Rich Overnight : CS 7643

Tianshi Wang Xiaoka Xiang Tianyu Yang Feng Zhu
Georgia Institute of Technology

$\{twang495, xxiang31, tyang372, fzhu47\}$@gatech.edu

Abstract

Financial market forecasting for the near future has draw intensive interest from professionals and retail investors. However, it is challenging to use the time series data e.g. prices and volumes for deep learning for two reasons. The first challenge is how to encode the time series data into a format a deep learning model can recognize the pattern within it. Secondly, the number of training examples usually is too small to train a complex deep learning model. In this project, we address the encoding problem by designing seven different layers each of which is constructed using different market data or encoding method in order to expose the recognizable patterns for the models. We use transfer learning to overcome the small number of training data for each stock leveraging the similarity between different stocks. Specifically, we firstly train our convolutional neural network (CNN) model on all stocks, then fix the weights of all the layers except the last one and fine-tune the model for each stock. With the trained model, we proposed several different trading strategies and test their performance. We compared the results with the benchmark strategy and discussed the strength and limitation of the model. Finally, we visualize the model to understand the learned patterns.

1. Introduction/Background/Motivation

1.1. Problem statement

Since the beginning of finance markets, predicting asset price move in the near future has attracted intensive attentions from investors and researchers. In recent years, the strong performance of the stock market further catalyzed the interest. The recent development of computer vision provides the possibility that the market data can be transformed to graph first and we can leverage the sophisticated computer vision models to recognize the patterns and predict market move. In this work, we are looking to successfully predict the market move in the next day and develop profitable strategies for stock investment combining encoding time series to graphs and deep learning technologies. From the prediction of each stock, we then determine the right time to performing day trade and make more profit. This work is inspired by Barra, et., al's paper *Deep Learning and Time Series-to-Image Encoding for Financial Forecasting* [4]

1.2. Current approach and its limitation

In *the Universal Principles of Successful Trading* [3], Brent Penfold introduced four trading styles: intraday, short term, medium term, or long term. Long term investors usually look for an undervalued company, but as US stock market is usually considered as a strong or semi-strong efficient one, it is no easy to identify undervalued companies for normal traders. Short term traders, on the other hand, studies the market trend by investigating technical indicators such as ADX, DMI, MACD, etc. or by looking at analysis of price, volume and patterns (such as Elliott wave principle or Dow theory) . Successful traders usually spend years to study those trends, and has been making some good profit, but for amateurs, it is just almost impossible to study these theory, and too easy to neglect their plan due to emotions. Because of that, for the past half century more people and organizations start using algorithm to help them making decision. Traditional machine learning algorithms for stock prediction usually use Time-series technical indicators as independent variable, and price as dependent variable to train a regression or classification model, recently more people start using deep learning.

The current approaches from literature have severe limitations. For the traditional machine learning approach, a major limitation is the lack of composition of features. For example, assuming an observed trade trends that *consecutive three days of profit suggests a higher chance of loss on the fourth day*, it can be successfully captured by the traditional time series analysis. However, more composition information, e.g. *consecutive three days of profit suggests a higher chance of loss in the following several days* requires transnational invariance which is a feature traditional machine learning models usually don't have. Therefore, recently many researchers start to apply deep learning models for stock investment strategy. Currently, some deep learning approaches from literature take into account of the transitional invariance and moreover considers the auto-correlation between prices of different dates. However, this approach only considers the price information of a single stock or index e.g. S&P500. In traditional financial strategy, trade volume data and its correlation with other stocks are also import factors which have been ignored by the current researches to the best of our knowledge. Besides the lack of leveraging volume and correlation data, current approaches are limited by the available training examples: only 1000 examples can be constructed from a 5 year daily data set which usually is not large enough for powering a deep learning model.

In this work, we try to overcome the limitations of current approach by encoding the time series data of price, volume, and correlation with S&P500 index for five tickers: *AAPL*, *QQQ*, *AMZN*, *INTC*, *MSFT*. The time series data are first transformed to seven layers each of which leverages different trade data i.e. price and volume as well as different encoding methods i.e. gram matrix dot product and recurrence matrix. We then train convolutional neural network (CNN) to learn the composition information of the low level layers. To leverage the correlation of different stocks, we perform transfer learning on all stocks and finely tune a layer of the model for each stock. Finally, based on the prediction, we propose different trading strategies and test on the test sets.

