Ludwig–Maximilians–Universität München

Institut für Statistik

Master's Thesis

A comprehensive analysis of the use of deep learning models for forecasting the cross-section of stock returns

Cem Öztürk

Munich, 08.05.2020

Supervision: Prof. Stefan Mittnik, PhD and Dr.Benjamin Moritz

Abstract

This thesis focuses on recent developments in neural networks and their predictive power for forecasting the cross-section of stock returns. Neural networks with architectures ranging from shallow to deep are used with different hyperparameter configurations. Company characteristics are considered as the feature set and relationship between features are explained. On the other hand, this work also challenges the recent conclusions about the applicability of neural networks in the financial domain with counter arguments and reasons are supported with the works from machine learning research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.0.1 Forecasting with Neural Networks

Recent advances in the computer hardware, especially in processing units have enabled researchers to handle data that was unfeasible to analyze in the past. These advances also let the machine learning algorithms such as artificial neural networks to regain popularity which was lost in the late 1990s because of their intensive computational requirements. Today, the availability of the modern and more specialized hardware (graphical processing units, tensor processing units etc.) allow researchers to work with more complex algorithms especially in the area of artificial neural networks and the term "deep neural networks" is popular in the machine learning research.

These advances enabled the researchers to conduct more sophisticated analyses with different types of data which is called alternative data.[4] It is a known fact that analysts have been attempting to forecast the decrease/increase in retail sales and economic activity using the satellite images from the parking lot of retail stores. There are also industry specific usages of alternative data such as monitoring mining facilities continuously to capture changes in mining inventories. Wireless signals in the crowded areas show where customers prefer to eat their meals along the day. Both the increase in the availability of alternative data and the models to analyze data with vast capacity of resources help

quantitative researchers to add precision to their portfolio forecasts. [1]

The developments in the hardware industry also made computing intensive methods popular again. As an example, the ancestor of the neural networks, the perceptron, was found in 1957.[24] However, the machines were significantly bigger in terms of size at that time and duration of computations were defined with not minutes but days. Today, it is possible to run a multi layer perceptron which has many parameters and is more complex compared to its ancestors, also known as artificial neural network, on a small computer. Additionally, availability of opportunities to use better hardware resources by remote access and cloud computing helps the researchers if they do not have enough computing power for their research.

Therefore, I used a wide variety of neural networks to forecast cross section of stock returns using the company characteristics as feature set. Neural networks are able to capture the non linear relationship between features through the non linear activation functions.[10] I compare the predictions that are obtained from different neural network architectures and provide ways to interpret them as well as paths to pick neural networks for long term investment strategies. Rest of the thesis goes as following: in the next chapter, I have provided a short literature review about both forecasting the cross-section of stock returns and a short introduction about neural networks. Then, the empirical approach and methodology are provided. I compare the results with some industry benchmarks as well as factor based models and derived some ways to interpret the models and explained their behaviour. In the end, suggestions are made to improve outcomes and indicate future directions in research.

[1] https://www.theatlantic.com/magazine/archive/2019/05/stock-value-satellite-images-investing/586009/

1.0.2 Main Findings

I have created artificial neural network models with hidden layers ranging from 1 layer neural network through 10 layer neural network and aimed to predict the change in the monthly stock price of large cap US companies using financial characteristics that are calculated each quarter from 1970 to 2018. I have compiled the results that I have found at the end of my experiments:

Deep networks work well, too The neural networks that have one or two hidden layers are often considered as shallow neural networks and since their architectures are less simple and less connections are present between layers, they take lesser time in terms of computations. The neural networks which have different number of layers and neurons on each layer focus on different features, which is expected. On the other hand, contrary to common belief, deep neural networks still have relatively good performance compared to shallow networks and they are able to learn from the non linear relationship of the financial characteristics of a company such as book to the market ratio, operating leverage etc. One should not rather use only shallow networks but also deep networks to average model performances and capture variations in their predictive powers.

Deep networks work well with small samples Increasing sample size usually improves the forecasts for the future since the model has more observations to learn from. On the other hand, more sample comes with a computational burden. Using less sophisticated machine learning models might be a solution, however predictive power of less sophisticated algorithms are usually limited. Neural networks are known to work with big samples and researchers are skeptical with the performance of neural networks. The recent findings[20] showed that there are deep neural networks that are able to perform well with small amount of observations and still able to learn from the sample with high predictive power. In this work, I produced reliable and promising results with a neural network that has more than hundred thousand parameters and the neural network predicted cross-section of stock returns successfully using the training samples that are around fifty

thousand observations.

Unconscious application of best practices may apply bias One point that I argue and tried to prove is that best configurations or best practices are not really necessary all the time when working with neural networks in the financial domain. One example is that for deep neural networks, it is usually suggested to employ batch normalization and dropout techniques as a regularization because of high number of layers as well as parameters[13][11]. However, batch normalization is not really necessary for the shallow neural networks since the effect of covariance shift between layers is negligible. Persisting the adaptation of batch normalization also may imply a bias on the neural network, therefore it reduces overall performance of neural networks. For deep neural networks, the effect is the same alike. Analogical architectures may lead to unexpected results when their activation functions, optimization methods, initial weights, learning rates are switched. In the end, these tunings have the possibility to amplify portfolio returns in the long term.

Neural networks are interpretable Neural networks are known as black-box models, inferring that it is hard to extract which features they weigh the most and the least from the predictions. I have used LIME[23] to explain model behaviours, however the problem is that LIME does not extract information about the model considering the whole sample but rather focuses explaining the behaviour using a subsample. On the contrary, it is possible to come up with a novel conclusion about the global model behaviour using the outcomes of multiple applications such as changes in turnover ratios, maximum drawdowns, coefficients that are obtained through the regression on factors and interpretations that are acquired through interpretability library. In this work, I have shown that neural networks usually put higher weights to similar features, but the small differences in the weights define their performance in the long run.

Neural networks are aggressive While forecasting cross-section of stock returns, I have used mostly the information that are calculated quarterly (except book to the market

ratio and price to the book ratio) from financial announcements. Neural networks make the major change in the portfolio every quarter since the majority of the feature set is updated. However, neural networks make small tunes to the portfolio using easy to calculate ratios such as book to the market ratio and price to the book ratio at the end of each month. Therefore, the aggressive behavior of stock returns might be restrained using that fact and asset managers may reduce trading costs when using neural networks as a helper in their decision making processes.

Chapter 2

Theory

2.1 Forecasting Returns

In financial econometrics, there are two ways to look at the expected returns. On one hand, time-series analysis examines the changes in the average returns over time. On the other hand, cross section analysis focuses on how average expected returns change across different stocks or portfolios. In this research, the aim is to find an answer to the question "Why stock A has a higher return expectation compared to stock B?" by trying to model the non-linear relationship between company characteristics using neural networks. Since the focus is on the difference of expected returns at the same point in time, this analysis is regarded as cross section analysis.[3]

There are already works that are done in the same field. Gu (2019) used the machine learning algorithms as well as less sophisticated neural networks, shallow neural networks, and found that forecasts that are made with machine learning algorithms and neural networks can provide higher alphas compared to the Fama - French 3 factor model portfolios[6] and Fama - French 5 factor model portfolios [7] [11]. In that work, it is also concluded that shallow networks perform better than deep networks (the most complex neural network has five layers utmost) in terms of forecast accuracy. Feng et al. (2018) also predicted the cross-section of stock returns[8] with neural networks and showed that forecasts for

monthly returns using deep learning are more accurate than forecasts using simple historical mean in predictions. This is an important finding due to the reason that Welch and Goyal (2008) claimed that predictive regression models are not able to beat historical mean.[28]

2.2 Neural Networks

Machine learning algorithms can be defined as the algorithms that can extract patterns from sample. According to[17]:
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

In this work, the task is defined as forecasting the cross section of returns using the financial ratios that are derived using company characteristics. In machine learning terms, the task is defined as a regression task with $f : R^n \rightarrow R$ where N features (financial ratios) are mapped to a target which is expected return of a stock.

The performance measure is different for each application and my focus is the application of the neural networks as an investment strategy. As a result, I have defined the performance measure P as the cross-sectional average return for each neural network and compared it with a benchmark which I selected as the monthly equally-weighted average large cap stock return between 1976 and 2019. The neural networks are trained on a different performance measure, mean absolute error, which I explain in the next section.

During the training phase, neural network learns from training sample provided realized returns and produces the forecasts as expected returns for observations in the validation sample and the test sample. Therefore, neural network learns from the experience by modelling the relationship between features and the target feature. In formal terms, the

experience E can be defined as supervised learning problem since company characteristics for stock i at time t is associated with a target feature, the change in the price from t to $t+1$, return $r_{i,t+1}$ at $t+1$ for stock i.

The performances in the training and the validation phases are useful when configuring the neural network and help to understand whether the model is working or not. However, these performances do not really reflect the ability of neural networks to learn. The performance of neural networks can be accurately measured using a sample, where neural networks are not exposed before. If neural networks are also able to perform well on unseen observations, they are claimed to generalize well on unseen sample. The error on unseen sample, or test sample is called as generalization error. A low generalization error shows the neural network is successful at modelling the relationship between input features.

2.2.1 Purpose

Neural networks are specific estimators that are designed using loose inspirations from human brain. Their goal is to approximate some function f^*. A neural network defines the mapping $y = f(x;\theta)$ and tries to estimate the parameters θ to find best function approximation f of f^*. In this work, I have used only feed forward neural networks, which enables information to pass from input $\mathbf{x}$ through some function f to output $\mathbf{y}$. There are no transition of feedback between the layers, therefore information goes only one way. When there exists a feedback connection between layers, this specific type of neural network is called as recurrent neural network. Recurrent neural networks are suitable for time-series analysis rather than cross sectional analysis.[10]

Neural networks consist of an input layer, hidden layers and an output layer. Hidden layers reside between input layer and output layer. Input layer consists of the features that are fed to the neural network. Output layer consists of the target that neural network aims to estimate. On each layer, at least one neuron is present.

During training phase, neural network tries to match $f(\mathbf{x})$ and $f^*(\mathbf{x})$. The training sample enables the network to learn from label $\mathbf{y}$ and the network tries to predict a close value to target $\mathbf{y}$ by tuning its own parameters on its layers. However, the behaviour of hidden layers in the neural network cannot be observed and interpretable.

In simpler terms, to understand the working mechanism of neural networks, one can start with thinking about linear models. In linear models, the results are obtained through either closed form solution or convex optimization. The problem with linear models is that the capacity is usually limited with linear functions, therefore it is almost impossible to capture non linear relationships between features. If one would like to go beyond linear model, one does not use the linear $\mathbf{x}$ but the nonlinear transformation of $\mathbf{x}$ which is $\phi(\mathbf{x})$ using the transformation ϕ. In that sense, $\phi(\mathbf{x})$ is a new representation of $\mathbf{x}$ which is high dimensional due to the nonlinear transformation function.

Neural networks try to learn the mapping function $\phi(\mathbf{x})$ to approximate f^*, in other words it decides the transformation of variables by itself. The model can be described as $y = f(\mathbf{x}; \theta, w) = \phi(\mathbf{x}; \theta)^T \mathbf{w}$. Parameter θ is used to learn ϕ and $\mathbf{w}$ maps the nonlinear transformation of $\mathbf{x}$ to the desired output. Here, the $\phi(\mathbf{x})$ describes the hidden layer of neural network.

On a practical level, neural networks can be described as nesting of functions. The linear regression can be expressed with one layer neural network as $f(x) = \mathbf{W}\mathbf{x}$ by defining the nonlinear transformation function as $\mathbf{I}$ where $\mathbf{I}$ is linear activation function or identity function. To describe it in a more general way, neural networks take a vector of inputs $\mathbf{x}$ and computes a transformation $z = \mathbf{W^T x} + \mathbf{b}$ and applies an element-wise nonlinear function $g(\mathbf{z})$. This nonlinear function $g(\mathbf{z})$ is the activation function that provides non-linearity to neural network.

As an example, one can describe multiple layer neural network assuming the activation function is defined as max function:

- 1-layer : $f = \mathbf{W_1 x}$
- 2-layers: $f = \mathbf{W_2} \max(0, \mathbf{W_1 x})$
- 3-layers: $f = \mathbf{W_3} \max(0, \mathbf{W_3} \max(0, \mathbf{W_1 x}))$

... up to N layers

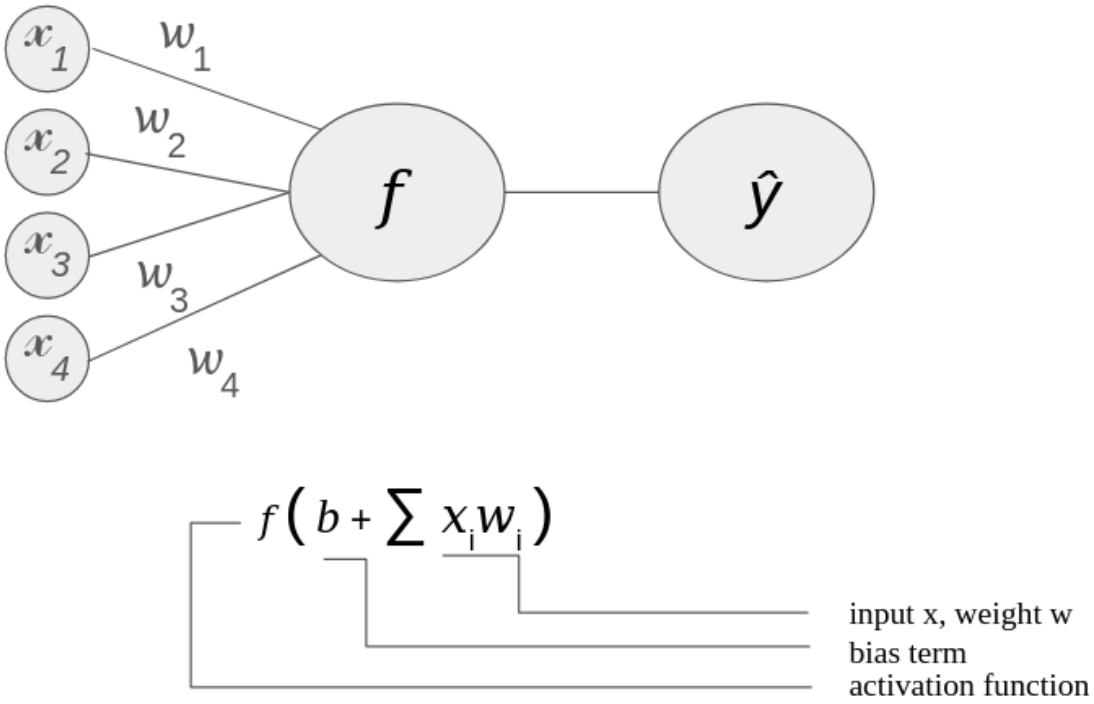
When neural networks take an input $\mathbf{x}$ and makes a forecast as $\hat{\mathbf{y}}$, information flows from input to the output layer. This is called as forward propagation. Forward propagation stops when a scalar cost $J(\theta)$ is calculated. During training, information flows from the output towards the input, which is called backpropagation. The backpropagation algorithm lets the information to flow through the network to compute the gradient of cost function. The aim of neural network at that point is to perturb the parameters in a way thus the decrease in the cost is maximum. This training iteration is repeated many times and each iteration is called as epoch. When the network repeats this process many times using a predefined number of epochs, it sets its parameters which pushes the cost to the minimum value within the number of epochs.

