Using Intra-Stock, Inter-Stock, and Temporal Aggregation to Predict Stock Market Returns and Stock Selections

Money Is All You Need
Prottoya Chowdhury (pchowdh4), Bailing Hou (bhou14),
Devesh Kumar (dkumar23), Haoyang Xu (hxu126)

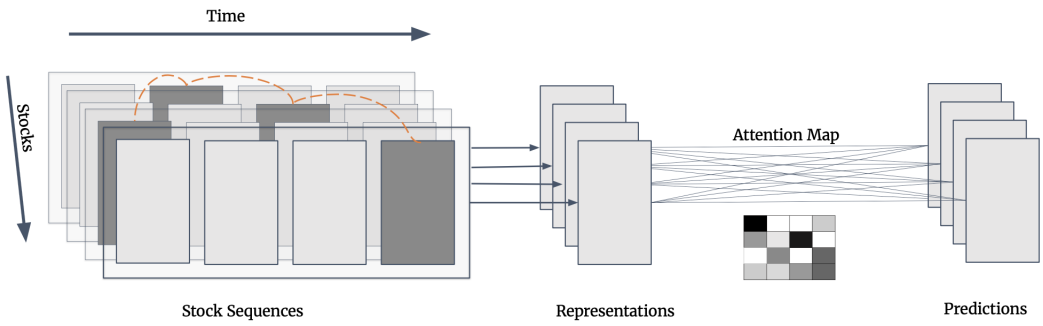
2 May 2025

1 Introduction

In this project, we re-implemented the following paper from the 2024 AAAI Conference on Artificial Intelligence, *MASTER: Market-Guided Stock Transformer for Stock Price Forecasting* by Li et al. 2024 [1], and predicted U.S. stock prices by training on the Global Factor Dataset. By solving the limitations of existing works, this paper aimed to tackle stock price forecasting, which has always been a challenging problem due to the market's high volatility and other factors.

As suggested by our team name, *Money Is All You Need*, we are deeply interested in the financial market, and stock price prediction is an important topic within this field with ongoing research. Incorporating a modified Transformer model to tackle this problem elegantly, Li et al. 2024 [1] used deep learning techniques and innovative methods that capture the stock correlation and automatically select relevant features with market information, making it an outstanding application of deep learning to tackle real-world challenges effectively and creatively.

Since the purpose of this model was to use Transformers to train a deep learning model to predict a series of future return data for different stocks, we classified the task of our model as a structured prediction problem.



Traditional Transformer-Based Stock Prediction Model

Traditional stock prediction models based on transformers have particular flaws. The models are typically based on an understanding of the Transformers' use in NLP, as seen above. Attention is used between various stock representations and the predictions, but the dashed lines show representations and connections that are not being properly recorded and modeled. We found that this kind of architecture does not factor in both how a single stock varies at various points in time, as well as how different stocks interact with each other at the same time. Thus, we implemented a new architecture to tackle those weaknesses. [1]

2 Methodology

The diagram of our improved model is shown below:

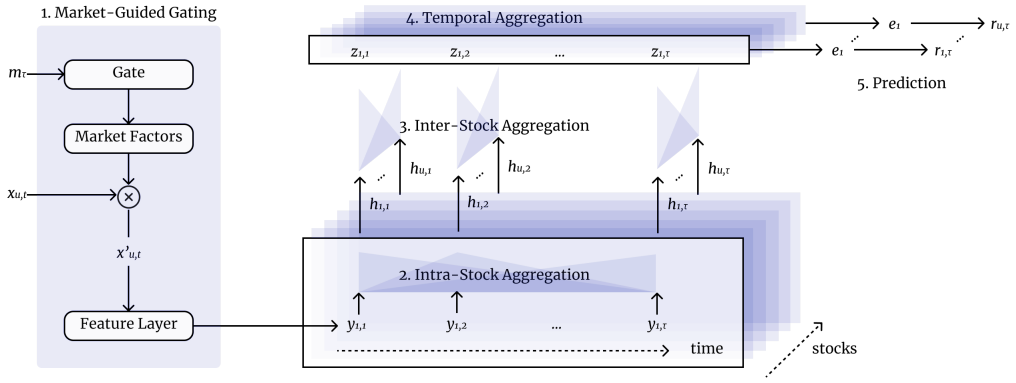


Figure 2: MASTER Model

There are 5 key components of our Transformer model. The **market-guided gating** element uses a market status vector to scale different features based on the market status vector that determines what features are the most important. **Intra-stock aggregation** creates an embedding across a period of time that takes into account the temporal context around a specific stock. **Inter-stock aggregation** involves using the overall market dynamics to create an embedding of how stocks compare to and interact with each other at a specific point in time. Attention is used in both intra-stock aggregation and inter-stock aggregation. Then, **temporal aggregation** puts temporal embeddings together to form an aggregated stock embedding. Lastly, the final **prediction layer** is a series of dense layers to help determine what the returns for a stock will be.

