Quantformer with US Stocks Data

Zhaofeng Zhang
Department of Mathematics
University of Michigan
Ann Arbor, United States
zhangzf@umich.edu

***Abstract*—Catching the complexities of financial markets is a persistent challenge in traditional quantitative trading. In recent years, researchers have worked on algorithm methods to solve this. To address this issue, this paper intorduces quantformer, an enhanced transformer-based neural network model, which is designed to construct effective investment factors. By transfer learning from sentiment analysis, quantformer not only retains the transformer model's strengths in long-range dependencies understanding and modeling data relationships, but also adapts to numerical inputs of financial markets. Previous stock data from the U.S. capital market from 2010 to 2020 are used to train the model and the model shows risk-profit balance ability in the final backtest. Full implementation details and source code are available on Github.**

***Index Terms*—quantformer, transformer, neural networks, quantitative finance, stock selection, portfolio optimization, market sentiment**

I. Introduction

Stock trading aims to optimize the return on investment in the capital market by trading financial assets. Traders gain profit from the difference of stock price in a period. However, stock prices are influenced by various information, which is a complicated system and makes it hard to make a profit for traders. Catching a stock's future trend is quite challenging due to the not-strong efficiency market. Based on this, a large number of strategies and methods have been designed for trading, and quantitative strategies have been playing an important role among them.

In recent years, Machine Learning (ML) has became a new tool in quantitative trading algorithms. ML methods allow systems to learn from previous dataset, and lend themselves particularly well to predicting the dynamic situation in stock markets. Although there have been several previous attempts to determine factors with ML methods in the quantitative finance field, there are two difficulties for this field. Firstly, in the field of sentiment analysis, which is a branch of Natural Language Processing (NLP), models are used to convert words in text to word vectors through word embeddings to serve as inputs. However, financial datasets contain categorical data instead of words such as industry types and quantitative data such as price fluctuation, turnover rate, and financial indicators. If the input comprises only categorical data, the time series can be treated as a sentence (1). In most cases, the input will involve numerical data, which cannot be transformed via word embeddings. Secondly, many of the NLP tasks can be considered as sequence-to-sequence (seq2seq) problems, such as machine translation and dialogue systems. As an example, the transformer architecture is based on the seq2seq architecture (2). To utilize the training set, decoders in transformer sequentially output samples and use masking operations to handle input sequences during training. However, in stock prediction, where the aim is often to accurately forecast future returns over a period, the transformer model is rarely used for such tasks.

To address these problems, we propose *quantformer*, a modified transformer architecture adapted to quantitative financial data, and use it as an investment factor. Quantformer is able to input numerical data directly, which refers to a method similar to sentiment analysis. In previous work (3), I trained quantformer by stock data from the Chinese stock market. At this time, I will train the model by the data of American stocks (Stocks-Daily-Price and Stocks-Quarterly-BalanceSheet from the Hugging Face) and experiment with them. My contributions and goals lie in following aspects:

- We propose quantformer, whose structure adapts to rolling stock-related time series data as inputs without positioning module and word embedding. The new structure with linear embedding shows better fitting to the numerical type inputs.
- According to our experiments, quantformer-based factors perform better in back-tests compared with other traditional factor strategies under the same time period with different trading frequencies. These experiments indicate a potential direction for applying transformer-based models to quantitative financial tasks.

II. Related Work

A. Stock data and market sentiment

In some cases, stock data can be considered to influence the future trend of a stock price (4). (5; 6) have indicated that investor feelings influence the market. This relationship is not uniform but varies with changes in investor sentiment. This also highlighted that investor psychology, especially during periods of high market stress can drastically impact market behavior and investor decisions. This link between sentiment and market performance is further affirmed by studies focusing on factors such as the turnover ratio, which are seen as reflective of market liquidity and investor behavior (7). (8) built a new factor based on sentiment analysis with the average of trading

Identify applicable funding agency here. If none, delete this.

signals from technical trading strategies to benchmark stocks of the S&P 500 index and DJIA. The sentiment factor shows correlation with stock returns. These findings underscore the mutual relationship between investor sentiment and market performance, demonstrating how psychological factors can drive market dynamics.

B. Transformer in time series prediction

Transformer (2) has received huge attention in the NLP field since it was introduced in 2017. According to its special encoder and decoder stacks with self-attention blocks, the transformer shows its advantage in robustness, speed, and long-term memory compared with other traditional models.

Some research proposed enhancements of the attention module. For example, Adversarial Sparse Transformer (AST) (9) introduced sparse attention using the α-Entmax function within adversarial training. Autoformer (10) replaced traditional self-attention with an auto-correlation mechanism that utilizes periodic patterns in frequency-domain time series analysis. Similarly, Informer (11) suggested ProbSparse attention that selects only the most relevant queries and introduces generative-style decoding to handle long-term dependencies. Gated Transformer Networks (GTN) (12) used separate attention for channel-wise and temporal-wise interactions for multivariate time series. Patch Time Series Transformer (PatchTST) (13) divided time series into patches to model local dependencies, and Quatformer (14) attempted to use rotation-based attention to capture periodic patterns in time series. AirFormer (15) introduceed dartboard spatial attention and causal temporal attention in air quality prediction. Rough transformer (16) facilitated multi-view signature attention and continuous-time path encoding for financial market modeling for complex financial time series prediction.

