# Final Project

## Improving Factor-Based Quantitative Investing By Forecasting Company Fundamentals

Team H

Jerry Fan
Rhea Ziran Gao
Casey Tirshfield
Eric Wang

January 2, 2019
IEOR 4720 Deep Learning

# Contents

**1. Inspiration** **3**

**2. Data Processing** **3**
2.1. Acquisition . . . . . 3
2.2. Cleaning . . . . . 4
2.3. Processing . . . . . 5

**3. Modelling** **6**
3.1. Linear Regression . . . . . 6
3.2. Deep Neural Networks . . . . . 6

**4. Final Model** **9**
4.1. Performance . . . . . 9
4.2. Explanation of Model Shortcomings and Possible Solution . . . . . 10
4.3. Portfolio . . . . . 11

**5. Conclusion** **12**

**A. Codebase** **12**

# 1. Inspiration

The focus of our project is to implement and expand upon the investing strategy presented in the paper, *Improving Factor-Based Quantitative Investing by Forecasting Company Fundamentals*, by John Alberg and Zachary Lipton [1]. While our methodologies diverge, the fundamental goal remains unchanged—namely, to build a successful portfolio based on a quantitative value investing-like strategy.

Value investing—as opposed to growth investing, for example—relies on the fundamental belief that over a sufficiently long time horizon, a company's market value reflects its intrinsic value, and that the best means by which to predict a company's intrinsic value is to look at its most recently announced fundamental data. The difficulty is that estimating the intrinsic value of a stock is quite challenging.

In an effort to employ technology to address this difficulty, Alberg and Lipton's paper uses deep learning not to predict the current intrinsic value of a stock, but to predict future company fundamentals. They propose that "the long-term success of an investment should depend on how well-priced the stock currently is with respect to its future fundamentals" [1, p. 2]. They have shown that a portfolio constructed based on clairvoyant knowledge of future fundamentals can drastically outperform the market. Thus, if deep learning can accurately predict future fundamentals, we can use those predictions to construct a highly profitable portfolio. This is the problem that we will expand on in our project.

# 2. Data Processing

## 2.1. Acquisition

We obtained our data exclusively through University of Pennsylvania's Wharton Research Data Services (WRDS). We pulled quarterly fundamental/market data from the CRSP/Compustat Merged Database.

Our raw data-set was built of the following variables:

| Compustat Variable Name | Compustat Variable Description |
|---|---|
| CONM | Company Name |
| TIC | Ticker Symbol |
| EXCHG | Stock Exchange Code |
| ACOQ | Current Assets – Other – Total |
| AOQ | Assets – Other – Total |
| APQ | Account Payable/Creditors – Trade |
| CHEQ | Cash and Short-Term Investments |
| COGSQ | Cost of Goods Sold |
| CSHOQ | Common Shares Outstanding |
| DLCQ | Debt in Current Liabilities |
| DLTTQ | Long-Term Debt – Total |
| INVTQ | Inventories – Total |
| LCOQ | Current Liabilities – Other – Total |
| LTQ | Liabilities – Total |

| | |
|---|---|
| NIQ | Net Income (Loss) |
| PPEGTQ | Property, Plant and Equipment – Total (Gross) – Quarterly |
| RECTQ | Receivables – Total |
| REVTQ | Revenue – Total |
| TXPQ | Income Taxes Payable |
| TXTQ | Income Taxes – Total |
| XINTQ | Interest and Related Expense – Total |
| XSGAQ | Selling, General and Administrative Expenses |
| CSHTRQ | Common Shares Traded – Quarter |
| MKVALTQ | Market Value – Total |
| PRCCQ | Price Close – Quarter |

Of the variables listed above, our sixteen features were:

| Feature | Compustat Variable Name |
|---|---|
| Revenue | REVTQ |
| Cost of Goods Sold | COGSQ |
| Selling, General and Administrative Expenses | XSGAQ |
| Earnings before interest and taxes | NIQ + TXTQ + XINTQ |
| Net Income | NIQ |
| Cash and Short-Term Investments | CHEQ |
| Receivables | RECTQ |
| Inventories | INVTQ |
| Other Current Assets | ACOQ |
| Property, Plant and Equipment | PPEGTQ |
| Other Assets | AOQ |
| Debt in Current Liabilities | DLCQ |
| Accounts Payable | APQ |
| Taxes Payable | TXPQ |
| Other Current Liabilities | LCOQ |
| Total Liabilities | LTQ |

and our target was:

| Target | Compustat Variable Name |
|---|---|
| EBIT/EV | $\text{EBIT} \cdot \text{CSHOQ}/\text{PRCCQ} + \text{DLTTQ} + \text{DLCQ} - \text{CHEQ}$ |

