mean portfolio return using Newey-West standard error with 20 lags. Portfolios are formed after excluding stocks in the bottom decile based on market value. Horizon 1 (2) refers to using return forecasts for day $t + 1$ to form portfolios for day $t + 1$ ($t + 2$). EW (VW) refers to equal-weighting (value weighting). Price Filter refers to the price level under which stocks are removed. The sample is daily from January 2001 to December 2023.

Horizon	Weight	Price Filter	Mean	t-Mean	SD	Mean/SD	Min	Max	MDD
1	EW	0	119.1	13.6	18.2	6.5	-9.8	18.3	-23.7
1	EW	1	110.4	13.8	17.5	6.3	-9.4	18.0	-25.4
1	EW	3	85.8	13.6	16.4	5.2	-8.9	17.2	-28.7
1	EW	5	73.8	13.6	15.8	4.7	-9.0	15.3	-28.5
1	VW	0	27.0	4.3	26.4	1.0	-15.0	20.3	-76.3
1	VW	1	25.4	4.3	25.6	1.0	-13.9	19.8	-74.2
1	VW	3	21.3	3.9	24.0	0.9	-13.2	17.2	-69.8
1	VW	5	18.8	3.7	23.2	0.8	-12.9	15.1	-73.0
2	EW	0	26.1	7.6	15.1	1.7	-7.3	15.4	-35.4
2	EW	1	25.6	7.7	14.8	1.7	-7.2	16.0	-33.8
2	EW	3	21.9	7.1	14.6	1.5	-8.6	14.8	-30.5
2	EW	5	19.5	6.7	14.3	1.4	-8.8	14.3	-26.5
2	VW	0	5.8	1.2	25.4	0.2	-16.6	18.7	-62.6
2	VW	1	5.8	1.2	24.7	0.2	-15.4	17.8	-63.1
2	VW	3	2.6	0.6	23.0	0.1	-15.4	16.1	-58.1
2	VW	5	1.8	0.4	22.0	0.1	-14.8	14.3	-57.9

Table 4: Daily Spanning Tests

This table reports results of the following spanning test

$$y_t = \alpha + \beta \times x_t + e_t$$

In Panel A, y_t is one of StockGPT-based portfolios and x_t are short-term reversal (ST_Rev), momentum (Mom), long-term reversal (LT_Rev), market (MKT), value (HML), size (SMB), profitability (RMW), and investment (CMA) from Fama and French (2015), as well as investment (R_IA), return on equity (R_ROE), and earnings growth (R_EG) from Hou et al. (2021). In Panel B, y_t is one of the factors and x_t is one of StockGPT-based portfolios. α is in annualized percentage points and R^2 is in percentage points. t_β is computed with Newey-West standard error using 20 lags. The sample is daily from January 2001 to December 2023.

Panel A: Stock Factors Span StockGPT

		β	t_β	β	t_β	β	t_β	β	t_β
EW	α	111.12	13.73	112.38	13.76	116.80	13.31	116.53	13.17
EW	ST_Rev	0.57	11.66	0.61	11.14				
EW	Mom	0.06	2.48	0.00	0.08				
EW	LT_Rev	-0.12	-2.33	-0.05	-1.38				
EW	MKT	0.10	5.54			0.23	7.99	0.24	8.32
EW	HML	0.09	1.84			0.05	0.80		
EW	SMB	0.04	0.86			0.01	0.13	0.02	0.26
EW	RMW	-0.02	-0.24			-0.05	-0.94		
EW	CMA	-0.08	-0.54			-0.20	-1.87		
EW	R_IA	0.14	1.16					-0.14	-1.65
EW	R_ROE	-0.13	-1.57					0.03	0.39
EW	R_EG	0.18	2.70					-0.05	-0.46
EW	R^2	27.26		25.74		7.64		7.47	
VW	α	17.41	3.11	18.64	3.25	25.52	4.14	25.60	4.07
VW	ST_Rev	0.85	12.81	0.84	11.45				
VW	Mom	0.09	1.97	0.10	2.27				
VW	LT_Rev	-0.18	-2.56	-0.13	-2.09				
VW	MKT	0.03	1.22			0.21	4.48	0.24	4.76
VW	HML	0.22	2.58			0.14	1.12		
VW	SMB	-0.15	-2.76			-0.21	-2.33	-0.18	-2.33
VW	RMW	0.02	0.25			0.14	1.44		
VW	CMA	-0.10	-0.44			-0.38	-2.07		
VW	R_IA	0.06	0.28					-0.26	-2.01
VW	R_ROE	0.15	1.41					0.40	3.69
VW	R_EG	0.10	1.05					-0.28	-1.55
VW	R^2	24.79		23.37		3.74		4.17	

