Quantitative trading based on AlphaNet

Abstract

In recent years, many machine learning algorithms have been successfully applied in the field of quantitative trading. However, most of them are multi-step methods, which means manual intervention is required by them. As end-to-end methods, deep learning models can overcome this shortcoming. Combined with GRU, the AlphaNet has been applied in quantitative trading and achieves good performance on some datasets. Motivated by attention based Transformer which has outperformed recurrent models like LSTM and GRU on many sequence processing tasks, we would like to replace AlphaNet's GRU with Transformer to improve model's performance on stock return prediction. Experiment results show that our model performs better than previous AlphaNet on both RMSE in return prediction and profits making in backtest. We also did analysis for history length and ablation experiments.

1 Introduction

At present, most quantitative trading algorithms rely on high-frequency trading, which requires high hardware facilities such as network speed. The recent application of machine learning algorithms in designing automated trading systems can help reduce latency requirements by predicting stock prices in advance based on historical data.(1)

However, many applied machine learning methods still requires manual intervention.(2) To solve this problem, some end-to-end deep learning methods have been applied in this field. For example, one of the famous computer vision models, AlphaNet (3), has been modified and applied to predict the stock return. (4) This modified AlphaNet (we call it previous AlphaNet in the following) relies on gated recurrent unit (GRU) to process the time sequential features and performs well on some quantitative trading datasets.

In the fields of natural language processing and sequential data processing, recurrent or convolutional neural networks were dominant models until Transformer was proposed.(5) Transformer eschewed recurrent structures and solely made use of attention mechanism, achieving much better performance than traditional recurrent models on many tasks.(5)(6)(7) Motivated by this, we would like to replace GRU in previous AlphaNet with Transformer to improve the model's ability of processing sequential features.

We trained both previous AlphaNet and our model on different time periods and predicted stock return. We used root mean squared error (RMSE) as the metric and did backtest with a trading strategy as well. For long term return, our model performs better than previous AlphaNet in return prediction, and in backtest, our model also makes more profits in general. In addition, history length analysis shows that the advantage of our model over previous AlphaNet is obvious when history length is not very large. And the ablation experiments verify the effectiveness of almost every part in our model, especially Transformer.

Preprint. Under review.

2 Related Work

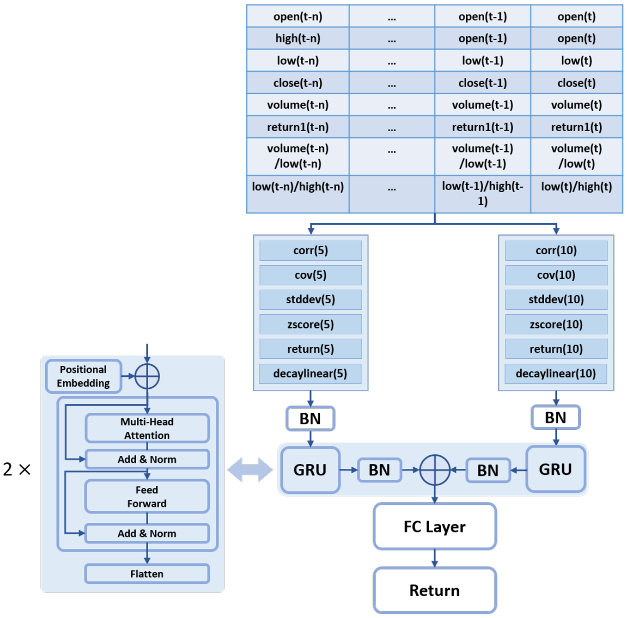
Many existing machine learning methods are being applied in the quantitative trading.(8)(9) K-nearest neighbor algorithm can be used for predicting stock price in business. The predictions can be extremely close and almost parallel to the actual prices. Not only it's robust with very small ratio according to the results, but also the results are rational and reasonable.(10) Boosting algorithm can also be applied in quantitative trading and has great performance on many datasets.(11) However, these models require manual intervention in the training, which might lead to information loss and affect prediction accuracy.

On the neural networks side, graph convolutional networks are used to capture stock market features. Combining the features extracted by the graph convolutional network with CNN, the stock market features and individual stock features are combined into joint features for stock trend prediction(12), which is an effective stock trend prediction method. There is also a lot of research on time series neural networks. The main content is to enable the convolution kernels in CNN to memorize long-term information(13). For example, the number of convolution kernels increases exponentially with the depth of the network. This type of convolutional network has been shown to perform well in extracting high-level features from structured data. However, the structure of the above models is too complex, resulting in poor performance in transfer learning or practical applications, such as running too slowly. We use the above models as inspiration to try to find a simple stock trend prediction model.