There are many considerations when training a neural network. Network depth (number of layers) and network width (number of neurons on each layer) are two important measures. Cost function is also a critical factor in training neural networks. Also, training the neural network with low number of epochs may apply bias on the model since the cost may not be minimized and a high number of epochs may lead to overfitting. The number of epochs needs to be picked carefully.

Neural networks are trained using maximum likelihood, therefore the cost function can be described as negative log-likelihood. The performance measure of neural network during training, mean absolute error cost, can be defined as:

$$J(\theta) = E_{\mathbf{x},\mathbf{y}\ \hat{p}_{sample}} |\mathbf{y} - f(\mathbf{x}; \theta)| + const.$$

In the cost function above, the $\hat{p}_{sample}$ refers to the training sample that neural network learns from and the discarded constant represents the variance of the distribution which is not parameterized. Mean absolute error is not differentiable and it has some bottlenecks during the calculation of gradients of the loss function. However, its robustness for outliers and initial experiments in the empirical analysis made me to proceed with mean absolute cost.



This neural network consists of one input layer with four neurons, one hidden layer with one neuron and one output layer with one neuron. The network adjusts weights for each features using backpropagation. The bias term is used to prevent problems (vanishing gradient) while training step. Since output is one dimensional, return for $t + 1$, the output layer has one neuron. Activation function is defined here with f and f can be anything, such as maximum, minimum, sigmoid or identity function etc. As an example, the problem turns to linear regression problem if the identity function is used.

Figure 2.1: One layer neural network with one neuron

2.2.2 Library

In the analysis phase, I have used Pytorch[21] library to design neural networks and carry out the analysis. Pytorch library is developed by Facebook and it is based on Torch[1] library and used mostly in natural language processing and computer vision.

2.2.3 Configurations

The neural networks that I have used are inspired from recent research in the deep learning field[20] since this deep neural network are proven to generalize on small samples. All of the neural networks I have used share the following configurations:

- Initialization Function: He initialization[12]
- Optimization: Adam optimization[14]
- Activation Function: ELU[2]

Initialization of Layers When neural networks are initialized and ready for training, the input vector $\mathbf{x}$ performs a dot product with the layers of weights $\mathbf{w}$. Therefore, these weights should be initialized with some number. If one initializes the weights with zeroes, the network does not learn properly. If the weights are initialized too small, the signal from the input gets much more smaller as it is passed through layers and in networks with many layers, deeper layers would not be able to learn from the sample. As a result, one needs to initialize the weights carefully. When using PyTorch library, weights are initialized with He initialization (Kaiming Uniform).[2].

In He initialization, the weights are picked from standard normal distribution $N(0, 1)$. Then they are scaled using $\frac{\sqrt{2}}{\sqrt{n}}$ where n defines number of neurons in the previous layer. The bias term b is set as zero. This is defined as Kaiming Uniform initialization of weights and proven as a good initialization strategy for neural networks.[12]

[1] https://github.com/torch

[2] Interested reader might check Xavier initialization as a starting point[9]

One of the most popular activation function in the deep learning research is ReLU (Rectified Linear Units) activation function. The popularity of ReLU arises from the fact that it allows to calculate gradients easily.[18] The reason that I have used ELU (Exponential Linear Units) activation function [2] rather than ReLU activation is that it generalizes better compared to ReLU activation function in deep neural networks that are trained with small samples.[20] The main difference of ELUs compared to ReLU is that it allows negative value for gradients and allowing small gradients results analogue to batch normalization with a lower computational complexity.

$$ReLU : f(x) = \max(0, x)$$

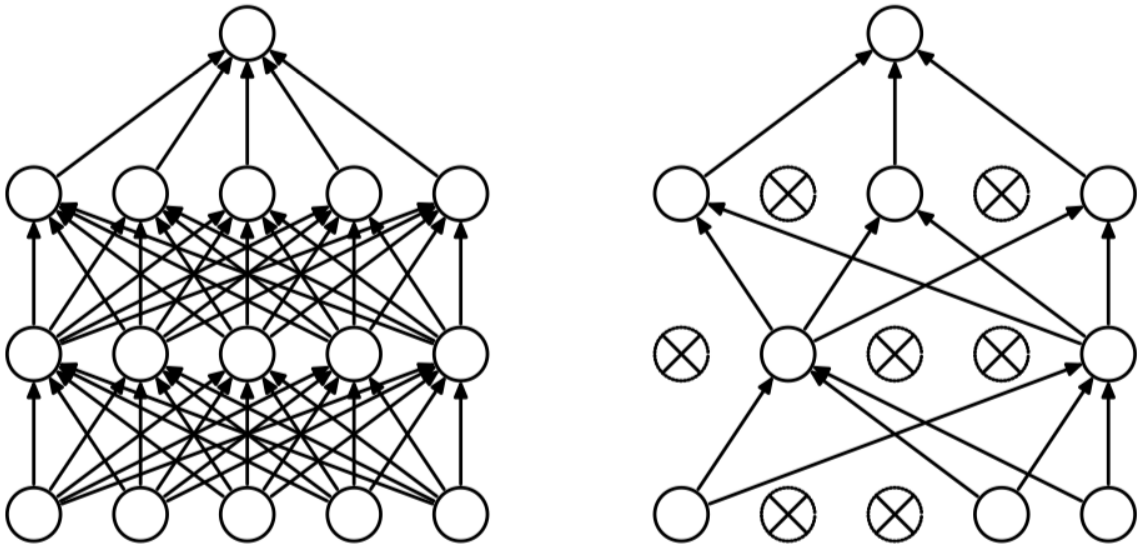
$$ELU : f(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}$$

For the optimization, I have used Adam[14] optimization technique. In contrast to the traditional methods in optimization such as Stochastic Gradient Descent (SGD) which has a learning rate that needs to be tuned for each optimization, Adam makes use of the moments of gradients μ and σ (first moment and second moment of gradients) by calculating the moving average of gradients to ensure a smoother learning compared to stochastic gradient descent.

I did not use other techniques such as batch normalization[13] since it is suggested for deep networks (such as in computer vision where neural networks consist of many layers) and not for shallow networks. Analyses that I made using batch normalization showed that it implies bias to the networks and the performances of the networks reduce in the end.

For deep neural networks with high number of parameters, dropout is usually suggested to be employed to reduce overfitting.[26] Overfitting happens when the network seems

to learn from training sample however neural networks cannot forecast the cross section of stock returns successfully when using the test sample. The term overfitting can be simplified as following: an overfitting neural network tries to mimic the output using the values in training sample instead of modelling the relationship between the features. Dropout deactivates random neurons in a neural network to prevent neural network to mimic the results and ensure learning is happening. In this work, I tuned the models using both with and without dropout and I have found that neural networks trained without dropout are still able to perform well on the test sample. This behaviour is also proven by different authors in the deep learning research.[20][22]



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

Figure 2.2: Dropout - Srivastava et al.(2014)

Chapter 3

Empirics

3.1 Data Description and Features

3.1.1 Universe

Monthly data for the empirical analysis is obtained from Center for Research in Security Prices (CRSP) for all stocks that are listed in the NYSE, NYSE MKT (AMEX) and NASDAQ. Company characteristics by firm level is available starting from January 1970, therefore the analysis spans the period between 1970 to 2019 (Appendix B). In order to make the analysis more applicable to the real world, only the stocks that have share codes 10 and 11 are used in the sample which defines ordinary common shares that are listed in these exchanges.

Stocks that are traded in these exchanges for a short period of time (less than 24 months) are removed from the sample.

3.1.2 Features

The initial sample has 2797065 rows and 76 columns including the target feature, return $r_{t+1,i}$ for the next month, defined as

$$r_{t+1} = \frac{p_{i,t+1} - p_{i,t}}{p_{i,t}} \tag{3.1}$$

where $p_{i,t}$ is the stock price of stock i at time t. Of 76 input features, two features are about trading information, trading volume and market cap information. Also, only two features are engineered; momentum factor for one month and momentum factor for last twelve months are calculated for the feature set to examine the impact of momentum factor. Rest of the feature set consists of company characteristics that are calculated quarterly except book to the market and price to the book ratios which are updated monthly.

It is decided to keep only large cap stocks whose market cap are higher than median market cap in the sample according to Kenneth French's data library.[1]After this reduction is applied, sample is reduced to 534840 observations over 48 years. The training window covers only 5 years and shifted quarterly, average training sample has about fifty thousand observations which is 10% of reduced sample size. This reduction of sample may raise the question about the effectiveness of neural networks because deep neural networks are known to generalize better with high number of observations, however results provided that neural networks with deep architectures are able to approximate the non linear relationships between company characteristics with small samples.

The reason that small cap and middle cap stocks are filtered is mainly due to the ambiguities in calculation of returns for delisted stocks. Even it affects our neural networks' performance in a positive way, I have removed these small cap and mid cap companies from sample. Number of large cap stocks in my sample changes between 643 and 1594 over time. The period where the number of large cap stocks reached its maximum is just

[1]https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/det_me_breakpoints.html

before dot com bubble which happened in late 1990s.

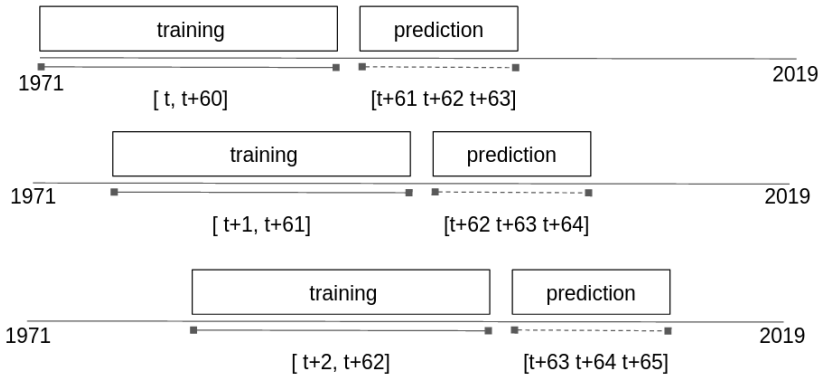


Figure 3.1: Walk forward testing for final models

In the final feature set, not all of the 76 features are used since it is discovered that features such as operating profit margin after depreciation (opmad), operating profit margin before depreciation (opmbd), cash flow margin (cfm) and pre-tax profit margin (ptpm) have almost 100% correlation with net profit margin (npm) feature. These features (Table 3.1) are removed from sample to speed up training phase. In the correlation matrix, there are still highly correlated features (over 90%) as it can be seen in the Figure 3.2 however further dimension reduction techniques are not applied to reduce feature set.

opmad	**opmbd**	**cfm**	**ptpm**
npm	npm	npm	npm

Table 3.1: Most correlated features with net profit margin

There are also few columns with high missing value percentages in the reduced sample. Even models are tested with walk forward testing and the missing percentages tend to change over time, the features with more than 30% missing values are removed from the feature set. The network configuration should be altered every single time another feature is added/removed to the input layer of neural network hence forth they are kept out of the analysis. The features that are removed because of high missing value ratios are

