Aalto University
School of Business

Predicting individual stock returns using optimized neural networks

Master's Thesis

Jukka Song

Aalto University School of Business

Master of Science in Finance

Spring 2019

Aalto University
School of Business

Aalto University, P.O BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author Jukka Song		
Title of thesis Predicting individual stock returns using optimized neural networks		
Degree Master of Science		
Degree programme Finance		
Thesis advisor(s) Matthijs Lof		
Year of approval 2019	**Number of pages** 57	**Language** English

Abstract

I investigate individual monthly U.S. stock return predictability through a comparative study on neural networks and ordinary least squares benchmarks, using a predictor set of 102 lagged firm characteristics and the market return from 1980 to 2018. I find monthly out-of-sample (OOS) R^2 of 0.80% for the best neural network, confirming similar findings of marginal predictability from existing literature applying machine learning to empirical finance. OOS R^2 increases to 7.12% for the best neural network, when considering average market return predictability using market return predictions constructed bottom-up from equal-weighting individual stock predictions. I also find significant monthly four-factor alphas of 1.55% and annualized Sharpe ratios of 2.62 on long-short top-bottom decile portfolios sorted on predicted returns – not taking into account trading costs. Investigating variable importances within neural networks reveals that networks using Rectifier as their activation function focus on momentum and liquidity variables, similar to existing findings, but networks using Maxout focus on firm fundamentals and risk measures instead – a new observation for the anomalies literature. Lastly, my findings confirm that return anomalies are stronger in small stocks, and prediction performance is generally stronger during market turbulence.

Keywords machine learning, neural networks, empirical finance, anomalies

Aalto-yliopisto
Kauppakorkeakoulu

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi

Maisterintutkinnon tutkielman tiivistelmä

Tekijä Jukka Song

Työn nimi Predicting individual stock returns using optimized neural networks

Tutkinto Kauppatieteiden maisteri

Koulutusohjelma Rahoitus

Työn ohjaaja(t) Matthijs Lof

Hyväksymisvuosi 2019 **Sivumäärä** 57 **Kieli** Englanti

Tiivistelmä

Tutkin Yhdysvaltalaisten yksittäisten osakkeiden kuukausittaisten tuottojen ennustettavuutta vertaillen neuroverkkoja pienimmän neliösumman menetelmään, käyttäen 102 yritysmuuttujan ja yhden markkinatuoton datasettiä vuosilta 1980-2018. Parhaan neuroverkon testiotannan R^2 on 0.80%, varmistaen samankaltaiset löydökset ennustettavuuden tasosta tutkimuksissa, jotka soveltavat neuroverkkoja empiiriseen rahoitustieteeseen. Testiotannan R^2 nousee 7.12%:iin parhaan neuroverkon tapauksessa, kun on kyseessä keskimääräisen markkinatuoton ennustettavuus käyttäen markkinatuoton ennustuksia, jotka ovat laskettu yksittäisten osake-ennustusten keskiarvoista. Osake-ennustusten perusteella järjestetty portfolio, joka ostaa korkeinta ja myy alinta desiiliä, tuottaa 1.55% merkittäviä kuukausittaisia neljän tekijän alfaa ja 2.62 Sharpen lukua vuositasolla – huomioimatta kaupankäyntikustannuksia. Tutkittaessa ennustajien tärkeyttä neuroverkoissa huomataan, että verkot, jotka käyttävät Rectifier-aktivointifunktiota keskittyvät momentum ja likviditeetti muuttujiin, kun taas verkot, jotka käyttävät Maxout-aktivointifunktiota keskittyvät yritysten perustekijöihin ja riskin mittareihin, tuoden uuden löydöksen anomalioiden kirjallisuuteen. Löydökseni vahvistaa, että tuottoanomaliat esiintyvät vahvempina pienissä osakkeissa, ja ennustettavuus on yleisesti parempaa markkinoiden ollessa turbulentteja.

