# AlphaNet: an explainable deep neural network for alpha mining

Sihan Yang

2025-05-20

# Introduction

This paper is a replication of a research report by Huatai Security, named “AlphaNet: A Neural Network for Factor Mining”. But in this report, we adopt some changes both in data preprocess, model construction and model interpretability. The result confirms that our model can generate stable alpha during both training and testing period. And different from some deep learning methods for alpha mining, our model has strong interpretability, which may be the source of our stable alpha. Finally, for this report structure, it would mainly consist of five parts. The first chapter is about data preprocess, which would tell you how we generate our feature figures and labels. Second is about our model structure, it is the core part for our report which would apply the concept of CNN and genetic algorithm for alpha mining. Model training and predicting is the third chapter and model interpretability is the fourth. Finally, chapter five would give a thorough factor testing using the sketch in our project 1.

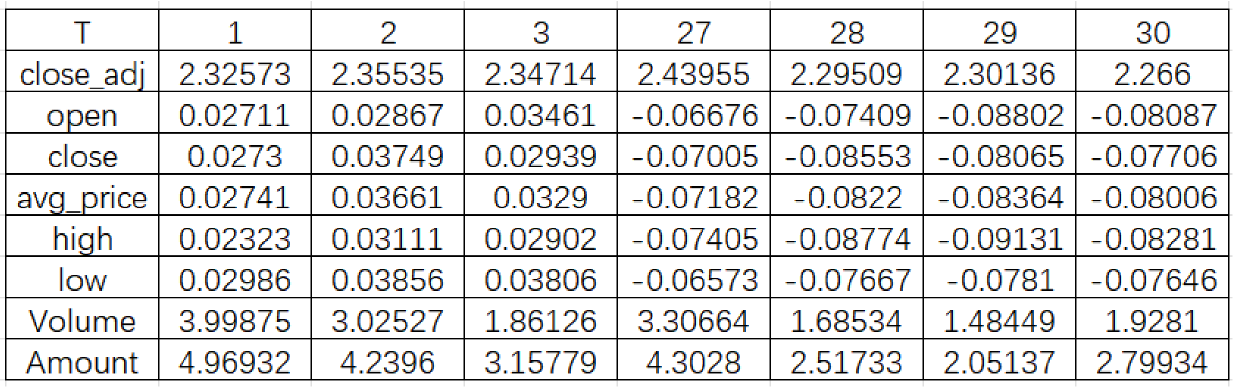
# 1. Data Preprocess

**Feature description:**

The original features fed for our AlphaNet is no difference from our former project. They are just normal end of day data, which consist of 8 columns: open, close, high, low, adjusted close, vwap, trade amount, trade volume. Meanwhile, we use the same time range and stock pool with our former project: January 2020 to February 2025 for time horizon and all A shares except for stocks end with ‘.BJ’.

**Feature to image:**

Inspired by the structure of convolutional neural networks (CNNs), AlphaNet organizes Eod data into a "data image" format for input into the network. The format for our feature image is shown in figure 1. Each day, for every stock, we set a rolling window (30 in our report) and collect all the Eod data during this window, finally we can get a feature image of size 8*30 for an individual stock. We first expand this feature image to cross-section and move our rolling window by 5 days, then we repeat this procedure throughout the time horizon. Finally, our X would be the shape of $\frac{TS}{5} * 8 * 30$, for example, if we have 5000 stocks and 1000 trading days, then we can collect X with shape 1000000*8*30. For labels, we use stock return in the next five days, which is consistent with our moving step.



