CS230

Convolutional Neural Network for Predicting Company Performance Based on Historical Financial Statement Information

Justin Amezquita[*]
Department of Computer Science
Stanford University
`jaa35@stanford.edu`

Abstract

This project aims to build a deep convolutional neural network (CNN) to predict future company performance based on historical financial statement information found in the company's annual 10-K and quarterly 10-Q reports. A 1D CNN was used where the channel depth corresponded to historical time periods such that convolutions were taken through time. We trained a 5 layer CNN which contained four convolution layers and one full-connected dense layer to predict stock returns for a select group of 523 publicly traded companies between 2010 and 2018. Our model achieved a 33% accuracy, which was 7% better than the baseline linear regression model. We also show that including more historical information increased the model accuracy to 54%.

1 Introduction

Investors in public equities generally fall into two camps: fundamental investing and quantitative investing. Fundamental investors reject the market efficiency hypothesis and instead believe that a firm's performance can be determined by intuition and insight gained from careful study of the firm's historical financial statements as well as close scrutiny of and relationships with management. Quantitative investors, on the other hand, instead collect as much data as possible (including daily or by-minute stock price/return, accounting information via quarterly and annual reports, relevant news articles, satellite images, weather data, etc.) and build mathematical models that use this data to predict stock returns usually with a short investing time horizon (i.e. intraday trading and high frequency trading). However, this brute force technique lacks the intuition and insight that has proved successful in the past by fundamental investors. This project hopes to bridge the two camps by training a deep learning model using features selected with a fundamental investor mindset. Fundamental investors use historical financial statement information to determine the health of a business and make predictions on future firm performance. Our deep learning model will therefore use this same information as input data and make predictions for firm performance, i.e. stock price, one year forward from the date at which the input data was made publicly available. Features will be selected based on signals shown to be statistically significant from academic studies in areas such as price momentum (Chan et. al. 1996), value (Piotroski 2002), quality, shareholder return, risk (Beneish 1999), safety, short interest (Fabozzi), to name a few. The goal of the project is to train an AI model with data and features selected with a fundamental investing mindset that can predict stock returns and relative ranking of stock returns with a given set of companies 1 year forward. This

[*]Alternate email: justin.amezquita@gmail.com

project idea is shared with another project I worked on concurrently for my Artificial Intelligence class. Both projects have the same goal. This project focuses on building a CNN to predict stock performance, while the other project focuses on building a deep fully-connected neural network as well as LSTM for a larger dataset of companies.

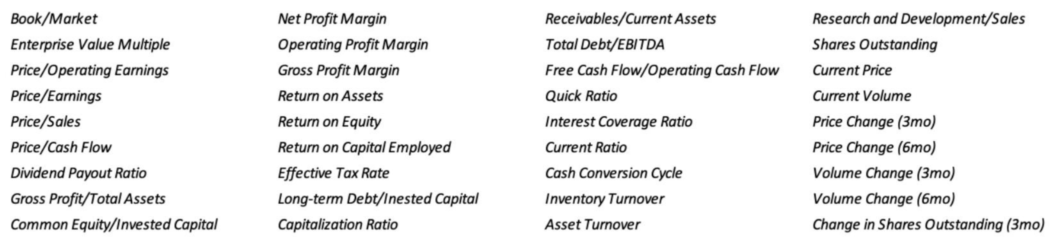
2 Related work

A lot of work has been done to predict stock prices using deep learning architectures. Most of them use recurrent neural networks such as LSTM [2] or support vector machines [3]. In either case, all of the previous published work I found uses either pure price information (daily stock prices, high/low price, bid/ask spread, volume, etc.) or alternative data such as news or twitter feed sentiment analysis [4], [5]. This project is different in that it uses a convolutional neural network, architected in such a way as to use the depth channel to perform convolutions through time. Additionally, this project uses only historical financial statement information and stock price information to predict long-term returns. This project was undertaken with a fundamental investor mind set, similar to how Fama approaches his investment research [6]. We hope that this model will be able to be used as an aid for fundamental investors throughout their investment research.

3 Dataset and Features

The dataset was taken from Wharton Research Data Services website [1], under the COMPUSTAT and CRSP databases. COMPUSTAT is a database for financial statement information relating to active and inactive companies. CRSP contains stock price information for active and inactive companies in North America. Quarterly data was collected from both databases for a subset of 523 companies listed in the S&P1500 Composite Index. Data for these companies was collected on a quarterly basis between January 2010 and January 2019. For each quarterly data between January 2010 and January 2018, historical financial information data was collected from COMPUSTAT and stock price information was collected from CRSP. Both datasets were merged based on the date and a unique identifier labeled GVKEY for each company. The subset of 523 companies represents the list of companies that were part of the S&P1500 Composite Index for every quarter between January 2010 and January 2019. This necessarily introduces selection bias. Companies may have been included in the S&P1500 Composite index in January 2010, but if they went bankrupt, delisted, or were removed from the index for any other reason before January 2019, then the company would not be included in the subset of 523 companies. Future work includes eliminating this bias by including all companies ever listed in both the COMPUSTAT and CRSP databases. Note, although this would reduce the selection bias previously described, it would not fully remove the bias for underlying biases in both databases. For example, to be included in the COMPUSTAT database, a company must have several years of publicly-released financial statement information. However, this selection bias does not significantly undermine the results of this project because the goal is the see whether historical financial information can be used to predict future performance. Therefore, we can interpret the results as whether historical financial information has predictive power given that a company will survive the time period under which it is examined.