In another direction, some works tried to make changes to the encoding or decoding of transformer. FEDformer (17) spited time series data into trend and seasonal components with frequency-enhanced modules. To improve computational efficiency, some models, including Multivariate Transformer (18) gave up using a decoder and focused solely on an encoder for feature extraction for forecasting tasks.

As a kind of revision sight, some research has proposed to replace or extend the positioning modules to better accommodate time series data. For instance, HFformer (19) gave up positioning modules, and uses activation functions with a lightweight linear decoder to revise the model. Noting that temporal order is inherently embedded in time series data, iTransformer (20) introduced a dimension-inverted structure and removes positioning. Transformer Hawkes Process (THP) (21) put time intervals into the positioning module directly. FX-Spot Transformer (22) used the Time to Vector (Time2Vec) function to handle irregular time intervals.

To overcome the limitations of standalone transformer models, an approach is to integrate them with other frameworks. ConvLSTM (23) merged Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) layers with transformer to simultaneously model spatial and temporal patterns. ProTran (24) combined state-space models (SSMs) with transformers to address non-Markovian dynamics in time series data, providing robust long-term forecasts. These hybrid approaches improved the versatility and domain adaptability of transformer-based architectures.

C. Transformer in quantitative trading

In the quantitative trading domain, a few research articles have tried transformer. (25) utilized transformer to predict stock market indices (including CSI 300, S&P 500, Hang Seng Index, and Nikkei 225) and concluded that the model can better catch the rules of stock market dynamics. (26) exploited transformer on trading sequences to classify stock price movements. Besides using a traditional transformer, other innovative methods of transformer are worth mentioning. (11) optimized the efficiency of time complexity and memory usage of transformer on extremely long time series by informer. (27) combined the advantages of CNNs and transformers to model short-term and long-term dependencies in financial time series. They illustrate the merits of this approach on intraday stock price prediction of S&P 500 constituents. (23) trained the model with stocks from the S&P 500 between 2004 and 2021, and performs better than other models such as VAR and ARIMA.

Similar to previous work, (28) introduced Adaptive Long-Short Pattern Transformer (ALSP-TF). Their model is structurally designed for stock price series at different context scales. With the help of a learnable function, they make self-attention aware of the weighted time intervals between patterns, to adaptively adjust their dependencies beyond similarity matching. In the end, they obtained more than 10% of annual return on average.

However, the transformer model can also have some disadvantages in practice. (29) mentioned that the global self-attention module focuses on point-wise token similarities without contextual insights. As fluctuations of stocks are conditioned on composite signals over manifold periods, lacking pattern-wise interaction hinders the adequate discrimination of stock tendency and is susceptible to noise points. On the other hand, (30) claimed that the basic query-key matching paradigm is position agnostic. Although position embedding is inserted into the sequential inputs, it may not be optimal because of the inability to reveal precise distances.

III. Model description

Fig1 1 shows the overview of the work. There are three major parts:
(i) *Data Initialization* to align the time series of stocks into a regular matrix;
(ii) Unlike the transformer, *Embedding* the training dataset and *training* them by a quantformer with word-embedding layer replaced by linear layer and with no Mask(opt.) layer;
(iii) Lastly, using the trained model to *predict* the possible trend of stocks.