Avainsanat koneoppiminen, neuroverkot, empiirinen rahoitustiede, anomaliat

Contents

1. Introduction

The asset pricing literature has found significant cross-sectional stock return predictability (Avramov and Chordia (2006), Rapach and Zhou (2013), Jordan, Vivian and Wohar (2014), and Dangl and Halling (2012)). In parallel, hundreds of anomaly variables that produce excess risk-adjusted returns and correlate with firms' subsequent stock returns have been discovered (Green, Hand, and Zhang (2013) and Hou, Xue, and Zhang (2017)). As traditional linear regression methods deal poorly with high-dimensional problems and model uncertainty, machine learning models that excel in such environments have been applied to enhance stock return predictability and gain new insights into which factors really provide independent information on stock returns (e.g. Gu, Kelly, and Xiu (2018)). My paper is largely motivated by the proliferation of anomaly variables discovered in the asset pricing literature, and the economic possibilities of understanding that information to predict stock returns with modern machine learning models.

In this paper, I study the predictability of individual stock returns based on a large set of fundamental predictor characteristics using neural networks. I use a set of 102 firm characteristics as Green et al. (2017) and the market return in a panel data to predict one-month-ahead U.S. returns for the sample period 1980 to 2018. My main focus is comparing various rigorously optimized neural network models to the baseline OLS regression predictions.

The paper contributes to the empirical asset pricing literature in three ways. First, I analyze the degree of stock return predictability for individual stocks, comparing machine learning to traditional methods. I evaluate predictability through statistical measures and economic profitability. From the statistical perspective, I find one of three chosen neural networks to consistently outperform OLS benchmarks across the measures with an average out-of-sample R^2 of 0.80%, average correlation between predicted and realized returns of 9.66%, and average directional accuracy of 52.96%. Across the three statistical measures, neural networks outperform in correlation and directional accuracy, but the results are more mixed for out-of-sample R^2. From the economic perspective, I construct stock portfolios sorted on predicted returns and evaluate the profitability of a long-short strategy on the top and bottom deciles. In terms of monthly returns and annualized Sharpe ratio, neural networks greatly outperform OLS benchmarks, with neural network portfolios returning as much as 1.51% on average per month, with a Sharpe ratio of 2.62, not accounting for trading costs. I also evaluate aggregate market return predictability by comparing the predicted and realized equal-weighted average returns of market portfolios constructed from individual stock returns. I find that the results mirror those of individual stock predictions, where a few neural networks outperform in R^2, and generally outperform in directional accuracy. Notably, the monthly out-of-sample R^2 at a portfolio level increases to an average of 7.12% for the best neural network. Upon further analysis, most of the predictability stems from the lagged market return predictor and its nonlinear interactions with firm predictors. Models trained without the market return predictor generally fail to achieve positive out-of-sample R^2. Additionally, upon analyzing result differences in 500 largest and smallest stocks, I find that all predictability measures confirm that predictability is stronger in small stock than big stocks (e.g. Green et al (2017)). I also analyze the time series of rolling out-of-sample R^2 and cumulative returns of the neural network portfolios, which reveal that prediction performance is generally better during times of market turbulence, aligning with results of Dangl and Halling (2012).

Second, I identify predictor variables that contribute the most to return prediction in neural networks, contributing to the return anomalies literature where hundreds of factors have been found as statistically significant predictors of cross-sectional returns. Since neural networks learn the relation between predictors and predictions without a priori information from the researcher and allow for nonlinear complex relations, it may identify predictive power eluding conventional linear methods. I follow the Gedeon (1997) method of calculating relative variable importance based on predictor weights in the neural network and find two different results from two different neural networks: one confirms earlier findings of Gu et al. (2018) and Messmer (2017), where momentum (e.g. 1-month or 6-month momentum) and liquidity (e.g. return volatility or zero trading days) variables are most important, the other finds new evidence of the importance of firm fundamentals (e.g. cash or volatility of cash flow) and risk-based measures (e.g. beta or idiosyncratic volatility).

Third, I provide a rigorous neural network comparison, by conducting extensive random grid searches for optimal model topologies and controlling for inherent randomness of neural networks by repeating the training of each optimized network 100 times and reporting summary statistics. I find that there are significant differences in neural network performance in the noisy data environment of stock returns. Neural networks using the popular Rectifier as their activation function produced highly varied results when measured by out-of-sample R^2, with standard deviations 0.42% compared to 0.12% of the next best neural network. Rectifiers also became unstable the most often during random grid search, where the majority of deeper Rectifier networks failed in training. The other two neural networks using so called Maxout and Tanh activation functions performed more stable, generating results in a denser spread. Additionally, based on relative variable importance per neural network, Rectifier and Maxout models prioritize very different data to extract predictive information. Rectifiers use the momentum and liquidity, while Maxouts use firm fundamentals and risk measures. Thus, this paper provides researchers looking to apply neural networks to empirical finance with detailed insights on model optimization choices and resulting performance.