Each data example consists of a vector of 103 input features and a label representing future company performance. The input features represent widely-used fundamental investing metrics and ratios derived from a company's financial statements (balance sheet, income statement, and statement of cash flows). See Figure 1 for a sample set of input features. The input data was normalized to be between 0 and 1.



Book/Market	*Net Profit Margin*	*Receivables/Current Assets*	*Research and Development/Sales*
Enterprise Value Multiple	*Operating Profit Margin*	*Total Debt/EBITDA*	*Shares Outstanding*
Price/Operating Earnings	*Gross Profit Margin*	*Free Cash Flow/Operating Cash Flow*	*Current Price*
Price/Earnings	*Return on Assets*	*Quick Ratio*	*Current Volume*
Price/Sales	*Return on Equity*	*Interest Coverage Ratio*	*Price Change (3mo)*
Price/Cash Flow	*Return on Capital Employed*	*Current Ratio*	*Price Change (6mo)*
Dividend Payout Ratio	*Effective Tax Rate*	*Cash Conversion Cycle*	*Volume Change (3mo)*
Gross Profit/Total Assets	*Long-term Debt/Inested Capital*	*Inventory Turnover*	*Volume Change (6mo)*
Common Equity/Invested Capital	*Capitalization Ratio*	*Asset Turnover*	*Change in Shares Outstanding (3mo)*

Figure 1: Sample set of input features

The output label was set to the quartile ranking for the one-year forward returns for the example company. For example, say the training example was for Apple in the second quarter of 2014. Then, the output label would be set by first calculated the actual returns for Apple stock between the second quarter of 2015 and 2014. These returns were then ranked across the one year forward returns for all 523 companies in the dataset between mid 2014 and 2015.

The CNN was trained on 10,041 examples. 3,347 examples were used for development and (hyper)parameter tuning, and 3,348 examples were used for testing. This represents a train/dev/test split of 60/20/20.

4 Methods

The approach uses a CNN to predict future company performance and compares the model accuracy against two baseline models: a random stock picker and a linear regression. The random stock picker selects predictions based on a normal distribution that is fit to the returns for a particular time period. For example, if predicting Apple's stock return for mid 2015, then the random stock picker randomly selects a return value from a distribution fit to the real returns observed for all 523 companies in the training set. After selecting a random return from this distribution, the output label is set to the quartile that corresponds to the selected return. The four quartiles are labeled 0 (0-25th percentile), 1 (26-50th percentile), 2 (51-75th percentile), and 3 (76th-100th percentile).

The other baseline model was a linear regression, which outputs a predicted returns quartile for a given example. Since the output is a continuous variable, the output is rounded to the nearest quartile 0-3.

The CNN has four 1-dimensional convolution layers, each of which is followed by batch normalization and a dropout layer. After the fourth convolutional layer, the neurons are flattened, and then a dense, fully-connected layer is applied, followed by batch normalization and a dropout layer. The output layer is a 4-class softmax layer. The four convolutional layers have input size of 128, 64, 32, and 16 hidden units, respectively. After the flatten layer, the dense layer has 32 hidden units.

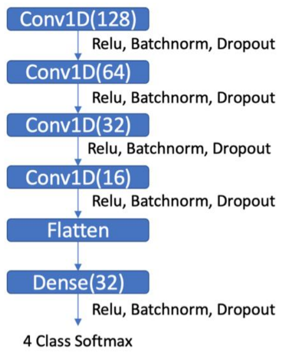


Figure 2: CNN with 4-class softmax output layer

A CNN architecture was used for the deep learning model because of its flexibility in folding in more historical time periods for training. In other words, the baseline CNN uses input data from only one fiscal quarter to predict 1 year forward returns. However, by using the depth channel in the CNN as the time dimension, we can instead choose to use any number of prior quarters as input data to predict 1 year forward returns. For example, the training set input to the baseline CNN has dimension 10041 x 1 x 103, where 10,041 is the number of training examples, 1 is the number of prior quarters to use as input, and 103 is the number of input features. However, to use a full year's worth of quarterly financials as input data, the input dimensions would be 2,510 x 4 x 103. Here, there are 2,510 training examples, where each example takes the past year (4 quarters) of historical quarterly financials as input to predict the next year's returns.

The deep learning model minimizes the categorical cross entropy loss function shown in Figure 3. There are 4 output classes which represent the quartile for the predicted returns.

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{C}(y_{ij} \cdot \log(\hat{y}_{ij}))$$

Figure 3: Categorical cross entropy loss function

Figure 4 shows the CNN loss on the training and development sets during the loss minimization iteration. Stochastic gradient descent was used to minimize the categorical cross entropy loss over 150 epochs.