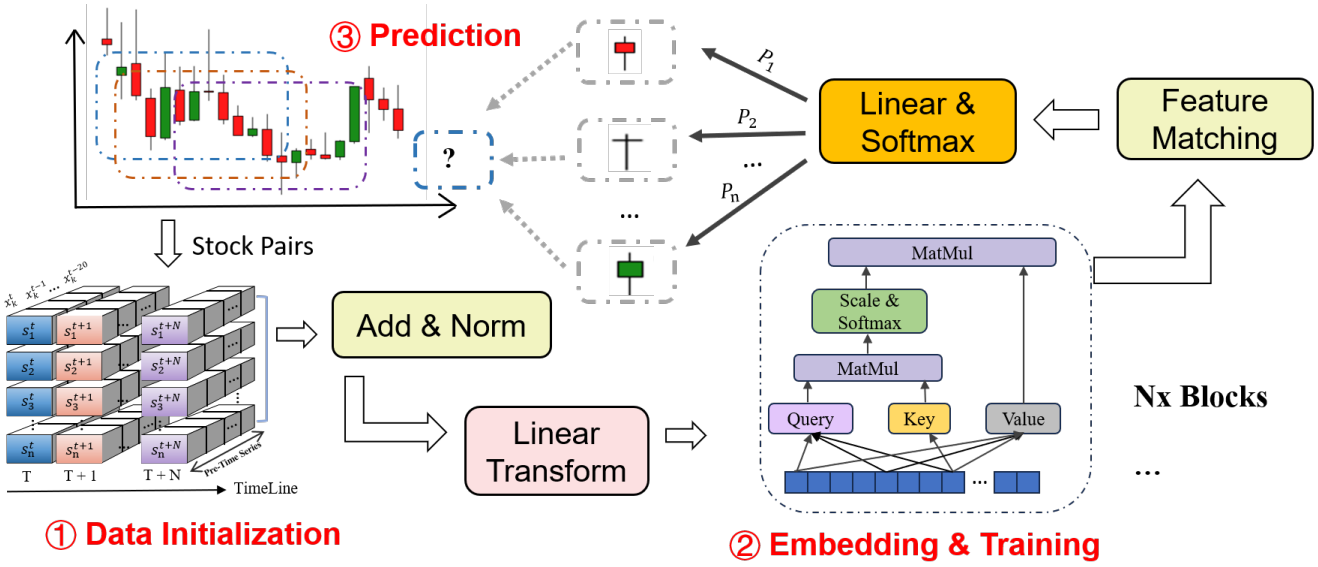

Figure 1: Overview of the work

A. *Overview of the dataset*

To obtain the target stock data, I select two datasets from the Hugging Face, which are https://huggingface.co/datasets/paperswithbacktest/Stocks-Daily-Price and https://huggingface.co/datasets/paperswithbacktest/Stocks-Quarterly-BalanceSheet.

The first dataset, called "Stock-Daily-Price," contains nearly 24M rows of daily data of 6644 stocks in the American stock market (the amount of the stock in the Hugging Face is different from the provided parquet files) from 1962 to 2025. Each row in the dataset includes the symbol, date, daily open, close price, volume and the adjust stock price.

The second dataset is called "Stocks-Quarterly-BalanceSheet". This dataset is the collection of stocks' balance sheet which announced each season, which includes 237k rows in addition. I will choose the column "*common_stock_shares_outstanding*" to calculate the turnover rate of the stock in each day that divide "column" in the first dataset with "*common_stock_shares_outstanding*" in the second dataset.

B. *Problem formulation*

Training input

Consider a candidate stock set S^t containing N stocks on the trading timestamp t:

$$S^t = \{s_n^{(t)}\}_{n=1}^N$$

For a stock s_n^t in the stock set S^t, where $n \in \{1, \ldots, N\}$, consider the two-dimensional historical 1-time step (such as one month) input feature matrix $\mathcal{X}_n^t \in \mathbb{R}^{20\times 2}$, consisting of 20 continuous timestamps:

$$\mathcal{X}_n^t = \{x_n^{t-m} | m = 19, 18, \ldots, 0\} \tag{1}$$

For each row vector x_n^{t-m} for $m = 0, \ldots, 19$, it contains two features. The first one is the accumulated daily profit rate r_n^{t-m} during the time step, where p_n^{t-m} is the close price of the last day at the timestamp $t-m$ and p_n^{t-m-1} is the close price of the last day at the timestamp $t-m-1$.

$$r_n^{t-m} = \frac{p_n^{t-m} - p_n^{t-m-1}}{p_n^{t-m-1}}, \quad v_n^{t-m} = \sum v_n^{t-m,i} \tag{2}$$

The second feature is the accumulated daily turnover rate v_n^{t-m} during the time step $t-m$ for stock n. $v_n^{t-m,i}$ is the daily turnover rate and i is the day index in the time period. To sum up, each $\mathcal{X}_n^t$ can be represented as:

$$\mathcal{X}_n^t = \begin{bmatrix} x_n^{t-19} & x_n^{t-18} & \ldots & x_n^t \end{bmatrix}^T \in \mathbb{R}^{20\times 2}, \quad x_n^{t-m} = [r_n^{t-m}, v_n^{t-m}] \tag{3}$$

Normalization

The next step is to normalize the input data for each time step by zero-mean, unit-variance normalization (Z-score). The normalization equation is shown in (4). For the row vector x_n^t ((1)), $\mathbb{E}[x^t]$ and $\mathrm{std}[x^t]$ represent the mean and standard deviation of all the to data x^t at time t, respectively, and the resulting values $\tilde{x}_n^t$ represent the normalized values of the row data x_n^t.

$$\tilde{x}_n^t = \frac{x_n^t - \mathbb{E}[x^t]}{\mathrm{std}[x^t]} \tag{4}$$

So, the normalized two-dimensional historical 1-time step input feature matrix can be represented in this way:

$$\tilde{\mathcal{X}}_n^t = \{\tilde{x}_n^{t-m} | m = 19, 18, \ldots, 0\} \tag{5}$$

In this way, the input sequences are normalized with zero mean and unit variance, which aims to reduce the influence

from outlying time points and allow different features to be comparable with each other (31).