## 2.2. Cleaning

The paper uses all companies from the NYSE, NASDAQ, and AMEX with at least 12 consecutive months of data anytime between 1970 and 2017, with 11815 companies satisfying these criteria. However, upon closer inspection of the selected companies, we found that most of them have very incomplete data. The paper addresses this issue with forward filling. This method is appropriate when NaN values were few and scattered. However, most of the 11815 stocks have consecutive years of missing data for multiple fundamentals; for many stocks over 50% of the data is missing. It is important to avoid filling time-series data as much as possible because doing so will introduce false step-like patterns, which the deep-learning models will try to learn.

To address the poor quality of data, we introduce stricter rules for stock selection. We consider all stocks from NYSE, NASDAQ, and AMEX with continuous data from January 1990 to present that have at most 6 missing values. Only 100 satisfy these criteria. Though this is a drastic decrease,

100 stocks is enough to achieve the purpose of this project: to explore using deep-learning to predict fundamentals.

## 2.3. Processing

To prepare our data for input into training, we must first decide how we will model the stocks. More specifically, should we create a new model for each stock or one universal model for all stocks? The paper does not mention which approach they used, but we believe that only the second method is viable.

The motivation behind the first approach is that different stocks may exhibit different patterns. While this is likely true, this method is not feasible because each stock has insufficient data to train a deep-learning model. Each stock only contains roughly 30 years of quarterly data, which amounts to less than 100 training samples (70/30 train/test split). For a Basic_RNN (we will use RNN to refer to the class of neural networks and Basic_RNN to refer to the basic RNN architecture), the number of training parameters is

$$n_{param} = n^2 + kn + nm$$

Where $n$, $k$, and $m$ are the dimensions of the hidden, output, and input layers. For illustration, let us choose extremely small values for $n, k = 4$. Our input size is 17 (discussed later), so for $n, k, m = 4, 4, 17$ $n_{param} = 100$. Even for such small $n, k$, we do not have enough samples to train a Basic_RNN.

Though the second approach may blur some unique stock dependent patterns, we will have much more data to train our model. This will allow our optimizer to actually find optimal model parameters. Moreover, this will allow for larger networks, which may further increase model performance.

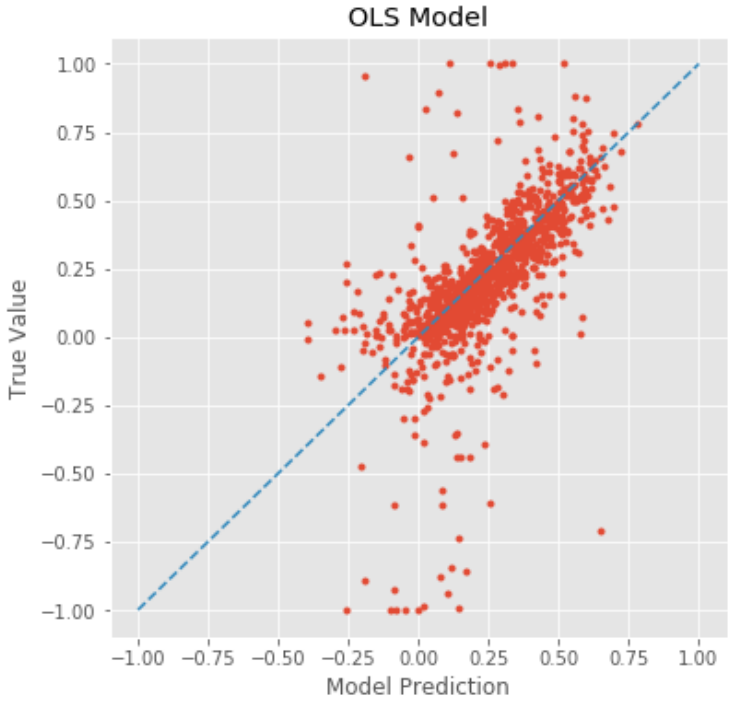
In order to train a universal model, we must carefully transform the data so that all stocks are comparable to one another. To do so we normalize each stock independently in a for loop. First, we scale the stock's fundamental features by dividing each fundamental by its largest absolute value (some of the fundamental entries are negative). In doing so, we capture the relative changes within each fundamental rather than the absolute changes. This will allow stocks with vastly different fundamental magnitudes to share a model. We also scale the response variable EBIT/EV using largest absolute value. The main advantage of this type of scaling is that it preserves percent changes for each fundamental.