1.3. Impact of this work

Professionals in finance and retail investors have been trying to predict the stock performance in the near future since the beginning of financial market. The research in this area is very active and challenging due to the uncertainty of the market. A better model helps market participants to understand the drive of the market and make wiser investment decisions. A successful model also will help us make multiple tradessuch that at the end of our analysis period our total return or unrealized gain and loss should be better compared to simply buy and hold.

1.4. Data

In this project, we use *YFinance package* [2] to get the past five-year end of day (EOD) data for the six tickers mentioned above and the S&P500 index. We use the adjusted closed price and volume daily adjusted close price for SPY as our key dataset. Value and closed price are the primary features in stock trading, and we want to only use the fundamental data.

2. Approach

We first transform the time series data to a graph of seven channels each of which leverages different trade data i.e. price and volume as well as different encoding methods i.e. gram matrix dot product and recurrence matrix. With gram matrix dot product, a time series $\{x_1, x_2, ..., x_n\}$ can be encoded into a layer C of size $n * n$ where an entry $C_{i,j}$ is defined as $x_i * x_j$. Similarly, with recurrence matrix encoding, the time series can also be transformed to a layer R of size $n * n$ with an entry $C_{i,j}$ equal to $abs(x_i - x_j)$. As follows, we list the data and encoding method for each layer in the graph encode:

- Layer 1: the gram matrix self dot product of stock daily change.
- Layer 2: the gram matrix dot product of stock weekly averaged change.
- Layer 3: the gram matrix dot product of stock daily change and daily volume.
- Layer 4: the gram matrix dot product of stock weekly change and weekly volume.
- Layer 5: the recurrence matrix of stock daily change.
- Layer 6: the gram matrix dot product of stock daily change and S&P500 daily change.
- Layer 7: the gram matrix dot product of stock weekly change and S&P500 weekly change.

Next, the time-series encoded graphs are used to train CNN models using transfer learning. Transfer learning is a machine learning technique that uses the pre-trained model from some other dataset and get retrained on the dataset we're interested in. transfer learning can be used when we don't have enough data. Here are the general steps for transfer learning. firstly, the model is trained on large-scale dataset, then we can take our custom data and initialize the network with weights trained in the first step, also, we need to change the last layer based on our prediction format, for example, with different category numbers. The last step is to train the model on the custom dataset and update the parameters. when there's no enough data, we'll freeze feature layer and take fixed-weight layer as feature extractors and update only last layer weights.

In our dataset, we have 1117 samples for each stock and there are more than 100 parameters in the model, so the general training may not have good performance, so we try the transfer learning to improve the performance. We firstly train our model on all stocks with all the layers (convolution layers and flatten layers) learning the parameters, then fix the weights of all the layers except the last one and retrain the model on each stock separately to get models for them.

Here only the last layer updates the weights. we'll provide the result and compare it with that from the general model.

Next, we visualize the result to understand the capacity and limit of the model by plotting input image and Saliency map which tells us the degree to which each pixel in the image affects the regression result for that image. To compute it, we calculate the gradient of the prediction with respect to the pixels of the image.

Finally, we propose trading strategy from the model performance and back tests the data. The performance are evaluated by the return

Our CNN contains a layer of 2D convolutional layer with ReLU as activation function, followed by a layer of 2D maxpooling, then another 2D convolutional layer followed by a layer of 2D maxpooling, plus a flatten, and 2 dense layers. There is no over-fitting in our model. We tuned the convolutional layer size and the flatten layer size, but it's difficult to find the most effective one.

2.1. Anticipated and encountered problems

The first problem we faced was how to encode trade data into a graph of multiple layers in a way that the hidden patterns of the trade data can be recognized. From literature review, we found the *GramianAngularField* module of the open source *Pyts* package [1] which performs Gramian angular fields (GAF) encoding. However, in GAF encoding, the price are first normalized during which the magnitude information is lost. To keep such information, we did not directly use the module to transform time series. Instead, we use different encoding methods (gram matrix dot product and recurrence matrix) as described previously.