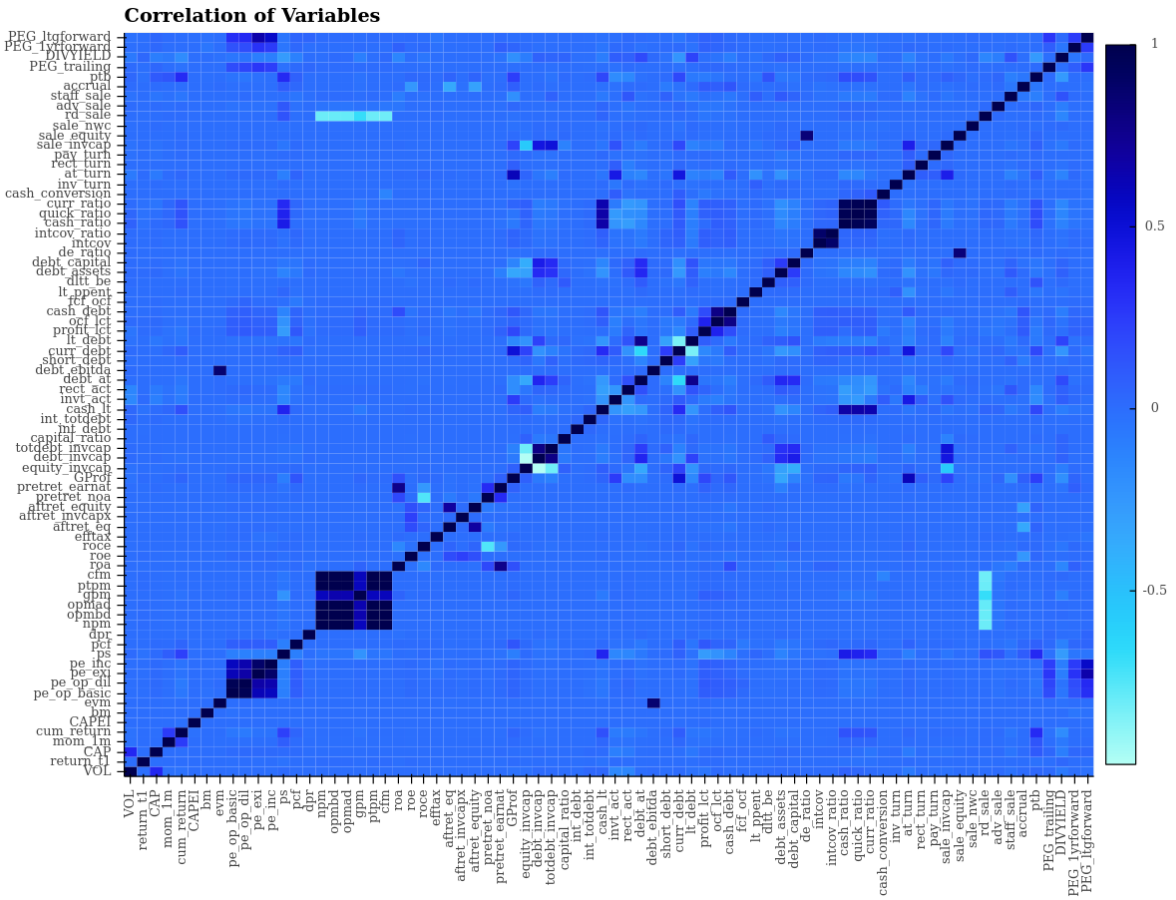


Figure 3.2: Correlation matrix of features

price/operating earnings (pe_op_basic), price/operating earnings diluted (pe_op_dil), forward p/e to long-term growth ratio (PEG_ltgforward) and trailing p/e to growth (PEG) ratio (PEG_trailing) and their missing value percentages are provided in the Table 3.2. Forward p/e to 1-year-growth ratio (PEG_1yrforward feature is also removed from sample since it is completely missing for the initial training period.

pe_op_basic	pe_op_dil	PEG_trailing	PEG_ltgforward
36.2%	57.8%	38.3%	34.4%

Table 3.2: Columns with missing values

After the elimination of features that are either highly correlated or present with high missing percentages, feature set is reduced to 67 columns including the target feature $r_{i,t+1}$.

3.2 Training Neural Networks

3.2.1 Training Procedure

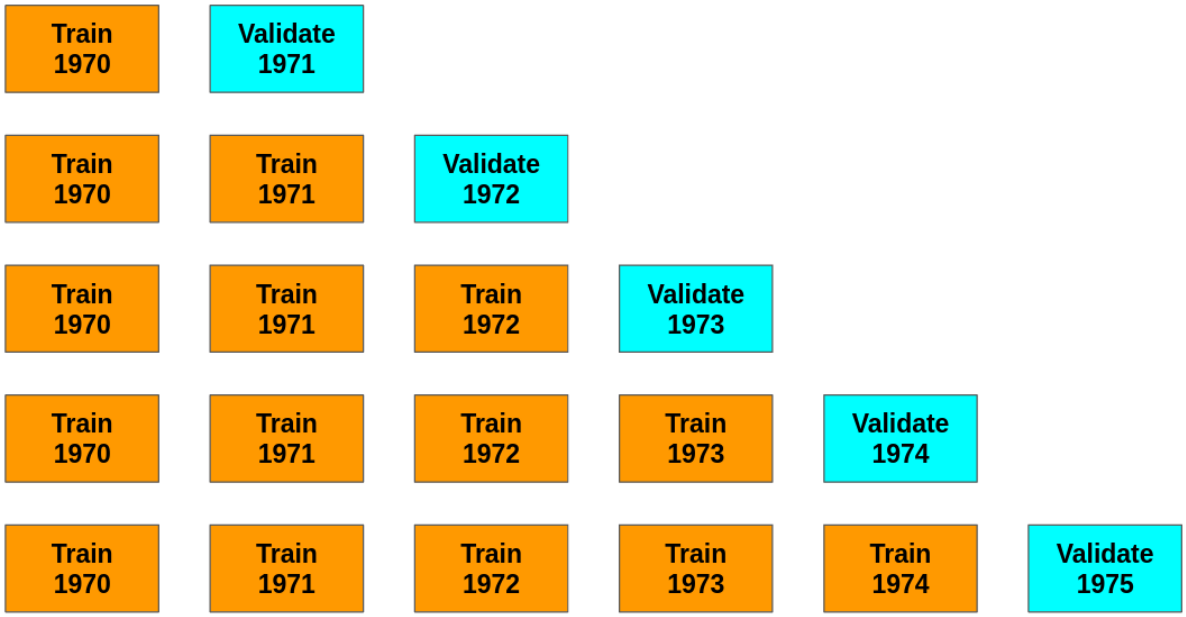
The training procedure consists of two sub-processes aiming to find best neural network architecture with best set of hyperparameters. The first step focuses on determining network depth (number of layers) and width (number of neurons per layer) and the second step is about finding the best hyperparameter configuration. Since there are no clear guidelines about how to construct neural network architectures and it is rather a trial and error procedure[10][2], depth and width of the neural network are two critical features which defines capacity of the model and plays a key role in the generalization performance.

For the first two sub-processes, architecture tuning and hyperparameter tuning phases, models are picked using the the walk forward method with expanding window. In walk forward method, initial sample is divided as training and hold out sample. In theory, if models really perform well, they should work independent of sample size. Therefore, I selected the period between 1970-1975 as training and validation sample. This subsample has 55998 observations. The rest of the sample, between 1976-2018 is used for walk forward testing with rolling window, where performance of final models, generalization performances, are compared.

In the training phase, after neural networks are trained in each window, forecasts are made for the hold out set. These forecasts for each validation sample are sorted and divided into 10 deciles. The average return of the decile having highest expected return (top decile) and lowest expected return (lowest decile) are calculated for each validation sample. Then it is assumed that neural network buys the top decile portfolio and sells the lowest decile

[2]There are approaches available such as NEAT. [27]

portfolio. (In other words, it creates a long short portfolio.) The neural networks that end up with the highest cumulative long-short portfolio return in the validation samples are picked for final forward testing phase.



Models are trained and hyperparameters are set according to the cumulative cross validation long-short performances

Figure 3.3: Walk forward cross validation design for training

3.2.2 Data Preprocessing

As can be found in Appendix B, the initial sample still has outliers as well as missing values after the columns with high missing percentages are removed. On the other hand, since the company characteristics have different scales, they need to be standardized within a range to speed up training phase.[1][15] Due to that reason, I have scaled all the features to $[-1, 1]$ range where -1 is the minimum value or vice versa.

The validation and test samples are scaled with training sample minimum and maximum rather than the test sample minimum and maximum. Since it is usually expected to have

lower variance for features in the training sample because of the size of the sample, I proceeded using the training sample's minimum and maximum to scale test sample.

To deal with outliers, I winsorized each feature both in each training sample and test sample to limit the effect of outliers. After the 1% and 99% percentiles of each feature are determined in each training sample, these values are used to limit the effect of extreme outliers that are present in validation sample (architecture and hyperparameter configuration) or the test sample(final walk forward performance comparison).

To fill the missing values, I have used cross-sectional median since most of the features have skewed distribution.(except DIVYIELD column, which is dividend to be paid to shareholders and filled with 0).

First Step: Finding the Architecture

To find out which neural network structures perform best for the analysis, I picked that there could be 15, 60 and 120 neurons in each layer. The reason for picking these numbers is that there are 66 features in the feature set therefore the aim is to find whether a specific neural network shape (e.g. constant number of neurons in each layer, decreasing number of neurons in deeper layers or vice versa) performs better than other architectures or not. For the first step of the analysis, 3281 neural networks are constructed having from no layer up to 10 layers with changing neurons in each layer. One specific neural network is included from the recent research in deep learning[20] which is ten layer neural network that is proven to generalize on small data sets.

For this phase, the learning rate is set as 0.003, dropout rate as 0% and neural networks are trained using 100 epochs.

Second Step: Finding the Hyperparameters

After the neural network architectures (capacities) are determined from no layer to 10 layers, the selected architectures are trained with different hyperparameter set (dropout rate, learning rate, number of epochs) to find best performing hyperparameters to ramp up the performances. The hyperparameters that are experimented with neural networks are defined as following:

- Dropout Rates: 0.0, 0.3, 0.7
- Learning Rates: 1e-2, 1e-3, 1e-4, 1e-5
- Epochs: 50, 100, 200

The combination of each hyperparameter setting is provided to candidate architectures and performances are compared again through walk forward methodology with expanding window. In the end, the best hyperparameter setting is picked for each neural network architecture and these settings are tested in the forward testing phase that is discussed in section 3.2.2.

The long-short portfolio results are provided in the Appendix C and according to the results, nine models are selected for forward testing phase. The models and their network architectures as well as hyperparameters are shown in Table 3.3.

The reason that the last two configurations are same is that I decided to compare the performance of the original research model[20] with default hyperparameter setting and optimized hyperparameter setting.

Step 3: Final Analysis with One Step Ahead Predictions

Following the architecture and hyperparameter configuration phases, neural networks are trained using walk forward methodology but this time with rolling window of five years

Name	Neurons and Layers	Dropout	Alpha	Epochs
NN1	66, 15, 1	0.0	1e-4	50
NN2	66, 120, 15, 1	0.7	1e-3	100
NN3	66, 60, 15, 15, 1	0.0	1e-4	50
NN4	66, 15, 15, 60, 120, 1	0.7	1e-5	200
NN5	66, 60, 15, 60, 120, 15, 1	0.3	1e-5	100
NN6	66, 15, 60, 60, 60, 60, 15, 1	0.0	1e-5	100
NN7	66, 15, 15, 15, 15, 60, 60, 15, 1	0.3	1e-5	100
NN8	66, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 1	0.0	1e-5	50
NN9	66, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 1	0.0	1e-3	200

Table 3.3: Network architectures and hyperparameters for forward testing

and forecasts are made covering next three months following the training window. Most of the company characteristics are calculated quarterly therefore training sample is shifted three months and neural networks are retrained with the new sample after predictions are made. Over 48 years, each neural network is trained and returns are estimated for 170 quarters.

After predictions are made for each month, they are sorted and divided into 10 deciles and companies that are going to be placed in the long portfolio (top decile) and short portfolio (lowest decile) are decided for each month. Neural networks' monthly cumulative performances over 40 years is calculated. As the benchmark portfolio and return, equally-weighted large cap portfolio and its return are used and the performances are compared with this benchmarks.

3.3 Forecast Evaluation

3.3.1 Comparison of Results

The first measure that I checked to evaluate neural networks is their cross-sectional average returns. During the hold out sample period that covers the period between January 1976 and December 2018, two neural networks showed significant performance. A shallow two layer neural network and ten layer neural network which is inspired from deep learning

research[20] showed relatively better performances compared to other networks. During this period, monthly equally weighted average large cap stock return is 1.07%. From the Table 3.4 it can be seen that top deciles of neural networks with two layers (NN2), five layers (NN5) and ten layers (NN9) performed better than the benchmark portfolio.

Deciles	Low	2	3	4	5	6	7	8	9	High	High-Low
NN1	1.09%	1.06%	1.07%	1.1%	1.05%	1.11%	1.1%	1.08%	1.0%	1.07%	-0.02%
NN2	0.67%	0.86%	1.03%	1.09%	1.1%	1.14%	1.12%	1.15%	1.29%	1.3%	0.62%
NN3	1.05%	0.97%	1.1%	1.04%	1.06%	1.13%	1.14%	1.08%	1.11%	1.06%	0.01%
NN4	1.0%	1.1%	1.17%	1.08%	1.07%	1.15%	1.1%	1.02%	1.02%	1.04%	0.04%
NN5	0.99%	1.05%	1.05%	0.99%	1.08%	1.06%	1.13%	1.05%	1.19%	1.15%	0.16%
NN6	1.08%	1.11%	1.11%	1.08%	1.05%	1.1%	1.04%	1.02%	1.1%	1.06%	-0.02%
NN7	0.96%	1.01%	1.15%	1.08%	1.18%	1.11%	1.14%	1.05%	1.03%	1.02%	0.06%
NN8	1.07%	1.02%	1.03%	1.09%	1.13%	1.19%	1.1%	1.06%	1.07%	0.98%	-0.09%
NN9	0.73%	0.94%	1.05%	1.13%	1.04%	1.07%	1.14%	1.2%	1.19%	1.24%	0.51%

Table 3.4: Cross-sectional average portfolio return per neural network

Deciles	Low	2	3	4	5	6	7	8	9	High	High-Low
NN1	0.02%	-0.01%	0.0%	0.03%	-0.02%	0.04%	0.03%	0.01%	-0.07%	0.0%	-1.09%
NN2	-0.4%	-0.21%	-0.04%	0.02%	0.03%	0.07%	0.05%	0.08%	0.22%	0.23%	-0.45%
NN3	-0.02%	-0.1%	0.03%	-0.03%	-0.01%	0.06%	0.07%	0.01%	0.04%	-0.01%	-1.06%
NN4	-0.07%	0.03%	0.1%	0.01%	0.0%	0.08%	0.03%	-0.05%	-0.05%	-0.03%	-1.03%
NN5	-0.08%	-0.02%	-0.02%	-0.08%	0.01%	-0.01%	0.06%	-0.02%	0.12%	0.08%	-0.91%
NN6	0.01%	0.04%	0.04%	0.01%	-0.02%	0.03%	-0.03%	-0.05%	0.03%	-0.01%	-1.09%
NN7	-0.11%	-0.06%	0.08%	0.01%	0.11%	0.04%	0.07%	-0.02%	-0.04%	-0.05%	-1.01%
NN8	0.0%	-0.05%	-0.04%	0.02%	0.06%	0.12%	0.03%	-0.01%	0.0%	-0.09%	-1.16%
NN9	-0.34%	-0.13%	-0.02%	0.06%	-0.03%	0.0%	0.07%	0.13%	0.12%	0.17%	-0.56%

Table 3.5: Cross-sectional average portfolio return minus benchmark portfolio return

However, the comparison of standard deviations of decile portfolios for each network shows that neural networks are aggressive in their stock pickings. Having a high degree of standard deviation in returns may fear the managers to invest in these portfolios. The standard deviation of benchmark portfolio return for the same period is 4.76%.

