Empirical Asset Pricing via Deep Learning Algorithms

Kaan Yolsever
20569159

Abstract: I use a Long Short-Term Memory Network (LSTM), a deep learning algorithm, which is designed for time series models to predict the stock returns 1-month in advance. My data set consists of 116 different fundamental and technical variables and almost 2 million observations spanning $\sim$ 30 years and $\sim$ 17000 companies. Preliminary results show that LSTM outperforms a vanilla neural network in terms of mean squared error of predicted one-month-ahead stock return predictions.

A thesis presented for the degree of
Bachelor of International Economics

Vancouver School of Economics
University of British Columbia
Canada
2019-03-02

Contents

List of Figures

1 Introduction

1.1 Motivation and Contributions

Banks, and investment firms have been using machine learning algorithms for a long time. However, academic research in the area is nascent. This paper contributes to the growing number of papers in the area by using a Long Short Term Memory(LSTM) network. LSTM network is of special importance to stock prediction because they are capable of learning long-term dependencies from data. They are explicitly designed to avoid the long-term dependency problem. Hochreiter and Schmidhuber (1997). LSTM's are explained further in the relevant section 3.1.2. The preliminary results show that LSTM tends to outperform vanilla neural networks and other traditional financial cross-sectional trading strategies.

1.2 Literature Review

Research in the area is burgeoning. Shihao et al. (2018) in their paper show that the deep learning algorithms can reach positive R^2 in predicting the stock prices, which outperform traditional linear models. Their work shows how machine learning algorithms outperform these methods both in terms of returns and the risk involved yielding in outstanding annualized Sharpe ratios. Also, they provide a theoretical and mathematical framework for using machine learning algorithms in a financial context. This paper borrows heavily from their work such as the algorithms used and the data preprocessing techniques because of computational difficulty of tuning hyper-parameters. However, this paper makes use of LSTM networks which are developed in the machine learning literature in 1997 by Hochreiter and Schmidhuber (1997). Shihao et al. (2018) suggests that complex methods outperform more naive methods. As a vanilla neural network with 3 hidden layers tops the list for the prediction accuracy among machine learning algorithms achieving a worst annual return of 13%, an average return of 33% over 60 years. So, adding further complexity to the methods could produce even better results. They conclude that the baseline patterns that OLS fares poorly, regularized linear models are an improvement, and non-linear models dominate carries over into sub-samples. Tree methods and neural nets are especially successful among large stocks, with R^2_{oos}'s ranging from 0.53% to 0.72%. This dichotomy provides reassurance that machine learning is not merely picking up small scale inefficiencies driven by illiquidity.

Guanhao et al. (2019) makes use of machine learning methods in assessing the predictive power of hundreds of metrics. Also, they provide a theoretical framework for selecting and evaluating which variables to use for better prediction. They show that traditional feature selection methods such as LASSO does not yield good results when the data is highly dimensional such as 100 variables and more. Traditional methods such as Fama and French (2015) form portfolios in medium-cap firms in the U.S. using the combinations 5 factors, namely size, book-to-market equity, investment factors (aggressive or conservative), market risk, and operating profitability. They observe that by using the recent 2, the observation of anomalous returns decreases meaning they can explain a bigger portion of the diversified portfolio returns compared to their 3-factor model which explained a high portion of the diversified portfolio returns. However, the Sharpe ratios they obtain do not compare with Shihao et al. (2018).

Keloharju et al. (2016) observe that a strategy which uses same-calendar-month returns over a long-time period was able to obtain a 13% return per year which is persistent across different stock markets, commodities and at the daily day frequency. They show that a meta-strategy that trades on 15 anomalies based on their same-calendar-month premiums make 1.88% per month but if the trade strategy instead uses anomalies based on other-calendar-month premiums, the strategy loses money and they also prove that the return seasonalities are economically significant. Thus, it is reasonable to expect that the results obtained by Shihao et al. (2018) are downplayed since they cannot account for such seasonal effects. This paper will take this into account when training the neural networks.

C.Cavalcante et al. (2016) showcases the state-of-the-art methods for financial prediction using machine learning techniques. They separate the process into 6 major steps: Data preparation, algorithm definition, training, forecasting evaluation, trading strategies, and money evaluation. They share the most relevant, and important papers about each step and what lacks in them. They also make a review of recent papers on "(i) the main goal of the primary study, (ii) the main application of the proposed intelligent, (iii) the input variables analyzed, (iv) the intelligent

techniques used to solve the problem and (v) whether the work proposed a trading system". The modern techniques for each stage selected from this survey will be used in my paper.

Pesaran and Timmermann (1995) shows that the predictability of the stock prices increases with market volatility. They observe that in 1960s when the markets were calm, the stock prices were almost non-predictable with the metrics available. Then, in 1970s the predictability rose to a level which can be exploited by the investors net of transaction costs. So, the literature review suggests that there is a growing interest in the area. Also, applying the modern techniques in computer science to a financial context will allow me to introduce new techniques for portfolio formation and to analyze the predictive power of different variables.