Throughout our training of the model, we measured the performance of our model in the following two ways:

1. **Measuring Returns**: We predicted the returns of different stock prices, which were measured in terms of percent changes of a stock's price. The performance of the model, in this part, was measured by the accuracy between the actual percent change and the predicted percent change using the standard mean-squared error (MSE). The goal of the model was to minimize the MSE such that our model's performance can approach the real-world data as closely as possible. Furthermore, even though we later used this accuracy measurement to select stocks and measure the model's selection skills (discussed below), the MSE loss in this measurement was the key value with which we fed back into the model to be trained and minimized.

2. **Measuring Stock Selection Skills**: While our model's core task was to predict percentage returns of a stock between two time periods, the core purpose of this model was to inform investors about how to allocate their capital into certain stocks that are likely to outperform the broader market. As such, to analyze the effectiveness of the model at predicting returns, we needed to quantify how much "investing skill"

the model has that is independent of the market performance. To infer the "investing skill" of the model, we used the model to construct a portfolio of the top 100 stocks with the highest predicted returns. Then, we evaluated the returns of these constructed portfolios using two metrics that were:

(a) Sharpe Ratio: This metric allowed us to identify how much return the portfolio generated for each percentage of standard deviation in the returns of the portfolio. Higher Sharpe ratios tend to indicate that the portfolio is able to generate more return while incurring lower levels of risk. The formula for the Sharpe ratio is the following:

$$\text{Sharpe Ratio} = \frac{E[R_a - R_f]}{\sigma_a}$$

In the formula above, R_a is the return of the stock, R_f is the risk-free rate, and σ_a is the standard deviation of returns of stock a.

(b) Information Ratio: This metric allowed us to identify whether using the model to generate a portfolio was able to produce higher returns as opposed to the overall return of the market. The formula was:

$$\text{Information Ratio} = \frac{\text{Portfolio Return - Market Return}}{\text{Tracking Error}}$$

Here, Tracking Error is the standard deviation of the difference between Portfolio Return and Market Return.

3 Results

Information Coefficient (IC)	0.0082
Rank Information Coefficient (RankIC)	0.0953
Information Ratio based IC (ICIR)	0.0271
Information Ratio based RankIC (RankICIR)	0.5296

Table 1: Evaluation of model with performance benchmarks

Information coefficient is the correlation between the predicted returns and the true returns. RankIC is similar to IC but uses rank, thus prioritizing the direction rather than magnitude. ICIR is a normalized version of IC. RankICIR normalizes RankIC.

Our model achieved an average MSE loss of around 1, which indicated that the model's prediction of monthly stock returns differed from the actual returns by about 1 standard deviation away. We also chose to investigate our results further by dynamically creating portfolios of stocks that consisted of the top third of stocks the model predicted as having the highest returns. We observed returns for these portfolios and found that on average they gained about 0.3 in a single month while maintaining an annual sharpe ratio of about 0.35, which is slightly low and suggests a bit of high volatility for the amount of returning earned.

This leads us to conclude that the model is able to identify and learn features that are predictive of stock returns for some stocks but not others. For stocks that the model predicts as having positive returns, there seems to be some degree of accuracy.

4 Challenges

There were a few hardest challenges that we faced throughout this project. Firstly, in the dataset we used, the dimensions of the parameters were often confusing, especially in

the case of stocks. To resolve this confusion, we mainly wrote comments and annotations of input sizes next to the necessary parameters. Then, the code we were writing had strong dependencies across different sections, so it was hard to distribute tasks (e.g., it was difficult to have one person implementing part A while another implementing part B that is strongly connected to part A). Our solution was that during the first few days of reading period, we met in person more frequently and made sure that everyone was understanding each part of the code (e.g., to deal with the remaining part of the data loading process), and we wrote better documentation of each function to help readability. Furthermore, we ran into some difficult issues during the pre-processing of the global factor data because the file is too big. So, while our teammate focused on unpacking the global factor data, we preprocessed open source data and had '.pkl' files ready to be trained such that we were able to move forward on other parts concurrently.

5 Reflection

How do you feel your project ultimately turned out? How did you do relative to your base/target/stretch goals?

Our base goal was to recreate the transformer model that was developed in the paper in the paper to predict stock returns. We successfully accomplished this base goal.

In terms of target goals, we hoped to predict returns and then form portfolios and observe the returns of the portfolio created by using the model we build. We were able to create a portfolio and demonstrate that the portfolio was profitable.

Finally, our stretch goal was to incorporate text data using our text database into our transformer model. This will incorporate a lot of latent data on sentiment surrounding financial markets that is likely to influence the trajectory of stock returns. We have made

partial progress towards the stretch goal. We created a script to calculate the public sentiment around various stocks. If we were to continue this project, we could incorporate this script's output with the input data for the model and train it. However, given the fact that it takes a long time to train and we have very limited time, we could not try this out yet.