As last step, when the Sharpe ratios[25] of the models are compared in Table 3.7, it can be seen that models have usually Sharpe ratios less than 1. Among all top deciles, only

Deciles	Low	2	3	4	5	6	7	8	9	High	High-Low
NN1	5.45%	5.01%	4.77%	4.63%	4.69%	4.83%	4.77%	4.81%	5.07%	6.15%	4.51%
NN2	7.22%	5.52%	4.93%	4.77%	4.6%	4.48%	4.46%	4.42%	4.54%	4.96%	4.52%
NN3	6.22%	5.16%	4.83%	4.67%	4.67%	4.6%	4.67%	4.69%	4.91%	5.52%	4.39%
NN4	5.94%	4.97%	4.88%	4.69%	4.72%	4.71%	4.75%	4.72%	5.05%	5.6%	4.11%
NN5	5.49%	5.0%	4.82%	4.68%	4.73%	4.7%	4.72%	4.82%	5.07%	5.61%	3.79%
NN6	6.02%	5.04%	4.84%	4.76%	4.7%	4.7%	4.72%	4.74%	4.89%	5.42%	4.11%
NN7	5.96%	5.14%	4.81%	4.72%	4.68%	4.73%	4.67%	4.77%	4.88%	5.23%	3.65%
NN8	5.52%	4.98%	4.68%	4.77%	4.78%	4.72%	4.76%	4.86%	5.06%	5.76%	3.71%
NN9	7.3%	5.48%	4.86%	4.61%	4.52%	4.41%	4.38%	4.43%	4.72%	5.44%	4.49%

Table 3.6: Std. deviation of cross-sectional average portfolio return per neural network

top deciles of 2 layer neural network and 10 layer neural network have higher Sharpe ratio compared to Sharpe ratio of benchmark portfolio which is 0.51.

Deciles	Low	2	3	4	5	6	7	8	9	High	High-Low
NN1	0.46	0.48	0.51	0.54	0.5	0.53	0.53	0.51	0.43	0.4	-0.3
NN2	0.15	0.31	0.46	0.52	0.55	0.59	0.59	0.61	0.7	0.65	0.19
NN3	0.38	0.41	0.52	0.5	0.51	0.57	0.57	0.52	0.52	0.43	-0.29
NN4	0.37	0.51	0.57	0.52	0.52	0.57	0.53	0.48	0.44	0.41	-0.28
NN5	0.39	0.47	0.48	0.46	0.52	0.51	0.56	0.49	0.56	0.48	-0.19
NN6	0.41	0.51	0.53	0.52	0.5	0.54	0.49	0.47	0.51	0.44	-0.33
NN7	0.35	0.43	0.56	0.52	0.6	0.54	0.57	0.49	0.47	0.43	-0.29
NN8	0.44	0.45	0.48	0.52	0.55	0.6	0.53	0.49	0.48	0.37	-0.42
NN9	0.17	0.36	0.49	0.57	0.51	0.55	0.61	0.65	0.6	0.56	0.11

Table 3.7: Cross-sectional Sharpe ratio per neural network

3.3.2 Model Check

It is a known fact that stock market returns has a symmetric distribution with fat tails.[5] In the last layer of neural networks, in order to scale the outputs of last hidden layer, an activation function (Tanh)[10] is used to limit the outputs to $[-1, 1]$ therefore the distributions are narrower than the realized returns' distribution. In the figure 3.4, only neural networks that have higher average returns than benchmark portfolio return are included to increase visibility. The Figure 3.4 shows that neural network with two hidden

layers and ten hidden layers have a reasonable expected return distribution however five layer neural network's distribution of expected returns does not seem quite good.

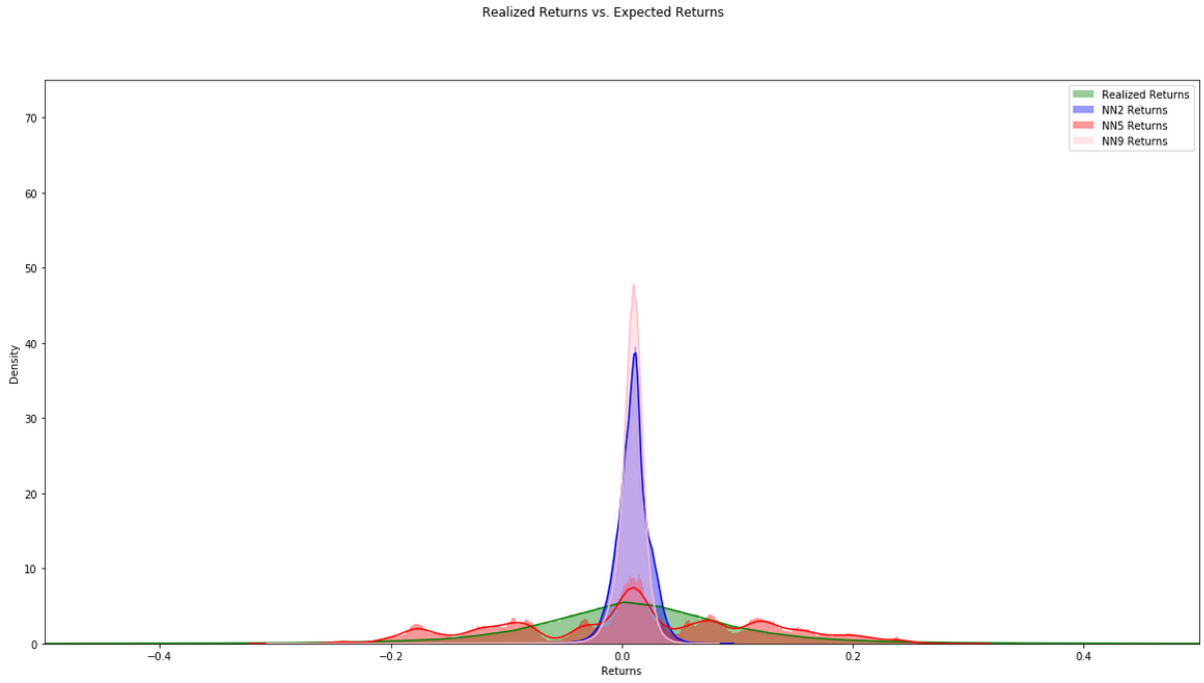


Figure 3.4: Distribution of expected returns for best performing networks

3.3.3 Trading Strategies

To evaluate the investment performances of neural networks, I have used long-short and long only portfolio strategies to compare performances. Figure 3.5 shows that only two layer neural network outperforms the benchmark portfolio return when long-short strategy is adapted. One important assumption that needs to be mentioned here is that since the universe consists of only large cap stocks which are not volatile in terms of return expectations, it may not be possible to capture huge losses often within the performance period therefore the short selling strategy may not be feasible.

When long only portfolios are compared, networks with two layers, five layers and ten layers perform better than the benchmark portfolio. The only issue that can be seen from

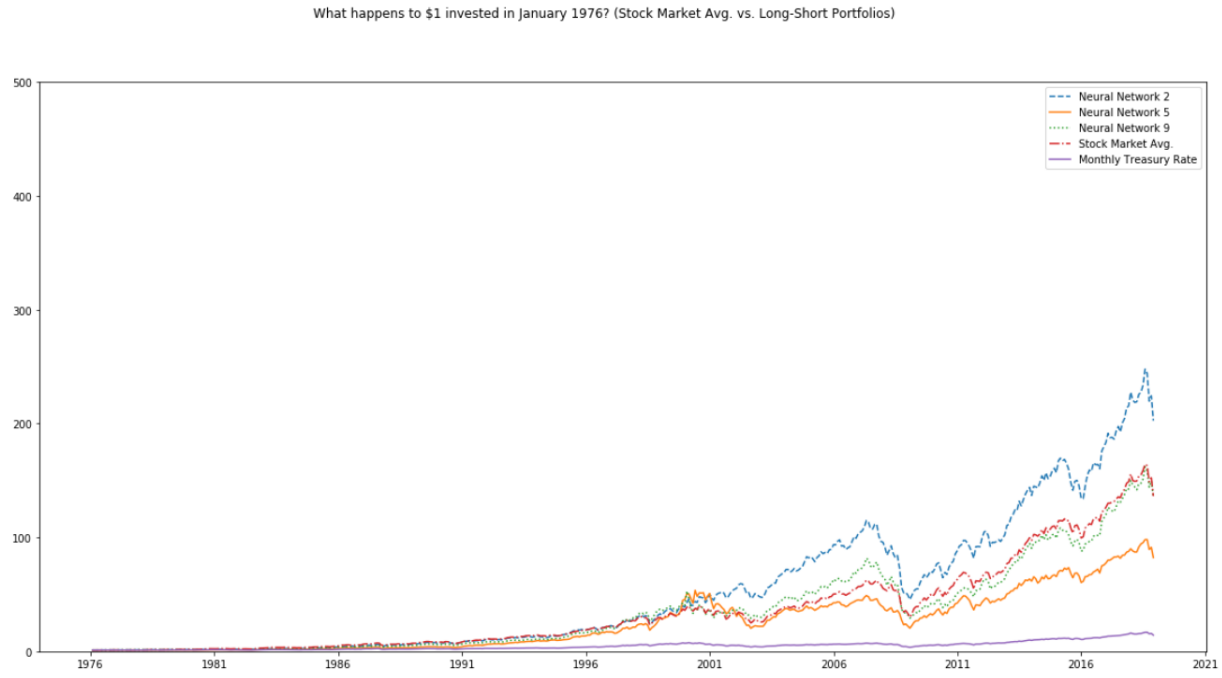


Figure 3.5: Cumulative performances of long short portfolios

the Figure 3.9 is that networks cannot predict recessions or cannot avoid huge market losses. Neural networks are forced to invest in long only portfolios, therefore stop-loss strategies can be used to prevent networks from investing when huge losses are observed at the investment period.

The maximum drawdowns (maximum loss from a peak to a trough of a portfolio) of each network and the benchmark portfolio point out that neural networks are usually exposed to higher losses compared to the benchmark poerdolio. However, as it can be seen from the Figure 3.9, the duration for recovery of losses are similar to the time span for the benchmark portfolio recovery and in the end neural networks are even able to bring higher returns compared to benchmark portfolio.

Portfolio turnovers show that neural networks are usually aggressive in their stock pickings. One reason of the high turnover rate is that neural networks' update the majority of the portfolios when financial announcements are made (each quarter) and they change