2 Data

2.1 Data set

The data set in this paper replicates Shihao et al. (2018)'s paper. The big part of the data set used in this paper is fetched using Green et al. (2012)'s SAS code which is publicly available on his personal website. The code combines COMPUSTAT, CRSP, and IBES through WRDS. The data set consists of 122 different fundamental and technical variables and almost 2 million observations spanning 1978 to 2017 and $\sim$ 17000 companies. The second data set is the 8 macroeconomic variables in Goyal and Welch (2004). Contrary to Shihao et al. (2018), I do not interact these 8 variables with the variables in Green et al. (2012). I also incorporate the investor sentiment data set by Baker and Wurgler (2006). The data set has 14 variables. This adds up to 94+8+14 = 116 variables in total as explained in the next subsection.

We only retain the stocks with the highest number of observations which is 474 months. So, at the end we end up up with 374 stocks. So, the data set is big enough for the training of the machine learning algorithms used in this paper.

2.2 Data Preprocessing

In Green's data set, after eliminating the redundant variables following Shihao et al. (2018), we end up with a time-series data set which includes 94 characteristics (61 of which are updated annually, 13 updated quarterly, and 20 updated monthly). Most of the characteristics are fundamental. The full list from Green et al. (2012) can be seen in 10. I drop the variables that are constant across time, or are not particularly useful for prediction as done in Shihao et al. (2018). The variables that are missing more than 80% of the values are dropped. The rest of the missing values in the data set are filled with the mean of the closest two observations which have data for the same stock.

Since the training process of neural networks is a delicate job, I scale the variables to the range of (-1,1), following the deep learning literature as in Goodfellow et al. (2016).

3 Empirical Methods

Following the machine learning literature as in Goodfellow et al. (2016): the data set is split into 3 parts: training, validation, and test sets. Given the computational difficulty, this paper uses 374 companies with the most number of observations to train the models on. We use 60% of the data for each stock as the training set, and 35% of the data for validation and 5% as test sets. This provides us with a lot of data that can be used to optimize the weights. The neural networks are trained on the training set to minimize mean squared predictions error (MSE). So, the loss function looks like:

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i) \tag{1}$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - max(0, w.X_i + b))^2 \tag{2}$$

Where the second line follows from substituting in the 'relu' activation. And w matrix is the matrix of weights in the neural network. The best approximation model is found using

gradient descent algorithm. Specifically, since each layer's output is the input of the next layer, the gradient is dependent on the weights of the previous layers at the later layers. Following Goodfellow et al. (2016), the gradients are calculated using back-propagation. It is illustrated below where $f(W^1, W^2, W^3, v)$ is the cost function. W_1 is the weight matrix of the first layer, W_2 is the weight matrix of the second layer, and W_3 is the third layer's.

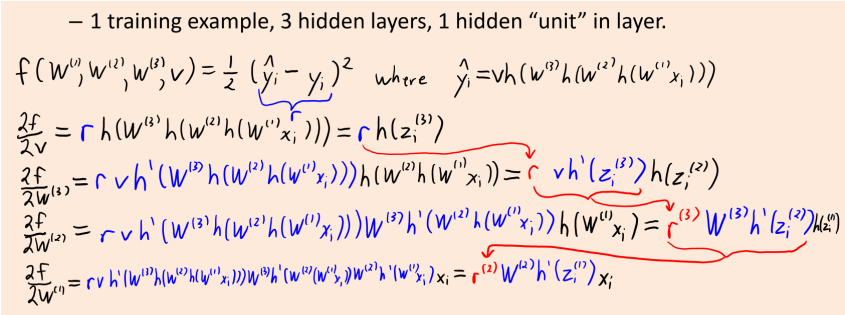


Figure 1: Back-propagation

However, in most cases, this results in over-fitting because of how complex neural networks can get. It is not rare that neural networks achieve close to 100% accuracy in the training set. Over-fitting is defined in statistics as "production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". To prevent over-fitting, we use a validation set. The validation set is used to check that the neural network is indeed learning the predictive parameter weights. While training, we concurrently evaluate the optimized model on the validation set, and we stop when the error in the validation set does not go down after 5 epochs[1]. This process of stopping the training before the maximum number of epochs allowed is called early-stopping. This is explained further in section 12. Finally, once the algorithm stops optimizing the weights, the predictions are made on the test set using the available data set.

As in Shihao et al. (2018), this paper describes an asset's excess returns as an additive prediction error model 3:

$$r_{i,t+1} = E_t(r_{i,t+1}) + \epsilon_{i,t+1} \tag{3}$$

where

$$E_t(r_{i,t+1}) = g(z_{i,t}) \tag{4}$$

and evaluates the prediction accuracy with the R^2 specified in that paper 5:

$$R^2_{oos} = 1 - \frac{\sum_{i,t\in\tau_3}(r_{i,t+1} - \hat{r}_{i,t+1})^2}{\sum_{i,t\in\tau_3} r^2_{i,t+1}} \tag{5}$$

The stocks are indexed by i and the time is indexed by t. g is the algorithm used when predicting the returns. And $z_{i,t}$ is the stock data available at month t, for stock i.

However, given the noisy nature of the stock market returns, reducing MSE is a difficult job. Also, LSTM and NN can learn the wrong weights since the stocks could behave differently from one time period to another, see 8. To handle these , I use ADAM optimizer to reduce the training MSE 11, and use early stopping 12 and dropout 13 to reduce over-fitting. See the appendix for further explanation about the algorithms.