Did your model work out the way you expected it to?

We had two core expectations for our model:

1. We expected the model to be able to predict directions of stock returns with some level of accuracy, generally above 50%.

2. We expected to predict the standard deviation of monthly stock returns from the mean monthyl return for each sample .

3. We expected the results of the model that were used in generating portfolios to yield a positive monthly return

In terms of the first expectation, we slightly underperformed our goal. On average, for the training phase of our model, it was able to predict the correction direction of stock returns about 50.4% of the time. This gave us some certainty that the model was not learning traits that were negatively correlated with predicting stock returns, but at the same time, it shows our model's limited ability to correctly learn features that were correlated with the direction returns.

In terms our second goal, our model demonstrated a good ability to predict the standard deviation of stock returns (compared to the mean return for each batch). The creators of the MASTER model intended for the model to output standard deviations of stock returns, and even though we used a completely different dataset to train our implementation of the model, we were able to achieve this. Throuhgout training our model, we constantly saw the MSE of the predicted standard deviations and the actual standard

deviations of returns decrease, suggesting that our model was able to identify absolute stock returns with more precision as we continued training.

In terms of our last goal, it appears that model was able to constantly predict normalized stock returns such that if we selected to form a portfolio of the top 1/6th of stocks with the highest predicted normalized returns, that portfolio was able to consistently produce positive returns. This insight inspired a few remarks regarding the model. First, we believe the model is able to identify certain traits that are predictive of monthly stock returns. This is backed up by fact that the portfolios generated from the model consistently achieve positive returns. Second, our model might be slightly negatively biased in that it expects more of the stock returns to be negative than positive. As such, we beleive that when the model does predict positive returns for a stock, there is a high likelihood that stock produces a positive monthly return, but the model is poor at discerning which stock will produce negative returns.

How did your approach change over time? What kind of pivots did you make, if any? Would you have done differently if you could do your project over again?

Our approach and the pivots we made are summarized as follows:

1. We initially believed that using a lot of features per stock observation would be the best approach to train our model. Our dataset included a total of 400+ features, but many of these features were not present for every single observation of stock and month. Thus, when creating our dataset, we were forced to eliminate a large portion of our data. This led to the model being unable to learn as well and to producing portfolios that had negative monthly returns. From this, we included that we needed to prioritize having more observations than features.

2. We then proceeded to update our data preprocessing mechanism to select fewer features (about 20). The larger training dataset appeared to give the model more learning capacity, and this is when we noticed our loss dropping and our predictions for returns becoming more accurate.

3. Next, we noticed that our model ability to update its weights and reduce loss plateaued around a prediction error of about 1.2 standard deviations. We chose to explore hyperparamter tuning at this stage, mainly by changing the number of Attention head, the learning rate of our optimizer, and the number of features we wanted to gate in Gating channel of our model. Doing so helped us to reduce the training loss to about 1 standard deviation and slightly lower in some cases.

One of the key limiting aspects to our model and training was the lack of integration of market information. While our dataset was large and included many observations, it didn't include information about the characteristics of the entire market or economy at a given time. The value of stocks is heavily influenced by current market dynamics and this information was not integrated. Thus, to improve upon this model, we would find a way to integrate market related information that is specific to each stock. This would allow the model to take into account the broader market and economic situation and how it could uniquely affect the returns for a stock. What do you think you can further improve on if you had more time?

If we had more time, we could train our model for longer and use more GPU compute. This would have the effect of improving our Sharpe ratio so that the model is more consistent. Also, our future data collection process can also take into account the market trend in the month that the data was recorded. Thus, we can determine the difference between the S&P500 index fund and our own portfolio performance. We could also use sentiment analysis with news feeds to try to get a sense of the public perception around a certain company, and around market conditions.

What are your biggest takeaways from this project/what did you learn?

One lesson we learned was that training a model on stock data requires a lot more com-

putations than what we expected. Our solution was to let the model run on OSCAR for long enough to get the task done, but in the future, we hope to find ways to further optimize this process. We trained our model mostly using data that is specific to a particular company. However, market and economic conditions strongly affect stock returns, and different types of companies might be impacted differently by market conditions (for example, during events such as the Great Depression, the COVID-19 Pandemic, or the surge of AI currently). Finding a way to merge our dataset with more relevant global economic, market, and social information would likely enhance our results. A lot of our dataset contains values that are missing, so, for the purpose of training our model, we dropped these pieces of data. Finding a strategy to integrate these pieces of information without biasing our results could introduce more complexity to our model. Those could all be possible ideas to further implement in future works related to this project.

6 Code Repository

Our GitHub is `https://github.com/sunnyhxu/MASTER`

References

[1] Tong Li, Zhaoyang Liu, Yanyan Shen, Xue Wang, Haokun Chen, and Sen Huang. Master: Market-guided stock transformer for stock price forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(1):162–170, Mar. 2024.