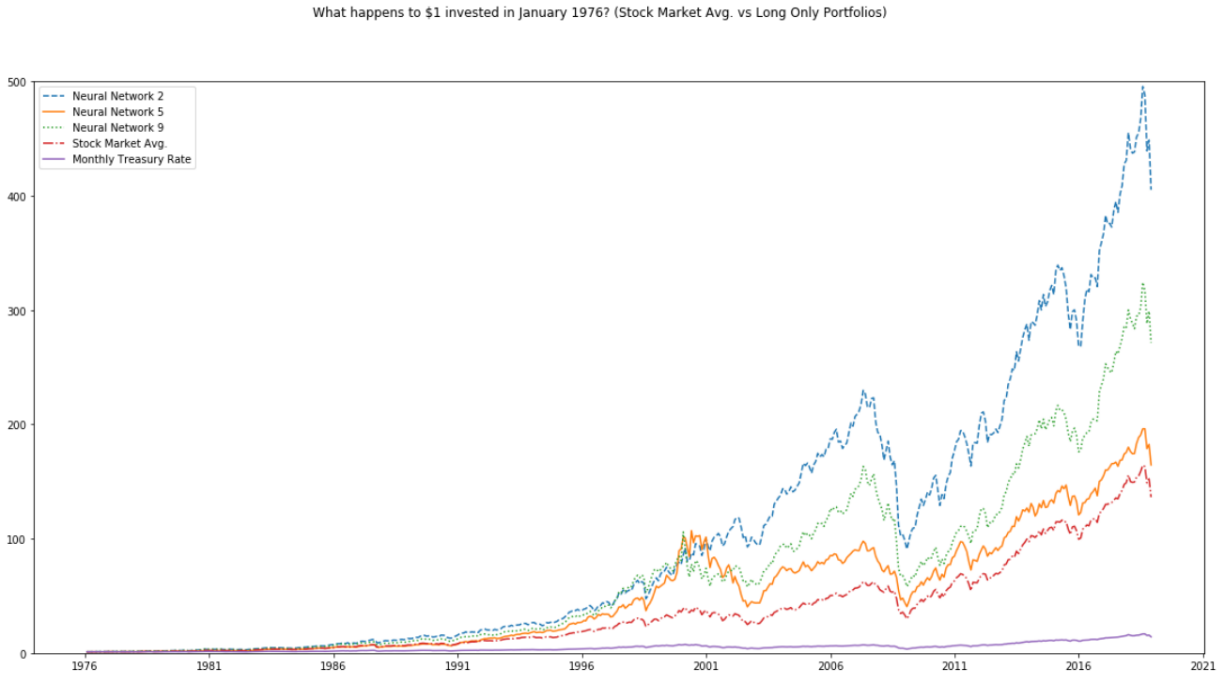


Figure 3.6: Cumulative performances of long only portfolios

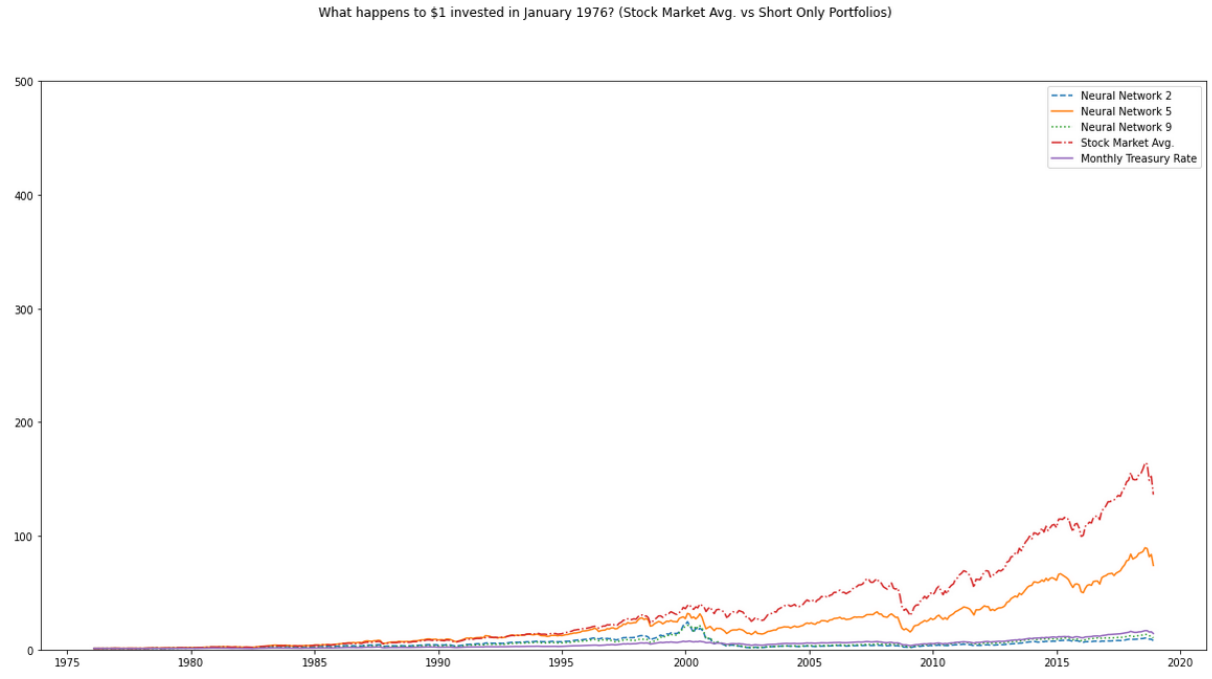


Figure 3.7: Cumulative performances of short only portfolios

NN Configuration	NN1	NN2	NN3	NN4	NN5	NN6	NN7	NN8	NN9	Stock Market
Max. Drawdown	-75.38%	-60.46%	-56.10%	-63.72%	-62.31%	-57.56%	-58.23%	-75.77%	-64.35%	-52.11%

Figure 3.8: Maximum market drawdown per network

portfolios only considering book to the market ratio and price to the book ratio between financial announcements. In the interpretability section (Section 4.2, I discuss it further that what can be the underlying reason for this behavior. The turnover ratios may seem high, however they can be tuned or neural networks can be constrained to construct the portfolio quarterly rather than monthly. However, to make a conclusion about this approach, further analyses should be made to compare the cumulative performances with constrained portfolio constructions.

A finding about the annual portfolio turnovers' figure is that best performing neural networks, two layer neural networks and ten layers neural networks have lesser turnover rates on average compared to other neural networks (Appendix E). However, their turnover ratios increase significantly during the market boom times, such as post financial crisis of 2008.

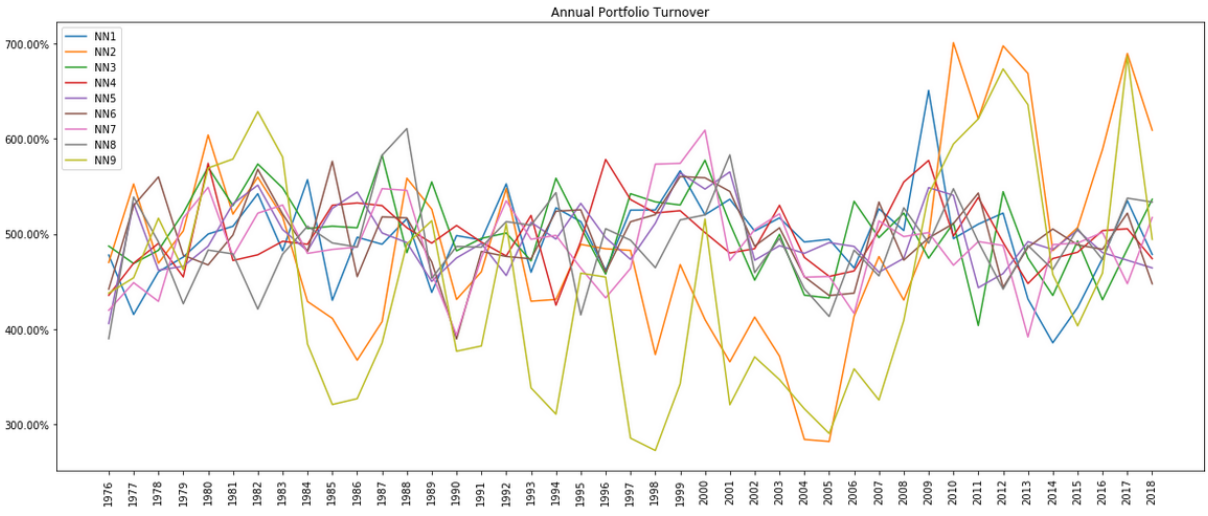


Figure 3.9: Changes in one-way portfolio turnovers

Chapter 4

Insights

4.1 Factor Based Analysis

When the portfolio returns for the top decile are regressed on three-factor model which is available on Kenneth French's data library[1][2], the measure R^2_{OOS} [3] shows that there is a high fit to the factors for both of the models.

When two layer neural network's top decile portfolio is regressed on the factors, its 87.6% of the monthly returns can be explained by the Fama-French 3 Factor model. It has a positive alpha which is 0.23% monthly and and 2.8% yearly, which means that it overperforms the benchmark factor model by 2.8% every year. Beta value is 1.03 since it consists of only equities. Coefficient of SmB is 0.22 and it can be concluded that portfolio consists of the the lower decile of the large cap stocks in terms of market cap. Coefficient of HmL signs that it is a growth fund.

On the other hand, ten layer neural network's top portfolios' 84% of monthly returns can be explained with Fama-French 3 Factor model. Its alpha is around 0.18% showing that it overperforms the benchmark factor model by 2.2% every year. The portfolio consists

[1] http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/f-f_5_factors_2x3.html
[2] http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/f-f_factors.html
[3] OOS: out of sample

	coef.	std. error	t	$P > \|t\|$	[0.025	0.975]
const	0.23	0.08	2.95	0.00	0.08	0.39
Mkt-RF	1.03	0.02	54.84	0.00	1.00	1.07
SMB	0.22	0.03	8.21	0.00	0.17	0.28
HML	0.07	0.03	2.28	0.02	0.01	0.12

	coef.	std. error	t	$P > \|t\|$	[0.025	0.975]
const	0.18	0.1	1.86	0.06	-0.01	0.37
Mkt-RF	1.02	0.02	44.04	0.00	0.98	1.07
SMB	0.43	0.03	12.72	0.00	0.36	0.50
HML	-0.08	0.04	-2.29	0.02	-0.15	-0.01

Table 4.1: Factor regression results for two layer and ten layer networks

of only equities similar to the portfolios that two layer neural network constructs which is also aligned with the stock universe. The interesting part is that coefficient of SmB is now higher than two layer neural network, pointing that portfolio usually consists of stocks have even lower market cap compared to two layer neural network. Coefficient of HmL is statistically significant, stating that portfolio is also a growth fund rather than a value fund. One conclusion that can be made is ten layer neural network expects higher returns from the companies who pass the median market cap recently (switched from being middle cap to large cap) in comparison to the two layer neural network.

4.2 Interpretability

One important problem about the neural networks is that they are usually black box models therefore it is really hard to interpret the model forecasts. There are some approaches exist such as LIME[23] and SHAP[16]. I have followed the approach of LIME, which focuses on the assumption that every non linear model behaves linear on a local scale. As an example, by looking and comparing the features of two similar observations (e.g. consider the stocks that are placed to the top decile by conditioning on neural network) one can come up with some conclusions about the behaviour of neural network. Therefore, LIME perturbs the feature set of similarly predicted data points (top decile stocks in this case) and evaluates the changes in the forecasts after perturbation of given observations. In the end, it comes up with a conclusion that how black box global model behaves on the local set of sample.

For the sample, since a neural network with same configuration is fitted with rolling window of five years, it is hard to come up with a global interpretation considering all the

features within given time frame because the weight of each feature changes over time as expected. However, I calculated the importance of features as following: for each predicted quarter, I have found the most important feature and least important feature for the top decile and normalized the feature set relatively to the most important feature and least important feature. Then I have basically summed up the importance of each feature (in a manner that, how many times feature A or feature B were picked as the most important feature over the hold out sample period) and obtained the Figure 4.1 for two layer neural network and Figure 4.2 for ten layer network.

The Figure 4.1 points out that book to the market ratio is the most positive feature for two layer neural network when it predicts the top decile. If this feature has a higher value, it means that stock is undervalued therefore neural network expects a higher return for a given stock then make predictions accordingly. On the other hand, price to the book ratio has a completely inverse effect and higher price to the book ratio weighs down the predictions of neural network.

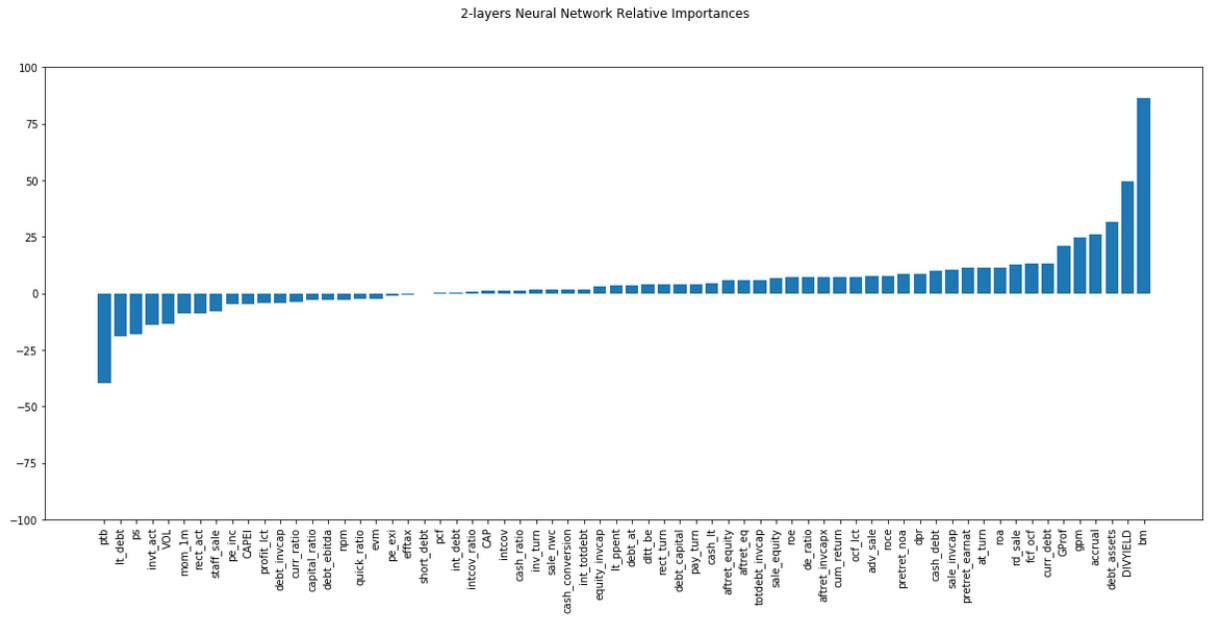


Figure 4.1: Feature importances for two layer neural network

One can ask the question that *why book to the market or price to the book features are more important?*. The answer might be that they are calculated more frequently than the rest of company characteristics. Majority of financial ratios or company characteristics are announced and calculated every quarter however book to the market or price to the book ratios are updated according to market price (stock price) since book value of the company does not change quite often. Therefore, neural networks update the majority of its portfolio each quarter and make small tweaks to the portfolio via book to the market and price to the book indicators (they sell stocks that they consider overvalued in the portfolio). The other important characteristics are dividend yield, debt to assets ratio, accruals as a fraction of total assets, gross profit margin and gross profit/total assets for top decile stocks. On the other hand, increase of long term debt to the total liabilities, price to the sales, inventory/current assets and volume features usually affect predictions negatively.

One can make many assumptions and conclusions using the interpretations. As an example, one can come up with a conclusion that higher debt shows the investment motivation of the top management, however only the short term debt is accepted as a good debt since long term debt to the liabilities ratio alters our predictions negatively. A company having high amount of inventory may show that the company is not successful in selling its products to customers therefore it may lead to problems in profitability. However, the assumptions may change depending on the market cycle therefore making a general conclusion would not be valid and accurate.