Even though it looks like the models are only learning on a stock-by-stock basis, the data set incorporates many elements related to macro factors such as industry momentum, recession, and features taken from Goyal and Welch (2004) and Baker and Wurgler (2006). So, the learning is both on the individual stock level, and cross-sectional.

[1]Epoch is when an entire data-set is passed forward and backward through the neural network only once.

3.1 Vanilla Neural Network

The explanation in this section summarizes Shihao et al. (2018)'s discussion on neural networks. Neural networks are the most powerful modeling device in machine learning. They have theoretical underpinnings as universal approximators for any smooth predictive association Hornik et al. (1989). They are currently the preferred approach for complex machine learning problems such as computer vision, natural language processing, and automated game-playing. At the same time, their complexity ranks neural networks among the least transparent, least interpretable, and most highly parameterized machine learning tools. The model used in this research paper is traditional feed-forward networks. These consist of an input layer of raw predictors, one or more hidden layers that interact and non-linearly transform the predictors, and an output layer that aggregates hidden layers into an ultimate outcome prediction.

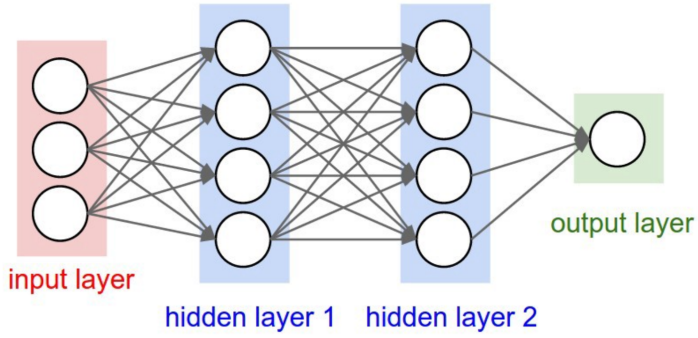


Figure 2: A fully-connected neural network with 8 hidden neurons in 2 hidden layers

The number of units in the input layer is equal to the dimension of the predictors, in our case it is 116.

A neural network with one hidden layer, one hidden unit, and a linear activation function is a simple linear regression. It simply finds the weights that optimize the linear predictions.

Each neuron in the structure above, does the following:

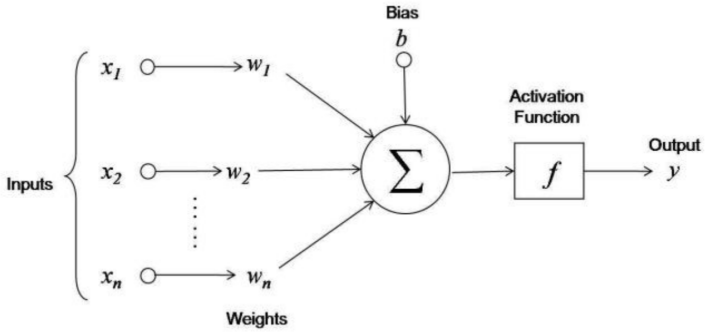


Figure 3: Neuron operation

First, it adds up the value of every neurons from the previous column it is connected to. On the Figure 1, there are 3 inputs coming to the neuron, so 3 neurons of the previous column are connected to our neuron.

This value is multiplied, before being added, by another variable called "weight" (w1, w2, w3) which determines the connection between the two neurons. Each connection of neurons has its own weight, and those are the only values that will be modified during the learning process.

Moreover, a bias value is added to the total value calculated which is the neural network equivalent of an intercept in linear regression. After all those summations, the neuron finally applies a function called "activation function" to the obtained value.

The activation function accounts for possible non-linear relationships in the data. There are many possible candidates such as: sigmoid, relu, tanh etc. In this paper I use 'relu' as in Shihao et al. (2018). A frequent problem with training neural networks is the vanishing gradient problem. This problem occurs because the gradient is calculated using the chain rule. Since the gradients of sigmoid, and tanh are bounded in (0,1), the gradient sometimes gets arbitrarily small, and the weights stop being updated. Since we scale the data set in the preprocessing state, the gradient values are very tiny and in a lot of the cases, they converge to zero, making the learning impossible. Relu prevents the vanishing gradient problem.

$$ReLU(x) = max(1, x)$$

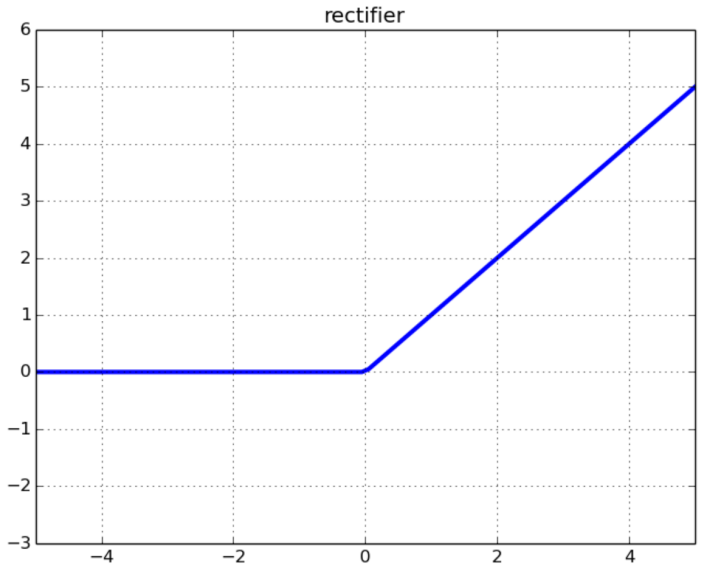


Figure 4: ReLU activation

As we add layers, the hidden layers act as change of basis for the number of variables. This paper uses a neural network with 3 hidden layers. In the first layer there are 32 neurons, second layer has 16, and the third layer has 8 neurons. The network is fully connected and after each layer we apply a 'relu' function.

