Market Prediction with Ensemble Deep Learning

Jingfeng Xiao[1] **Tongfei Zhou**[1] **Yichen Ji**[1]

[1]**University of Toronto**
{vjf.xiao, tongfei.zhou, yc.ji}@mail.utoronto.ca

Abstract

We trained a large ensemble of deep learning frameworks where FFNNs, CNNs and Transformer models are included to forecast the return rates of 3773 investment assets in the market. We also utilized this model in the Ubiquant Market Prediction competition on Kaggle to learn 300 anonymized features and achieved 46/2893 in the public leaderboard ranking (top 2%).

1 Introduction

The efficient market hypothesis (EMH) states that the financial market is informationally efficient if any accessible information about future capital values is reflected in the market price(2). Consequently, any analysis based on publicly available data, such as historical share prices, macroeconomic variables, company-specific disclosure information, will not generate consistent excess returns. However, anomalies could occur once or repeatedly in the real market, which deviates from the rules of EMH(4). In recent years, state-of-the-art deep learning techniques have captivated greater attention in quantitative finance due to their prominent expressibility and potential power in capturing anomaly signals and alpha-generating features(3).

In this paper, we propose an ensemble framework of deep learning architectures, including Feed-Forward Neural Networks (FFNNs), Convolutional Neural Networks (CNNs) and Transformer models, to seek better generalization performance in market return prediction.

2 Related Work

Our model is inspired by lots of recent empirical work in return prediction problems and deep learning frameworks for financial time series. (6) shed light on using ensemble methods such as bagging and stacking for better out-of-sample generalization in stock prediction. The blending ensemble model proposed by (5) outperformed the best existing prediction model substantially in terms of various evaluation metrics such as mean-squared error (MSE), F1-score, precision rate, etc..

(8) proposed to convert 1D technical analysis factors to 2D images for trend forecasting, which was an important motivation for us to incorporate CNN-based architectures into our ensemble model. The dataset provided by the Kaggle competition can be transformed into a 3D tensor where each dimension represents `time_id`, `investment_id` and `feature_id` respectively. Therefore, 2D convolutional layers can perfectly capture spatial information in such image data.

Moreover, due to the time-dependence property of our data, attention-based models can be applied for feature learning and extraction. (11) showed that the attention mechanism could adaptively assign different weights to each input feature sequence to choose the most relevant features automatically.

3 Dataset

The dataset contains anonymized features of investments derived from real historical data, and our task is to predict the investments' return rates based on the anonymized features in a future time range. The first column in the training dataset is `time_id`, denoted with a number i ranging from 0 to 1219. The second column is `investment_id`, with number i indicating the i-th investment, ranging from 0 to 3773. The third column is `target`, which indicates the investment return rates for an investment at a certain time. The rest of the columns are anonymized features generated from market data. All the feature and target columns have been normalized. There are 3141410 rows in the training data ordered in time, with each row a return rate for an investment at a certain time, together with 300 anonymized features. The preprocessing details can be found in the **Appendix** section.

4 The Ensemble Deep Learning Method

In this section, we discuss three deep learning architectures in our ensemble model and illustrate how we integrate them together as our ensemble deep learning method in the algorithm box.

4.1 Feed-Forward Neural Network (FFNN)

Two variants of Feed-Forward Neural Network(FFNN) were proposed to extract the distinct of multivariate financial time series: (i) a feed-forward neural network with bi-way input of `investment_id` embedding and feature data (called `FFNN-dr`) (ii) a feed-forward network that uses `time_id` as feature augmentation besides `investment_id` and features (called `FFNN-ft`).