For ten layer neural network, the importance of features for the top decile slightly differ from two layer neural network. It gives higher weights to debt to assets ratio and relatively less weight to dividend yield that company has paid. Also, one interesting finding is that it relies more on momentum factor for 12 months as it can be seen from the Figure 3.9. Ten layer neural network gets affected badly between the years 2000 and 2003 because of the reason that it followed high momentum stocks during during the dot com bubble. It

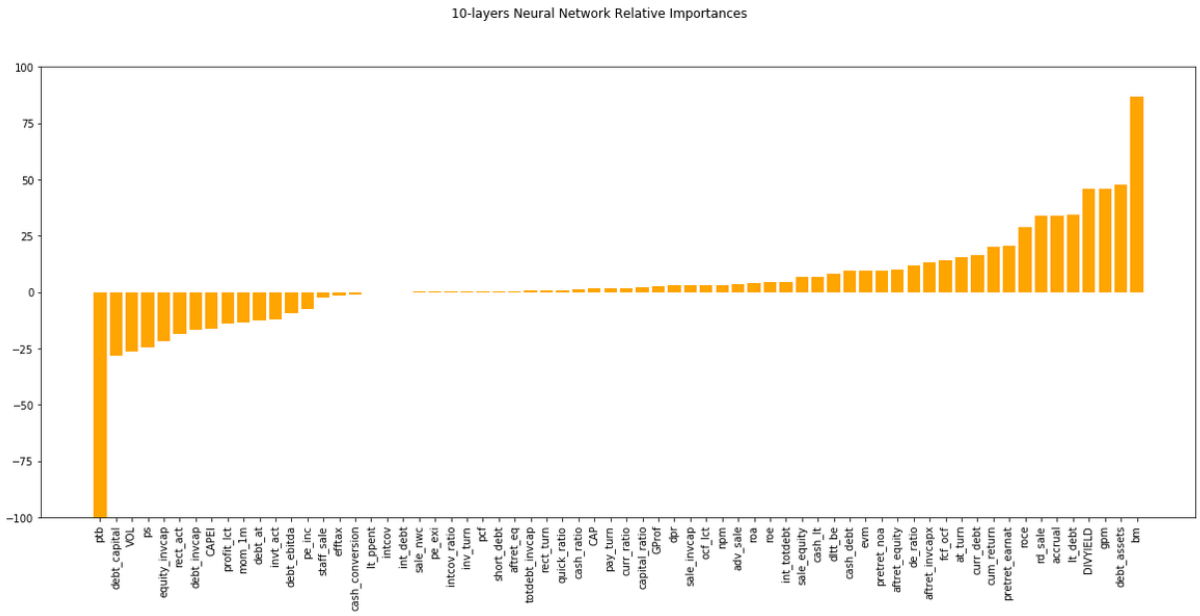


Figure 4.2: Feature importances for ten layer neural network

also gives higher weights to research and development expenses to sales ratio and return on capital employed, which can be related to operation leverage and innovation strength of the company.

The monthly changes in the weights of features with the time also provides interesting insights and would be interesting for future research. These interpretations can be used when model averaging or ensemble methods are used since different neural networks evaluate each of the features in a different way.

Chapter 5

Summary

The research in deep learning is still one of the prospective areas in the future and many questions await to be answered in the field. Therefore, it might be early to make conclusions about the performance of neural networks in general. As it is shown in the analysis, a network that can be regarded as deep neural network may have higher monthly average return compared to equally-weighted average large cap stock return. On the other hand, both shallow and deep neural networks outperform the average return of large cap stocks. These outcomes indicate that there might be still better networks in the field or different networks may evaluate the feature set completely different.

The analysis in this work is a regression analysis. The output is a continuous variable. The performances are compared using long short portfolios therefore the expected returns are discretized and top performers are compared with worst performers. The research consists of only large cap stocks where companies are regarded as usually less risky compared to other listed stocks. One future direction that I can point out is that instead of sorting and dividing expected returns into ten deciles, the network can be trained directly on discretized (already divided) realized returns. That means the discretization of returns at time t is done before the training phase rather than after training phase. Thus the problem turns to a multi class classification problem[10] with ten classes referring to the deciles.

In Chapter 2, the cost function and importance of it is discussed. The mean absolute cost is not differentiable therefore it is not the perfect selection as a cost function. On the other hand, mean squared cost is sensitive to the outliers and did not bring good results during empirical research. When the problem is changed to a multi class problem, a different cost function, cross-entropy loss function can be used to improve performance.

The research is limited to large cap stocks and the increase of the universe to mid cap stocks may increase the performance of neural networks, since the changes in the financials might be more observable and these businesses are more shaky compared to the large cap stocks. However, filtering mid cap and small cap stocks is also an advantage since transaction costs are higher for illiquid stocks.[19] In addition to that, I have used only financials that are used quarterly, therefore increasing the feature set might be a good idea to amplify strategy returns.

Neural networks can be used and ensembled to forecast cross-section of returns. And in that sense, one does not need many training samples to make use of them. The process of finding a good neural network might be a computer intensive process, however the networks that generalize well for an shorter period of time (three to five years) usually generalize in the long term as well. The neural networks that has out performed the benchmark portfolio return from 1976 to 1980 has continued to behave in the same way from 1980 to 2018. In this thesis, the aim is to challenge findings about neural networks and their application in the forecasting cross section of stock returns, accordingly it is decided to set the aim as explainability of applications rather than outperforming some benchmarks.

The cross section and the time series applications can be merged by adding feedback connections to the neural network and this is also an interesting area to discover. In this work, only feed forward neural networks are discussed where the transition of feedback is not present between the layers. However, recurrent neural networks and convolutional neural networks with feedback connections which can carry the previous market cycle

information to the recent time periods might be an interesting area to explore.

Appendix A

Data Descriptions

[1]

[1] source: https://wrds-www.wharton.upenn.edu/

Financial Ratio	Variable Name	Category	Formula
Capitalization Ratio	capital_ratio	Capitalization	Total Long-term Debt as a fraction of the sum of Total Long-term Debt, Common/Ordinary Equity and Preferred Stock
Common Equity/Invested Capital	equity_invcap	Capitalization	Common Equity as a fraction of Invested Capital
Long-term Debt/Invested Capital	debt_invcap	Capitalization	Long-term Debt as a fraction of Invested Capital
Total Debt/Invested Capital	totdebt_invcap	Capitalization	Total Debt (Long-term and Current) as a fraction of Invested Capital
Asset Turnover	at_turn	Efficiency	Sales as a fraction of the average Total Assets based on the most recent two periods
Inventory Turnover	inv_turn	Efficiency	COGS as a fraction of the average Inventories based on the most recent two periods
Payables Turnover	pay_turn	Efficiency	COGS and change in Inventories as a fraction of the average of Accounts Payable based on the most recent two periods
Receivables Turnover	rect_turn	Efficiency	Sales as a fraction of the average of Accounts Receivables based on the most recent two periods
Sales/Stockholders Equity	sale_equity	Efficiency	Sales per dollar of total Stockholders' Equity
Sales/Invested Capital	sale_invcap	Efficiency	Sales per dollar of Invested Capital
Sales/Working Capital	sale_nwc	Efficiency	Sales per dollar of Working Capital, defined as difference between Current Assets and Current Liabilities
Inventory/Current Assets	invt_act	Financial Soundness	Inventories as a fraction of Current Assets
Receivables/Current Assets	rect_act	Financial Soundness	Accounts Receivables as a fraction of Current Assets
Free Cash Flow/Operating Cash Flow	fcf_ocf	Financial Soundness	Free Cash Flow as a fraction of Operating Cash Flow, where Free Cash Flow is defined as the difference between Operating Cash Flow and Capital Expenditures
Operating CF/Current Liabilities	ocf_lct	Financial Soundness	Operating Cash Flow as a fraction of Current Liabilities
Cash Flow/Total Debt	cash_debt	Financial Soundness	Operating Cash Flow as a fraction of Total Debt
Cash Balance/Total Liabilities	cash_lt	Financial Soundness	Cash Balance as a fraction of Total Liabilities
Cash Flow Margin	cfm	Financial Soundness	Income before Extraordinary Items and Depreciation as a fraction of Sales
Short-Term Debt/Total Debt	short_debt	Financial Soundness	Short-term Debt as a fraction of Total Debt

Financial Ratio	Variable Name	Category	Formula
			term Liabilities, and Total Capital is defined as the sum of Total Debt and Total Equity (common and preferred)
After-tax Interest Coverage	intcov	Solvency	Multiple of After-tax Income to Interest and Related Expenses
Interest Coverage Ratio	intcov_ratio	Solvency	Multiple of Earnings Before Interest and Taxes to Interest and Related Expenses
Dividend Payout Ratio	dpr	Valuation	Dividends as a fraction of Income Before Extra. Items
Forward P/E to 1-year Growth (PEG) ratio	PEG_1yrforward	Valuation	Price-to-Earnings, excl. Extraordinary Items (diluted) to 1-Year EPS Growth rate
Forward P/E to Long-term Growth (PEG) ratio	PEG_ltgforward	Valuation	Price-to-Earnings, excl. Extraordinary Items (diluted) to Long-term EPS Growth rate
Trailing P/E to Growth (PEG) ratio	PEG_trailing	Valuation	Price-to-Earnings, excl. Extraordinary Items (diluted) to 3-Year past EPS Growth
Book/Market	bm	Valuation	Book Value of Equity as a fraction of Market Value of Equity
Shillers Cyclically Adjusted P/E Ratio	capei	Valuation	Multiple of Market Value of Equity to 5-year moving average of Net Income
Dividend Yield	divyield	Valuation	Indicated Dividend Rate as a fraction of Price
Enterprise Value Multiple	evm	Valuation	Multiple of Enterprise Value to EBITDA
Price/Cash flow	pcf	Valuation	Multiple of Market Value of Equity to Net Cash Flow from Operating Activities
P/E (Diluted, Excl. EI)	pe_exi	Valuation	Price-to-Earnings, excl. Extraordinary Items (diluted)
P/E (Diluted, Incl. EI)	pe_inc	Valuation	Price-to-Earnings, incl. Extraordinary Items (diluted)
Price/Operating Earnings (Basic, Excl. EI)	pe_op_basic	Valuation	Price to Operating EPS, excl. Extraordinary Items (Basic)
Price/Operating Earnings (Diluted, Excl. EI)	pe_op_dil	Valuation	Price to Operating EPS, excl. Extraordinary Items (Diluted)
Price/Sales	ps	Valuation	Multiple of Market Value of Equity to Sales
Price/Book	ptb	Valuation	Multiple of Market Value of Equity to Book Value of Equity

Financial Ratio	Variable Name	Category	Formula
After-tax Return on Invested Capital	aftret_invcapx	Profitability	Net Income plus Interest Expenses as a fraction of Invested Capital
Gross Profit Margin	gpm	Profitability	Gross Profit as a fraction of Sales
Net Profit Margin	npm	Profitability	Net Income as a fraction of Sales
Operating Profit Margin After Depreciation	opmad	Profitability	Operating Income After Depreciation as a fraction of Sales
Operating Profit Margin Before Depreciation	opmbd	Profitability	Operating Income Before Depreciation as a fraction of Sales
Pre-tax Return on Total Earning Assets	pretret_earnat	Profitability	Operating Income After Depreciation as a fraction of average Total Earnings Assets (TEA) based on most recent two periods, where TEA is defined as the sum of Property Plant and Equipment and Current Assets
Pre-tax return on Net Operating Assets	pretret_noa	Profitability	Operating Income After Depreciation as a fraction of average Net Operating Assets (NOA) based on most recent two periods, where NOA is defined as the sum of Property Plant and Equipment and Current Assets minus Current Liabilities
Pre-tax Profit Margin	ptpm	Profitability	Pretax Income as a fraction of Sales
Return on Assets	roa	Profitability	Operating Income Before Depreciation as a fraction of average Total Assets based on most recent two periods
Return on Capital Employed	roce	Profitability	Earnings Before Interest and Taxes as a fraction of average Capital Employed based on most recent two periods, where Capital Employed is the sum of Debt in Long-term and Current Liabilities and Common/Ordinary Equity
Return on Equity	roe	Profitability	Net Income as a fraction of average Book Equity based on most recent two periods, where Book Equity is defined as the sum of Total Parent Stockholders' Equity and Deferred Taxes and Investment Tax Credit
Total Debt/Equity	de_ratio	Solvency	Total Liabilities to Shareholders' Equity (common and preferred)
Total Debt/Total Assets	debt_assets	Solvency	Total Debt as a fraction of Total Assets
Total Debt/Total Assets	debt_at	Solvency	Total Liabilities as a fraction of Total Assets
Total Debt/Capital	debt_capital	Solvency	Total Debt as a fraction of Total Capital, where Total Debt is defined as the sum of Accounts Payable and Total Debt in Current and Long-

Financial Ratio	Variable Name	Category	Formula
Profit Before Depreciation/Current Liabilities	profit_lct	Financial Soundness	Operating Income before D&A as a fraction of Current Liabilities
Current Liabilities/Total Liabilities	curr_debt	Financial Soundness	Current Liabilities as a fraction of Total Liabilities
Total Debt/EBITDA	debt_ebitda	Financial Soundness	Gross Debt as a fraction of EBITDA
Long-term Debt/Book Equity	dltt_be	Financial Soundness	Long-term Debt to Book Equity
Interest/Average Long-term Debt	int_debt	Financial Soundness	Interest as a fraction of average Long-term debt based on most recent two periods
Interest/Average Total Debt	int_totdebt	Financial Soundness	Interest as a fraction of average Total Debt based on most recent two periods
Long-term Debt/Total Liabilities	lt_debt	Financial Soundness	Long-term Debt as a fraction of Total Liabilities
Total Liabilities/Total Tangible Assets	lt_ppent	Financial Soundness	Total Liabilities to Total Tangible Assets
Cash Conversion Cycle (Days)	cash_conversion	Liquidity	Inventories per daily COGS plus Account Receivables per daily Sales minus Account Payables per daily COGS
Cash Ratio	cash_ratio	Liquidity	Cash and Short-term Investments as a fraction of Current Liabilities
Current Ratio	curr_ratio	Liquidity	Current Assets as a fraction of Current Liabilities
Quick Ratio (Acid Test)	quick_ratio	Liquidity	Quick Ratio: Current Assets net of Inventories as a fraction of Current Liabilities
Accruals/Average Assets	Accrual	Other	Accruals as a fraction of average Total Assets based on most recent two periods
Research and Development/Sales	RD_SALE	Other	R&D expenses as a fraction of Sales
Avertising Expenses/Sales	adv_sale	Other	Advertising Expenses as a fraction of Sales
Labor Expenses/Sales	staff_sale	Other	Labor Expenses as a fraction of Sales
Effective Tax Rate	efftax	Profitability	Income Tax as a fraction of Pretax Income
Gross Profit/Total Assets	GProf	Profitability	Gross Profitability as a fraction of Total Assets
After-tax Return on Average Common Equity	aftret_eq	Profitability	Net Income as a fraction of average of Common Equity based on most recent two periods
After-tax Return on Total Stockholders' Equity	aftret_equity	Profitability	Net Income as a fraction of average of Total Shareholders' Equity based on most recent two periods

