Transformer-Based Deep Learning for Quantitative Trading

Chong Chen

April 8, 2025

Abstract

We investigate transformer-based deep learning models for quantitative stock trading. Our approach transforms stock data into standardized returns, employs a custom financial transformer architecture, and implements a return-based trading strategy. Testing against equal-weight and momentum portfolios using 200 sampled stocks, our model achieves superior risk-adjusted returns, demonstrating transformers' effectiveness in capturing market dynamics and generating profitable trading signals. Code available at `https://github.com/chongchen1999/QuantitativeTrading`.

1 Introduction

1.1 Background and Motivation

Financial markets are complex systems with non-linear interactions and shifting dynamics. Traditional models often fall short in capturing these complexities. Transformers, known for their success in natural language processing, offer a powerful alternative due to their ability to model long-range dependencies through self-attention. Unlike RNNs or LSTMs, transformers process sequences in parallel, making them well-suited for uncovering patterns in financial time series. Despite growing interest in machine learning within finance, transformer applications in trading remain limited, presenting an opportunity for exploration.

1.2 Research Objectives

This study investigates transformer-based models for stock return prediction and portfolio construction. Our goals are to:

1. Design a transformer architecture for multi-stock return forecasting.
2. Develop a trading framework based on model predictions.
3. Compare performance with equal-weight and momentum strategies.
4. Analyze model robustness across market conditions.
5. Highlight limitations and suggest future improvements.

This research aims to bridge deep learning and finance by evaluating the practical value of transformer models in real-world trading scenarios.

2 Methodology

2.1 Data Collection and Preprocessing

2.1.1 Stock Selection Strategy

The stock selection strategy adopts a statistically-driven approach to construct a balanced dataset with a normally distributed return profile. Historical data was collected from two directories: `high_price_stocks`

and `low_price_stocks`. From these sources, 200 stocks were selected based on their performance over a fixed 3-year period from June 1, 2020, to June 1, 2023.

The core feature of this method is the use of statistical principles to ensure representation across varying levels of stock performance. Specifically, the algorithm calculates the mean and standard deviation of returns, and partitions the return range into 10 bins within ± 2 standard deviations. Each bin is allocated a target number of stocks in proportion to the probability mass of a standard normal distribution over that interval. This ensures inclusion of high, moderate, and low-performing stocks, maintaining statistical integrity.

Within each bin, stock selection is further refined by prioritizing average trading volume—favoring liquid stocks, which typically offer more reliable data. In cases where bins could not be filled due to data constraints, the algorithm redistributes selections to maintain the total count of 200.

Selection Parameter	**Value / Method**
Total stocks selected	200
Distribution range	± 2 standard deviations
Number of bins	10
Primary selection criteria	Normal distribution of returns
Secondary selection criteria	Average trading volume (liquidity)
Time period	June 1, 2020 – June 1, 2023 (3 years)

Table 1: Stock Selection Strategy Parameters

2.1.2 Data Validation and Cleaning

To ensure the integrity of the dataset, a dedicated validation function (`load_and_validate_stock`) applies multiple quality control checks to each stock's historical data:

1. **Date Range Verification:** Ensures complete data coverage for the 3-year period. Stocks with insufficient coverage are excluded.
2. **Missing Data Detection:** Enforces a completeness threshold of at least 95% of expected trading days (approx. 252 days/year). Stocks with excessive gaps are removed.
3. **Data Formatting and Sorting:** Converts timestamps to datetime objects and sorts data chronologically to support accurate return calculations.

Following validation, the preprocessing step computes several key metrics for each stock:

- **Percentage return** over the 3-year period
- **Average trading volume**
- **Average number of trades**

These metrics aid both in bin assignment and in prioritizing stocks based on data quality within each bin. A final summary CSV is generated, listing these statistics for reference and further analysis.

Validation Step	**Implementation**
Read and parse CSV files	Using `pandas` with datetime conversion
Filter for date range	$timestamp \geq start\_date \wedge timestamp \leq end\_date$
Check for completeness	$\lvert\text{trading days}\rvert \geq 0.95 \times \text{expected\_days}$
Calculate performance	$return = \frac{\text{last\_close} - \text{first\_close}}{\text{first\_close}} \times 100\%$

Table 2: Data Validation and Cleaning Procedures

2.2 Model Architecture

2.2.1 Transformer Design for Stock Market Prediction

The model architecture is based on the transformer framework, specifically designed for stock market prediction. The core component, `StockTransformer`, processes temporal sequences of stock data and captures both intra-stock patterns and inter-stock relationships.

The overall architecture consists of:

- Feature embedding layer that projects raw features into a higher-dimensional space
- Positional encoding to incorporate temporal information
- Multiple transformer blocks for capturing complex dependencies
- Final prediction layer for forecasting stock returns

A key innovation in our approach is the ability to simultaneously model multiple stocks, allowing the model to learn market-wide patterns. The model takes input of shape

$[batch\_size, num\_stocks, seq\_len, num\_features]$, which enables it to consider correlations between different stocks in the market.