3 Method and Approach

3.1 Overall model

The overall structure of our model is shown in Figure 1. For each stock, we construct a data image based on several features of it, which is the input of our model. Two feature expansion layers consisting of six functions will process the input features. The expanded features will be normalized by batch normalization layers. Since the obtained features are still time sequential features, they are processed by sequential models. In the previous AlphaNet, these features are processed by GRU. In our model, we replace GRU with two Transformer encoders to improve the model's ability of processing sequential data. Then, the processed features from two Transformers are processed by batch normalization layers. Next, the normalized features are flattened or pooled after being concatenated. Last, a fully connected layer receives the concatenated feature to predict the label.



open(t-n)	...	open(t-1)	open(t)
high(t-n)	...	open(t-1)	open(t)
low(t-n)	...	low(t-1)	low(t)
close(t-n)	...	close(t-1)	close(t)
volume(t-n)	...	volume(t-1)	volume(t)
return1(t-n)	...	return1(t-1)	return1(t)
volume(t-n) /low(t-n)	...	volume(t-1) /low(t-1)	volume(t) /low(t)
low(t-n)/high(t-n)	...	low(t-1)/high(t-1)	low(t)/high(t)

Figure 1: model structure

3.2 Data image

The data image of each stock is shown in the top of Figure 1. In the data image, each column denotes a time step (a single day), and each row refers a different feature. There are eight features in the data image, open (opening price), high (highest price), low (lowest price), close (closing price), volume, return1 (one-day return), and two ratios volume/low and low/high.

3.3 Feature expansion operator

In our model, there are two feature expansion layers consisting of six operator functions. The definitions of these functions are shown in Table 1. The stride of functions in these two layers are 5 and 10 respectively. The window length is the same as the stride. For each stock, there are 8 different features at each time step. Since corr and cov are calculated for all the pairs of features and others are calculated for a single feature, there are $2 \times \binom{8}{2} + 4 \times 8 = 88$ features for each time window after expansion.

Table 1: Operator Functions

Functions	Definition
corr(X, Y, d)	Correlation coefficients between sequential array X and Y in last d days
cov(X, Y, d)	Covariance between sequential array X and Y in last d days
stddev(X, d)	Standard deviation of sequential array X in last d days
zscore(X, d)	Quotient of average divided by standard deviation of sequential array X in last d days
return(X, d)	(X - delay(X, d))/delay(X, d) - 1, delay(X, d) is the value of X d days ago
decaylinear(X, d)	Weighted average of sequential array X in last d days, weight of last i_{th} day is 2(d + 1 - i)/(d(d + 1))

3.4 GRU

GRU is a simplified variation of LSTM. Its structure is shown in Figure 2. GRU is a recurrent model which takes the memory of the previous step $\mathbf{H}_{t-1}$ as part of the input of the current step t. It consists of a reset gate and an update gate. These two gates takes the current input $\mathbf{X}_t$ and $\mathbf{H}_{t-1}$ as input to compute $\mathbf{R}_t$ and $\mathbf{Z}_t$.

$$\mathbf{R}_t = \sigma(\mathbf{W}_r \cdot [\mathbf{H}_{t-1}, \mathbf{X}_t]) \qquad \mathbf{Z}_t = \sigma(\mathbf{W}_z \cdot [\mathbf{H}_{t-1}, \mathbf{X}_t])$$

Then, the reset gate combines the current input with previous memory.

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{W} \cdot [\mathbf{R}_t * \mathbf{H}_{t-1}, \mathbf{X}_t])$$

Last, the update gate decides the ratio of $\tilde{\mathbf{H}}_t$ and $\mathbf{H}_{t-1}$ contributing to the current output and memory $\mathbf{H}_t$.

$$\mathbf{H}_t = (1 - \mathbf{Z}_t) * \mathbf{H}_{t-1} + \mathbf{Z}_t * \tilde{\mathbf{H}}_t$$

3.5 Transformer

In the fields of natural language processing and sequential data processing, Transformer has good performance and performs much better than recurrent models like LSTM and GRU on many datasets, which shows Transformer's strong ability of processing sequential data.(5)(6)(7) In our model, the expanded features obtained from feature expansion layers are still time sequential features. Therefore, we would like to replace GRU with Transformer to improve the model's performance.