Target labels

For a stock set S^t on the trading timestamp t, each stock s_n^t has a profit rate r_n^{t+1} on the trading timestamp $t+1$, where the calculation method is the same as for r_n^t in the sequence of inputs shown in (2). The set $\mathcal{G}^{t+1}$ contains the next-time stamp's profit for N stocks in the timestamps $t+1$. It is defined as $\mathcal{G}^{t+1} = \{r^{t+1_n}\}_{n=1}^N$. Each r_n^{t+1} in the list $\mathcal{G}^{t+1}$ is then ranked by $\Psi(r_n^{t+1})$, which is the empirical quantile CDF for r_n^t:

$$\Psi(r_n^{t+1}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{r_i^{t+1} \leq r_n^{t+1}\}} \tag{6}$$

For the target label for each feature s_n^t in $\mathcal{G}^{t+1}$, we have a one-shot vector $y_n^t \in \mathbb{R}^\varrho$ based on $\Psi(r_n^{t+1})$. y_n^t is partitioned into ϱ equal bins, where ϱ is set to 3 or more. Each bin corresponds to a $\varphi \times 100\%$ of the stocks in the set, where $\varphi\varrho \leq 1$. And the boundary term ξ is used to ensure non-overlapping intervals.

$$\xi = \frac{1-\varphi\varrho}{\varrho - 1}$$

Then y_n^t is represented as:

$$y_n^t = [\mathbf{1}_{\{(i-1)(\varrho+\xi) \leq \Psi(r_n^{t+1}) < i\varrho + (i-1)\xi)} | i = 1, \ldots, \varrho] \in \mathbb{R}^\varrho$$

For the empirical quantile CDF $\Psi(r_n^{t+1})$ of the stock in the range $[i\varphi, (i+1)\xi)$ for $i = 1, \ldots, \varrho$, y_n^t is represented as $\mathbf{0}_\varrho$. For example, suppose $\varrho = 3$, $\varphi = 0.2$; for the stocks s_n^t whose profit values r_n^t ranked in the top, middle, bottom $\varphi \times 100\% = 20\%$ respectively, as ($\xi = 0.2$ in this case):

$$y_n^t = \begin{cases} [1,0,0]^T & \text{if } \Psi(r_n^{t+1}) \in [0, 0.2) \\ [0,1,0]^T & \text{if } \Psi(r_n^{t+1}) \in [0.4, 0.6) \\ [0,0,1]^T & \text{if } \Psi(r_n^{t+1}) \in [0.8, 1.0) \\ [0,0,0]^T & \text{otherwise} \end{cases}$$

The stocks in the intermediate ranges ($\Psi(r_n^{t+1}) \in [0.2, 0.4), [0.6, 0.8)$) are marked as $[0,0,0]^T$. Alternatively, when $\varrho = 5$ and $\varphi = 0.2$, the stocks are divided into five equally sized quantiles, with no null labels ($[0,0,0,0,0]^T$), where each part contains 20% of the stock in S^t:

$$y_n^t \in \{[1,0,0,0,0]^T, [0,1,0,0,0]^T, \ldots, [0,0,0,0,1]^T\}$$

C. Quantformer encoder

Then, a quantformer encoder structure is designed as described in this subsection. The representation hierarchy consists of L blocks of multi-head self-attention layers. Taking initialized stock embedding sequences $\mathcal{X} = \{\tilde{\mathcal{X}}_n^{(t)}\}_{n=1}^N \in \mathbb{R}^{N \times 20 \times 2}$ as inputs, the canonical self-attention layers introduced in (2) can perform information exchange between every time points for each stock s_n^t.

To process both categorical and numerical data, the word embedding layer is replaced by a standard linear layer, utilizing linear transformations to substitute the process of word embedding. The linear embedding is shown in 7, where $\mathbf{W}_E \in \mathbb{R}^{2\times d}$ is the trainable weight matrix and $\theta_E \in \mathbb{R}^d$ is the bias and d is the dimension of the hidden feature space. The resulting sequence $\mathcal{X}_i'$

$$\mathcal{X}_i' = \mathcal{X}_i \mathbf{W}_E + \theta_E \tag{7}$$

In stock prediction, we aim to accurately forecast the return for a future period, thus the model's output is generally a single value representing the probability of price increase or decrease. Therefore, the decoder is simplified by removing the autoregressive prediction mechanism (in the decoder) and the masking operations. For the attention head $h = 1, \ldots, H$, the query, key, and value metrics for each $\mathcal{X}_i \in \mathbb{R}^{20\times 2}$ are computed using separate learned linear projections:

$$\mathbf{Q}_{i,h} = \mathcal{X}_i' \mathbf{W}_h^Q, \quad \mathbf{K}_{i,h} = \mathcal{X}_i' \mathbf{W}_h^K, \quad \mathbf{V}_{i,h} = \mathcal{X}_i' \mathbf{W}_h^V \tag{8}$$

where the trainable weights $\mathbf{W}_h^Q$, $\mathbf{W}_h^Q$, $\mathbf{W}_h^Q \in \mathbb{R}^{d\times\varrho}$ for the h-th head. Then the attention for each head is computed using scaled dot-producted attention:

$$\text{Attention}(\mathbf{Q}_{i,h}, \mathbf{K}_{i,h}, \mathbf{V}_{i,h}) = \text{softmax}(\mathbf{Q}_{i,h}, \mathbf{K}_{i,h}^T/\sqrt{d})\mathbf{V}_{i,h} \tag{9}$$