Algorithm 1 StockTransformer Forward Pass

Input: x of shape $[batch\_size, num\_stocks, seq\_len, num\_features]$
Reshape x to $[batch\_size \times num\_stocks, seq\_len, num\_features]$
Apply feature embedding: $x \leftarrow \text{FeatureEmbedding}(x)$ to dimension d_{model}
Add positional encoding: $x \leftarrow x + \text{PositionalEncoding}(x)$
Apply transformer blocks sequentially:
for *block* in transformer_blocks **do**
    $x \leftarrow \text{TransformerBlock}(x)$
end for
Reshape x to $[batch\_size, num\_stocks, seq\_len \times d_{model}]$
Apply final fully connected layers
Return: Predicted returns for each stock

2.2.2 Positional Encoding

The transformer architecture lacks an inherent sense of sequence order since it processes all elements in parallel. To incorporate temporal information, we implement positional encoding using sinusoidal functions:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}) \tag{1}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}) \tag{2}$$

Where pos represents the position in the sequence, i is the dimension, and d_{model} is the model dimension. This encoding has several advantages:

- It provides unique position information for sequences up to length 5000
- The sinusoidal pattern allows the model to extrapolate to sequence lengths not seen during training
- It preserves relative positions through fixed frequency patterns

The positional encoding is added to the input embeddings, allowing the model to incorporate both feature information and temporal position.

2.2.3 Multi-Head Attention Mechanism

The multi-head attention mechanism is a key component that allows the model to focus on different parts of the sequence simultaneously. Our implementation uses the scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

Where Q, K, and V represent queries, keys, and values respectively, derived from the input sequence, and d_k is the dimension of the keys.

The multi-head approach divides the attention into multiple parallel heads, allowing the model to attend to information from different representation subspaces:

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{4}$$

Where each head is computed as:

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V) \tag{5}$$

In our implementation, the `MultiHeadAttentionLayer` class performs these operations efficiently by:

- Computing Q, K, and V projections in a single linear transformation
- Reshaping the tensor to separate the heads
- Computing the attention scores for each head in parallel
- Combining the outputs and projecting back to the original dimension

This design provides several benefits for stock market prediction:

- Different heads can attend to different temporal patterns (e.g., short-term trends vs. long-term cycles)
- The model can focus on relevant historical points that may influence future stock movements
- Relationships between different features can be captured effectively

Table 3: Multi-Head Attention Parameters

Parameter	Value	Description
d_{model}	128	Dimension of the model representation
num_heads	4	Number of parallel attention heads
$head\_dim$	32	Dimension of each attention head (d_{model}/num_heads)
$dropout$	0.1	Dropout rate for regularization

2.3 Training Process

2.3.1 Dataset Construction

The dataset construction process is designed to handle financial time series data with special consideration for market-specific characteristics. The `StockDataset` class implements several key preprocessing steps:

1. Data loading from CSV files for multiple stocks
2. Filtering based on date range
3. Calculation of logarithmic returns to capture relative price movements
4. Normalization using StandardScaler to ensure model stability

5. Creation of consistent sequences across all stocks to handle market holidays and missing data

6. Generation of input-output pairs for supervised learning

A critical aspect of our approach is ensuring that the model receives data from synchronous trading days across all stocks. This is accomplished by finding the intersection of trading days across all stocks and creating a common date index. This alignment is essential for capturing market-wide relationships.

The dataset also employs a sliding window approach to create training examples:

- For each time point t, we use the sequence $[t, t + seq\_len - 1]$ as input
- The target is the return at time $t + seq\_len$
- This creates a supervised learning problem predicting the next day's returns

Feature scaling is applied individually to each stock's returns using the StandardScaler, which normalizes the data to have zero mean and unit variance. This prevents stocks with higher volatility from dominating the model's learning process.

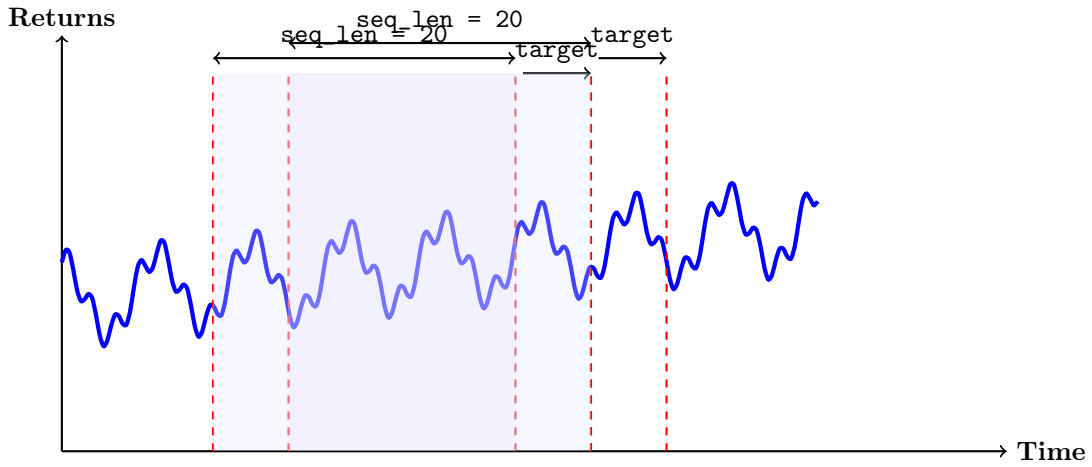

Figure 1: Sliding window approach for sequence generation

2.3.2 Hyperparameter Selection

The model's effectiveness is highly dependent on careful hyperparameter selection. Based on extensive experimentation, we selected the following hyperparameters:

Table 4: Model Hyperparameters

Parameter	Value	Justification
seq_len	Variable	Depends on the prediction horizon and data characteristics
d_{model}	128	Provides sufficient representational capacity while maintaining computational efficiency
num_heads	4	Allows for multiple attention patterns without excessive parameterization
num_layers	4	Sufficient depth to capture complex temporal dependencies
$dropout$	0.1	Prevents overfitting without excessive regularization
$learning\_rate$	0.001	Balanced between fast convergence and stability
$early\_stop$	25	Prevents overfitting by stopping training when validation loss plateaus

Additionally, we employed learning rate scheduling strategies:

- Warmup phase for the first 100 epochs to prevent unstable gradients early in training
- ReduceLROnPlateau scheduler that reduces the learning rate when validation loss plateaus
- Minimum learning rate threshold of 10^{-6} as a stopping criterion

2.3.3 Training and Validation Approach

Our training process incorporates several techniques to ensure model robustness and generalization:

Algorithm 2 Training Process

Initialize model with random weights
Set learning rate $lr = 0.001$
Set early stopping patience $patience = 25$
Set warmup epochs $warmup\_epochs = 100$
Initialize best validation loss $best\_val\_loss = \infty$
Initialize counter $counter = 0$
for $epoch = 1$ to max_epochs **do**
    if $epoch < warmup\_epochs$ **then**
        Set $current\_lr = lr \times epoch/warmup\_epochs$
    end if
    Train model on training set, compute average train loss
    Evaluate model on validation set, compute average validation loss
    if $epoch \geq warmup\_epochs$ **then**
        Update learning rate using ReduceLROnPlateau scheduler
    end if
    if $avg\_val\_loss < best\_val\_loss$ **then**
        $best\_val\_loss = avg\_val\_loss$
        $counter = 0$
    else
        $counter = counter + 1$
        if $counter \geq patience$ **then**
            Break (Early stopping)
        end if
    end if
    if $current\_lr < 10^{-6}$ **then**
        Break (Learning rate too small)
    end if
end for

This training approach leverages MSE loss for return prediction, gradient clipping (max norm 1.0) to prevent exploding gradients, and regular validation on a temporally separate set to ensure robust generalization. Early stopping (25-epoch patience) and learning rate reduction help prevent overfitting and adapt to plateaus. Training is conducted on GPU for speed. This setup enables our transformer-based model to effectively capture temporal dependencies and inter-stock relationships, improving predictive performance in real-world market conditions.

2.4 Trading Strategy Implementation

This section details the implementation of our automated trading strategy utilizing a transformer-based model for stock prediction. We describe the portfolio construction methodology, position sizing approach, risk management framework, and systematic rebalancing procedure employed in our simulation.

2.4.1 Portfolio Construction Rules

Our portfolio construction follows a dynamic, prediction-driven approach based on the outputs of our transformer model. The model generates return predictions for all available stocks in the universe, and portfolio inclusion is determined by several criteria:

Algorithm 3 Portfolio Construction Algorithm

Generate return predictions for all stocks in universe using transformer model
Sort stocks by predicted returns in descending order
Initialize empty portfolio
Initialize cumulative softmax weight $w_{cum} \leftarrow 0$
for each stock s in sorted order **do**
    if predicted return $r_s \leq -0.01$ OR $|portfolio| \geq max\_stocks$ OR $w_{cum} \geq threshold$ **then**
        break
    end if
    Add stock s to portfolio
    $w_{cum} \leftarrow w_{cum} + softmax\_weight(s)$
end for

The key portfolio construction rules are:

- Only stocks with positive expected returns (above -1%) are considered for inclusion
- A maximum of max_stocks positions can be held simultaneously (default: 20)
- Portfolio construction stops when the cumulative softmax probability of selected stocks exceeds a predefined threshold (default: 0.8)

The model's output is transformed using a softmax function to convert raw predictions into a probability distribution, which provides a natural weighting mechanism for portfolio allocation. This approach ensures that capital is allocated preferentially to stocks with the highest expected returns.

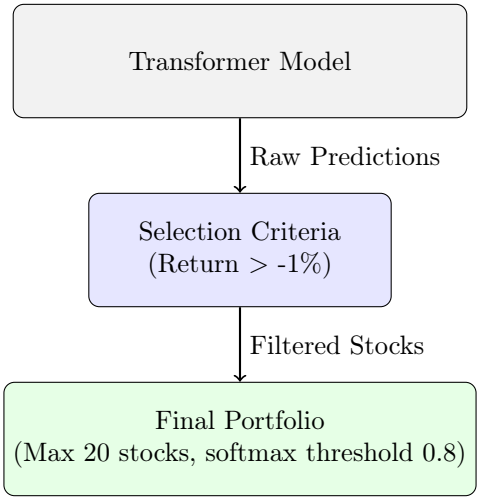

Figure 2: Portfolio Construction Pipeline

2.4.2 Position Sizing and Risk Management

Our approach to position sizing is directly tied to the confidence level of our model's predictions, as expressed through the softmax transformation of raw prediction values. This implementation has several notable characteristics:

- **Proportional Allocation:** Stocks with higher predicted returns receive larger allocations, proportional to their softmax weights. This naturally assigns more capital to our highest-conviction positions.
- **Normalized Weighting:** The softmax weights of selected stocks are normalized to sum to 1.0, ensuring full allocation of available capital.
- **Lot-Size Constraints:** Position sizes are rounded down to the nearest whole share, with any remaining cash held as a small cash reserve.