| T | 1 | 2 | 3 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|
| close_adj | 2.32573 | 2.35535 | 2.34714 | 2.43955 | 2.29509 | 2.30136 | 2.266 |
| open | 0.02711 | 0.02867 | 0.03461 | -0.06676 | -0.07409 | -0.08802 | -0.08087 |
| close | 0.0273 | 0.03749 | 0.02939 | -0.07005 | -0.08553 | -0.08065 | -0.07706 |
| avg_price | 0.02741 | 0.03661 | 0.0329 | -0.07182 | -0.0822 | -0.08364 | -0.08006 |
| high | 0.02323 | 0.03111 | 0.02902 | -0.07405 | -0.08774 | -0.09131 | -0.08281 |
| low | 0.02986 | 0.03856 | 0.03806 | -0.06573 | -0.07667 | -0.0781 | -0.07646 |
| Volume | 3.99875 | 3.02527 | 1.86126 | 3.30664 | 1.68534 | 1.48449 | 1.9281 |
| Amount | 4.96932 | 4.2396 | 3.15779 | 4.3028 | 2.51733 | 2.05137 | 2.79934 |

Figure 1: Feature Image

## 2. Model Structure

**Genetic Algorithm for alpha mining:**

Genetic Algorithm is a very classical algorithm for alpha mining. The main idea for this algorithm is to try different combinations of operators and features as many as possible and find the ones with the best performance. To be clear, in each step Genetic Algorithm will construct many symbolic trees. As shown in figure 2, the figure tree represents the factor:

$$ts\_mean(ts\_corr(close, open))$$

and in the next generation, the symbolic trees with the highest fitness will be reserved as parent, then by inheritance and mutation, they would produce the next generation. One drawback of Genetic Algorithm is that it exhibits a high degree of randomness, as each generation involves probabilistic processes of inheritance and mutation. So, the final performance of our best population would somehow depend on this probabilistic process. Besides, libraries for Genetic Algorithm, such as gplearn, is not based on cuda so that we can not utilize our GPU for acceleration. Considering these, we propose a new model, which corporates Genetic Algorithm into deep neural network.

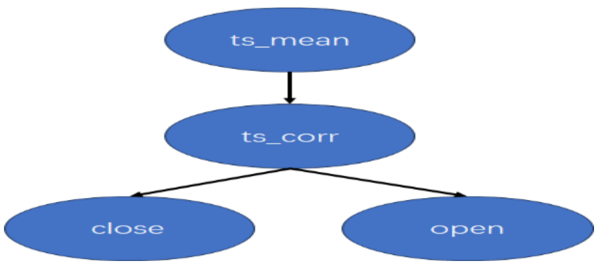


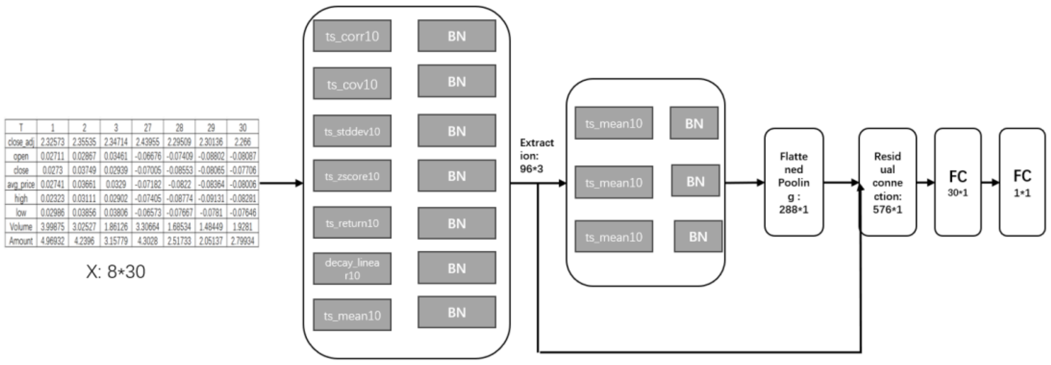
Figure 2: Symbolic Tree for Genetic Algorithm

**AlphaNet Structure:**

In our AlphaNet, we designed seven operators, each implemented as a distinct layer within the Network:

$ts\_corr(X, Y, d)$: the correlation coefficient of feature X and Y in the past d days.