For the nature of time series training, we also face the challenge of avoiding using later data to predict a earlier result. In this work, we always use a consecutive block of data to predict the next day return to avoid expose later data to the model.

Another challenge is that the number of examples are not enough to train a deep CNN model. This is a common issue for the application of deep learning in financial market. At first, we created 500 training examples using a data set of 2-year daily trading data. In comparison, a several layer CNN model contains more than 100 tunable parameters. This causes severe model overfitting. The lack of a large number of training examples greatly limit the predictability of the model. To address the issue, we first increase the look-back years from 2 years to 5 years to get 1000 training data for each stock. Secondly, we perform transfer learning leveraging the observance that different stocks though performs differently share many similar patterns which is known in finance. By doing this, the number of examples grow to 6000 for the initial run if using 6 stocks. Next, a finer tune is performed for each stock in which only a few dozen parameters are need to be refined. This approach greatly reduced over-fitting of the deep learning model.

3. Experiments and Results

3.1. Saliency maps

The major measure of success for this project is the profit and loss (P&L) by implementing the suggested strategy. Our trained model predicts the return of each stock in the next day using encoded historic trade data. Based on the return, we set a threshold and purchase stock for the specific stock if its predicted return exceeds the threshold. If purchased, the position would be sold at the time market closed the next trade day. One trade day can only purchase or sell once, but can buy after sell, assuming buy happens after sell. All trades are compound trading.

Besides trying to achieve high accuracy on the prediction, we also want to know what the models have learned and what kind of features play the key role for the prediction. In Figure 1, we show the input and saliency maps for each of the seven layers of a encoded image for *INTC* stock. A saliency map tells us the degree to which each pixel in the image affects the classification score for that image. For each pixel in the layer, this gradient tells us the amount by which the prediction will change if the pixel changes by a small amount, which shows the critical pixels for the prediction. It's expecting to get some object segmentation in the well-performed Saliency map.

We select the model and image which give the closest prediction to show the saliency map. In general saliency map, it takes the max absolute value of the gradients from different channels and show them in the image. Here we want to identify the critical pixels on each layer, so we provide the matrix image of each layer and their corresponding prediction gradients. Based on the seven groups, we find that the maps on each layer are quite similar on the shapes, however, the relationship of layer matrix and the Saliency map is difficult to determine. The weights in our model seem to be similar for each element in the same layer. We consider it's related to our small dataset and model and need further work to address the problem.

3.2. Buy and sell strategy

In order to apply our result to the real stock market, we developed the strategy that we will buy the target stock at last date's close price and sell it at the current's date close price if the analyzing result is positive. Besides, we also set different threshold to make comparisons with each other. The reason why we set this strategy is that when we collected the data and trained the model we both use close price in the market for our training dataset. The principle we used is that if the predict value is positive, it is worth buying on that date, otherwise, it is not worth buying. The larger the predict result is, the more worth it should buy on