Panel B: StockGPT Spans Stock Factors

		ST_Rev	Mom	LT_Rev	MKT	HML	SMB	RMW	CMA	R_IA	R_ROE	R_EG
EW	α	-40.19	9.23	5.33	-25.98	-0.89	-0.90	10.20	6.72	6.68	9.37	8.85
EW	t_α	-6.11	1.57	1.44	-3.47	-0.18	-0.26	4.30	3.05	2.66	3.27	2.71
EW	β	0.42	-0.06	-0.04	0.29	0.02	0.03	-0.05	-0.04	-0.04	-0.05	-0.03
EW	t_β	9.19	-1.92	-2.13	7.51	0.59	1.39	-4.75	-4.05	-3.50	-3.59	-1.82
EW	R^2	25.69	0.45	0.66	7.28	0.04	0.27	1.27	1.26	1.09	0.96	0.60
VW	α	2.17	1.96	1.21	5.30	0.74	3.03	4.50	2.61	2.38	3.45	5.10
VW	t_α	0.75	0.48	0.52	1.28	0.26	1.46	2.46	1.66	1.39	1.52	2.64
VW	β	0.27	-0.00	-0.04	0.12	0.01	-0.02	-0.00	-0.02	-0.02	0.01	-0.00
VW	t_β	8.18	-0.14	-2.50	3.32	0.53	-1.51	-0.12	-2.99	-2.65	0.99	-0.28
VW	R^2	22.82	-0.01	0.95	2.38	0.05	0.28	-0.02	0.85	0.65	0.12	0.00

Table 5: Monthly Fama-MacBeth Regression

This table reports the time series averages of slopes and adjusted R^2 's of the following cross-sectional regression

$$x_{it+1} = a_t + b_t \times \hat{x}_{it+1} + e_{it+1}$$

where x_{it+1} is the actual realized return of stock i in month $t + 1$ and \hat{x}_{it+1} is its StockGPT return forecast. Returns are in basis points and R^2 in percentage points. t is the t -statistic of time-series mean of b_t computed using Newey and West (1987) standard error with 4 lags. Horizon 1 (2) means comparing return forecasts for month $t + 1$ with actual returns in month $t + 1$ ($t + 2$). The sample is monthly from February 2001 to December 2023.

Horizon	b	t	R^2
1	3.01	2.49	0.55
2	-0.08	-0.08	0.43

Table 6: Monthly Portfolio Statistics

This table reports the return statistics of the monthly long-short StockGPT-based portfolios. Mean and SD (standard deviation) are in annualized percentage points; Mean/SD (Sharpe ratio) is annualized; Min, Max, and MDD (max drawdown) are in percentage points; and t-Mean is t -statistic of the mean portfolio return using Newey-West standard error with 4 lags. Portfolios are formed after excluding stocks in the bottom decile based on market value. Horizon 1 (2) refers to using return forecasts for month $t + 1$ to form portfolios for month $t + 1$ ($t + 2$). EW (VW) refers to equal weighting (value weighting). Price Filter refers to the price level under which stocks are removed. The sample is monthly from February 2001 to December 2023.

Horizon	Weight	Price Filter	Mean	t-Mean	SD	Mean/SD	Min	Max	MDD
1	EW	0	13.5	3.8	14.9	0.9	-14.3	18.4	-35.0
1	EW	1	13.5	4.2	13.5	1.0	-14.3	17.1	-27.2
1	EW	3	13.1	4.1	13.6	1.0	-15.6	12.0	-20.6
1	EW	5	12.5	3.9	13.8	0.9	-14.5	12.7	-25.5
1	VW	0	11.6	2.3	24.4	0.5	-24.8	27.0	-54.7
1	VW	1	8.6	1.7	23.9	0.4	-20.8	26.9	-50.4
1	VW	3	9.6	2.1	22.4	0.4	-20.5	20.2	-39.5
1	VW	5	9.1	2.0	21.5	0.4	-20.7	18.1	-37.2
2	EW	0	7.5	2.0	15.8	0.5	-15.3	19.0	-27.9
2	EW	1	6.3	2.0	14.0	0.4	-18.3	15.7	-28.3
2	EW	3	5.7	1.8	13.8	0.4	-17.1	11.0	-32.9
2	EW	5	5.6	1.8	13.9	0.4	-17.2	12.7	-35.4
2	VW	0	12.6	2.1	28.2	0.4	-36.7	32.1	-61.8
2	VW	1	9.8	1.8	26.9	0.4	-36.7	25.6	-62.7
2	VW	3	7.3	1.5	24.5	0.3	-33.5	19.8	-64.2
2	VW	5	5.7	1.2	23.5	0.2	-30.4	19.4	-61.7

A Daily Model Trained with All Stocks

Table A.1: Daily Portfolio Statistics

This table reports the return statistics of the daily long-short StockGPT-based portfolios. Mean and SD (standard deviation) are in annualized percentage points; Mean/SD (Sharpe ratio) is annualized; Min, Max, and MDD (max drawdown) are in percentage points; and t-Mean is t -statistic of the mean portfolio return using Newey-West standard error with 20 lags. Portfolios are formed after excluding stocks in the bottom decile based on market value. Horizon 1 (2) refers to using return forecasts for day $t + 1$ to form portfolios for day $t + 1$ ($t + 2$). EW (VW) refers to equal-weighting (value weighting). Price Filter refers to the price level under which stocks are removed. The sample is daily from January 2001 to December 2023.