The structure of Transformer is shown in Figure 3.(5) In the left part (encoder part) of Transformer, for input at every step, there is an embedding layer to map each input to embedding vectors. The reason for this is that in field like natural language processing, the inputs are usually words or their indices in a vocabulary list which can't be used for training directly, and embedding layer can transform these values to trainable vector features. In our model, the expanded features are trainable features already, so we don't need to do embedding for them. And our experimental results also prove that embedding is not suitable for our model.

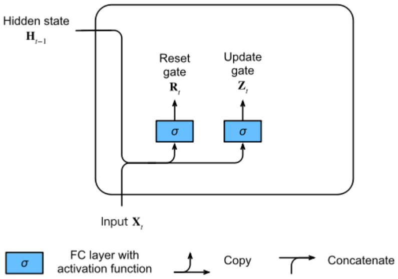


Figure 2: structure of GRU

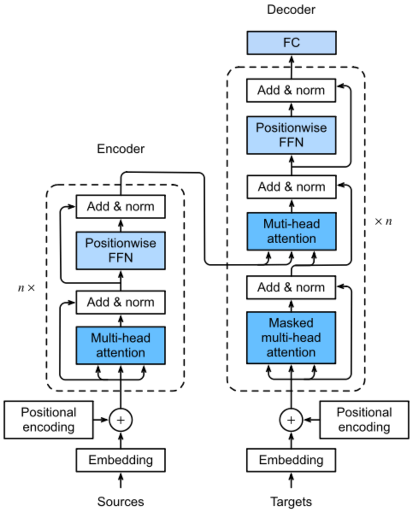


Figure 3: structure of Transformer

After the embedding layer, the embedding vectors are added with positional embedding vectors, since there is no recurrent units in Transformer and adding positional embedding can introduce order of sequence into the model.

Next, the embedded features are fed into a multi-head self-attention layer. For query matrix Q, key matrix K, and value matrix V, the attention output is

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

where d_k is the dimension of query and key. If $Q = K = V$, then it is self-attention. Self-attention allows every position in the input attend to all the positions. In this way, correlation between different parts of the input can be extracted and then used in the following prediction. Instead of doing a single self attention, multi-head attention is used in Transformer to attend different subspaces' information at the same time. For Q, K and V, the multi-attention of them is

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned}$$

where W^O, W^Q, W_K and W_V are all matrices of tunable parameters.

Then, after the multi-head attention layer, the obtained features are added with the input of the multi-head attention layer, and the sum features are normalized and fed into a feed-forward network which consists of two linear transformations with a ReLu activation between them.

The right part (decoder part) of Transformer consists of several parts similar to those introduced above. Since we only use the encoder part of Transformer, we don't talk about the decoder in detail.

3.6 Backtest

In order to evaluate the performance of our model in the real world, a trading strategy and a backtesting framework need to be designed. We designed a simple trading strategy to test our model.

Our trading strategy is to sell a stock when the forecasted return is negative, i.e. predict that the stock will fall, and buy the next time the forecasted return is positive, and vice versa. In our strategy, we hold at most one stock at a time, alternating buying and selling.

Figure 4 is the trading signal chart of the strategy, and we judge the effectiveness of the model in reality through backtesting with the strategy.

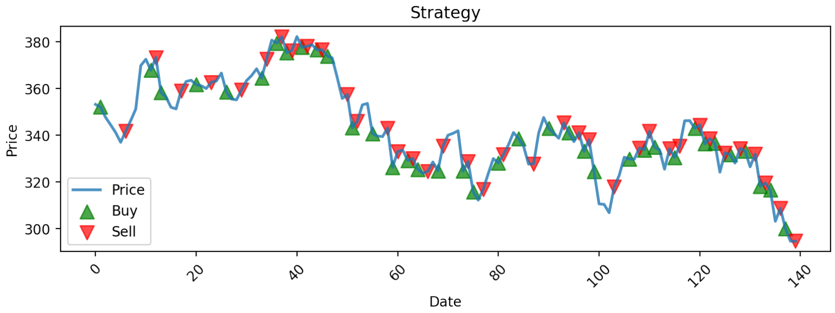


Figure 4: Example of trading strategy

4 Experiments and Results

4.1 Experiments Setup

4.1.1 Data collection

We collect 100 stocks' daily data from Yahoo Finance. The time range is from 01/01/2015 to 10/31/2021. The data we collect includes opening price, highest price, lowest price, closing price and volume.