After scaling the stock, we reshape its time series data to sequences which will be fed into the neural network. Our goal is to use the past 12 quarters (3 years) to predict next quarter's EBIT/EV value. At any given time, we have the present and historical 16 fundamentals and EBIT/EV values. Thus, the input dimension for our network is is $12 \times 17$ and our prediction is a single value. After performing normalization and reshaping for each stock, we combine all the sequences into one large dataset.

# 3. Modelling

## 3.1. Linear Regression

The baseline model for comparison purposes is Multiple Linear Regression. This is the benchmark that the paper uses to gauge the efficacy of neural network models. However, the paper does not describe what the inputs for the regression are. We believe there are two likely options. First, the authors might have used the current 16 fundamentals and EBIT/EV to predict the future EBIT/EV. However, this does not capture any time-dependency in the data. The second, and more likely option, is to regress solely on historical EBIT/EV; the response is a linear combination of past EBIT/EV values. More specifically $y_1 = k + \sum_{-11}^{0} \beta_i \times y_i$ where the $y$ = EBIT/EV and the subscript represents quarters from present. We use the latter regressions approach to train our model.




Since the OLS model is a linear combination of past values, it captures the general trend of the historical values. This seems to perform moderately well, as shown by the clustering of points along $y = x$. The testing MSE and MAE are respectively 0.0330 and 0.1018 and the $R^2 = 0.5464$.

## 3.2. Deep Neural Networks

We will use recurrent neural networks to try to better model the evolution of EBIT/EV. Since we will use the past 12 quarters of data to forecast one quarter into the future, our RNNs will have 12 cells. To determine the best architectures of the RNN, we will perform a grid-search over difference

possible architecture combinations. We will first test Basic_RNN/GRU/LSTM, hidden state sizes $16, 32, 64, 128$, and network depths $1, 2, 3$. By network depth $n$, we mean an $n$ stacked RNN. We will use `batch size` $= 4$ and `epochs` $= 200$, these two parameters will be further tuned after we've determined the best architectures. The table below summarizes the results of our search.

| n_Hidden | Depth | MSE | Sharpe Ratio |
|---|---|---|---|
| 16 | 1 | 0.069531 | 1.197995 |
| 32 | 1 | 0.064954 | 0.167129 |
| 64 | 1 | 0.067856 | 1.155646 |
| 128 | 1 | 0.081782 | 1.181163 |
| 16 | 2 | 0.061235 | 1.085747 |
| 32 | 2 | 0.063940 | 0.692549 |
| 64 | 2 | 0.068056 | 0.644313 |
| 128 | 2 | 0.068091 | 1.232470 |
| 16 | 3 | 0.063191 | 1.014650 |
| 32 | 3 | 0.059944 | 1.188904 |
| 64 | 3 | 0.066994 | 0.647497 |
| 128 | 3 | 0.067439 | 1.102478 |

(a) *Basic_RNN*

| n_Hidden | Depth | MSE | Sharpe Ratio |
|---|---|---|---|
| 16 | 1 | 0.058276 | 0.932585 |
| 32 | 1 | 0.057798 | 1.429266 |
| 64 | 1 | 0.062059 | 0.532301 |
| 128 | 1 | 0.071014 | 0.569725 |
| 16 | 2 | 0.062017 | 1.318058 |
| 32 | 2 | 0.065163 | 1.095102 |
| 64 | 2 | 0.060357 | 0.782527 |
| 128 | 2 | 0.063070 | 1.465788 |
| 16 | 3 | 0.062101 | 0.803267 |
| 32 | 3 | 0.063417 | 1.104489 |
| 64 | 3 | 0.065190 | 1.053502 |
| 128 | 3 | 0.061826 | 1.195497 |