$ts\_cov(X, Y, d)$: the covariance matrix of feature X and Y in the past d days.
$ts\_stddev(X, d)$: the standard error for X in the past d days.
$ts\_zscore(X, d)$: Normalized X using the past d days sample.
$ts\_return(X, d)$: Percentage change for X during past d days.
$decay\_linear(X, d)$: Weighted X in the past d days with the weight: $[\frac{1}{\sum_1^d i}, \cdots \frac{d}{\sum_1^d i}]$.
$ts\_return(X, d)$: Mean value for X during past d days.
The input of these layers is our eight original features, but the output would exhibit different shapes depending on our operator. For $ts\_corr$ and $ts\_cov$, the output will be 28, which is $C_8^2$ since we tried every combination of 2 features and operators. For the remaining, the output remains 8. Since for each sample, the X is a feature image, which is a 2-dimension input with shape 8*30, so the output is also a 2-dimension features whose shape is depend on d. Here we set d=10, so the final columns for our output would be $\frac{30}{10} = 3$.
Once constructing all the operators, we embed them all into our network as the first layer, which is called features extracting layer. Throughout this layer, we can obtain $28 * 2 + 8 * 5 = 96$ features in total. Then, we process these features by a Batch Normal layer.
After the feature extraction layer is the pooling layer. Pooling method derives from CNN, it uses different pooling functions such as max, min and mean to extract the significant information from the features after convolution layer. In AlphaNet, we tried both max, min and mean function so that our features number will increase to $96 * 3 = 288$ for each sample. The same as the former layer, we add Batch Normal after the pooling.
Afterwards, we connect the flattened features after extraction layer, which is $96 * 3 = 288$ and features after the pooling together, which is also 288 in length. This is called residual connection method, which is meant to learn information both from features and the difference between each layer' s output. The residual connection output would then be fed into a fully connection layer with output length of 30. After activation function and dropout, we add another fully connection layer and finally get our prediction. The whole structure of AlphaNet is shown in Figure 3.



| T | 1 | 2 | 3 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|
| close_adj | 2.32573 | 2.35535 | 2.34714 | 2.43955 | 2.29509 | 2.30136 | 2.266 |
| open | 0.02711 | 0.02867 | 0.03461 | -0.06676 | -0.07409 | -0.08802 | -0.08087 |
| close | 0.0273 | 0.03749 | 0.02939 | -0.07005 | -0.08553 | -0.08065 | -0.07706 |
| avg_price | 0.02741 | 0.03661 | 0.0329 | -0.07182 | -0.0822 | -0.08364 | -0.08006 |
| high | 0.02323 | 0.03111 | 0.02902 | -0.07405 | -0.08774 | -0.09131 | -0.08281 |
| low | 0.02986 | 0.03856 | 0.03806 | -0.06573 | -0.07667 | -0.0781 | -0.07646 |
| Volume | 3.99875 | 3.02527 | 1.86126 | 3.30664 | 1.68534 | 1.48449 | 1.9281 |
| Amount | 4.96932 | 4.2396 | 3.15779 | 4.3028 | 2.51733 | 2.05137 | 2.79934 |

Figure 3: AlphaNet Structure

# 3. Training and Predicting

## Data Set division:

The time horizon for our dataset is 2020-01~2025-02, we divide the first 3 years into train set, 2023-01~2024-01 into validation set, 2024-01~2025-02 into test set. Batch size is set to be 1000, which means model parameters are learned from the 1000 samples every batch. Considering the time correlation, we do not shuffle our data set.