Considering the temporality of investment feature data and the unique time series footprint of individual investment instruments, `investment_id` and feature are used as two-way input for the `FFNN-dr`. The first block of the architecture contains two sets of blocks that take in the `investment_id` embedding and feature input, consisting of three groups of one linear layer and one dropout layer. In the second block of `FFNN-dr`, the two tensors output from the previous block sets are concatenated and fed forward through another four groups of linear layers with a dropout layer to produce the final prediction. However, as investment instruments in the dataset appear to have variable footprints, some short-lived investments can negatively disturb the model's out-of-sample performance. Such, `FFNN-ft` expands on this idea by introducing `time_id` to the feed-forward neural network. This model is primarily a shallow one, as we use only one set of linear layers with per-layer dropout implementation for `investment_id` and `time_id`, and two sets of such layers for the feature input. Afterward, the three tensors are concatenated to compute through the final linear layer for prediction.

4.2 Convolutional Neural Networks (CNN)

Although financial time series are sequential by nature, some spatial relationships may exist within the investment instrument pools. To detect such connections, Convolutional Neural Networks(CNN) can be exploited to improve forecasting capacity on this complex multivariate time series (7). We constructed two variants of the CNN architecture to improve feature extraction ability: (i) CNN model with Gaussian noise layers and 1D convolutional layers (called `CNN-gn`); (ii) CNN hybrid model with 1D and 2D convolutional layers (called `CNN-2d`).

The architecture of the CNN can be grouped into three blocks as follows. The first block aims to prepare the input for convolution and is thus constituted by a linear layer, a dropout layer, and a reshape layer. An additional Gaussian-Noise layer is added to reduce overfitting in `CNN-gn`. The second block consists of five groups of 1D convolutional layers, a batch-normalization layer that normalizes the values of the previous layer's features to stabilize training, and a leakyReLU layer to add non-linearity. In this block of `CNN-2d`, the 1D convolution layer of the last three groups is replaced by a 2D convolutional layer after reshaping the previous layer. The last block aims to unify outputs of feature maps across all the convoluted tensors. Such, we flattened the tensors and utilized another fully connected layer with dropout regularization to produce our return prediction.

4.3 Attention-based Model

Besides the CNN architectures described above, the relationships among features can also be extracted through an attention mechanism. For each investment at certain time, we first applied an embedding layer for the `investment_id`, and then we applied a single traditional Transformer-encoder(9) block to 300 features for an investment. The learned context vector is further fed into the `FFNN_dr` described above to predict the investment return rate. We trained this model with five different initialization so that we may arrive at a better optimal point in the loss space with reduced variance. The final result is the average of five predicted return rates.

4.4 Algorithm for the Ensemble Deep Learning Method

The hyperparameter settings for all contributing models are illustrated in **Table 1**, and pseudo-codes for our ensemble deep learning model is shown in **Algorithm 1**. We applied the `ensemble()` function to each contributing model by training each model with five different initialization. Notice that the competition will score our prediction by Pearson Correlation Coefficient, but we still chose `Mean_Squared_Error` as the loss function. After minibatch experiments, we found that using Pearson Correlation Coefficient directly as the loss function resulted in extremely unstable training, so we used `Mean_Squared_Error` as a proxy metric but with more stable training outcomes.

Models	Batch Size	Dropout Rate	Optimizer	Loss Function	Epochs
`FFNN-dr`	4096	0.4	Adam with learning rate 0.001	`MSE`	30
`FFNN-ft`	4096	-	RMSProp with learning rate 0.001	`MSE`	30
`CNN-gn`	4096	-	Adam with learning rate 0.001	`MSE`	30
`CNN-2d`	4096	-	Adam with learning rate 0.001	`MSE`	30
`Attention-based`	512	-	Adam with learning rate 0.0005	`MSE`	30