The outputs from all heads are concatenated and sent back to the original dimension using $\mathbf{W}^O \in \mathbb{R}^{\varrho H \times d}$, and the final multi-head attention output $\mathbf{F}_i$:

$$\begin{aligned} \mathbf{F}_i &= \text{Multihead}(\mathbf{Q}_{i,h}, \mathbf{K}_{i,h}, \mathbf{V}_{i,h}) \\ &= \big(||_{h=1}^{h=H} \text{Attention}(\mathbf{Q}_{i,h}, \mathbf{K}_{i,h}, \mathbf{V}_{i,h})\big)\mathbf{W}^O \end{aligned} \tag{10}$$

where $||$ represents the concatenation operator. After applying attention to all inputs, all the output from attention modules are represented in the form of:

$$\mathbf{F} = [\mathbf{F}_1; \mathbf{F}_2; \ldots \mathbf{F}_N] \in \mathbb{R}^{N\times 20\times \varrho H}$$

which are fed into feed-forward layers.

D. Output process

After processing through the multi-head self-attention and feed-forward layers, with the encoder output F_i, the output layer can be represented as:

$$\mathbf{Z}^t = \text{Softmax}(\mathbf{F}_i \mathbf{W}_Z + \theta_Z) = \{z_1^t, z_2^t, \ldots z_N^t\} \in \mathbb{R}^\varrho \tag{11}$$

where $\mathbf{W}_Z \in \mathbb{R}^{\varrho\times d}$ and $\theta_Z \in \mathbb{R}^d$. With the softmax function, the predicted probability of the output can be represented as:

$$\hat{Y}_n^t = \{\hat{y}_{n,i}^t\}_{i=1}^\varrho = \left\{\frac{\exp(z_i^t)}{\sum_{i=1}^\varrho \exp(z_i^t)}\right\}_{i=1}^\varrho \in \mathbb{R}^\varrho \tag{12}$$

In (12), the $\hat{Y}_n^t$ takes the values between 0 and 1 and sum to 1, which means that they can be interpreted as the probability of the stock's performance.

E. Prediction

The Mean Squared Error Loss (MSELoss) are used to quantify the loss. It is defined as the average of the squares of the differences between the predicted value and the actual prices:

$$\text{MSELoss} = \frac{1}{N}\sum_{i=1}^{N} ||(y_i^t - \hat{y}_i^t)||_2^2 \tag{13}$$

where $||\cdot||_2^2$ denotes the squared Euclidean distance between predicted and target class probability vectors.

IV. Experiments

A. Dataset

Our training data contains 6644 stocks listed either on the US stock market. The training time period ranges from January 2010 to December 2024. The data has been obtained from the Hugging Face, which is stock trading data and stock balance sheet. The training period is from January 2010 to December 2019 and the testing period starts from January 2020.

Closing price adjustments, such as dividends, stock splits, and other corporate actions that can affect a stock's price, are applied to the stock training prices. These adjustments are essential for stock price analysis, especially over a long period, as they provide a more accurate picture of a stock's value and performance over time, which is more appropriate for financial backtesting. (32).

B. Implementation details

The model and its training process are implemented with PyTorch (33). The hyperparameters of the model are optimized by grid search, as it is simple to implement and reliable in low dimensional spaces (34). Here the input dimension is 2, and the dimension of the hidden feature space d is 16. The number of multi-head attention modules is 16, and the number of layers of encoder and decoder is 6. The model was trained on an NVIDIA GeForce RTX 2070 GPU and NVIDIA A100 Tensor Core GPU using the Adam optimizer (35) for 50 epochs. The learning rate is 0.001, and the batch size is 64.

C. Trading strategy

Algorithm 1 shows the pseudocode of the trading strategy. Before the first trade date of the timestamp t, all sequences $\mathcal{X}_n^t$ from the stock set S^t are put in the model and the list of outputs $\hat{Y}^t$ is required. The portfolio value at time t, denoted as P^t is updated based on the previous period's weights and returns as follows:

$$P^t = P^{t-1}(\sum_{n=1}^{N} w_n^{t-1}(1 + r_n^{t-1})) \tag{14}$$

where w_n^t is the weight of stock s_n^t and r_n^t is the return of the stock and $\sum_{n=1}^{N} w_n^t = 1$. To determine the weight of each stock, a trading strategy $\Phi \in \mathbb{R}^{\varrho}$ and the decision factor $\mathbf{b}$ is used at the strategy, where $\mathbf{b}$ determines how many quantile groups are selected, satisfying $1 \leq \mathbf{b} < \varrho$. The strategy does not allow short-sell (the Chinese stock market does not allow short-sell either), so Φ only contains 0 and 1.

$$\Phi \in \{0,1\}^{\varrho}, \quad ||\Phi||_0 = \mathbf{b}$$

For example:

- If $\varrho = 3$, $\mathbf{b} = 1$, then $\mathbf{\Phi}$ could be one of: $[1,0,0]$, $[0,1,0]$, or $[0,0,1]$.
- If $\mathbf{b} = 2$, then possible values of $\mathbf{\Phi}$ include: $[1,1,0]$, $[1,0,1]$, or $[0,1,1]$.