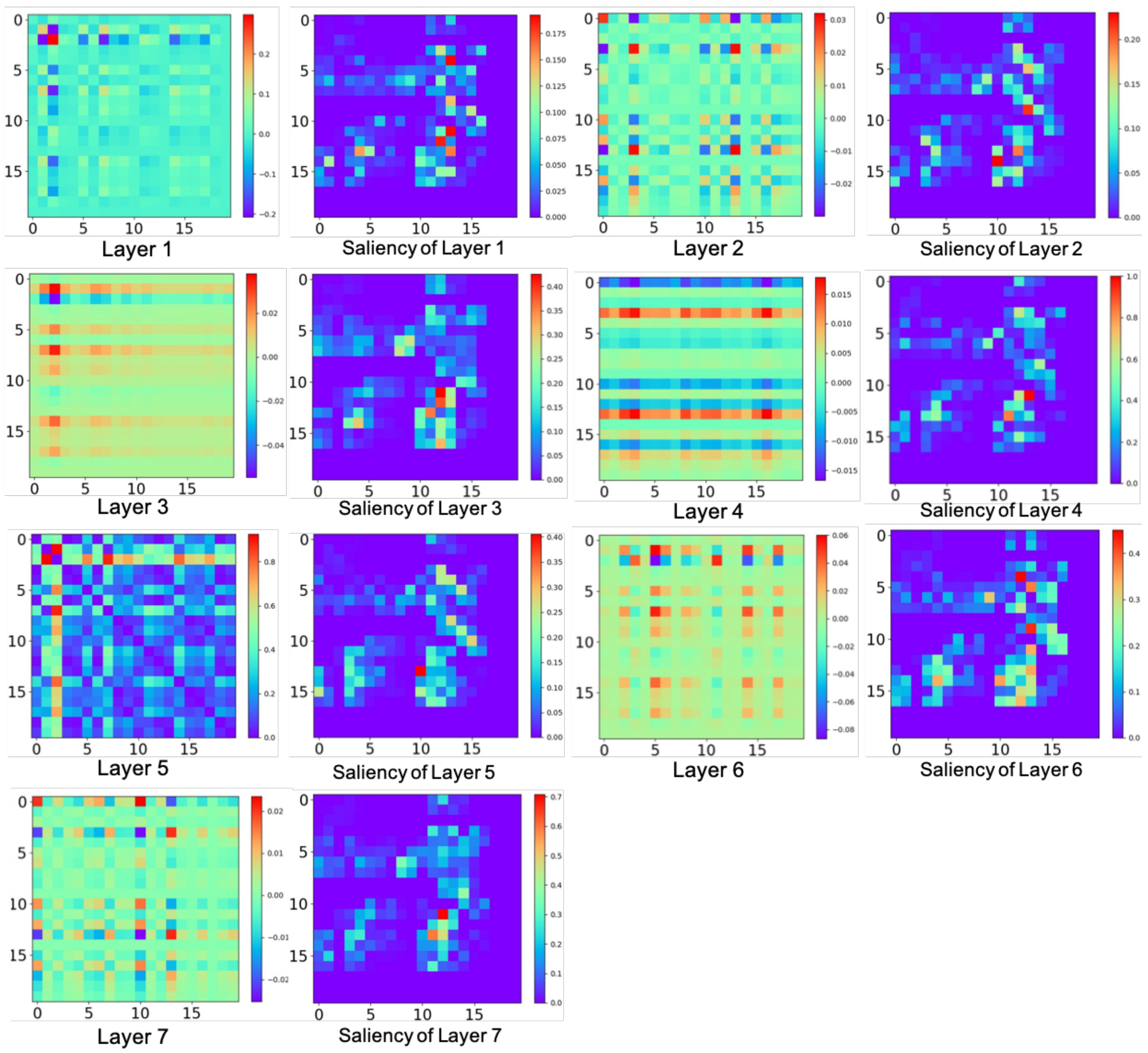

Figure 1. Input and saliency map for each channel of encoded image of *INTC*

that date. For our strategy is to buy stocks and sell stocks within one day, we planned to use different threshold to control whether this stock is worth to be operated buying and selling in this strategy.

To illustrate our model, we choose five large-cap stocks or index, i.e. Apple (AAPL), Invesco QQQ (QQQ), Amazon (AMZN), Microsoft (MSFT), and Intel (INTC). We assume the original fund should be 50,000 dollars and divide it equally among the stocks. Therefore, for each stocks, the initial fund is 10,000 dollars. We set the start date to be 2020-11-10 and the end date to be 2021-04-30 to make the prediction for 117 trade dates. Here is the result for adjusted threshold and profits of total combinations, as shown in table 1.

As the matter of result, after comparison with the result displayed on the table above, we can conclude that the predicted profit is stable when the threshold is within 0.2 %. The profit becomes unpredictable with a higher threshold because when we increase the threshold, we miss many opportunities in a bull market. When the trade times is decrease, the variance of the result become larger so that the model will predict over-fitted result. When comparing with normal training profit, we can conclude that transfer learning model can generate a better and stable result. In this particular model, we can conclude that we can have over 10 % of profile benefits if we set the threshold within 0.10 %. However, the bench strategy, which is to buy at the start date and sell at the end date, returns a total profit of 9743.92 or

Threshold	Normal training profit	Normal Profile change	Transfer learning profit	Transfer Profile change
0.00 %	2822.74	5.65%	5506.39	11.01%
0.02 %	2553.31	5.11%	5345.85	10.69%
0.05 %	1240.07	2.48%	6142.03	12.28%
0.10 %	181.25	0.36%	4629.96	9.26%
0.15 %	-1886.39	-3.77%	3635.91	7.27%
0.20 %	3104.13	6.21%	6676.81	13.35%
0.50 %	1256.95	2.51%	-588.49	-1.12%
1.00 %	1498.28	3.00%	283.36	0.57%

Table 1. Profits for the large-cap share.