4.1.2 Pre-processing

First, we use the daily closing price to compute 1-day return, 5-day return, 10-day return, 20-day return and 30-day return. For the t_{th} day, n-day return is

$$\text{Return}(t) = \frac{(p_t - p_{t-n})}{p_{t-n}}$$

where p_t and p_{t-n} is the stock price on the t_{th} and $(t-n)_{\text{th}}$ day. In addition, since we can't compute n-day return for the first n days, we eliminate data on these days. Then, we calculate the ratio of volume and lowest price and the ratio of lowest price and highest price.

4.1.3 Training details

For 5-day return, 10-day return, 20-day return and 30-day return, we use one of them as the label. For each of these labels, we construct datasets every three months from 01/31/2015. We use the data of 800 days (history length) from the beginning date as the training set, and the following 150 days as validation set, and the following 150 days as test set. We use these 9 datasets in different time periods to do the training individually, and we do the training for all the 4 labels.

The hyperparameters we have tuned are shown in Table 2. For each hyperparameter, the bold value(s) is(are) optimal. "#feature extraction layers" refers to the number of branches of feature expansion layers with different strides. "#Transformer block layers" refers to the number of Transformer blocks stacked in each feature expansion branch. "Pooling/Flatten" refers to the operation we do to features obtained from Transformers before concatenation. We set n in data image as 30, which means for each day we use the data from past 30 days to predict its label. The loss function is mean squared error (MSE). Adam is used as optimizer. Learning rate is set to 0.0001. The number of training epochs is 100. The batch size is 200.

4.1.4 Test details

We use the trained model of each time period to do the prediction for test set in the same time period. We use root mean squared error (RMSE) to evaluate the results. The definition of RMSE is

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_{\text{pre}} - y_{\text{true}})^2}$$

Table 2: Hyperparameters

hyperparameters	Values
#heads of the first multi-head attention layer	1, 2, **4**, 8
#heads of the first multi-head attention layer	1, **2**, 4, 8
#feature extraction layers	**2**, 3
stride of feature extraction layers	3, **5**, **10**, 15
#Transformer block layers	1, **2**, 3
Pooling/Flatten	globalAverage, globalMax, **Flatten**
dropout	0, **0.1**, 0.2

where N is the number of stocks, y_{pre} is the predicted label, y_{true} is the true label. In addition, we do backtest for the prediction results, as mentioned in Section 3.6. We do the training and test three times for datasets at each time period and use the average results as test results.

4.2 RMSE

For 5-day, 10-day, 20-day and 30-day return, RMSE of test sets in different time periods are shown in Figure 5. We can see that for 5-day and 10-day return, our model's performance is similar to previous AlphaNet's. For 20-day and 30-day return, our model performs better than previous AlphaNet, especially for 30-day return, our model outperforms for all the beginning dates. The reason for this might be that Transformer is good at extracting very long term dependency in the input, so Transformer performs similar to GRU when the predicting label only relies on short and medium term input (i.e. 5-day and 10-day return), and the advantage of Transformer is obvious only when long term input influences the result (i.e. 20-day and 30-day return).

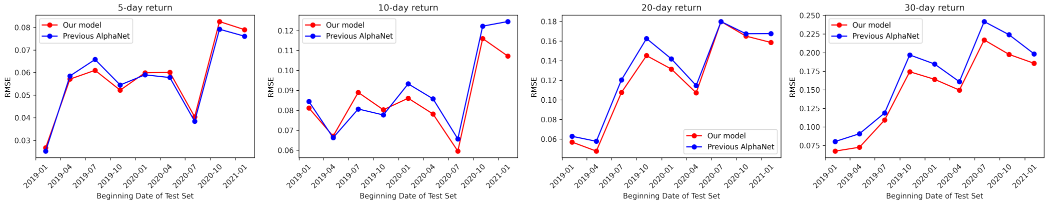


Figure 5: RMSE of different time periods for 5-day, 10-day, 20-day and 30-day return

The prediction and label values of 20-day return from 04/05/2019 to 11/06/2019 and 30-day return from 08/05/2020 to 03/10/2021 of stock "FB" are shown in Figure 6 and Figure 7 respectively. The RMSE of the former is 0.0478 and 0.0578 for our model and previous AlphaNet, and the RMSE of the latter is 0.2171 and 0.2418 for our model and previous AlphaNet. We can see that smaller RMSE corresponds to better regression performance.