Recall the predicted output $\hat{y}_n^t = \{\hat{y}_{n,i}^t\}_{i=1}^{\varrho}$, we sort S^t by $\Psi(\hat{y}_{n,1}^t)$, which is the empirical quantile CDF for $\hat{y}_{n,1}^t$. Similar to the 6, $\Psi(\hat{y}_{n,1}^t)$ is computed in this way:

$$\Psi(\hat{y}_{n,1}^t) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{\hat{y}_{i,1}^t \leq \hat{y}_{n,1}^t\}} \tag{15}$$

The sorted predicted vector $\tilde{y}_n^t$ for stock s_n^t at time t is shown below.Here the stocks are ranked into ϱ parts and each part contains $\varphi \times 100\%$ of the stocks, and the parameters ϱ and φ satisfy $\varphi\varrho \leq 1$:

$$\tilde{y}_n^t = [\mathbf{1}_{\{(i-1)(\varrho+\xi) \leq \Psi(\hat{y}_{n,1}^t) < i\varrho\ +(i-1)\xi)\}} | i = 1, \ldots, \varrho] \in \mathbb{R}^{\varrho}$$

Then the weight for each selected stock is computed by the 16 shown below, and all the chosen stocks are equal-weighted in this way:

$$w_n^t = \frac{1}{\mathbf{b}\cdot\varrho}(\tilde{y}_n^{t-1}\Phi^T\tilde{y}_n^t)^T \cdot \mathbf{1} \tag{16}$$

The same method is run repeatedly during the subsequent periods. The backtest starts from January 2020, in other words, the result of the sequences from May 2018 to December 2019 will be used as the first stock pool to trade. The transaction fee is set as 0.3%, where it is 0.00278% in NYSE (36).

Algorithm 1 Trading Strategy

Require: Feature sequence χ_n^t for each stock $s_n^t \in S^t$, initial cash P^0, trading strategy Φ, decision factor $\mathbf{b}$
Ensure: Final portfolio and asset weights w_n^t

Procedure RUN_FREQUENCY(trade, frequency, start-day=1, time='open')
  Function PREDICT(χ_n^t)
    $\hat{y}_n^t \leftarrow$ quantformer(χ_n^t)
  Function SORTLABEL($\hat{y}_{n,1}^t$)
    $\tilde{y}_n^t \leftarrow \left[\mathbf{1}\left\{\frac{i-1}{\varrho} \leq \Psi(\hat{y}_{n,1}^t) < \frac{i}{\varrho}\right\}\right]_{i=1}^{\varrho}$
  Function COMPUTEWEIGHT($\tilde{y}_n^{t-1}$, $\tilde{y}_n^t$, Φ)
    numerator $\leftarrow \tilde{y}_n^{t-1} \cdot \Phi^{\top} \cdot \tilde{y}_n^t$
    denominator $\leftarrow \sum_{k=1}^{N} \tilde{y}_k^{t-1} \cdot \Phi^{\top} \cdot \tilde{y}_k^t$
    $w_n^t \leftarrow \frac{1}{\mathbf{b}\cdot\varrho} \cdot \frac{\text{numerator}}{\text{denominator}}$
  Function TRADE(w_n^t, P^t)
  for each $s_n^t \in S^t$
  order(s_n^t, $w_n^t P^t$) else order =0

D. Metrics

Formally, the Sharpe Ratio (SR) (37) will be used to test the performance of the strategy. The SR is a measure of risk-adjusted return that describes the additional earnings an investor receives for each standard deviation unit increase (shown in (17)), where R_p is the return of the portfolio and R_f is the risk-free rate.

$$\text{SR} = \frac{\mathbb{E}[R_p] - R_f}{\text{std}[R_p]} \tag{17}$$

Value at risk (VaR) is a method to summarize the total risk in a portfolio (38).

$$\text{VaR}_\alpha = \inf\{x : P(\mathcal{L} > x) \leq \alpha\} \tag{18}$$

(18) shows the calculation of VaR, where $\mathcal{L}$ is the loss of the holding period T,then the $\text{VaR}(\alpha)$ is the αth upper quantile of $\mathcal{L}$. In the measurement of the portfolio, 99% VaR is used to estimate the maximum loss during the period with 99% confidence.

Max Drawdown (MD) means the potential worst-case scenario or the most extreme possible loss from the previous peak. For the trough value of a portfolio A_{trough} and the previous peak value of the portfolio A_{peak}, the MD is calculated in the following way:

$$\text{MD} = \frac{A_{peak} - A_{trough}}{A_{peak}}$$

V. Results and Discussion

Fig 2 shows the result of the quantformer, which is compared with the S&P 500 index in the US market. The detailed comparison is shown in Table I with the sight of annual return, SR, MD and 95% VaR from 2020 to 2024.

The quantformer strategy shows risk-adjusted performance from 2020 to 2024, as reflected by its Sharpe ratio ranging from 1.17 to 2.14. While its raw returns were lower than those of the S&P 500 in bullish years like 2020, 2021, and 2023, quantformer outperformed during periods of market downturn, particularly in 2022 when it delivered a positive return of 13.51% compared to the S&P 500's substantial loss of -36.78%.