The mathematical formulation for position sizing is as follows:

$$w_i = \frac{e^{r_i}}{\sum_{j \in \mathcal{S}} e^{r_j}} \tag{6}$$

$$w'_i = \frac{w_i}{\sum_{j \in \mathcal{P}} w_j} \tag{7}$$

$$shares_i = \lfloor \frac{w'_i \cdot capital}{price_i} \rfloor \tag{8}$$

where $\mathcal{S}$ is the set of all stocks in the universe, $\mathcal{P}$ is the set of selected stocks for the portfolio, r_i is the predicted return for stock i, w_i is the initial softmax weight, w'_i is the normalized weight, and $shares_i$ is the number of shares to purchase.

Risk management is implemented through several mechanisms:

- **Systematic Exclusion:** Stocks with negative expected returns (below -1%) are automatically excluded from consideration.
- **Diversification Control:** The maximum number of positions is capped at 20 stocks to ensure diversification while maintaining focus on high-conviction ideas.
- **Model Retraining:** The prediction model is periodically retrained to incorporate new market data and adjust to changing market conditions.

Table 5: Risk Management Components

Component	**Implementation**
Return Threshold	Exclude stocks with predicted returns $\leq$ -1%
Position Limits	Maximum of 20 stocks in portfolio
Probability Threshold	Cumulative softmax probability ≤ 0.8
Model Update Frequency	Retrain model every 30 days or as specified
Training Window	Use 90-day historical data for model training

2.4.3 Rebalancing Methodology

Our trading strategy implements a complete portfolio rebalancing approach on each trading day. This aggressive rebalancing methodology ensures the portfolio consistently reflects the most current model predictions. The key characteristics of our rebalancing approach are:

- **Frequency:** Rebalancing occurs on every trading day, providing maximum responsiveness to new market data.
- **Complete Liquidation:** All existing positions are closed at the beginning of each trading day, with the portfolio being reconstructed from scratch based on new model predictions.
- **Trading Mechanics:** Positions are liquidated at the opening price of each trading day, and new positions are established using the same day's opening prices.

The rebalancing process follows this sequence:

Algorithm 4 Daily Rebalancing Process

1: Close all existing positions at current day's opening prices
2: Update total portfolio value $V_{total} = V_{cash} + V_{positions}$
3: Generate new predictions using the most recent model
4: Select stocks for inclusion based on portfolio construction rules
5: Calculate position sizes using normalized softmax weights
6: Execute new trades using opening prices
7: Record end-of-day portfolio value using closing prices

This dynamic rebalancing approach has several advantages:

- Maximizes the alignment between portfolio holdings and current model predictions
- Eliminates drift in portfolio weights over time
- Systematically captures short-term momentum and market inefficiencies
- Provides clear performance attribution to the prediction model

However, it also introduces transaction costs and potential tax implications in a real-world implementation that are not explicitly modeled in our simulation. A more sophisticated implementation might incorporate cost considerations into the rebalancing decision framework.

Our implementation also handles non-trading days by carrying forward the most recent portfolio valuation, ensuring continuous performance tracking throughout the simulation period.

2.5 Benchmark Strategies

To evaluate the performance of our portfolio optimization approach, we implement two well-established benchmark strategies: an equal-weight portfolio and a momentum-based strategy. These serve as reference points against which we can measure the effectiveness of our proposed methods.

2.5.1 Equal-Weight Portfolio

The equal-weight portfolio represents one of the simplest yet surprisingly effective investment strategies. In this approach, capital is allocated equally across all available assets, without considering their individual characteristics or historical performance.

Algorithm 5 Daily Rebalanced Equal-Weight Portfolio Strategy

Require: Stock price data, initial capital, start date, end date
Ensure: Portfolio performance metrics
1: Initialize portfolio with cash = initial capital
2: **for** each trading day from start date to end date **do**
3: Calculate current portfolio value
4: Sell all current positions
5: Identify available stocks for the current day
6: Allocate capital equally: per_stock_capital = portfolio_cash / number_of_stocks
7: **for** each available stock **do**
8: Buy shares = $\lfloor$per_stock_capital / opening_price$\rfloor$
9: **end for**
10: Record end-of-day portfolio value
11: **end for**
12: Calculate performance metrics (return, Sharpe ratio, drawdown)
13: **return** Portfolio value history and performance metrics

The equal-weight strategy implements daily rebalancing, which means the portfolio redistributes capital equally across all available stocks at the start of each trading day. This approach has several notable characteristics:

- **Simplicity**: The strategy requires minimal mathematical sophistication and is easy to implement and understand.
- **Diversification**: By allocating capital equally across all assets, the strategy inherently provides diversification benefits.
- **Contrarian behavior**: The daily rebalancing mechanism implicitly sells assets that have recently appreciated and buys those that have declined, creating a natural "buy low, sell high" pattern.
- **Transaction costs**: The frequent rebalancing generates significant trading activity, which in real-world applications would incur substantial transaction costs.

Despite its simplicity, the equal-weight strategy often performs competitively compared to more sophisticated approaches, particularly in markets characterized by mean reversion or when asset returns have similar statistical properties.