(b) *GRU*

| n_Hidden | Depth | MSE | Sharpe Ratio |
|---|---|---|---|
| 16 | 1 | 0.063431 | 1.098120 |
| 32 | 1 | 0.059451 | 0.971624 |
| 64 | 1 | 0.067985 | 1.282112 |
| 128 | 1 | 0.065505 | 1.409189 |
| 16 | 2 | 0.061568 | 1.318661 |
| 32 | 2 | 0.063230 | 1.146608 |
| 64 | 2 | 0.066301 | 1.099996 |
| 128 | 2 | 0.060858 | 0.816400 |
| 16 | 3 | 0.061741 | 1.132993 |
| 32 | 3 | 0.062758 | 1.243679 |
| 64 | 3 | 0.061545 | 0.696659 |
| 128 | 3 | 0.064904 | 1.300401 |

(c) *LSTM*

Figure 1: *Performance of various RNN architectures. MSE refers to testing mean squared error and Sharpe Ratio refers to the performance of the portfolio constructed based on the model output. We will discuss this in more detail later.*

From the search, we can see that the Basic_RNN favors larger networks while GRU/LSTM favor simpler networks (in terms of hidden size and depth). This observation was made by grouping the results by depth or hidden size. We believe this to be the case because Basic_RNN is inherently simpler than GRU/LSTM. Thus, Basic_RNN requires additional hidden size/depth to capture the same patterns as GRU/LSTM.

We believe that we should typically shy away from larger networks because we do have a tremendous amount of data; we only have $\sim 3000$ training samples. When we work with large networks such as `depth` $= 3$ and `n_hidden` $= 128$, we simply do not have enough data to adequately train these models. Due to our data limitations, we should focus on smaller networks.

Though our deep learning models certainly have predictive power, none of the architectures so far have outperformed our baseline OLS. The efficacy of our OLS model inspired us to add basic attention to our networks. The rationale being that basic attention will take a weighted average of historical hidden states just like how OLS simply takes a weighted average of historical values. We hoped that adding can increase our reduce the testing MSE while simultaneously improving portfolio performance. We run the same search as before but now we add an attention layer on top of the RNN.

| n_Hidden | Depth | MSE | Sharpe Ratio |
|---|---|---|---|
| 16 | 1 | 0.065475 | 0.621899 |
| 32 | 1 | 0.064857 | 1.368926 |
| 64 | 1 | 0.065280 | 1.449125 |
| 128 | 1 | 0.065783 | 1.409822 |
| 16 | 2 | 0.060263 | 0.666955 |
| 32 | 2 | 0.062259 | 0.998888 |
| 64 | 2 | 0.061217 | 1.378524 |
| 128 | 2 | 0.065983 | 0.869195 |
| 16 | 3 | 0.066462 | 0.764989 |
| 32 | 3 | 0.062782 | 1.299194 |
| 64 | 3 | 0.060900 | 1.494304 |
| 128 | 3 | 0.064178 | 1.036547 |

(a) *Basic_RNN*

| n_Hidden | Depth | MSE | Sharpe Ratio |
|---|---|---|---|
| 16 | 1 | 0.060861 | 1.342689 |
| 32 | 1 | 0.060550 | 0.957384 |
| 64 | 1 | 0.063342 | 1.202311 |
| 128 | 1 | 0.061662 | 0.854509 |
| 16 | 2 | 0.058907 | 1.440239 |
| 32 | 2 | 0.062806 | 1.120805 |
| 64 | 2 | 0.065147 | 1.271288 |
| 128 | 2 | 0.061545 | 1.397114 |
| 16 | 3 | 0.060304 | 0.549188 |
| 32 | 3 | 0.062821 | 1.422639 |
| 64 | 3 | 0.061682 | 1.387477 |
| 128 | 3 | 0.062896 | 0.743212 |

(b) *GRU*

| n_Hidden | Depth | MSE | Sharpe Ratio |
|---|---|---|---|
| 16 | 1 | 0.058478 | 1.194309 |
| 32 | 1 | 0.063036 | 0.949507 |
| 64 | 1 | 0.063637 | 1.207028 |
| 128 | 1 | 0.061621 | 1.494700 |
| 16 | 2 | 0.059923 | 1.062025 |
| 32 | 2 | 0.064444 | 1.597992 |
| 64 | 2 | 0.060233 | 1.140768 |
| 128 | 2 | 0.059582 | 0.814396 |
| 16 | 3 | 0.062691 | 1.167111 |
| 32 | 3 | 0.060624 | 1.345288 |
| 64 | 3 | 0.068329 | 1.356667 |
| 128 | 3 | 0.063591 | 1.551531 |