Year	Strategy	Return (%)	SR	MD (%)	95% VaR (%)
2020	Quantformer	5.60	1.96	0.41	0.27
2020	S&P 500	39.60	1.31	17.18	8.75
2021	Quantformer	7.29	1.72	0.43	0.43
2021	S&P 500	11.59	0.97	5.62	3.64
2022	Quantformer	13.51	2.14	0.44	0.41
2022	S&P 500	-36.78	-1.37	26.49	11.74
2023	Quantformer	8.26	1.94	0.51	0.39
2023	S&P 500	38.2	2.03	10.41	4.14
2024	Quantformer	3.15	1.17	0.71	0.48
2024	S&P 500	24.85	2.25	4.51	2.75

TABLE I: Comparasion between quantformer strategy and S&P 500

In terms of volatility control, quantformer maintained lower maximum drawdowns, consistently under 1%, while the S&P 500 saw drawdowns as high as 26.49% in 2022. Furthermore, the 95% VaR for Quantformer remained low across all years, highlighting its robustness under stress conditions. This expresses that quantformer offers a more stable and risk-managed return profile.

VI. Conclusion

This study proposes a new neural network architecture, *quantformer*, inspired by the transformer architecture, for quantitative stock prediction and trading. I address the need for handling numerical input data rather than text, and adapted the model for forecasting tasks rather than sequence-to-sequence problems common in NLP. To enable direct processing of numerical time series data, we replace the word embedding layer with a standard linear layer and removed the output

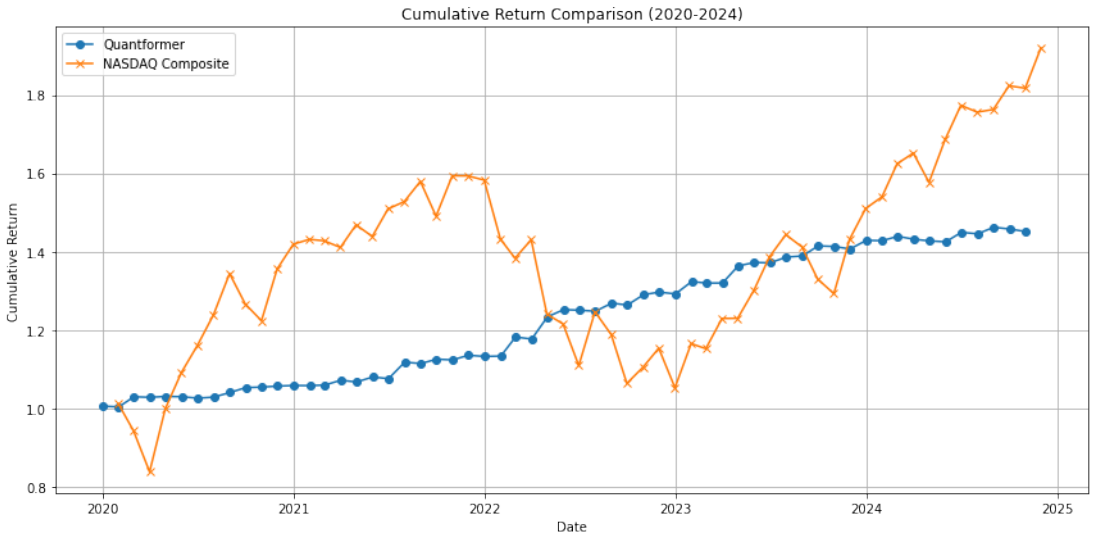

Figure 2: Comparasion between quantformer and S&P 500

masking operations. We also simplify the decoder to produce a probability distribution over future price movements rather than autoregressively generating token sequences.

Our experimental results demonstrate the promise of this approach. The quantformer-based trading strategies can deliver substantial returns over the benchmark in various market situations, with Sharpe Ratios indicating sound risk-adjusted performance. The positive alphas affirm the strategy's ability to outperform expected returns after accounting for risk factors.

Overall, our work illustrates the ability of the quantformer to handle financial time series, making a profit in a boom market environment and avoiding huge drawdowns in downward market. It is a novel approach for further analysis and investigations of the proposed ML-based quantitative methods. The implementation code of quantformer is available at https://github.com/zhangmordred/STATS507_QFwithUSStocks.

Acknowledgment

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Y. Gorishniy, I. Rubachev, and A. Babenko, "On embeddings for numerical features in tabular deep learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 991–25 004, 2022.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, 2017, p. 6000–6010.

[3] Z. Zhang, B. Chen, S. Zhu, and N. Langrené, "Quantformer: from attention to profit with a quantitative transformer trading strategy," 2024, arXiv:2404.00424.

[4] C. S. Asness, "The power of past stock returns to explain future stock returns," *SSRN:2865769*, 1995.

[5] Y. Chen, H. Zhao, Z. Li, and J. Lu, "A dynamic analysis of the relationship between investor sentiment and stock market realized volatility: evidence from China," *PlOS One*, vol. 15, no. 12, p. e0243080, 2020.

[6] H. P.H and A. Rishad, "An empirical examination of investor sentiment and stock market volatility: evidence from India," *Financial Innovation*, vol. 6, no. 1, pp. 1–15, 2020.