There are many choices to make when structuring a neural network, including the number of hidden layers, the number of neurons in each layer, which units are connected, regularization strength, kernel initialization and etc.

3.2 Recurrent Neural Networks and Long Short Term Memory

This section summarizes the discussion by Olah (2010). Recurrent neural networks are networks with loops in them, allowing information to persist. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data. In the last few years, there have been incredible success applying RNN's to a variety of problems: speech recognition, language modeling, translation, image captioning.Essential to these successes is the use of "LSTMs," a very special kind of recurrent neural network which works, for many tasks, much much better than the standard version. Almost all exciting results based on recurrent neural networks are achieved with LSTM's. Long Short Term Memory networks usually just called "LSTM's" are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber (1997), and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems, and are now widely used.

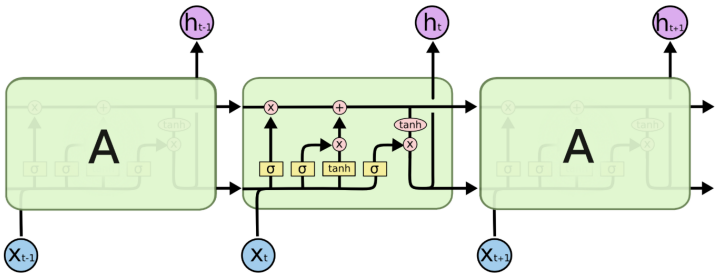


Figure 5: An example of a LSTM network

The first step in LSTM is to decide what information is going to be thrown away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents "completely keep this" while a 0 represents "completely get rid of this." In the next step we the algorithm decides what new information is going to be stored in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values are updated. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. In the next step, these two are combined to create an update to the state. This is done by multiplying the old state by f_t, forgetting the things that were decided to be forgotten earlier. Then the algorithm adds $i_t \times \tilde{C}t$. These are the new candidate values, scaled by how much we decided to update each state value. Finally, the output will be based on the cell state, but will be a filtered version. First, sigmoid layer decides what parts of the cell state is going to be outputted. Then, cell state goes through tanh (to push the values to be between -1 and 1) and is multiplied by the output of the sigmoid gate, so the algorithm only outputs the parts we decided to.

4 Results

From the preliminary results, the LSTM network tend to outperform the neural network in terms of stability and prediction accuracy. As Shihao et al. (2018) has shown, the vanilla neural networks outperform all other traditional machine learning algorithms and have positive R^2 for out of sample predictions. So, LSTM's are a promising source for stock return prediction. Two of the most successful prediction estimations are below. This is not the case in general as explained later in this chapter.

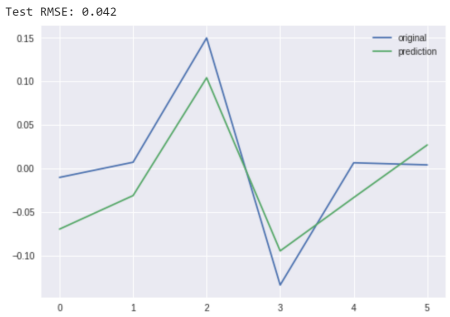


Figure 6: LSTM return predictions for 6 months

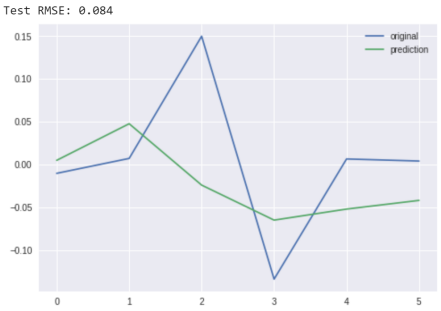


Figure 7: Vanilla Neural Network return predictions for 6 months

The financial data is notoriously noisy. So,often times a stock behaves very differently in from a time period to another, making the learning (or optimizing the weights) difficult as can be seen below where the error starts oscillating.

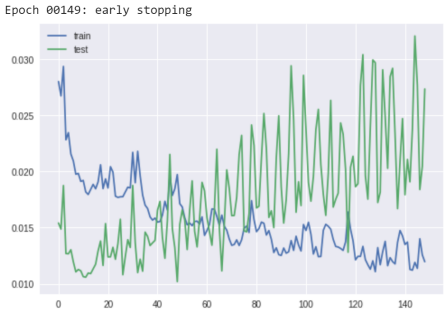


Figure 8: Training mean squared error(MSE) versus test MSE over epochs

However, luckily this is not the case for most stocks, the test error and the training error tend to move together. So, generally the learning happens.