Horizon	Weight	Price Filter	Mean	t-Mean	SD	Mean/SD	Min	Max	MDD
1	EW	0	82.5	10.6	16.4	5.0	-9.0	13.8	-24.4
1	EW	1	71.1	10.3	15.7	4.5	-8.4	13.4	-25.9
1	EW	3	48.0	8.9	15.0	3.2	-9.3	14.1	-37.4
1	EW	5	36.3	7.8	14.7	2.5	-9.3	13.7	-47.7
1	VW	0	18.7	3.5	24.9	0.8	-16.9	17.7	-61.5
1	VW	1	17.1	3.3	24.4	0.7	-14.0	15.7	-62.8
1	VW	3	12.6	2.6	23.2	0.5	-13.4	17.7	-57.7
1	VW	5	10.4	2.3	22.4	0.5	-12.5	15.9	-58.7
2	EW	0	16.3	4.9	13.9	1.2	-8.6	13.7	-40.8
2	EW	1	14.7	4.6	13.7	1.1	-9.2	14.3	-37.4
2	EW	3	12.9	4.3	13.5	1.0	-9.9	13.9	-31.7
2	EW	5	10.8	3.8	13.4	0.8	-10.1	13.5	-29.2
2	VW	0	-0.6	-0.1	23.7	-0.0	-12.3	17.4	-75.1
2	VW	1	-1.1	-0.2	23.0	-0.0	-11.6	17.0	-78.5
2	VW	3	-3.6	-0.8	21.8	-0.2	-11.1	15.3	-84.7
2	VW	5	-4.7	-1.0	21.2	-0.2	-11.0	15.2	-87.1

References

- Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Arango, S. P., Kapoor, S., et al. (2024). Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*.
- Baldassarre, M. T., Caivano, D., Fernandez Nieto, B., Gigante, D., & Ragone, A. (2023). The social impact of generative ai: An analysis on ChatGPT. *Proceedings of the 2023 ACM Conference on Information Technology for Social Good*, 363–373.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877–1901.
- De Bondt, W. F., & Thaler, R. (1985). Does the stock market overreact? *Journal of Finance*, 40(3), 793–805.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Fama, E. F. (1970). Efficient capital markets. *Journal of finance*, 25(2), 383–417.
- Fama, E. F., & French, K. R. (2015). A five-factor asset pricing model. *Journal of financial economics*, 116(1), 1–22.
- Fama, E. F., & MacBeth, J. D. (1973). Risk, return, and equilibrium: Empirical tests. *Journal of political economy*, 81(3), 607–636.
- Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5), 2223–2273.
- Han, Y., Yang, K., & Zhou, G. (2013). A new anomaly: The cross-sectional profitability of technical analysis. *Journal of Financial and Quantitative Analysis*, 48(5), 1433–1461.
- Hou, K., Mo, H., Xue, C., & Zhang, L. (2021). An augmented q-factor model with expected growth. *Review of Finance*, 25(1), 1–41.
- Jegadeesh, N. (1990). Evidence of predictable behavior of security returns. *Journal of Finance*, 45(3), 881–898.
- Jegadeesh, N., & Titman, S. (1993). Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency. *Journal of Finance*, 48(1), 65–91.
- Jiang, J., Kelly, B., & Xiu, D. (2023). (Re-) Imag (in) ing Price Trends. *The Journal of Finance*, 78(6), 3193–3249.
- Jiang, J., Kelly, B. T., & Xiu, D. (2022). Expected returns and large language models. *Available at SSRN*.
- Kelly, B., Xiu, D., et al. (2023). Financial machine learning. *Foundations and Trends® in Finance*, 13(3-4), 205–363.
- Kirtac, K., & Germano, G. (2024). Sentiment Trading with Large Language Models. *Finance Research Letters*.
- Lopez-Lira, A., & Tang, Y. (2023). Can ChatGPT forecast stock price movements? Return predictability and large language models. *arXiv preprint arXiv:2304.07619*.

- Mannuru, N. R., Shahriar, S., Teel, Z. A., Wang, T., Lund, B. D., Tijani, S., Pohboon, C. O., Agbaji, D., Alhassan, J., Galley, J., et al. (2023). Artificial intelligence in developing countries: The impact of generative artificial intelligence (AI) technologies for development. *Information Development*.
- Newey, W. K., & West, K. D. (1987). Hypothesis testing with efficient method of moments estimation. *International Economic Review*, 777–787.
- Sætra, H. S. (2023). Generative AI: Here to stay, but for good? *Technology in Society*, 75, 102372.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Wang, G., Yang, G., Du, Z., Fan, L., & Li, X. (2023). Clinicalgpt: Large language models finetuned with diverse medical data and comprehensive evaluation. *arXiv preprint arXiv:2306.09968*.
- Wu, S., Irsoy, O., Lu, S., Dabrowski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., & Mann, G. (2023). BloombergGPT: A large language model for finance. *arXiv preprint arXiv:2303.17564*.
- Yang, Y., UY, M. C. S., & Huang, A. (2020). FinBERT: A Pretrained Language Model for Financial Communications.
- Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. (2022). OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.