Table 1: Hyperparameter settings of each contributing model

Algorithm 1 Ensemble Deep Learning Method

Input:
$\texttt{features} \leftarrow (\texttt{batch\_size}, \texttt{num\_feature} = 300)$
$\texttt{investment\_id} \leftarrow (\texttt{batch\_size}, 1)$
$\texttt{time\_id} \leftarrow (\texttt{batch\_size}, 1)$
Process:
$\texttt{output}_1 \leftarrow \texttt{ensemble}(\texttt{FFNN\_dr}(\texttt{investment\_id}, \texttt{features}))$
$\texttt{output}_2 \leftarrow \texttt{ensemble}(\texttt{FFNN\_ft}(\texttt{investment\_id}, \texttt{features}, \texttt{time\_id}))$
$\texttt{output}_3 \leftarrow \texttt{ensemble}(\texttt{CNN-gn}(\texttt{features}))$
$\texttt{output}_4 \leftarrow \texttt{ensemble}(\texttt{CNN-2d}(\texttt{features}))$
$\texttt{output}_5 \leftarrow \texttt{ensemble}(\texttt{Attention-based}(\texttt{investment\_id}, \texttt{features}))$
$\texttt{return\_rate} \leftarrow \sum_{i=1}^{5} \texttt{weight}_i \times \texttt{output}_i$
Output: `return_rate`

5 Experiments

We tested several deep learning models on the prediction task without ensemble, and compared the performance of each individual model with our ensemble model as described in Section 4. The prediction quality is measured by Pearson Correlation Coefficient Score (PCC) in the evaluation phase of the Kaggle competition.

Model	PCC
XGBoost (n_estimator: 800, max_depth: 12, learning_rate: 0.05, subsample = 0.9)	0.1222
Attention-based	0.1379
Convolution Net	0.1538
Ensemble Deep Learning Model	**0.1567**

Table 2: Pearson Correlation Scores of benchmark models and ensemble model

Table 2 shows the performance of individual benchmark models and our ensemble model. Without utilizing the ensemble technique, the CNN architecture had the best performance. The reason might be that the convolution kernels have captured the relationship among different investments and the relationship across time. The single Transformer-encoder block did not capture such information very nicely, and the XGBoost is too simple to capture such complicated relationships.

6 Conclusions, Limitations and Future Work

6.1 Conclusion

We propose an ensemble deep learning model that incorporates sequential and spatial perspectives of financial time series data. By stacking three types of common deep learning architectures and fine-tuning the weights of predicted return values, the ensemble model achieved better generalization performance than any single contributing model in the Pearson Correlation Coefficient (PCC).

6.2 Limitations

It is worth noting several limitations in our model and deep learning applications in finance in general.

Data with low signal-to-noise ratio. Financial time series data naturally consists of large amounts of noises from irrational behaviors of market participants. However, deep learning architectures are designed to recognize the underlying patterns of data. Consequently, models may fall into data dredging or fishing random noises instead of detecting features with clear economic interpretations.

Feature engineering. Lack of fine-grained data preprocessing and feature engineering may result in the flaw of "Garbage-In-Garbage-Out (GIGO)". It is because anomalies emerging in the financial market are rapidly changing over time, and signals among features captured by models should also change. Well-behaved models in the past can have unsatisfying performance in the future, inferring low consistency and reliability of deep learning models in such prediction tasks.

6.3 Future Work

Try other architectures. Temporal Convolutional Networks (TCN)(10) have the potential to combine the convolutional networks and time-series data together. Since our dataset consists of different investments with 300 features given fixed `time_id`, we may consider feature values of all the investments at each time as an image and the entire dataset as a video. However, since the task here is to predict the return rate of each investment at a certain time, the domain of the model has shifted, and so further research on the domain transfer and domain shift is in our interests as well

Feature learning and AutoML using AutoGluon. The features in the dataset are anonymized, so it is nearly impossible to find real-life explanations of feature values, which require more subtle feature extraction and representation learning. Moreover, the generalization performance of the ensemble framework heavily depends on which contributing models to utilize. We attempted to use AutoGluon, a convenient automated machine learning module(1), but it requires an enormously large RAM size to run. If adequate compute is available, we will use this tool to contribute to model selection.

References

[1] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola, *Autogluon-tabular: Robust and accurate automl for structured data*, arXiv preprint arXiv:2003.06505 (2020).