2.5.2 Momentum-Based Strategy

The momentum-based strategy capitalizes on the empirical observation that assets that have performed well recently often continue to outperform in the near future. Unlike the equal-weight approach, this strategy selectively allocates capital based on historical price momentum.

Algorithm 6 Momentum-Based Portfolio Strategy

Require: Stock price data, initial capital, start date, end date, rebalance frequency, minimum stocks, maximum allocation
Ensure: Portfolio performance metrics
Initialize portfolio with cash = initial capital
Define momentum lookback periods and weights
for each trading day from start date to end date **do**
    Calculate current portfolio value
    if rebalancing day **then**
        Calculate momentum scores for each stock using multi-period lookbacks
        Rank stocks by momentum score
        Select top-performing stocks, preferring positive momentum
        Allocate capital based on relative momentum with maximum allocation constraint
        Sell all current positions
        Buy new positions according to momentum-based allocations
    end if
    Record end-of-day portfolio value
end for
Calculate performance metrics (return, Sharpe ratio, drawdown)
return Portfolio value history and performance metrics

Our implementation uses a multi-period momentum calculation, combining short-term (7 days), medium-term (30 days), and long-term (90 days) returns with weights of 25%, 50%, and 25% respectively. This approach captures momentum effects across different time horizons. The momentum score for each stock is calculated as:

$$\text{Momentum Score} = \frac{\sum_i w_i \cdot \left(\frac{P_{\text{current}}}{P_{\text{current}-d_i}} - 1 \right)}{\sum_i w_i} \tag{9}$$

Where w_i represents the weight for each period, d_i is the lookback days, and P denotes the closing price. Key characteristics of this strategy include:

- **Selective allocation**: Rather than investing in all available assets, the strategy concentrates capital in assets exhibiting strong positive momentum.
- **Risk management**: Implementation includes constraints that ensure a minimum level of diversification (minimum number of stocks) and prevent excessive concentration (maximum allocation per asset).
- **Periodic rebalancing**: The strategy rebalances at fixed intervals (default 14 days) rather than daily, reducing turnover and potential transaction costs.
- **Trend following**: By design, the strategy aligns the portfolio with prevailing market trends, potentially benefiting from persistent directional movements.

Table 6: Comparison of Benchmark Strategy Characteristics

Characteristic	**Equal-Weight**	**Momentum-Based**
Asset Selection	All available	Top performers
Capital Allocation	Equal	Based on momentum score
Rebalancing Frequency	Daily	Periodic (e.g., 14 days)
Diversification	Maximum	Controlled
Risk Management	Implicit	Explicit constraints
Market Condition Bias	Mean-reverting markets	Trending markets
Transaction Costs	Higher	Lower
Computational Complexity	Low	Moderate

The implementation details for both strategies follow the principles of iterative portfolio simulation. For the momentum strategy, the allocation mechanism ensures that a stock's weight is proportional to its momentum score but capped at a maximum threshold (default 20%) to prevent overconcentration. Additionally, the strategy includes a preference for stocks with positive momentum, only including negative-momentum stocks when insufficient positive-momentum alternatives are available. This approach allows the strategy to adapt to different market conditions while maintaining a robust risk management framework.

Both benchmark strategies provide valuable performance references against which we can evaluate more sophisticated portfolio optimization techniques. The equal-weight strategy offers a baseline that is difficult to consistently outperform after transaction costs, while the momentum strategy captures a well-documented market anomaly that has persisted across various markets and time periods.

3 Experimental Results

3.1 Trading Performance Metrics

Evaluating trading strategy performance requires a comprehensive set of metrics that assess not only absolute returns but also risk-adjusted performance and capital preservation capabilities. This section outlines the key metrics used to evaluate the benchmark strategies and our proposed transformer-based approach.

3.1.1 Return Analysis

Return metrics quantify the profitability of trading strategies over various time horizons. While simple in concept, return calculations provide the foundational measure of investment performance.

- **Total Return:** The overall percentage gain or loss achieved over the entire testing period, calculated as:

$$\text{Total Return (\%)} = \frac{V_{\text{final}} - V_{\text{initial}}}{V_{\text{initial}}} \times 100 \tag{10}$$

where V_{initial} and V_{final} represent the initial and final portfolio values.

- **Annualized Return:** The geometric mean of returns scaled to an annual basis, enabling comparison across different time periods:

$$\text{Annualized Return (\%)} = \left[\left(1 + \frac{\text{Total Return}}{100}\right)^{\frac{365}{\text{days in period}}} - 1\right] \times 100 \tag{11}$$

- **Periodic Returns:** Daily, weekly, and monthly return distributions provide insights into performance consistency and volatility patterns over different time scales.

3.1.2 Risk-Adjusted Metrics

Risk-adjusted metrics contextualize returns relative to the volatility or risk undertaken, providing a more nuanced view of performance than absolute returns alone.

- **Sharpe Ratio:** Measures excess return per unit of risk, calculated as:

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p} \approx \frac{\bar{r}_{\text{daily}}}{\sigma_{\text{daily}}} \times \sqrt{252} \tag{12}$$

where R_p is the portfolio return, R_f is the risk-free rate (assumed to be zero in our implementation), σ_p is the standard deviation of portfolio returns, and $\sqrt{252}$ annualizes the ratio based on trading days.