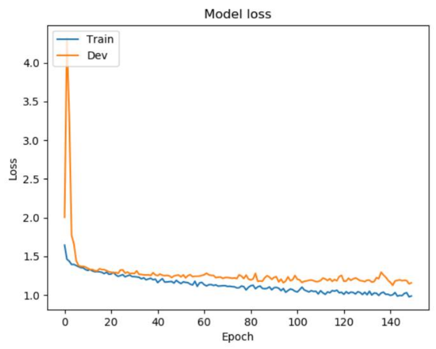


Figure 4: Training and development loss

5 Experiments/Results/Discussion

The hyperparameters that were tuned for the CNN are listed in Figure 5. The dropout rate was set to 30% because this value balanced the tradeoff between overfitting vs. underfitting to the training data. The batch size of 64 kept computing time low while still achieving high testing performance. Stochastic gradient descent was chosen for the optimizer, and the learning rate, decay rate, and momentum values were set to reduce the computing time without sacrificing the results.

Hyperparameter	Value
dropout rate	30%
batch size	64
epochs	150
optimizer	SGD
learning rate	0.10
decay rate	1.00E-06
momentum	0.90

Figure 5: CNN hyperparameter tuning

Accuracy

Model	Train	Test
Random Selector	N/A	0.23
Linear Regression	0.26	0.26
CNN	0.34	0.32
CNN 1-yr priors	0.52	0.37
CNN 2-yr priors	0.75	0.48
CNN 8-yr priors	1.00	0.54

(a) Model Accuracy

True\Predicted	Quartile 1	Quartile 2	Quartile 3	Quartile 4
Quartile 1	384	3	273	171
Quartile 2	295	4	340	167
Quartile 3	311	3	337	225
Quartile 4	314	3	241	277

(b) Confusion matrix

Figure 6: Performance Results

The most tuning and iteration occurred with respect to the neural network architecture. Different numbers of convolutional layers as well as number of hidden units in each layer were tested by first starting with one layer and then adding successive layers to improve performance on the development dataset. A dense layer was added right before the softmax output layer in order to further reduce the number of hidden units before the final output layer, with the aspiration that higher-level patterns would be constructed in this layer.

Figure 6 shows the performance results for the CNN compared to the two baseline models. Accuracy was chosen as the performance metric because the problem was set up as a 4-class classification and the goal was to accurately classify the quartile in which a company's returns would fall one year forward. Figure 5(a) shows the model accuracy. The CNN had an accuracy of 32%, which was 9% better than the random selector and 6% better than the linear regression base line model.

In addition to the standard CNN, which uses financial data from one quarter to predict returns, additional CNN were trained using one, two, and eight years of input data. As expected, the accuracy improved with longer look back periods. For the 8-yr priors CNN, the testing accuracy was 54%. However, the model clearly overfit to the data. We used a higher dropout rate to mitigate against overfitting, but it did not work as well for the 8-year model.

The classification report for the CNN is shown in Figure 7. Interestingly, the CNN classifies with a higher precision for the top quartile. This discrepancy increases as more historical data is used for prediction. This model trait suggests that healthy company fundamentals in prior quarters is a good sign that the company will continue to outperform in the future.

CNN Classification Report

Class (quartile)	Precision	Recall	F1-score
1 (low)	0.29	0.46	0.36
2	0.31	-	0.01
3	0.28	0.38	0.33
4 (high)	0.33	0.33	0.33

Figure 7: CNN classification report

6 Conclusion/Future Work

The CNN outperformed the baseline linear regression model by 6%. When more historical data was given as input to train the CNN, the accuracy improved from 32% with a one quarter look-back to 54% with an eight year look-back. In the future, experimentation with recurrent neural network architectures may help increase accuracy. Additionally, only 523 companies were used as training data over an eight year time period. Collecting data from more companies (e.g. there have been over 29,000 publicly listed companies in North America since 1950), can help improve performance.

7 Contributions

This was a solo project. I was the only contributor, although advice from the TAs, in particular Hoormazd Rezaei, was very helpful.

References

See [8] for a link to my github repository for this project.

[1] Wharton Research Data Services: https://wrds-web.wharton.upenn.edu/wrds/

[2] Chen, K., Zhou, Y., and Dai, F. (2015). A lstm-based method for stock returns prediction: A case study of china stock market. In *Big Data* (Big Data0, 2015 IEEE International Conference on, pages 2823-2824. IEEE.

[3] Kim, K. -j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307-319.

[4] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1?8, 2011.

[5] Sanjiv R Das and Mike Y Chen. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375?1388, 2007.

[6] Eugene F Fama. The behavior of stock-market prices. *The journal of Business*, 38(1):34?105, 1965.

[7]. Several essential python libraries were used in this project, namely: Tensorflow, Keras, Sklearn, Sci kit, numpy, pandas.

[8] Link to my github repository for this project: https://github.com/jaa35/cs230_project.git