One promising result is that the model is indeed learning which variables have predictive power. I use eli5 library from python to analyze the importance of each of 117 variables that are present in the research paper. Since neural networks are difficult to interpret, the way this paper overcomes this issue is following Shihao et al. (2018) and Breiman (2001). The method recommended in Breiman (2001) is to replace the entries of one feature with noise, then see how the predictive power of the model changes. This yields in the table below:

Weight	Feature
0.0010 ± 0.0034	mom36m
0.0010 ± 0.0037	mom6m
0.0008 ± 0.0005	salerec
0.0005 ± 0.0007	pps
0.0004 ± 0.0009	mve_m
0.0004 ± 0.0007	mom12m
0.0002 ± 0.0004	quick
0.0002 ± 0.0003	roeq
0.0001 ± 0.0002	depr
0.0001 ± 0.0001	Returns
0.0001 ± 0.0001	prccq
0.0001 ± 0.0002	aeavol
0.0001 ± 0.0006	std_turn
0.0001 ± 0.0002	dy
0.0001 ± 0.0001	pchsale_pchxsga
0.0001 ± 0.0000	cinvest
0.0000 ± 0.0004	roavol
0.0000 ± 0.0001	pchcapx_ia
0.0000 ± 0.0001	invest
0.0000 ± 0.0001	cash
… 107 more …	

Figure 9: The feature importance determined by the algorithm proposed in Breiman (2001)

To create the table I train a neural network on the portion of the data set with stocks with the most number of observations per stock. The algorithm cross-validates the results on 5 sets randomly created from the big data set. The first number is the importance associated with the variable. It is a coefficient of significance since there is no one number that explains how the deep learning model uses this variable by itself.

The results are in parallel with the financial wisdom. First 5 are,respectively, 36 month-momentum, 6-month-momentum,sales-to-receivables, financial statement score, size. Momentum as pointed out in Carhart (1997) has significant predictive power. Comparing these to Shihao et al. (2018), LSTM seems to be making similar decisions with a simple neural network.

5 Conclusion

The results are premature. However, LSTM looks like a better alternative than a Vanilla Neural Network given the results. In this paper, I did not use a stateful LSTM network but the simple LSTM as specified in the Keras library of Python language. LSTM with 32-16-8 hidden neurons in 3 hidden layers was able to obtain comparable results to a simple neural network for almost every stock. Also, the results support that LSTM has a self-regularizing effect. LSTM's performance with or without regularization was similar. However, a simple neural network became more predictive once I introduced L1-regularization.

- Stateless LSTM predictions generally align with those of simple NN. This means that the additional predictive power coming from using a more complicated model is not enough to justify the significantly higher computational cost.
- I observed that the interactions terms which add 800 features to the data set as in Shihao et al. (2018) do not contribute significantly to the results.
- The networks are able to learn which of the variables are most important for prediction in parallel with what financial literature would predict.
- I had to introduce $L1$-regularization for a simple neural network which was not necessary for LSTM. This suggests that LSTM has a self-regularizing effect.

- I was not able to obtain the results obtained by Shihao et al. (2018). I believe this could be because:
  - They have a bigger data set starting from 1940's.
  - They have more computational power so they can train the algorithms multiple times with randomly initialized neuron kernels
  - They could be able to optimize the weights better because of the computational power

A References

Works Cited

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". *Neural Networks* 2.5. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8 (1989): 359–366. Web. <http://www.sciencedirect.com/science/article/pii/0893608089900208>.

Pesaran, M. Hashem and Allan Timmermann. "Predictability of Stock Returns: Robustness and Economic Significance". *The Journal of Finance, vol. 50, no. 4* (1995). Print.

Carhart, Mark M. "On Persistence in Mutual Fund Performance". *The Journal of Finance* 52.1. ISSN: 00221082, 15406261 (1997): 57–82. Web. <http://www.jstor.org/stable/2329556>.

Hochreiter, Sepp and Jürgen Schmidhuber. "Long Short-term Memory". *Neural computation* 9. DOI: 10.1162/neco.1997.9.8.1735 (Dec. 1997): 1735–80. Print.

Breiman, Leo. "Random Forests". *Machine Learning* 45.1. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324 (Oct. 2001): 5–32. Web. <https://doi.org/10.1023/A:1010933404324>.

Goyal, Amit and Ivo Welch. "A Comprehensive Look at the Empirical Performance of Equity Premium Prediction". Working Paper Series 10483. DOI: 10.3386/w10483 (May 2004). Web. <http://www.nber.org/papers/w10483>.

Baker, Malcolm and JEFFREY Wurgler. "Investor Sentiment and the Cross-Section of Stock Returns". *The Journal of Finance* 61.4. DOI: 10.1111/j.1540-6261.2006.00885.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.2006.00885.x (2006): 1645–1680. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.2006.00885.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.2006.00885.x. Web. <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.2006.00885.x>.

Olah, Chris. "Understanding LSTM Networks". 2010. Web. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

Ribeiro, Bernardete, et al. "Financial Distress Model Prediction using SVM". *World Conference on Computational Information, Barcelona* (2010). Print.

Green, Jeremiah, John R. M. Hand, and Frank Zhang. "The Supraview of Return Predictive Signals". *Review of Accounting Studies* 18. DOI: 10.2139/ssrn.2062464 (May 2012). Print.

Kingma, Diederik and Jimmy Ba. "Adam: A Method for Stochastic Optimization". *International Conference on Learning Representations* (Dec. 2014). Print.

Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". *Journal of Machine Learning Research* 15 (2014): 1929–1958. Web. <http://jmlr.org/papers/v15/srivastava14a.html>.

Fama, Eugene F. and Kenneth R. French. "Dissecting Anomalies with a Five-Factor Model". *The Review of Financial Studies* 29.1. ISSN: 0893-9454. DOI: 10.1093/rfs/hhv043. eprint: http://oup.prod.sis.lan/rfs/article-pdf/29/1/69/24450717/hhv043.pdf (Aug. 2015): 69–103. eprint: http://oup.prod.sis.lan/rfs/article-pdf/29/1/69/24450717/hhv043.pdf. eprint: http://oup.prod.sis.lan/rfs/article-pdf/29/1/69/24450717/hhv043.pdf. Web. <https://dx.doi.org/10.1093/rfs/hhv043>.

C.Cavalcante, Rodolfo, et al. "Computational Intelligence and Financial Markets: A Survey and Future Directions". *Expert Systems with Applications 55 194-211* (2016). Print.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016. Print.

Keloharju, Matti, Juhani Linnainmaa, and Peter Nyberg. "Return Seasonalities". *Journal of Finance Vol. LXXI, No 4* (2016). Print.

Shihao, Gu, Kelly Bryan T., and Dacheng Xiu. "Empirical Asset Pricing via Machine Learning". *Chicago Booth Research Paper No. 18-04; 31st Australasian Finance and Banking Conference* (2018). Print.

Guanhao, Feng, Giglio Stefano, and Dacheng Xiu. "Taming the Factor Zoo". *Fama-Miller Working Paper; Chicago Booth Research Paper No. 17-04* (2019). Print.

B Additional Tables and Figures

B.1 Variables List

The data is collected by Green et al. (2012). And the explanatory tables below are taken from Shihao et al. (2018)

No.	Acronym	Firm characteristic	Paper's author(s)	Year, Journal	Data Source	Frequency
1	absacc	Absolute accruals	Bandyopadhyay, Huang & Wirjanto	2010, WP	Compustat	Annual
2	acc	Working capital accruals	Sloan	1996, TAR	Compustat	Annual
3	aeavol	Abnormal earnings announcement volume	Lerman, Livnat & Mendenhall	2007, WP	Compustat+CRSP	Quarterly
4	age	# years since first Compustat coverage	Jiang, Lee & Zhang	2005, RAS	Compustat	Annual
5	agr	Asset growth	Cooper, Gulen & Schill	2008, JF	Compustat	Annual
6	baspread	Bid-ask spread	Amihud & Mendelson	1989, JF	CRSP	Monthly
7	beta	Beta	Fama & MacBeth	1973, JPE	CRSP	Monthly
8	betasq	Beta squared	Fama & MacBeth	1973, JPE	CRSP	Monthly
9	bm	Book-to-market	Rosenberg, Reid & Lanstein	1985, JPM	Compustat+CRSP	Annual
10	bm_ia	Industry-adjusted book to market	Asness, Porter & Stevens	2000, WP	Compustat+CRSP	Annual
11	cash	Cash holdings	Palazzo	2012, JFE	Compustat	Quarterly
12	cashdebt	Cash flow to debt	Ou & Penman	1989, JAE	Compustat	Annual
13	cashpr	Cash productivity	Chandrashekar & Rao	2009, WP	Compustat	Annual
14	cfp	Cash flow to price ratio	Desai, Rajgopal & Venkatachalam	2004, TAR	Compustat	Annual
15	cfp_ia	Industry-adjusted cash flow to price ratio	Asness, Porter & Stevens	2000, WP	Compustat	Annual
16	chatoia	Industry-adjusted change in asset turnover	Soliman	2008, TAR	Compustat	Annual
17	chcsho	Change in shares outstanding	Pontiff & Woodgate	2008, JF	Compustat	Annual
18	chempia	Industry-adjusted change in employees	Asness, Porter & Stevens	1994, WP	Compustat	Annual
19	chinv	Change in inventory	Thomas & Zhang	2002, RAS	Compustat	Annual
20	chmom	Change in 6-month momentum	Gettleman & Marks	2006, WP	CRSP	Monthly
21	chpmia	Industry-adjusted change in profit margin	Soliman	2008, TAR	Compustat	Annual
22	chtx	Change in tax expense	Thomas & Zhang	2011, JAR	Compustat	Quarterly
23	cinvest	Corporate investment	Titman, Wei & Xie	2004, JFQA	Compustat	Quarterly
24	convind	Convertible debt indicator	Valta	2016, JFQA	Compustat	Annual
25	currat	Current ratio	Ou & Penman	1989, JAE	Compustat	Annual
26	depr	Depreciation / PP&E	Holthausen & Larcker	1992, JAE	Compustat	Annual
27	divi	Dividend initiation	Michaely, Thaler & Womack	1995, JF	Compustat	Annual
28	divo	Dividend omission	Michaely, Thaler & Womack	1995, JF	Compustat	Annual
29	dolvol	Dollar trading volume	Chordia, Subrahmanyam & Anshuman	2001, JFE	CRSP	Monthly
30	dy	Dividend to price	Litzenberger & Ramaswamy	1982, JF	Compustat	Annual
31	ear	Earnings announcement return	Kishore, Brandt, Santa-Clara & Venkatachalam	2008, WP	Compustat+CRSP	Quarterly