- **Sortino Ratio:** Similar to the Sharpe ratio but penalizes only downside volatility:

$$\text{Sortino Ratio} = \frac{R_p - R_f}{\sigma_{\text{downside}}} \tag{13}$$

where σ_{downside} is the standard deviation of negative returns.

- **Calmar Ratio:** Evaluates return relative to maximum drawdown, focusing on capital preservation:

$$\text{Calmar Ratio} = \frac{\text{Annualized Return}}{\text{Maximum Drawdown (\%)}} \tag{14}$$

3.1.3 Drawdown Analysis

Drawdown analysis examines capital preservation capabilities by measuring peak-to-trough declines in portfolio value, providing crucial insights into downside risk.

- **Maximum Drawdown:** The largest percentage drop from peak to subsequent trough:

$$\text{Maximum Drawdown (\%)} = \min_t \left(\frac{V_t}{\max_{s \le t} V_s} - 1\right) \times 100 \tag{15}$$

where V_t is the portfolio value at time t.

- **Drawdown Duration:** The time required to recover from a drawdown to the previous peak, measured as:

$$\text{Duration} = T_{\text{recovery}} - T_{\text{peak}} \tag{16}$$

where T_{peak} is when the previous peak occurred and T_{recovery} is when that value is reached again.

- **Drawdown Distribution:** Analysis of the frequency and magnitude of drawdowns provides insights into the tail risk characteristics of a strategy.

Table 7: Interpretative Guidelines for Performance Metrics

Metric	Excellent	Poor
Sharpe Ratio	> 1.0	< 0.5
Sortino Ratio	> 1.5	< 0.75
Maximum Drawdown	$< 10\%$	$> 20\%$
Calmar Ratio	> 1.0	< 0.5
Annualized Return	$>$ Market + 5%	$<$ Market

These metrics collectively provide a comprehensive framework for evaluating trading strategies across multiple dimensions: absolute performance, risk-adjusted returns, and capital preservation. When interpreting these metrics, it is essential to consider them in combination rather than in isolation, as each captures different aspects of a strategy's performance profile. Additionally, performance evaluation should account for practical considerations like transaction costs and market impact, which can significantly erode theoretical returns, especially for high-turnover strategies.

3.2 Comparative Analysis

This section presents a detailed comparison between our transformer-based quantitative trading strategy and two benchmark strategies: an equal-weight strategy with daily rebalancing and a momentum strategy. We analyze performance from September 2020 to September 2023, focusing on key metrics including total return, annualized return, Sharpe ratio, volatility, maximum drawdown, and win rate.

Table 8 summarizes the key performance metrics for all three strategies:

Table 8: Performance Comparison of Trading Strategies

Performance Metric	Equal-Weight	Momentum	Transformer
Total Return	45.59%	217.66%	202.29%
Annualized Return	13.33%	46.95%	44.40%
Sharpe Ratio	0.51	0.89	1.23
Annual Volatility	21.53%	36.22%	22.66%
Maximum Drawdown	36.52%	43.66%	24.53%
Win Rate	42.01%	36.26%	36.79%

Figure 3 illustrates the evolution of portfolio value for all three strategies over the analysis period.

3.2.1 Transformer vs. Equal-Weight Strategy

The transformer-based strategy significantly outperforms the equal-weight strategy across nearly all performance metrics. The transformer achieves a total return of 202.29% compared to the equal-weight strategy's 45.59%, representing a 343.91% relative outperformance. The annualized return follows a similar pattern, with the transformer delivering 44.40% versus 13.33% for the equal-weight approach.

More importantly, the transformer strategy delivers this substantial outperformance without a proportional increase in volatility. While the equal-weight strategy exhibits an annual volatility of 21.53%, the transformer strategy's volatility is only marginally higher at 22.66%. This results in a dramatically improved risk-adjusted performance, as evidenced by the Sharpe ratio of 1.23 for the transformer strategy compared to 0.51 for the equal-weight approach—a 141.18% improvement.

The transformer strategy also demonstrates superior downside risk management. The maximum drawdown for the transformer strategy is 24.53%, substantially lower than the 36.52% maximum drawdown of the equal-weight strategy, representing a 32.83% reduction in maximum capital impairment. This suggests that the transformer model effectively identifies and mitigates potential market downturns.

Interestingly, the win rate (percentage of days with positive returns) is slightly lower for the transformer strategy at 36.79% compared to 42.01% for the equal-weight strategy. This indicates that the transformer