## Loss Function:

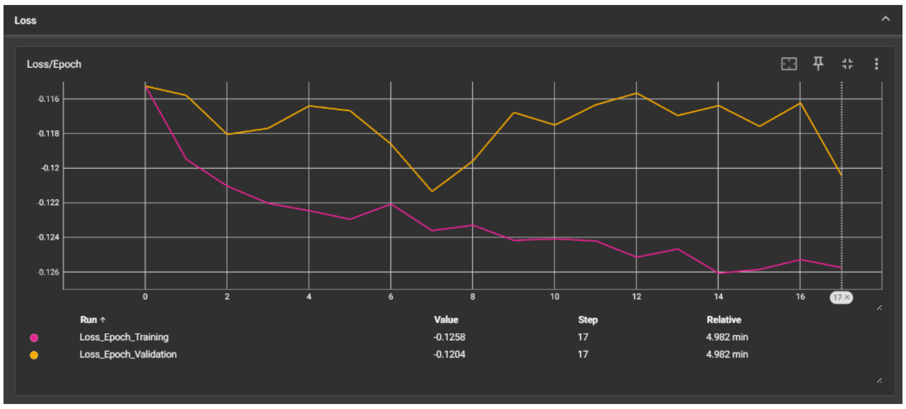
Since our target is to mine the best performed alpha, so we are not necessarily need to predict the return as accurate as possible. From what we have learned in this class, IC is good criteria to evaluate a factor’ s performance. Here we set our loss function as:

$$Loss = -abs(IC)$$

For every batch, we calculate the loss of the 1000 samples, added it to the total loss. Our target is to find the parameters with the smallest loss in validation set (which is the highest IC essentially). Since correlation coefficient is a differentiable function, and there are ways to calculate the gradient of absolute values, so our loss function can be found minimal using gradient descent methods.

## Model training:

We have 50 epochs in total, which means our model would update its’ parameters from all the input through 50 times. To avoid overfitting and time wasting, we set an early stopping of 10 epochs. If the loss in validation set does not decrease for more than 10 epochs, our model will stop training. Learning rate is set 0.001. Training and Validation Loss is shown in Figure 4.



| | Run ↑ | Value | Step | Relative |
|---|---|---|---|---|
| ● | Loss_Epoch_Training | -0.1258 | 17 | 4.982 min |
| ● | Loss_Epoch_Validation | -0.1204 | 17 | 4.982 min |

Figure 4: Training and Validation Loss

The figure clearly displays training and validation loss movement with epochs. Throughout the training process, the training loss keeps decreasing while validation loss reaches the minimal at epoch 7 with value -0.1258, so model parameter stops updating at epoch 17 where early stopping is triggered.

## Model predicting:

Since we do not shuffle our dataset when dividing dataset, we can easily match each feature and predictions with the corresponding date and tickers. After training, the best performed model will be saved in a .pt file, by loading this set of parameters into an AlphaNet model and input features with corresponding date and ticker, the prediction for both training, validation and prediction data set can be obtained.

# 4. Model Interpretability

**From perspective of factor investment:**

Let’ s review what Genetic Algorithm do for factor mining. Actually, this algorithm will finally output a symbolic tree with the best fitness throughout all the generations, and one symbolic tree represents one alpha. This is the first procedure during factor investment: single factor investment. The next procedure is factor combination, which combines different factor into a synthetic factor with a better performance by our criteria. Look back to AlphaNet, the extraction and pooling layer actually have done the factor mining procedure, while the fully connection layer plays a role in factor combination. It combines factors from genetic operators together in order to get a smaller loss. In general, AlphaNet serves as an integrated framework for both factor discovery and factor combination.

**Gradient methods:**

Recalling the structure of our model, before the fully connection layer, our residual connection is shaped as 576*1, and the last 288 features are generated from the pooling layer. Once we can define the importance of residual connection output, we can define the most significant alphas upon matching their symbolic combination. One way to define the importance is to calculate output’ s gradient.

In Neural Network, model parameters are updated through backward propagation, where gradient is initially calculated from the loss and propagated backwards until reaching the input X. During this process, gradients during the intermediate layers are also calculated. Smaller negative gradient means greater impact in reducing the loss.