No.	Acronym	Firm characteristic	Paper's author(s)	Year, Journal	Data Source	Frequency
32	egr	Growth in common shareholder equity	Richardson, Sloan, Soliman & Tuna	2005, JAE	Compustat	Annual
33	ep	Earnings to price	Basu	1977, JF	Compustat	Annual
34	gma	Gross profitability	Novy-Marx	2013, JFE	Compustat	Annual
35	grCAPX	Growth in capital expenditures	Anderson & Garcia-Feijoo	2006, JF	Compustat	Annual
36	grltnoa	Growth in long term net operating assets	Fairfield, Whisenant & Yohn	2003, TAR	Compustat	Annual
37	herf	Industry sales concentration	Hou & Robinson	2006, JF	Compustat	Annual
38	hire	Employee growth rate	Bazdresch, Belo & Lin	2014, JPE	Compustat	Annual
39	idiovol	Idiosyncratic return volatility	Ali, Hwang & Trombley	2003, JFE	CRSP	Monthly
40	ill	Illiquidity	Amihud	2002, JFM	CRSP	Monthly
41	indmom	Industry momentum	Moskowitz & Grinblatt	1999, JF	CRSP	Monthly
42	invest	Capital expenditures and inventory	Chen & Zhang	2010, JF	Compustat	Annual
43	lev	Leverage	Bhandari	1988, JF	Compustat	Annual
44	lgr	Growth in long-term debt	Richardson, Sloan, Soliman & Tuna	2005, JAE	Compustat	Annual
45	maxret	Maximum daily return	Bali, Cakici & Whitelaw	2011, JFE	CRSP	Monthly
46	mom12m	12-month momentum	Jegadeesh	1990, JF	CRSP	Monthly
47	mom1m	1-month momentum	Jegadeesh & Titman	1993, JF	CRSP	Monthly
48	mom36m	36-month momentum	Jegadeesh & Titman	1993, JF	CRSP	Monthly
49	mom6m	6-month momentum	Jegadeesh & Titman	1993, JF	CRSP	Monthly
50	ms	Financial statement score	Mohanram	2005, RAS	Compustat	Quarterly
51	mvel1	Size	Banz	1981, JFE	CRSP	Monthly
52	mve_ia	Industry-adjusted size	Asness, Porter & Stevens	2000, WP	Compustat	Annual
53	nincr	Number of earnings increases	Barth, Elliott & Finn	1999, JAR	Compustat	Quarterly
54	operprof	Operating profitability	Fama & French	2015, JFE	Compustat	Annual
55	orgcap	Organizational capital	Eisfeldt & Papanikolaou	2013, JF	Compustat	Annual
56	pchcapx_ia	Industry adjusted % change in capital expenditures	Abarbanell & Bushee	1998, TAR	Compustat	Annual
57	pchcurrat	% change in current ratio	Ou & Penman	1989, JAE	Compustat	Annual
58	pchdepr	% change in depreciation	Holthausen & Larcker	1992, JAE	Compustat	Annual
59	pchgm_pchsale	% change in gross margin - % change in sales	Abarbanell & Bushee	1998, TAR	Compustat	Annual
60	pchquick	% change in quick ratio	Ou & Penman	1989, JAE	Compustat	Annual
61	pchsale_pchinvt	% change in sales - % change in inventory	Abarbanell & Bushee	1998, TAR	Compustat	Annual
62	pchsale_pchrect	% change in sales - % change in A/R	Abarbanell & Bushee	1998, TAR	Compustat	Annual