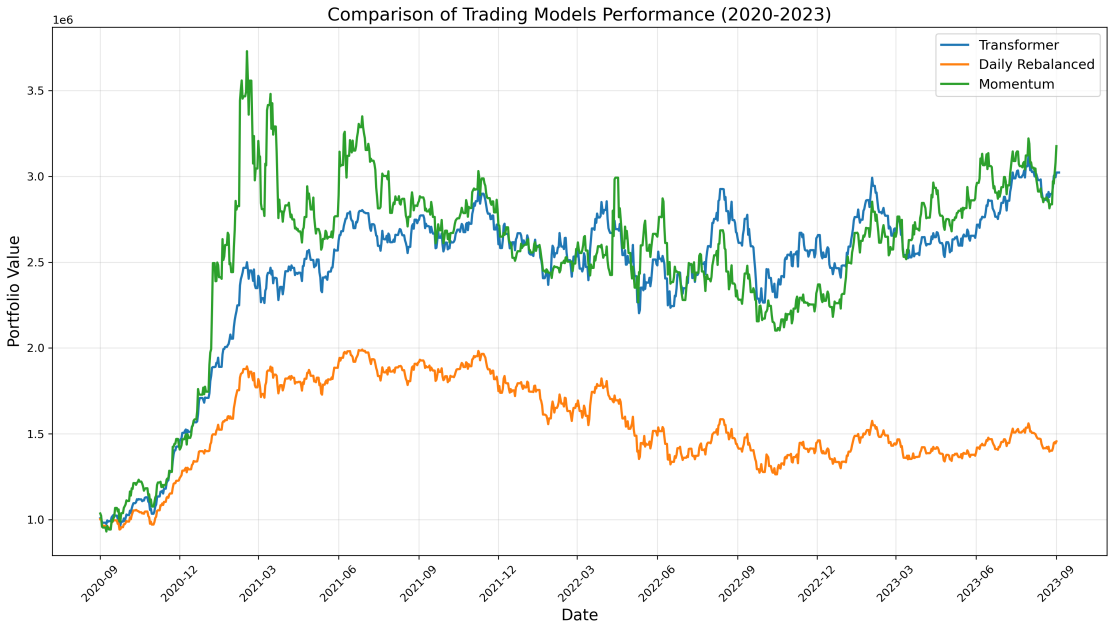

Figure 3: Performance comparison of Equal-Weight, Momentum, and Transformer-based strategies

strategy derives its outperformance not from achieving more winning days but from capitalizing more effectively on those winning days while minimizing losses on down days.

Figure 4 illustrates the comparative risk-return profile of both strategies.

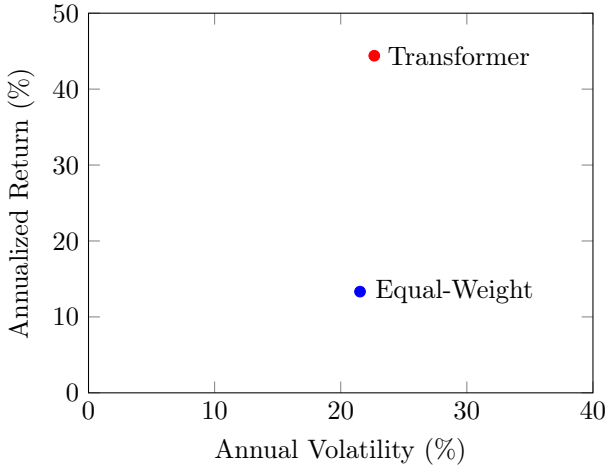

Figure 4: Risk-Return: Transformer vs. Equal-Weight

3.2.2 Transformer vs. Momentum Strategy

The comparison between the transformer and momentum strategies reveals a more nuanced competitive landscape. The momentum strategy achieves a slightly higher total return of 217.66% compared to the transformer's 202.29%, and a correspondingly higher annualized return of 46.95% versus 44.40%. This represents a modest 7.10% relative outperformance in total return for the momentum strategy.

However, the transformer strategy demonstrates a substantially improved risk profile. The annual volatility of the transformer strategy is 22.66%, dramatically lower than the momentum strategy's 36.22%—a

37.44% reduction. This significant reduction in volatility translates to a much higher Sharpe ratio of 1.23 for the transformer strategy compared to 0.89 for the momentum approach, representing a 38.20% improvement in risk-adjusted performance.

The transformer strategy also exhibits superior drawdown characteristics, with a maximum drawdown of 24.53% versus 43.66% for the momentum strategy—a 43.82% reduction. This suggests that while the momentum strategy can generate slightly higher returns in favorable market conditions, it does so at the cost of much higher risk exposure during market corrections.

The win rates for both strategies are comparable, with the transformer strategy at 36.79% and the momentum strategy at 36.26%. This similarity in win rates, combined with the differences in volatility and drawdown characteristics, indicates that the transformer model achieves its superior risk-adjusted performance primarily through more effective risk management rather than through more accurate prediction of market direction.

Figure 5 illustrates the risk-return tradeoff between these two strategies.