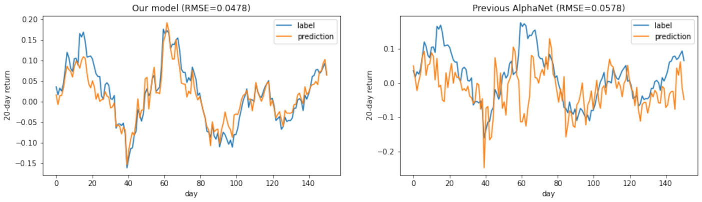


Figure 6: 20-day return predictions and labels from 04/05/2019 to 11/06/2019 of stock "FB"

4.3 Backtest

Figure 8 is the backtest results of the model, the x-axis is the test set start date, and the y-axis is the profit, in dollars per share. This profit is the average profit of all the backtested stocks. The four subfigures represent models that predict 5-day, 10-day, 20-day, and 30-day return, respectively. It can

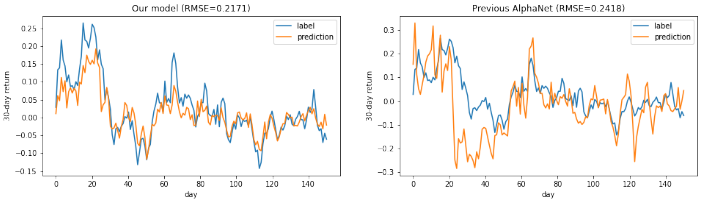


Figure 7: 30-day return predictions and labels from 08/05/2020 to 03/10/2021 of stock "FB"

be seen that our model performs better than the previous AlaphaNet most of the time. From this we demonstrate that our model is effective in reality, and the important role of attention mechanism in processing sequential financial data.

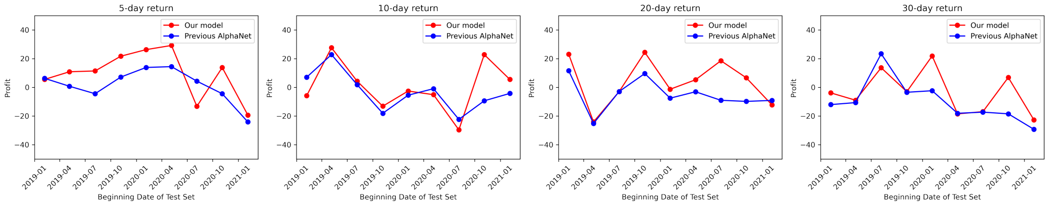


Figure 8: Backtesting profit

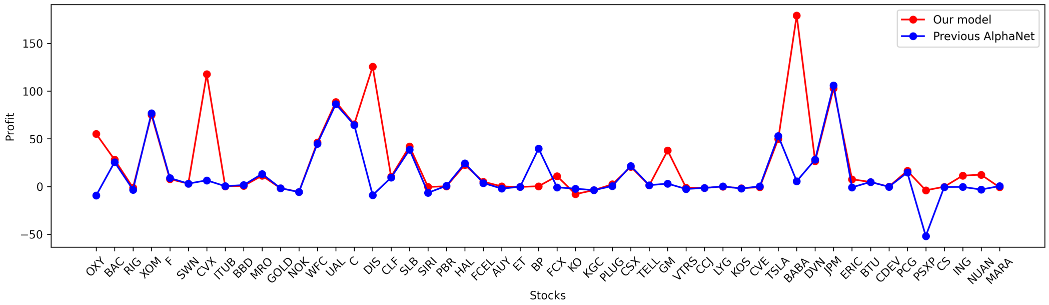


Figure 9: All stock backtesting profit comparison

Figure 9 is an example of the profits of all listed stocks of test set from 01/07/2020 to 08/10/2020 with 20-day return as label, with stock names on the x-axis and backtested profits on the y-axis. As can be seen, our model is similar to the previous AlphaNet in most cases, but achieves much higher profits than the previous AlphaNet on some stocks.