Appendix B

Descriptive Statistics

feature	count	mean	std	min	1%	25%	50%	75%	99%	max
VOL	50881	3,368.50	4,005.77	0.00	95.00	1,005.00	2,110.00	4,173.00	19,387.00	78,441.00
return_t1	55156	0.00	0.11	-1.00	-0.26	-0.06	0.00	0.06	0.32	1.26
CAP	55156	809,842.54	2,236,210.80	61,106.63	80,247.31	178,384.92	328,584.38	670,992.75	8,758,155.38	50,592,470.50
mom_1m	54739	0.01	0.11	-0.53	-0.24	-0.06	0.00	0.06	0.34	1.26
cum_return	52380	0.06	0.40	-0.89	-0.63	-0.19	0.01	0.24	1.37	7.41
CAPEI	50878	20.62	55.05	-685.53	2.49	9.63	13.92	23.40	101.27	5,487.54
bm	51929	0.84	0.64	0.01	0.08	0.39	0.70	1.09	3.06	8.65
evm	52151	8.45	48.41	-4,300.50	1.49	5.61	7.97	11.18	33.34	4,073.53
pe_exi	49746	15.90	15.45	-125.00	0.24	7.64	12.34	19.90	67.03	288.75
pe_inc	47722	16.60	15.58	-114.06	-7.89	8.41	12.77	20.57	69.75	293.75
ps	52250	1.31	1.52	0.02	0.09	0.43	0.83	1.55	8.60	13.56
pcf	51630	12.14	36.23	-271.55	-125.20	4.19	7.55	15.95	154.90	399.68
dpr	51971	0.15	2.71	-348.90	0.00	0.00	0.02	0.23	0.93	35.16
npm	52250	0.09	0.37	-0.63	-0.01	0.04	0.07	0.11	0.33	32.52
opmbd	52184	0.22	1.21	-0.84	0.03	0.11	0.17	0.26	0.69	149.71
opmad	52201	0.17	1.20	-0.90	0.01	0.08	0.13	0.21	0.54	149.71
gpm	52180	0.33	0.50	-76.15	0.03	0.24	0.31	0.42	0.81	1.00
ptpm	52241	0.15	0.54	-0.66	-0.01	0.07	0.12	0.18	0.51	48.22
cfm	48398	0.14	0.15	-0.48	0.02	0.07	0.11	0.17	0.55	7.01
roa	51435	0.17	0.10	-1.52	0.01	0.11	0.15	0.22	0.49	2.06
roe	51548	0.20	2.31	-26.90	-0.07	0.10	0.13	0.17	0.63	235.30
roce	47786	0.18	0.13	-0.34	0.00	0.10	0.15	0.23	0.63	3.88
efftax	51588	0.38	0.34	-19.52	-0.06	0.33	0.44	0.48	0.57	3.08
aftret_eq	51897	0.14	0.08	-1.70	-0.02	0.10	0.13	0.17	0.39	1.25
aftret_invcapx	51946	0.13	0.50	-0.29	0.00	0.08	0.10	0.15	0.47	31.11
aftret_equity	51613	0.14	0.08	-1.70	-0.02	0.10	0.13	0.17	0.39	1.25
pretret_noa	47736	0.21	0.26	-2.54	0.00	0.11	0.18	0.27	0.78	14.68
pretret_earnat	47774	0.16	0.15	-0.23	0.01	0.09	0.13	0.20	0.54	3.41
GProf	52186	0.32	0.25	-3.37	0.02	0.13	0.28	0.45	1.08	1.95
equity_invcap	52417	0.71	0.34	-7.42	0.27	0.54	0.71	0.86	1.27	4.00
debt_invcap	51959	0.29	0.20	0.00	0.00	0.14	0.28	0.44	0.73	5.62
totdebt_invcap	51833	0.43	0.45	0.00	0.00	0.22	0.38	0.56	2.25	23.93
capital_ratio	51949	0.29	0.19	0.00	0.00	0.14	0.28	0.43	0.70	1.00
int_debt	44954	0.18	1.60	0.00	0.03	0.07	0.08	0.12	1.00	94.10
int_totdebt	45507	0.07	0.04	0.00	0.02	0.06	0.07	0.08	0.20	2.35
cash_lt	48299	0.29	1.39	-1.06	0.01	0.06	0.12	0.25	2.47	63.35
invt_act	44689	0.40	0.19	0.00	0.00	0.27	0.42	0.52	0.81	2.05
rect_act	44694	0.38	0.14	0.00	0.03	0.30	0.38	0.45	0.84	0.98
debt_at	51697	0.26	0.17	0.00	0.00	0.13	0.25	0.38	0.65	1.20
debt_ebitda	51585	2.47	5.29	-210.57	0.00	0.80	1.78	3.47	13.03	400.00
short_debt	49743	0.27	0.26	0.00	0.00	0.08	0.18	0.37	1.00	1.01
curr_debt	48663	0.48	0.24	-0.94	0.08	0.28	0.46	0.64	1.00	1.68
lt_debt	51829	0.40	0.25	-0.35	0.00	0.20	0.42	0.59	0.87	1.19
profit_lct	48505	1.03	0.94	-4.21	0.12	0.58	0.86	1.27	3.56	32.02
ocf_lct	48201	0.63	3.33	-3.58	-0.35	0.22	0.46	0.75	2.77	181.34
cash_debt	51289	0.28	3.08	-1.20	-0.20	0.07	0.14	0.26	1.41	155.45
fcf_ocf	39902	-1.18	23.42	-1,425.70	-14.42	-0.66	0.02	0.41	0.94	1.01
lt_ppent	51816	7.75	27.40	-2.02	0.29	0.71	1.07	1.82	150.49	408.11
dltt_be	51565	0.60	1.32	0.00	0.00	0.16	0.39	0.79	2.42	93.82
debt_assets	52165	0.50	0.18	-0.33	0.12	0.39	0.50	0.61	0.95	1.00
debt_capital	46890	0.41	0.21	0.00	0.04	0.27	0.39	0.53	0.96	0.99
de_ratio	52120	2.09	4.39	-0.26	0.15	0.63	0.98	1.55	22.91	184.74
intcov	46249	17.86	216.50	-4,226.10	0.76	2.62	4.08	7.81	149.31	11,669.00
intcov_ratio	46186	29.50	341.63	-46.32	0.18	3.28	5.97	12.49	256.57	18,815.70

cash_ratio	44861	0.52	1.58	0.00	0.03	0.14	0.26	0.53	3.54	63.35
quick_ratio	44744	1.42	1.86	0.00	0.24	0.88	1.19	1.57	5.04	71.50
curr_ratio	48610	2.23	1.95	0.00	0.38	1.42	2.06	2.73	6.17	72.04
cash_conversion	36146	191.42	8,572.74	0.05	4.96	58.50	99.84	151.79	749.12	1,151,806.00
inv_turn	41962	22.88	1,228.86	0.05	1.08	3.00	4.48	7.51	78.85	133,596.00
at_turn	51647	1.16	0.90	0.00	0.06	0.49	1.09	1.52	5.07	8.88
rect_turn	46457	11.56	27.45	0.00	0.11	5.15	6.83	9.54	142.42	618.21
pay_turn	38440	14.04	10.25	-91.22	-0.55	8.31	12.25	17.41	48.45	267.18
sale_invcap	52222	1.81	7.02	0.00	0.21	0.81	1.47	2.12	8.95	901.77
sale_equity	51899	2.32	1.82	0.00	0.38	1.24	1.98	2.85	9.99	74.91
sale_nwc	41888	13.16	149.98	-959.46	0.61	3.36	4.88	8.06	96.65	8,353.83
rd_sale	52500	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.06	0.23
adv_sale	52169	0.01	0.03	0.00	0.00	0.00	0.00	0.00	0.12	0.85
staff_sale	52169	0.11	0.16	0.00	0.00	0.00	0.00	0.23	0.49	4.67
accrual	51555	0.01	0.07	-1.01	-0.19	-0.01	0.02	0.04	0.15	5.00
ptb	51929	2.15	2.13	0.13	0.33	0.89	1.39	2.50	10.85	17.31
PEG_trailing	33705	2.57	7.03	-0.41	0.01	0.43	1.12	2.36	30.41	183.30
DIVYIELD	55156	0.03	0.03	0.00	0.00	0.01	0.03	0.05	0.11	0.30

Table B.1: Descriptive statistics of the feature set

Appendix C

Performance Comparison

	Deciles	1	2	3	4	5	6	7	8	9	10	High-Low
Neural Network Name	Configs											
NN4	[66, 15, 15, 60, 120, 1]	-0.12%	1.30%	2.30%	1.48%	1.27%	1.59%	4.14%	2.82%	2.61%	3.77%	3.89%
NN6	[66, 15, 60, 60, 60, 60, 15, 1]	-0.11%	1.77%	1.26%	2.04%	2.47%	2.17%	2.89%	2.50%	2.89%	3.28%	3.39%
NN7	[66, 15, 15, 15, 15, 60, 60, 15, 1]	0.79%	0.97%	1.99%	2.41%	1.59%	2.17%	2.71%	2.07%	2.28%	4.16%	3.37%
NN5	[66, 60, 15, 60, 120, 15, 1]	0.58%	2.10%	1.86%	1.15%	3.25%	1.48%	2.89%	3.20%	1.68%	2.94%	2.36%
NN2	[66, 120, 15, 1]	1.24%	2.47%	1.58%	2.80%	1.59%	1.12%	1.34%	2.90%	3.18%	2.94%	1.71%
NN3	[66, 60, 15, 15, 1]	1.16%	0.61%	2.30%	2.30%	0.49%	3.46%	2.97%	3.17%	2.23%	2.47%	1.31%
NN1	[66, 15, 1]	2.10%	1.35%	1.99%	2.02%	2.17%	2.16%	2.58%	2.63%	2.03%	2.14%	0.05%
NN8&NN9	[66, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 1]	2.61%	2.24%	1.41%	2.34%	0.74%	2.10%	2.11%	2.90%	2.89%	1.79%	-0.81%

Table C.1: Portfolio returns for neural networks architectures during training phase

	Deciles	1	2	3	4	5	6	7	8	9	10	High-Low
Neural Network Name	Configs											
NN6	[66, 15, 60, 60, 60, 60, 15, 1]	-2.06%	2.56%	4.07%	5.18%	5.59%	4.78%	5.72%	5.93%	6.18%	5.07%	7.13%
	D: 0 - R:1e-05 - E: 100											
NN8	[66, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 1]	1.63%	3.72%	2.76%	3.15%	3.16%	3.53%	5.10%	3.79%	7.55%	8.46%	6.83%
	D: 0 – R:1e-05 – E: 50											
NN4	[66, 15, 15, 60, 120, 1]	-0.20%	1.73%	2.26%	1.07%	2.20%	2.50%	2.30%	2.03%	2.71%	4.60%	4.80%
	D: 0.7 – R:1e-05 – E: 200											
NN1	[66, 15, 1]	1.72%	2.86%	4.36%	4.23%	3.82%	5.43%	4.14%	4.76%	5.89%	5.54%	3.82%
	D: 0 – R:0.0001 – E: 50											
NN5	[66, 60, 15, 60, 120, 15, 1]	0.77%	0.04%	1.25%	2.09%	2.67%	2.02%	1.78%	3.68%	2.80%	4.11%	3.34%
	D: 0.3 – R:1e-05 – E: 100											
NN3	[66, 60, 15, 15, 1]	2.45%	4.01%	5.46%	4.58%	4.17%	3.42%	4.20%	4.25%	4.49%	5.71%	3.26%
	D: 0 – R:0.0001 – E: 50											
NN7	[66, 15, 15, 15, 15, 60, 60, 15, 1]	0.07%	2.45%	0.96%	2.77%	1.34%	2.26%	2.76%	2.66%	2.80%	3.11%	3.03%
	D: 0.3 – R:0.0001 – E: 50											
NN2	[66, 120, 15, 1]D: 0.7R:0.001E: 100	0.24%	1.43%	1.00%	2.34%	1.82%	1.96%	2.36%	3.73%	3.17%	3.12%	2.88%
	D: 0.7 – R:0.001 – E: 100											
NN9	[66, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 1]	6.59%	5.53%	2.59%	3.73%	3.19%	2.55%	5.15%	5.62%	3.50%	4.29%	-2.30%
	D: 0 – R:0.001 – E: 200											

Table C.2: Portfolio returns for neural networks hyperparameters during training phase

Appendix D

Fama-Macbeth Regression Results

Deciles	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
const	-0.49	-0.24	-0.07	0.00	0.02	0.08	0.07	0.10	0.24	0.23
	(0.00)	*(0.00)*	*(0.29)*	*(1.00)*	*(0.79)*	*(0.20)*	*(0.26)*	*(0.08)*	*(0.00)*	*(0.00)*
Mkt-RF	1.30	1.13	1.06	1.03	1.00	0.98	0.98	0.96	0.97	1.03
	(0.00)	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*
SMB	0.50	0.27	0.24	0.22	0.21	0.19	0.18	0.20	0.21	0.22
	(0.00)	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*
HML	-0.39	-0.07	0.10	0.15	0.20	0.21	0.20	0.19	0.17	0.07
	(0.00)	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.02)*

Table D.1: Fama-Macbeth 3 Factor regression results for two layer network

Deciles	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
const	-0.49	-0.19	-0.04	0.07	-0.03	0.03	0.09	0.17	0.15	0.18
	(0.00)	*(0.01)*	*(0.53)*	*(0.28)*	*(0.68)*	*(0.66)*	*(0.11)*	*(0.00)*	*(0.04)*	*(0.06)*
Mkt-RF	1.37	1.15	1.04	1.00	0.99	0.97	0.97	0.96	0.96	1.02
	(0.00)	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*
SMB	0.40	0.25	0.24	0.15	0.16	0.16	0.15	0.20	0.31	0.43
	(0.00)	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*
HML	-0.25	0.04	0.16	0.17	0.20	0.19	0.21	0.14	0.06	-0.08
	(0.00)	*(0.11)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.03)*	*(0.02)*