**Alpha expression match:**

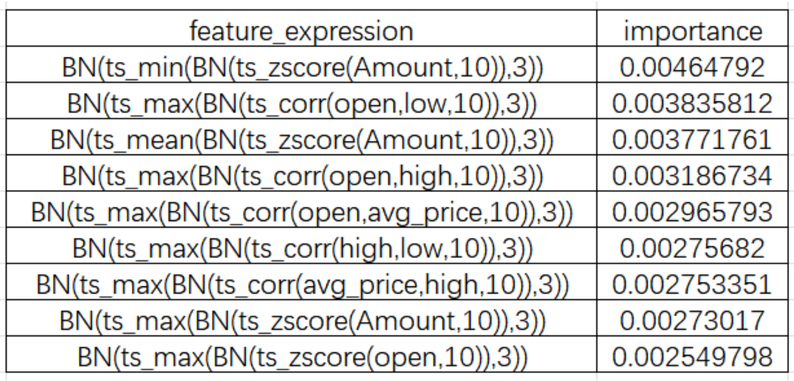
In the feature extraction layer, we do not generate the combination randomly, instead, by some orders. For example, the first feature here is ‘close_adj’ , the second feature is ‘open’ , and the first operator is ‘ts_corr’ . Knowing these, we can deduce the first three alphas in the last 288 columns of residual connection:

1: $BN(ts\_mean(BN(ts\_corr(close_{adj}, open))))$

2: $BN(ts\_max(BN(ts\_corr(close_{adj}, open))))$

3: $BN(ts\_min(BN(ts\_corr(close_{adj}, open))))$

The other 285 expressions can be deduced in the same way. Upon knowing the alpha expression and there corresponding gradient, we can find the expressions with the smallest gradient, which is of great importance to our model. Top 10 important alphas are shown here:



| feature_expression | importance |
|---|---|
| BN(ts_min(BN(ts_zscore(Amount,10)),3)) | 0.00464792 |
| BN(ts_max(BN(ts_corr(open,low,10)),3)) | 0.003835812 |
| BN(ts_mean(BN(ts_zscore(Amount,10)),3)) | 0.003771761 |
| BN(ts_max(BN(ts_corr(open,high,10)),3)) | 0.003186734 |
| BN(ts_max(BN(ts_corr(open,avg_price,10)),3)) | 0.002965793 |
| BN(ts_max(BN(ts_corr(high,low,10)),3)) | 0.00275682 |
| BN(ts_max(BN(ts_corr(avg_price,high,10)),3)) | 0.002753351 |
| BN(ts_max(BN(ts_zscore(Amount,10)),3)) | 0.00273017 |
| BN(ts_max(BN(ts_zscore(open,10)),3)) | 0.002549798 |

Figure 5: Feature Importance

## 5. Factor Backtesting

**Backtesting rules:**

Since our model use the returns in the next 5 days as labels, so the rebalanced day for our factor is naturally set 5. During each rebalanced day, we drop out stocks whose high equal to low or trade state equal to False. Dividing them into 5 groups and calculate hierarchical returns and long-short returns. The backtest horizon covers all the training, validation and testing dataset, from 2020-01 to 2025-02.

**Alpha hierarchical performance:**

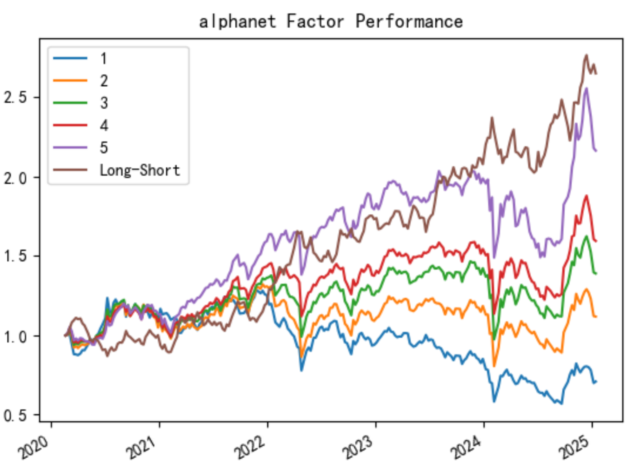


Figure 6: Hierarchical Performance

Hopefully, AlphaNet factor performs well both in training and testing dataset. This factor exhibits clear stratification with 5 the best and 1 the worst. What’ s more, LS return is not generated from the bad performed layer 1. Layer 5 has very desiring long return, nearly 150% in 5 years and 50% in training period.