[7] S. Naseem, M. Mohsin, W. Hui, G. Liyan, and K. Penglai, "The investor psychology and stock market behavior during the initial era of COVID-19: a study of China, Japan, and the United States," *Frontiers in Psychology*, vol. 12, p. 626934, 2021.

[8] W. Ding, K. Mazouz, O. Ap Gwilym, and Q. Wang, "Technical analysis as a sentiment barometer and the cross-section of stock returns," *Quantitative Finance*, vol. 23, no. 11, pp. 1617–1636, 2023.

[9] S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang, "Adversarial sparse transformer for time series forecasting," *Advances in neural information processing systems*, vol. 33, pp. 17 105–17 115, 2020.

[10] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: decomposition transformers with auto-correlation for long-term series forecasting," *Advances in neural information processing systems*, vol. 34, pp. 22 419–22 430, 2021.

[11] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[12] M. Liu, S. Ren, S. Ma, J. Jiao, Y. Chen, Z. Wang, and W. Song, "Gated transformer networks for multivariate time series classification," 2021, arXiv:2103.14438.

[13] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: long-term forecasting with transformers," in *The Eleventh International Conference on Learning Representations (ICLR 2022)*, 2022.

[14] W. Chen, W. Wang, B. Peng, Q. Wen, T. Zhou, and L. Sun, "Learning to rotate: quaternion transformer for complicated periodical time series forecasting," in *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 2022, pp. 146–156.

[15] Y. Liang, Y. Xia, S. Ke, Y. Wang, Q. Wen, J. Zhang, Y. Zheng, and R. Zimmermann, "Airformer: predicting nationwide air quality in China with transformers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 14 329–14 337.

[16] F. Moreno-Pino, A. Álvaro, H. Waldon, D. Xiaowen, and C. Álvaro, "Rough transformers: lightweight and continuous time series modelling through signature patching," 2024, arXiv:2405.20799.

[17] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: frequency enhanced decomposed transformer for long-term series forecasting," in *International conference on machine learning*. PMLR, 2022, pp. 27 268–27 286.

[18] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 2114–2124.

[19] F. Barez, P. Bilokon, A. Gervais, and N. Lisitsyn, "Exploring the advantages of transformers for high-frequency trading," 2023, arXiv:2302.13850.

[20] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "iTransformer: inverted transformers are effective for time series forecasting," in *The Twelfth International Conference on Learning Representations (ICLR 2024)*, 2024.

[21] S. Zuo, H. Jiang, Z. Li, T. Zhao, and H. Zha, "Transformer Hawkes process," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 692–11 702.

[22] T. Fischer, M. Sterling, and S. Lessmann, "Fx-spot predictions with state-of-the-art transformer and time

embeddings," *Expert Systems with Applications*, vol. 249, p. 123538, 2024.
[23] S. Kim, S.-B. Yun, H.-O. Bae, M. Lee, and Y. Hong, "Physics-informed convolutional transformer for predicting volatility surface," *Quantitative Finance*, vol. 24, no. 2, pp. 203–220, 2024.
[24] B. Tang and D. S. Matteson, "Probabilistic transformer for time series analysis," *Advances in Neural Information Processing Systems*, vol. 34, pp. 23 592–23 608, 2021.
[25] C. Wang, Y. Chen, S. Zhang, and Q. Zhang, "Stock market index prediction using deep Transformer model," *Expert Systems with Applications*, vol. 208, p. 118128, 2022.
[26] Q. Ding, S. Wu, H. Sun, J. Guo, and J. Guo, "Hierarchical multi-scale Gaussian transformer for stock movement prediction." in *IJCAI*, 2020, pp. 4640–4646.
[27] Z. Zeng, R. Kaur, S. Siddagangappa, S. Rahimi, T. Balch, and M. Veloso, "Financial time series forecasting using CNN and transformer," 2023, arXiv:2304.04912.
[28] H. Wang, T. Wang, S. Li, J. Zheng, S. Guan, and W. Chen, "Adaptive long-short pattern transformer for stock investment selection," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 2022, pp. 3970–3977.
[29] K. Xu, Y. Zhang, D. Ye, P. Zhao, and M. Tan, "Relation-aware transformer for portfolio policy learning," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 4647–4653.
[30] C. Wu, F. Wu, and Y. Huang, "DA-transformer: distance-aware transformer," 2020, arXiv:2010.06925.
[31] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
[32] P. A. Diamond, "A model of price adjustment," *Journal of Economic Theory*, vol. 3, no. 2, pp. 156–168, 1971.
[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: an imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[34] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 2, 2012.
[35] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.
[36] New York Stock Exchange, "Nyse price list 2025," https://www.nyse.com/publicdocs/nyse/markets/nyse/NYSE_Price_List.pdf, 2025, accessed: April 19, 2025.
[37] W. Sharpe, "Capital asset prices: a theory of market equilibrium under conditions of risk," *Journal of Finance*, vol. 19, pp. 425–442, 1964.
[38] J. Hull, *Risk management and financial institutions*. Hoboken: John Wiley & Sons, 2012, vol. 733.