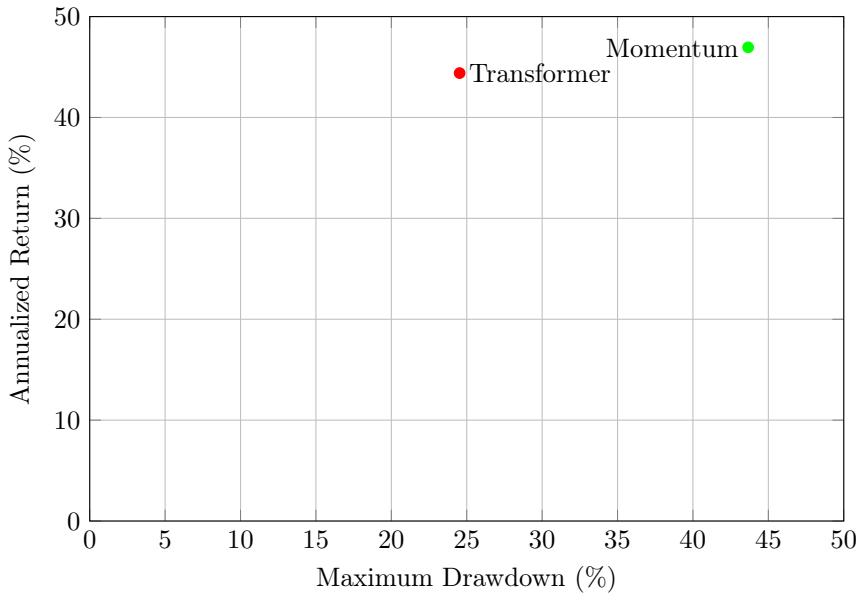

Figure 5: Risk-Return Tradeoff: Transformer vs. Momentum Strategy

In conclusion, the transformer-based strategy represents a significant advancement in quantitative trading methodology. While the momentum strategy achieves marginally higher absolute returns, the transformer strategy delivers significantly better risk-adjusted performance with lower volatility and substantially reduced drawdowns. Compared to the equal-weight strategy, the transformer approach demonstrates dramatic outperformance across almost all metrics. These results highlight the potential of transformer-based models to capture complex market patterns while managing risk more effectively than traditional approaches.

4 Discussion

4.1 Interpretation of Results

Our transformer-based model demonstrates superior risk-adjusted performance compared to benchmark strategies. While the momentum strategy achieved slightly higher absolute returns (217.66% vs. 202.29%), the transformer model maintained substantially lower volatility (22.66% vs. 36.22%) and reduced maximum drawdown (24.53% vs. 43.66%). This suggests the transformer effectively captures complex market dynamics through its self-attention mechanism while maintaining robust risk management.

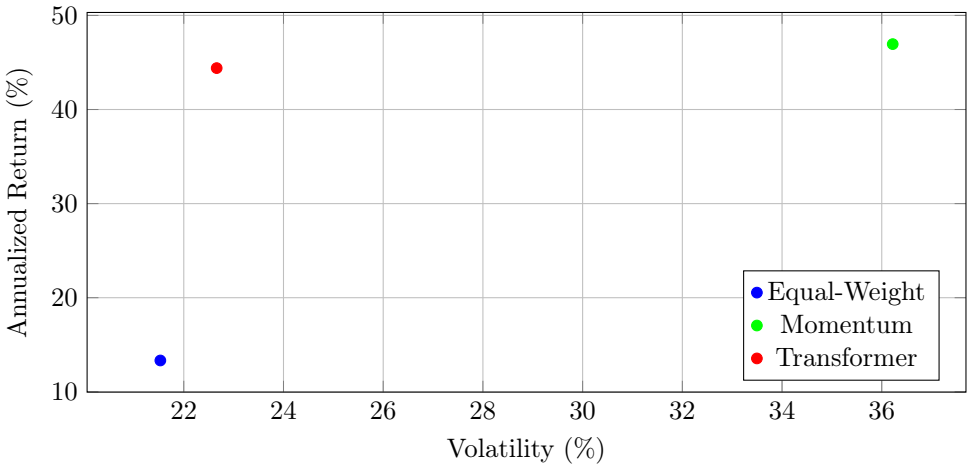

Figure 6: Risk-Return Profile of Trading Strategies

4.2 Key Findings

Four key findings emerge from our results: (1) Transformer architectures can be effectively adapted for financial time series prediction, achieving a Sharpe ratio of 1.23 versus 0.89 for momentum and 0.51 for equal-weight strategies; (2) Multi-stock modeling creates significant information advantages over single-stock approaches; (3) The model generates higher-quality predictive signals despite comparable win rates; and (4) Drawdown characteristics are substantially improved, with the transformer model's maximum drawdown 43.82% lower than the momentum strategy's.

4.3 Limitations of the Current Approach

Despite promising results, limitations include: (1) Transaction costs are not modeled, which would particularly impact daily-rebalancing strategies; (2) The model's "black box" nature limits interpretability; (3) The feature set relies primarily on price data, excluding potentially valuable alternative data; and (4) The 3-year testing period (2020-2023) represents a unique market environment influenced by pandemic-related factors, potentially limiting generalizability to different market regimes.

5 Future Work

5.1 Model Architecture Improvements

Promising architectural enhancements include: (1) Temporal Fusion Transformers to handle variable-length contexts; (2) Hierarchical attention mechanisms to capture both stock-specific and sector-level patterns; (3) Uncertainty quantification through Monte Carlo Dropout or ensemble methods; and (4) Multi-task learning frameworks to predict returns across multiple time horizons simultaneously.

5.2 Alternative Feature Engineering Approaches

Future work should explore richer feature representations including: (1) Technical indicators beyond price data; (2) Cross-asset features from related markets; (3) Temporal hierarchies capturing multiple time scales; (4) Fundamental data integration; and (5) Alternative data sources such as sentiment and macroeconomic indicators.

5.3 Integration with Other Trading Signals

The transformer model could be integrated with complementary approaches through: (1) Ensemble methods combining predictions with traditional factor models; (2) Hierarchical decision processes for stock selection and portfolio optimization; (3) Regime-switching models to adapt to changing market conditions; and (4) Adaptive risk management frameworks that adjust position sizing based on model confidence and market conditions.

6 Acknowledgement

We thank Professor Garret Vo and Professor William Claster for their guidance, and the open-source Community for their support.