4.4 History Length Analysis

To analize the influence of history length, we construct datasets with history length of 400, 500, 600, 700, 800, 900, 1200, 1500 and 1800. These datasets has the same test set in the period from 08/05/2020 to 03/09/2021 with 30-day return as label, so their training set's time period are different. The results are shown in Figure 10.

As history length becomes larger, RMSE of both our model and previous AlphaNet decrease in general, because more data contributes to better performance generally. When history length is not very large, our model performs much better than previous AlphaNet. When history length is large, two model's performance becomes similar, and previous AlphaNet even outperforms our model sometimes. This shows that our model's advantage over previous AlphaNet might be obvious only when history length is not very large.

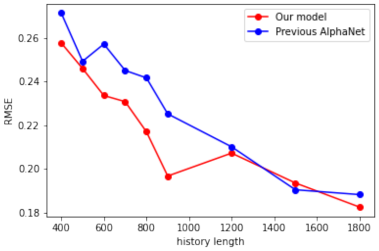


Figure 10: History length analysis

4.5 Ablation Experiments

To analyze the contribution of each component in our model, we did ablation experiments for each part of our model. At each time, we remove one of the following components, feature expansion layer, BN1 (batch normalization layer before Transformer), Transformer, BN2 (batch normalization layer after Transformer), and we train the modified model on the dataset beginning at 04/30/2015 with history length 800 and 20-day return as label. The results are shown in Table 3.

Table 3: Ablation Experiments

Removed component	RMSE
None	0.0478
Feature Expansion Layer	0.0640
BN1	0.1217
Transformer	19.1082
BN2	0.0483

We can see that feature expansion layer has a little contribution to our model, while BN1 is important to the overall performance because batch normalization can help to stabilize training and shorten the time to converge. Transformer is critical to our model since we need sequential model to process the sequential expanded features. In addition, BN2 nearly contributes nothing to the performance. This might because there is a normalization layer at the end of Transformer block. Therefore, additional normalization after Transformer is unnecessary.

5 Conclusions

Overall, our new model performs better than the previous AlphaNet, and the new model with the addition of the attention mechanism based Transformer achieves smaller RMSE in return prediction and more profits in backtest for multiple time periods in 2019-2021 and multiple returns of different duration. This demonstrates the effectiveness of our model and the importance of attention mechanism in financial time series data processing. In addition, history length analysis shows the advantage of our model is more obvious when history length is not too large. And the effectiveness of most components in our model is validated by the ablation experiments.

References

[1] Israel, Ronen, Bryan T. Kelly, and Tobias J. Moskowitz. "Can Machines' Learn'Finance?." Journal of Investment Management (2020).

[2] Lin, Chen, et al. "Mining of stock selection factors based on genetic programming." Financial engineering research of Huatai Securities. 2019.

[3] Sharma, Rishab, Rahul Deora, and Anirudha Vishvakarma. "AlphaNet: An Attention Guided Deep Network for Automatic Image Matting." 2020 International Conference on Omni-layer Intelligent Systems (COINS). IEEE, 2020.

[4] Lin, Chen, et al. "AlphaNet: Factor Mining Neural Networks." Financial engineering research of Huatai Securities. 2020.

[5] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[6] Boogaart, E. "The GRU and Transformer explaining Human Behavioural Data represented by N400." (2019).

[7] Reza, Selim, et al. "A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks." Expert Systems with Applications (2022): 117275.

[8] Nair, Binoy B., V. P. Mohandas, and N. R. Sakthivel. "A decision tree-rough set hybrid system for stock market trend prediction." International Journal of Computer Applications 6.9 (2010): 1-6.

[9] Desai, V.S., Bharati, R. A comparison of linear regression and neural network methods for predicting excess returns on large stocks. Annals of Operations Research 78, 127–163 (1998).

[10] Alkhatib, K., Najadat, H., Hmeidi, I., Shatnawi, M. K. A. (2013). Stock price prediction using k-nearest neighbor (kNN) algorithm. International Journal of Business, Humanities and Technology, 3(3), 32-44.

[11] Rasekhschaffe, K. C., Jones, R. C. (2019). Machine learning for stock selection. Financial Analysts Journal, 75(3), 70-88.

[12] Chen, Wei, et al. "A novel graph convolutional feature based convolutional neural network for stock trend prediction." Information Sciences 556 (2021): 67-94.

[13] Lea, Colin, et al. "Temporal convolutional networks: A unified approach to action segmentation." European Conference on Computer Vision. Springer, Cham, 2016.