Table D.2: Fama-Macbeth 3 Factor regression results for ten layer network

Deciles	1	2	3	4	5	6	7	8	9	10
const	-0.04	-0.11	-0.07	-0.04	-0.03	0.00	0.00	0.04	0.18	0.20
	(0.70)	*(0.16)*	*(0.27)*	*(0.48)*	*(0.57)*	*(0.98)*	*(0.92)*	*(0.44)*	*(0.01)*	*(0.01)*
Mkt-RF	1.20	1.10	1.05	1.04	1.01	1.00	0.99	0.97	0.98	1.03
	(0.00)	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*
SMB	0.20	0.20	0.26	0.26	0.26	0.23	0.22	0.24	0.26	0.27
	(0.00)	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*
HML	-0.32	-0.04	0.08	0.12	0.16	0.12	0.13	0.14	0.11	0.06
	(0.00)	*(0.20)*	*(0.01)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.14)*
RMW	-0.92	-0.25	0.03	0.11	0.11	0.13	0.14	0.12	0.12	0.12
	(0.00)	*(0.00)*	*(0.40)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.20)*
CMA	-0.14	-0.11	-0.01	0.00	0.00	0.11	0.09	0.03	0.05	-0.07
	(0.06)	*(0.04)*	*(0.79)*	*(0.91)*	*(0.92)*	*(0.00)*	*(0.03)*	*(0.40)*	*(0.36)*	*(0.24)*

Table D.3: Fama-Macbeth 5 Factor regression results for two layer network

Deciles	1	2	3	4	5	6	7	8	9	10
const	-0.01	-0.12	-0.10	-0.02	-0.12	-0.1	0.00	0.13	0.18	0.28
	(0.90)	*(0.12)*	*(0.15)*	*(0.78)*	*(0.04)*	*(0.08)*	*(0.96)*	*(0.03)*	*(0.02)*	*(0.00)*
Mkt-RF	1.24	1.12	1.05	1.02	1.01	1.00	0.99	0.97	0.96	1.00
	(0.00)	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*
SMB	0.13	0.24	0.29	0.23	0.23	0.22	0.21	0.22	0.28	0.36
	(0.00)	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*
HML	-0.06	0.08	0.10	0.13	0.13	0.07	0.13	0.09	0.00	-0.13
	(0.27)	*(0.02)*	*(0.01)*	*(0.00)*	*(0.00)*	*(0.01)*	*(0.00)*	*(0.00)*	*(0.99)*	*(0.01)*
RMW	-0.89	-0.09	0.12	0.20	0.19	0.20	0.17	0.09	-0.07	-0.20
	(0.00)	*(0.01)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.00)*	*(0.06)*	*(0.00)*
CMA	-0.40	-0.15	0.03	0.00	0.07	0.19	0.09	0.04	0.04	0.01
	(0.00)	*(0.00)*	*(0.50)*	*(0.87)*	*(0.09)*	*(0.00)*	*(0.02)*	*(0.32)*	*(0.40)*	*(0.84)*

Table D.4: Fama-Macbeth 5 Factor regression results for ten layer network

Appendix E

Turnovers

Year	NN1	NN2	NN3	NN4	NN5	NN6	NN7	NN8	NN9
1976	477.73%	469.99%	487.19%	435.47%	405.7%	441.98%	419.7%	389.9%	437.7%
1977	415.27%	552.44%	468.85%	469.15%	532.13%	529.76%	448.73%	538.76%	453.85%
1978	460.18%	469.21%	482.74%	490.04%	461.87%	559.87%	429.04%	491.46%	516.53%
1979	475.65%	503.11%	521.99%	454.75%	465.95%	477.56%	516.58%	426.55%	462.38%
1980	499.86%	603.94%	570.11%	574.16%	487.03%	467.38%	548.96%	482.98%	569.23%
1981	508.00%	520.76%	529.74%	472.03%	531.6%	498.81%	476.83%	478.92%	578.67%
1982	542.23%	559.55%	573.44%	478.04%	551.24%	567.9%	521.77%	420.96%	628.54%
1983	482.25%	517.93%	548.24%	492.21%	504.52%	520.88%	530.35%	478.74%	580.71%
1984	556.98%	428.95%	505.58%	489.05%	482.13%	481.54%	479.47%	508.02%	384.21%
1985	430.17%	411.18%	508.08%	530.1%	526.87%	576.31%	483.66%	490.46%	320.63%
1986	496.5%	367.23%	506.41%	532.54%	543.96%	455.06%	486.06%	486.21%	326.83%
1987	489.1%	407.73%	582.71%	529.74%	500.99%	517.88%	547.55%	582.61%	385.33%
1988	515.92%	558.49%	480.23%	506.46%	490.82%	516.81%	545.75%	610.72%	489.41%
1989	438.52%	525.95%	554.63%	490.3%	450.05%	470.48%	458.89%	453.19%	513.84%
1990	498.21%	431.08%	482.15%	508.83%	474.77%	389.54%	393.56%	486.81%	376.66%
1991	493.59%	460.21%	495.24%	491.28%	489.9%	481.57%	475.05%	485.98%	382.21%
1992	552.56%	547.67%	500.96%	476.76%	456.22%	476.26%	534.8%	512.86%	510.72%
1993	459.63%	429.21%	471.55%	519.41%	512.13%	473.83%	494.34%	509.1%	338.08%
1994	527.43%	431.11%	558.52%	425.04%	494.53%	523.81%	498.47%	543.39%	310.54%
1995	512.96%	488.99%	507.16%	492.06%	532.11%	525.32%	464.08%	414.8%	458.74%
1996	461.7%	484.32%	457.67%	578.27%	495.99%	459.95%	432.73%	505.49%	454.62%
1997	524.98%	482.63%	542.26%	536.13%	473.24%	512.73%	463.69%	493.39%	285.3%
1998	525.11%	373.1%	533.69%	522.00%	511.17%	520.28%	573.34%	464.44%	272.3%
1999	566.39%	467.75%	530.36%	524.39%	564.2%	560.24%	574.12%	514.67%	342.33%
2000	520.05%	409.9%	577.41%	501.37%	547.1%	558.94%	609.01%	519.99%	515.63%
2001	536.41%	365.46%	510.98%	479.9%	565.17%	544.61%	472.07%	583.04%	320.37%
2002	502.96%	412.6%	451.4%	484.42%	472.32%	487.15%	504.51%	459.3%	370.71%
2003	516.84%	371.47%	499.52%	530.04%	487.57%	506.35%	521.18%	495.37%	346.76%
2004	491.45%	283.9%	435.54%	475.28%	479.53%	455.02%	454.46%	442.36%	316.13%
2005	494.29%	281.76%	432.54%	455.29%	490.81%	435.17%	455.25%	413.18%	290.15%
2006	464.14%	413.31%	534.28%	461.07%	486.83%	437.68%	416.44%	482.93%	358.21%
2007	526.42%	476.16%	496.06%	503.49%	459.66%	533.47%	513.65%	455.83%	325.36%
2008	503.51%	430.38%	521.88%	554.5%	474.33%	472.42%	497.17%	527.43%	408.87%
2009	650.81%	496.41%	474.41%	577.23%	548.65%	495.72%	501.34%	490.22%	541.52%
2010	495.00%	701.00%	511.46%	497.77%	540.57%	510.82%	467.08%	547.46%	594.5%
2011	510.21%	621.16%	403.72%	538.3%	443.46%	542.93%	491.94%	491.66%	620.84%
2012	521.87%	697.63%	544.37%	490.89%	458.31%	443.8%	487.76%	441.88%	673.39%
2013	431.52%	668.65%	474.65%	447.78%	491.8%	485.89%	391.68%	487.83%	635.77%
2014	385.5%	482.4%	435.26%	474.07%	483.55%	505.13%	488.7%	462.46%	457.41%
2015	422.53%	506.67%	492.61%	480.75%	503.79%	488.36%	490.41%	505.51%	403.34%
2016	474.26%	588.81%	430.73%	503.4%	480.37%	483.75%	501.55%	473.00%	458.81%
2017	534.86%	689.58%	483.98%	505.39%	472.29%	521.8%	447.65%	537.59%	686.57%
2018	478.36%	609.01%	536.36%	473.93%	464.27%	447.58%	517.23%	533.45%	494.28%
Average	497.02%	488.34%	503.41%	498.9%	495.1%	496.79%	488.99%	491.18%	446.46%

Appendix F

Electronic Appendix

- data (folder):
  - breakpoints (folder):
    - ME_Breakpoints.csv: includes Fama-French market breakpoints
  - factors (folder):
    - F-F_Research_Data_5_Factors_2x3.csv: Fama-French 3 Factors
    - F-F_Research_Data_Factors.csv: Fama-French 5 Factors
  - README.txt: README for data processing steps
  - final_data_lc.csv: data including only large cap stocks after preprocessing
  - market_data_19602018.csv: market data from CRSP
  - fin_ratio_19702018.csv: company characteristics data from CRSP
- preprocess (folder):
  - cleaner.py : script that is used to create final_data_lc.csv and performs preprocessing
- optimizer (folder):
  - solver_comparison.py : solver that performs optimization of neural network

  - models.py : model that initializes feedforward neural networks automatically

- results (folder):
  - interpretability_best (folder):
    - data.json: includes feature importances for 2 layer neural network
  - interpretability_second (folder)
    - data.json: includes feature importances for 10 layer neural network
  - layers (folder):
    - returns.csv: includes the result of neural network architecture search
  - one step (folder):
    - returns.csv: includes the result of neural network hyperparameter search
  - best config(folder):
    - returns.csv: includes the result of forward testing

- Forecasting_Returns_Step1.py : performs the neural network architecture search and extracts CSV file
- Forecasting_Returns_Step2.py : performs the neural network hyperparameter search and extracts CSV file
- Forecasting_Returns_Step3.py : performs the neural network forward testing for final models and extracts CSV file
- Forecasting_Returns_Step4.py : performs the interpretation of top decile and extracts JSON file
- Reproduce Tables and Plots : A Jupyter Notebook that includes methods to reproduce results in the thesis
- requirements.txt: list that lists the packages that needs to be installed

Bibliography

[1] C.M. Bishop: *Neural Networks for Pattern Recognition.* Oxford University Press, USA, 1995.

[2] D.A. Clevert, T. Unterthiner S. Hochreiter: *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs).* ICLR 2016, 2016.

[3] J. Cochrane: *Asset Pricing.* McGraw-Hill, New York, pg. 435, 2005.

[4] Eagle Alpha: *Alternative Data Use Cases*, 2018.

[5] E. Fama: *The Behavior of Stock-Market Prices.* The Journal of Business, Vol. 38, No. 1. (Jan., 1965), pp. 34-105, 1965.

[6] E.F. Fama K.R. French: *Common risk factors in the returns on stocks and bonds.* Journal of Financial Economics 33 (1993) pp. 3-56, 1993.

[7] E.F. Fama K.R. French: *A Five-Factor Asset Pricing Model.* Journal of Financial Economics 116 (2015), pp.1-22, 2013.

[8] G. Feng, N.G. Polson J. Xu: *Deep Learning in Characteristics-Sorted Factor Models*, 2018.

[9] X. Glorot Y. Bengio: *Understanding the difficulty of training deep feedforward neural networks.* Volume 9 of JMLR pg.249-256, 2010.

[10] I. Goodfellow, Y. Bengio A. Courville: *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[11] S. Gu, B. Kelly D. Xiu: *Empirical Asset Pricing via Machine Learning.* The Review of Financial Studies Vol. 33, pp. 2223-2273, 2020.

[12] K. He, X. Zhang, S. Ren J. Sun: *Delving Deep into Rectifiers:Surpassing Human-Level Performance on ImageNet Classification.* Proceedings of the IEEE internationalconference on computer vision, pg. 1026–1034, 2015.

[13] S. Ioffe C. Szegedy: *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* Proceedings of the 32nd International Conference on Machine Learning. JMLR: WCP volume 37.

[14] D.P. Kingma J.L. Ba: *Adam: A Method For Stochastic Optimization.* ICLR 2015, 2015.

[15] Y. LeCun, L. Bottou, G. Orr K.R. Muller: *In: Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700.* Springer, 2012.

[16] S.M. Lundberg S.I. Lee: *A Unified Approach to Interpreting Model Predictions.* 31st Conference on Neural Information Processing Systems (NIPS 2017), 2017.

[17] T. Mitchell: *Machine Learning.* McGraw-Hill, New York, 1997.

[18] V. Nair G.E. Hinton: *Rectified Linear Units Improve Restricted Boltzmann Machines.* ICLR 2010, 2010.

[19] R. Novy-Marx M. Velikov: *A Taxonomy of Anomalies and Their Trading Costs.* The Review of Financial Studies, Volume 29, Issue 1, pp. 104–147, 2016.

[20] M. Olson, A.J. Wyner R. Berk: *Modern Neural Networks Generalize on Small DataSets.* 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), 2018.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai S. Chintala: *PyTorch: An Imperative Style, High-Performance Deep Learning Library.* Curran Associates, Inc., 2019.

[22] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda Q. Liao: *Why and When Can Deep – but Not Shallow – Networks Avoid the Curse of Dimensionality: a Review.* International Journal of Automation and Computing volume 14, pp. 503–519, 2017.

[23] M.T. Ribeiro, S. Singh C. Guestrin: *"Why Should I Trust You?" Explaining the Predictions of Any Classifier.* KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2016, pp. 1135–1144, 2016.

[24] F. Rosenblatt: *The Perceptron: A Perceiving and Recognizing Automaton.* Report 85, 60, 1, Cornell Aeronautical Laboratory, 1957.

[25] W.F. Sharpe: *The Sharpe Ratio.* The Journal of Portfolio Management Fall 1994, Issue 21, pp.49-58, 1995.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever R. Salakhutdinov: *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* Journal of Machine Learning Research 15 pp. 1929-1958, 2014.

[27] K.O. Stanley R. Miikkulainen: *Evolving Neural Networks through Augmenting Topologies.* Evolutionary Computation 10(2): pp. 99-127, 2002.

[28] I. Welch A. Goyal: *A Comprehensive Look at The Empirical Performance of Equity Premium Prediction.* The Review of Financial Studies, Volume 21, Issue 4, pp. 1455–1508, 2008.