(c) *LSTM*

Figure 2: *Performance of various Attention RNN architectures.*

Once again, we notice that RNN attention architectures favor larger networks while GRU/LSTM favor smaller networks. Morevoer, we notice that adding attention improved both the MSE and Sharpe Ratio in most cases. To better visualize the improvement of attention, we average the performances different architecture classes.

| Cell Type | Attention | | No Attention | |
|---|---|---|---|---|
| | MSE | Sharpe | MSE | Sharpe |
| Basic_RNN | 0.0638 | 1.1132 | 0.0669 | 0.9425 |
| GRU | 0.0619 | 1.1407 | 0.0627 | 1.0235 |
| LSTM | 0.0622 | 1.2401 | 0.0633 | 1.1264 |

Based on these observations, we selected several candidate architectures for further tuning. Since, GRU/LSTM outperform Basic_RNN significantly, we will focus on GRU/LSTM models. For these models, we determined that smaller networks are more feasible given our data and more favorable in terms of performance. Thus, we will only further tune GRU/LSTM `depth` $= 1$ `n_hidden` $= 16$, `depth` $= 1$ `n_hidden` $= 32$, and `depth` $= 2$ `n_hidden` $= 16$. For this tuning we will adjust `epochs` $= 100, 150, 200, 250, 300$, `batch size` $= 4, 8, 16$ and `learning rate` $= 0.0001, 0.001, 0.01, 0.1$. The top 5 models with the lowest MSE are:

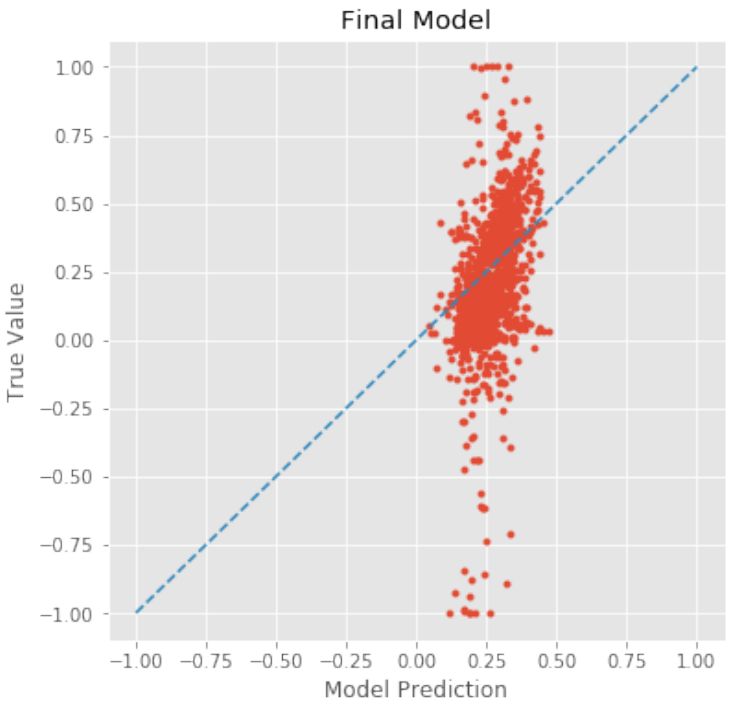
| Epochs | n_Hidden | Depth | Batch Size | Learning Rate | Attention | MSE | Sharpe Ratio | Cell |
|---|---|---|---|---|---|---|---|---|
| 200 | 16 | 1 | 16 | 0.0001 | True | 0.046042 | 1.242165 | LSTM |
| 200 | 16 | 2 | 16 | 0.0001 | False | 0.047333 | 1.180516 | LSTM |
| 250 | 16 | 2 | 16 | 0.0001 | False | 0.047845 | 1.338276 | LSTM |
| 300 | 16 | 1 | 16 | 0.0001 | True | 0.047998 | 1.300277 | LSTM |
| 250 | 16 | 2 | 16 | 0.0001 | False | 0.048808 | 1.236754 | GRU |

Interestingly, all the best performing models all use `batch size` $= 16$ and `learning rate` $= 0.0001$. When attention is used a single layer RNN is best. When there is no attention, a double layer RNN is best. Since top model has a significantly smaller MSE than the other models, it is the final product of our modelling. Thus, our final model is Attention LSTM with `depth` $= 1$ and `n_hidden` $= 16$, trained with `epoch` $= 200$, `batch size` $= 16$ and `learning rate` $= 0.0001$.