**IC performance:**

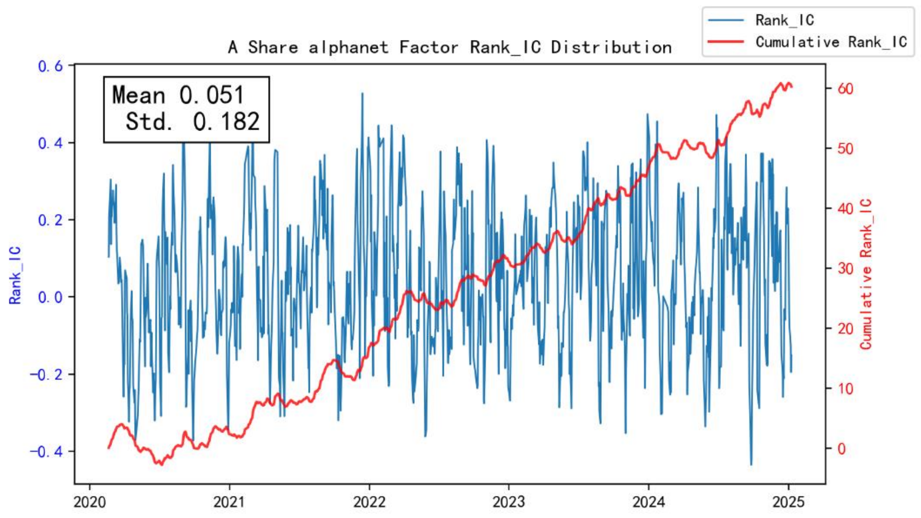


Figure 7: IC Performance

Since we use the IC function in our project 1 where daily frequency alpha is required, we have to fill our prediction with the nearest former one to get daily frequency, which contribute some error in calculating the IC. However, the IC is still satisfying, especially for alpha generated by machine learning methods. This IC indicate that our model in testing set is as same robust as our training set.

**Hit rate:**

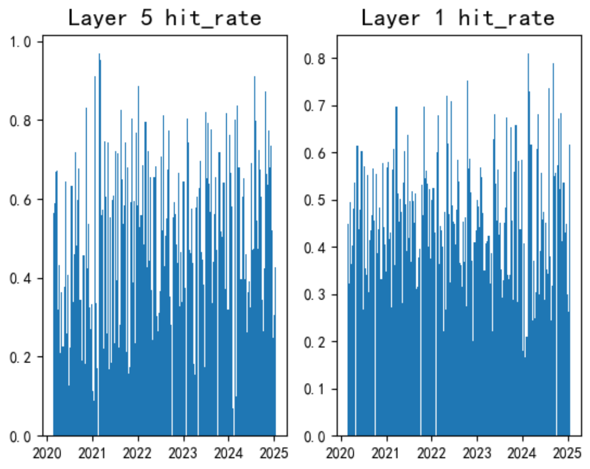


Figure 7: Hit Rate

Since stock pool is All A share, we use ‘choice_a’ as our benchmark. Clearly, layer 5’ hit rate remains higher than layer 1 throughout the whole period.

## Conclusion

AlphaNet presents a novel and interpretable deep learning framework for alpha factor mining by integrating convolutional neural networks with genetic algorithms. Through structured data preprocessing, custom feature extraction layers, and a unique loss function based on the Information Coefficient, the model effectively identifies and combines predictive factors. Extensive backtesting demonstrates AlphaNet’ s ability to generate stable and robust alpha across training and testing periods. Its interpretability, achieved through gradient analysis and symbolic expression matching, sets it apart from traditional black-box models. AlphaNet offers a promising direction for combining machine learning with financial domain knowledge in quantitative investing.

## Reference

AlphaNet: A Neural Network for Factor Mining, Huatai Security