19.49 %. However, it is worth noting that averagely the market return is 4-5% in average. The results indicate that the model have limitation due to the lack of input of economy and policies as well as company core values, profits and long-term goals. However, it can be potentially extended by adding more data layers or using an adjustable strategy. That's why our strategy is not good enough to fit the expectation of the share market. However, our model is still meaningful, which is because it can product stable profits for the stock when people are doing short-term investment. After adjusting for the increasing threshold, using transfer learning can produce a stable profit. This type of profit will not be influence by the market. It only depends on recent days result.

In conclusion, our training model can put out a good prediction for the stock market. What's more, after doing for transfer learning, the result will be more stable and can obtain a steady increasing for the profile value.

4. Other Sections

The code is available at https://github.com/tianshi-wang/rich_overnight.

Problem structure

There are three main parts in our problem. The first one is to make image from the time series dataset, the second one is to train CNN model on the images to make prediction, and the final part is to build up the exchange strategy and use the regression predictions to make decisions.

Learned parameters

We firstly train our model with all the layers (convolution layers and flatten layers) learning the parameters, then fix the weights of all the layers except the last one and retrain the model on each stock separately to get models for them. Here only the last layer updates the weights.

Input and output representation

The input of the neural network are 7-layer 2D matrix, and the out put of the model is the regression prediction. In our experiment, the prediction is the percentage of gain/loss of the next day. We don't use classification to predict gain or loss because we are not only interested in the trend of the market, but also want to know how much is the gain or loss, which helps us for the different exchange strategies. Mean Absolute Error (MAE) is used as loss function for our regression model. We chose MAE over mean squared error (MSE) firstly to reduce the effect of outliers. Secondly, MAE is in consistence with the investment return which is also linear with respect to stock price.

Hyperparameters and optimization

Due to the limition of input dataset, there is overfit in our model. However, we mitigate overfit by transfer learning and use models with less number of learnable parameters. The hyperparameters of the model include the layer numbers of the model and the shape of each model. We use adam optimizer.

Deep Learning framework

In this work, Tensorflow-kera was used. Our CNN contains a layer of 2D convolutional layer with ReLU as activation function, followed by a layer of 2D maxpooling, then another 2D convolutional layer followed by a layer of 2D maxpooling, plus a flatten, and 2 dense layers.

future work

Given that deep learning models usually requires large volume of data, intraday price and volume data, preferably including pre-market and after-market could be used instead of daily data. We wonder if more dataset could improve the accuracy of the model, as well as make the Saliency map more related to the features of the input.

5. Work Division

The delegation of work among team members is show in Table 2.

Student Name	Contributed Aspects	Details
Tianshi Wang	Data Creation and Implementation	Scraped the dataset for this project, transformed data into graph layers. Proof reading entire paper
Xiaoka Xiang	Implementation	Implemented CNN of encoder and tune the hyper parameters.
Tianyu Yang	Implementation and Analysis	Implemented training function, trading strategies and analysis, analyzed the result.
Feng Zhu	Implementation	Implemented Transfer Learning and tune transfer learning layer, implemented Saliency Map.

Table 2. Contributions of team members.

6. Miscellaneous Information

References

[1] Pyts Package. https://github.com/johannfaouzi/pyts, 2021. 3

[2] Yfinance Package. https://pypi.org/project/yfinance/, 2021. 2

[3] Brent Penfold. *The Universal Principles of Successful Trading: Essential Knowledge for All Traders in All Markets*. 2010. 1

[4] Andrea Corriga Alessandro Sebastian Podda Silvio Barra, Salvatore Mario Carta and Diego Reforgiato Recupero. Deep learning and time series-to-image encoding for financial forecasting. 1