[2] Eugene F Fama, *Efficient capital markets: A review of theory and empirical work*, The journal of Finance **25** (1970), no. 2, 383–417.

[3] JB Heaton, Nicholas G Polson, and Jan Hendrik Witte, *Deep learning in finance*, arXiv preprint arXiv:1602.06561 (2016).

[4] Madiha Latif, Shanza Arshad, Mariam Fatima, and Samia Farooq, *Market efficiency, market anomalies, causes, evidences, and some behavioral aspects of market anomalies*, Research journal of finance and accounting **2** (2011), no. 9, 1–13.

[5] Yang Li and Yi Pan, *A novel ensemble deep learning model for stock prediction based on stock prices and news*, International Journal of Data Science and Analytics **13** (2022), no. 2, 139–149.

[6] Isaac Kofi Nti, Adebayo Felix Adekoya, and Benjamin Asubam Weyori, *A comprehensive evaluation of ensemble learning for stock-market prediction*, Journal of Big Data **7** (2020), no. 1, 1–40.

[7] Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman, *Stock price prediction using lstm, rnn and cnn-sliding window model*, 2017 international conference on advances in computing, communications and informatics (icacci), IEEE, 2017, pp. 1643–1647.

[8] Omer Berat Sezer and Ahmet Murat Ozbayoglu, *Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach*, Applied Soft Computing **70** (2018), 525–538.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in neural information processing systems **30** (2017).

[10] Renzhuo Wan, Shuping Mei, Jun Wang, Min Liu, and Fan Yang, *Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting*, Electronics **8** (2019), no. 8, 876.

[11] Xuan Zhang, Xun Liang, Aakas Zhiyuli, Shusen Zhang, Rui Xu, and Bo Wu, *At-lstm: An attention-based lstm model for financial time series prediction*, IOP Conference Series: Materials Science and Engineering, vol. 569, IOP Publishing, 2019, p. 052037.

7 Appendix

Here are the supplement details and explanations for the dataset, models' architectures and training procedures.

7.1 Preprocessing on Dataset

Since the competition did not provide the dataset for testing the deep learning model, and therefore, we applied a stratifed k-fold with $k = 5$ on the training dataset to be able to train and validate the deep learning models we described above. After the training and validation process, as the test dataset has only `time_id`, `investment_id` and 300 features, we need to feed these inputs into our model and added the predicted `target` column to the test dataset, and sumbit this dataset to the platform for scoring. Also, as our dataset preserves the time order, the stratified k-fold is actually a Combinatorial Purged Cross-Validation trick.

Specifically for this competition, the host set up public and private datasets with different `investment_id`. Since we only had the access to the public datasets, we applied a **random-mask** trick to the `investment_id` column so that we enforce the embedding layer of the `investment_id` to see Out-of-Vocabulary (OOV) `investment_id`. In the implementation, we did this by introducing an `investment_id` with number 0, which is not existed in the original training dataset. Such trick may help our contributing models to learn some "common knowledge" of the `investment_id`.

In the training dataset, it is however not guaranteed that for each day, there is an id assigned to an investment, i.e., not every row has a value for `investment_id`. To solve such issue, we applied an IntergerLookUp Layer to map the `investment_id` into a contiguous region before we applied the embedding layer.

7.2 Model Repositories

One can find the architectures of all contributing models and ipynb notebook files in our project repository

7.3 Contribution

Jingfeng Xiao

Jingfeng designed the general ensemble framework for the competition, preprocessed the dataset, documented all the code and submitted the final version to the competition for scoring. He also designed and tuned the FFNN and CNN architectures.

Tongfei Zhou

Tongfei designed and tuned the Attention-based model, incorporated the Attention-based model into the general framework and compared different models' performance with the benchmark model.

Yichen Ji

Yichen designed and tested the AutoGluon method, analyzed the difference among different loss functions. He also critiqued the limitations and future work of this ensemble framework.