No.	Acronym	Firm characteristic	Paper's author(s)	Year, Journal	Data Source	Frequency
63	pchsale_pchxsga	% change in sales - % change in SG&A	Abarbanell & Bushee	1998, TAR	Compustat	Annual
64	pchsaleinv	% change sales-to-inventory	Ou & Penman	1989, JAE	Compustat	Annual
65	pctacc	Percent accruals	Hafzalla, Lundholm & Van Winkle	2011, TAR	Compustat	Annual
66	pricedelay	Price delay	Hou & Moskowitz	2005, RFS	CRSP	Monthly
67	ps	Financial statements score	Piotroski	2000, JAR	Compustat	Annual
68	quick	Quick ratio	Ou & Penman	1989, JAE	Compustat	Annual
69	rd	R&D increase	Eberhart, Maxwell & Siddique	2004, JF	Compustat	Annual
70	rd_mve	R&D to market capitalization	Guo, Lev & Shi	2006, JBFA	Compustat	Annual
71	rd_sale	R&D to sales	Guo, Lev & Shi	2006, JBFA	Compustat	Annual
72	realestate	Real estate holdings	Tuzel	2010, RFS	Compustat	Annual
73	retvol	Return volatility	Ang, Hodrick, Xing & Zhang	2006, JF	CRSP	Monthly
74	roaq	Return on assets	Balakrishnan, Bartov & Faurel	2010, JAE	Compustat	Quarterly
75	roavol	Earnings volatility	Francis, LaFond, Olsson & Schipper	2004, TAR	Compustat	Quarterly
76	roeq	Return on equity	Hou, Xue & Zhang	2015, RFS	Compustat	Quarterly
77	roic	Return on invested capital	Brown & Rowe	2007, WP	Compustat	Annual
78	rsup	Revenue surprise	Kama	2009, JBFA	Compustat	Quarterly
79	salecash	Sales to cash	Ou & Penman	1989, JAE	Compustat	Annual
80	saleinv	Sales to inventory	Ou & Penman	1989, JAE	Compustat	Annual
81	salerec	Sales to receivables	Ou & Penman	1989, JAE	Compustat	Annual
82	secured	Secured debt	Valta	2016, JFQA	Compustat	Annual
83	securedind	Secured debt indicator	Valta	2016, JFQA	Compustat	Annual
84	sgr	Sales growth	Lakonishok, Shleifer & Vishny	1994, JF	Compustat	Annual
85	sin	Sin stocks	Hong & Kacperczyk	2009, JFE	Compustat	Annual
86	sp	Sales to price	Barbee, Mukherji, & Raines	1996, FAJ	Compustat	Annual
87	std_dolvol	Volatility of liquidity (dollar trading volume)	Chordia, Subrahmanyam & Anshuman	2001, JFE	CRSP	Monthly
88	std_turn	Volatility of liquidity (share turnover)	Chordia, Subrahmanyam, &Anshuman	2001, JFE	CRSP	Monthly
89	stdacc	Accrual volatility	Bandyopadhyay, Huang & Wirjanto	2010, WP	Compustat	Quarterly
90	stdcf	Cash flow volatility	Huang	2009, JEF	Compustat	Quarterly
91	tang	Debt capacity/firm tangibility	Almeida & Campello	2007, RFS	Compustat	Annual
92	tb	Tax income to book income	Lev & Nissim	2004, TAR	Compustat	Annual
93	turn	Share turnover	Datar, Naik & Radcliffe	1998, JFM	CRSP	Monthly
94	zerotrade	Zero trading days	Liu	2006, JFE	CRSP	Monthly

Figure 10: Variables List

C Algorithms in Detail

C.1 Optimizer

To minimize the MSE, I use the ADAM optimizer from Kingma and Ba (2014) following Shihao et al. (2018).

The pseudo-code is provided below:

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Figure 11: ADAM optimizer pseudo-code, Kingma and Ba (2014)

C.2 Early Stopping

Following Shihao et al. (2018), I use early stopping algorithm which introduces regularization to the algorithm against over-fitting. I use patience := 100 and learning rates specified by Python Keras library.

Algorithm Early Stopping

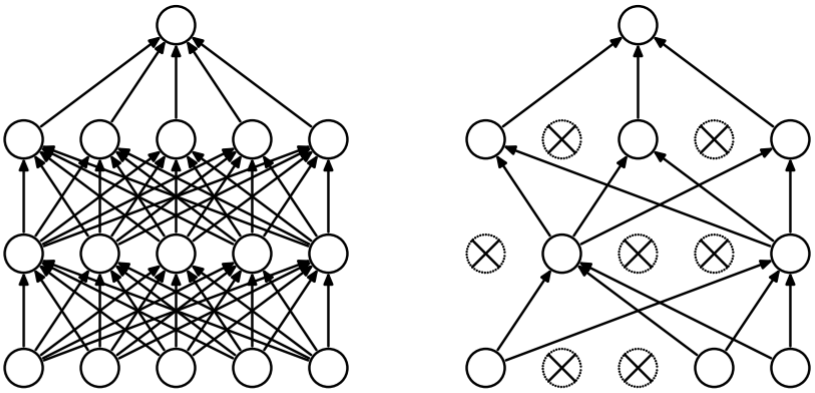
Initialize $j = 0$, $\epsilon = \infty$ and select the patience parameter p.
while $j < p$ do
  Update θ using the training algorithm
  Calculate the prediction error from the validation sample, denoted as ϵ'.
  if $\epsilon' < \epsilon$ then
    $j \leftarrow 0$.
    $\epsilon \leftarrow \epsilon'$.
    $\theta' \leftarrow \theta$.
    else
    $j \leftarrow j + 1$.
  end
end
Result: The final parameter estimate is θ'.

Figure 12: Early Stopping Algorithm, Shihao et al. (2018)

C.3 Dropout

To further increase regularization, I introduce dropout between layers. Dropout randomly selects p portion of the nodes in the layer and prevents them from being used in the training.

Dropout significantly reduces over-fitting and gives major improvements over other regularization methods. It improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets Srivastava et al. (2014).



(a) Standard Neural Net (b) After applying dropout.

Figure 13: Dropout in action, Srivastava et al. (2014)

C.4 Batch Normalization

Even though batch normalization is used by Shihao et al. (2018), this paper found that its use makes the training process less accurate. So, it is not used in this study. However, the pseudo-code can be found below in case it is used in the final submission.

Algorithm Batch Normalization (for one Activation over one Batch)

Input: Values of x for each activation over a batch $\mathcal{B} = \{x_1, x_2, \ldots, x_N\}$.

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{\mathcal{B}})^2$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta := \mathrm{BN}_{\gamma,\beta}(x_i)$$

Result: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i) : i = 1, 2 \ldots, N\}$.

Figure 14: Batch normalization, Shihao et al. (2018)

Batch normalization is applied after each activation layer. It normalizes the output of the activation layer such that it has mean zero, and standard deviation 1.