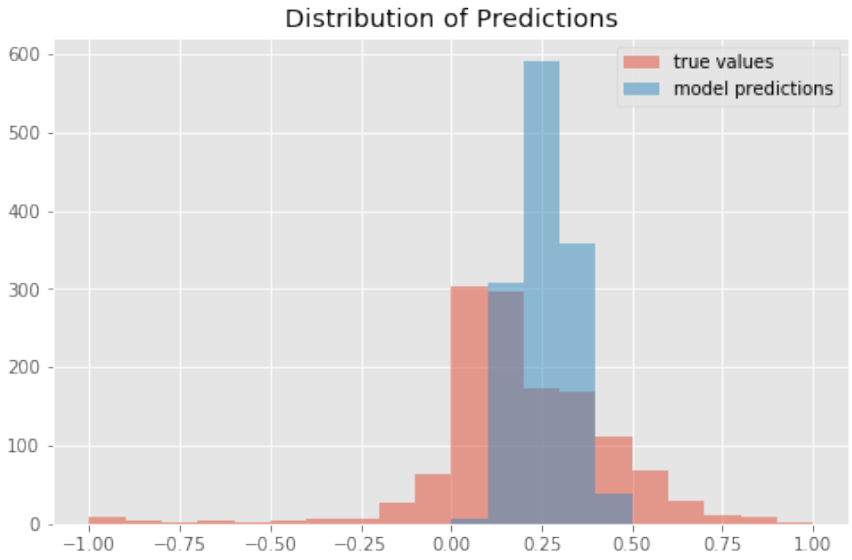
# 4. Final Model

## 4.1. Performance

To better understand what our model actually does, we compared its predictions to the testing set's true values.




It's clear that our model's predictions are much less variable than the true values. Our predictions have `mean` = 0.258 and `std` = 0.072, whereas the true values have `mean` = 0.199 and `std` = 0.258. The discrepancy between the model and the true values can be visualized clearly using the histogram below.




The histogram confirms that our model is not only biased, but does not adequately capture the variability in the data, just as. Finally, let us look at some stocks individually to further understand how our model behaves.

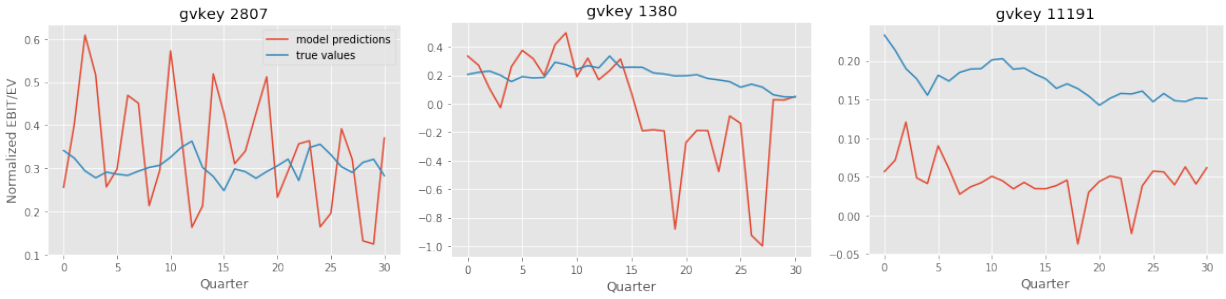



Figure 3: *Model predictions for three stocks. gvkey is an unique stock identifier.*

In the first subplot, we can clearly see how our model does capture the variability in the data. In the second subplot, we see that our model recognizes the drop in EBIT/EV after quarter 15 by trending downwards. However, it is once again unable to capture the severity of the drop due to its invariability. In the last subplot, we see that our model completely misses the mark. This is due to the fact that our model almost always predicts values near the mean of 0.258. Thus, when the true EBIT/EV values are far from 0.258, our model fails.

## 4.2. Explanation of Model Shortcomings and Possible Solution

As mentioned previously, our model was not only biased but also did not adequately capture the variability of EBIT/EV. In fact, a simple OLS using the historical EBIT/EV beat our best model. Theoretically, our attention LSTM should be at least as good as OLS; its architecture can replicate OLS, but can also capture more complex dependencies.

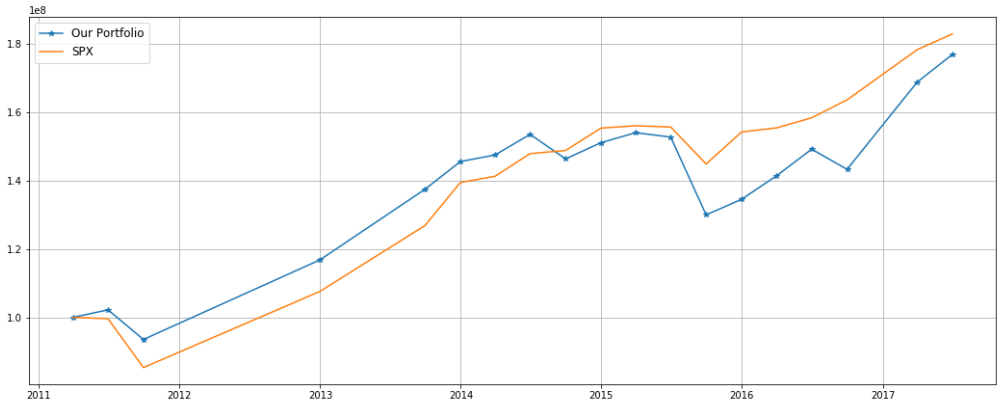
We believe that the reason for our model's shortcomings is that there are too many inputs and not enough data. Many of the input fundamentals are likely redundant. When these features are included in the model, they increase the number of parameters that need to be trained without adding much to the model. If we had infinite data, our optimizer should eventually be able to determine that these extraneous features are useful and assign them very little weights. However, we only have around 3000 training samples, which is likely not enough for the model to learn what is important and distinguish all the extraneous features.

Since, the data source is rather low quality, it will be difficult to obtain a tremendous amount of data. Thus, we believe that feature selection is a more practical remedy this problem. Perhaps, one could start with a simple attention/no attention LSTM using only historical EBIT/EV as inputs (input vector length 1). Then one can add an additional fundamental (input vector length 2) to see if the model performance increase. After determining the best second fundamental, one can try to add a third (input vector length 3). One can repeat this until model performance stops improving. This method is exactly like forward-selection for multiple regression. We hope that this will reduce the input vector dimension to only include salient features.

Of course, we should still try to obtain more data. Our standards for data were quite high. By reducing the requirement, one could likely double or even triple the number of training/testing samples. However, one must be careful when relaxing data requirements as it could lead to bad data, which could undermine the benefits of more data.

## 4.3. Portfolio

The trading strategy proposed by Alberg and Lipton is to hold the stocks with the highest EBIT/EV. Our model predicted normalized EBIT/EV, which we can easily transform back to EBIT/EV. Since we used far fewer stocks than Alberg and Lipton, we will divide our portfolio resources into only 100 stocks. The performance of our portfolio in the testing period is plotted below.




As we can see, our portfolio roughly follows the S&P500. The purpose of aim of our project is to use deep learning to predict time-series of fundamental data, not necessarily constructing the most

profitable portfolio. Thus, we will not worry too much about the performance of the portfolio. We have implemented the algorithm so that an user can satisfy his/her curiosity and check the portfolio's performance.

## 5. Conclusion

Inspired by Alberg and Lipton, our group attempted to utilize deep learning to forecast future EBIT/EV ratio. Because of our strict data requirements, our data-set was much smaller than that used by Alberg and Lipton. Morevoer, our data processing differed significantly from Alberg and Lipton because our goal was to train an universal model that was independent of the individual stock. We tested a variety of architectures and found that smaller networks performed better. Our final model is Attention LSTM with `depth` = 1, `n_hidden` = 16, trained with `epoch` = 200 `batch size` = 16 `learning rate` = 0.0001. However, even our best model could not outperform OLS with only historical EBIT/EV values. We believe that the source of our model's shortcoming is the redundancy of many input features, which is particularly problematic because our data-set is rather limited. The next step in this endeavor would be to perform feature selection; forward-selection starting with only historical EBIT/EV as inputs would be a good approach. Finally, we have implemented the trading algorithm proposed by Alberg and Lipton which allows us to use our forecast of EBIT/EV to construct and rebalance a portfolio.

## A. Codebase

Our project's codebase can be found in the following GitHub Repository.

# References

[1] John Alberg and Zachary C. Lipton. Improving Factor-Based Quantitative Investing by Forecasting Company Fundamentals. *arXiv e-prints*, November 2017. arXiv:1711.04837.