2. Literature review

In the asset pricing literature, results for stock return predictability are mixed. Stock return predictability has been examined both through cross-sectional regressions that seek to model differences in expected returns across stocks on firm-level characteristics, and through time-series regressions on aggregate market returns on macroeconomic variables. There is an enormous literature (see for example Rapach and Zhou's (2013) overview paper) documenting how various variables have predictive power of aggregate stock returns, such as the popular dividend-price ratio (Fama and French (1988)). However, the evidence for such predictability is predominantly in-sample. Goyal and Welch (2008) show in their influential paper for numerous economic variables in the literature that out-of-sample equity premium forecasts fail to consistently outperform the simple historical average benchmark forecasts in terms of mean square forecast error (MSFE).

2.1. *Predictability of stock returns*

Stock returns inherently contain a large unpredictable component, so that even the best forecasting models can explain only a small part of returns. In addition, market efficiency requires that when successful forecasting models or factors are discovered, they will also be adopted by others, which leads to the eventual disappearance of such predictability. On this point, McLean and Pontiff (2016) document a 58% decrease in post-publication returns based on a study of 97 variables and their out-of-sample post-publication return predictability. The maximum level of predictability in terms of monthly R^2 is 8%, according to a loose analysis by Rapach and Zhou (2013) of the typical predictive regression model. In practice, forecasting results for stock market aggregate returns are generally in the 1% neighborhood for in-sample tests, and lower out-of-sample: Fama and French (1988) report monthly R^2 statistics of 1% or less for predictive regression models based on dividend-price ratios, Zhou (2010) reports 1% or less for individual predictive regressions based on 10 popular economic variables, and Jordan et al. (2014) report over 2% out-of-sample for several European countries based on a combination of fundamental, macroeconomic and technical variables. These papers calculate R^2 statistics comparing predicted returns to the historical average prediction. In comparison, a study by Gu et al. (2018) using machine learning algorithms to predict individual stock returns use a R^2 statistic that compares predicted returns to a zero-return prediction, because the historical average comparison underperforms the zero-return prediction when considering individual stocks. Their study yields monthly R^2 values of less than 1% at best. However, the literature agrees on that even small improvements in prediction accuracy can result in significant economic benefits (Campbell and Thompson (2008), Rapach and Zhou (2013)).

2.2. *Proliferation of anomaly factors and the curse of dimensionality*

The proliferation in the number of published anomaly variables that appear significant in explaining cross-sectional stock returns has led to the questioning of the validity of traditional methods in evaluating whether new characteristics really provide independent information about average returns, as asserted by Cochrane's (2011) presidential address. Since the establishing of the CAPM of Sharpe (1964) and Lintner (1965), discoveries have been published on significant factors such as the size (Banz (1981)), value (Rosenberg, Reid, and Landstein (1985), Fama and French (1992)), and momentum (Jegadeesh and Titman (1993)) that have become mainstays in the widely used Fama and French (1993) three-factor and Carhart (1997) four-factor models. More recently in the past decades, the discovery of new anomaly variables has accelerated, resulting in hundreds of anomaly factors being published, as documented by Green, Hand, and Zhang (2013), where they analyze 333 return predictive signals, or Hou, Xue, and Zhang (2017), who conduct a replication study on 447 anomaly variables. However, these aforementioned critical studies are unified in their conclusion that most of the analyzed predictive signals do not appear significant, or their scale is much lower than originally published. Thus, there is serious evidence of p-hacking and data snooping in empirical finance (Chordia, Goyal, and Saretto (2018)).

To tame the factor zoo, many econometric methods have been applied to deal with the curse of dimensionality. For example, Giglio and Xiu (2016) and Kelly et al. (2017) use dimension reduction methods to test and estimate factor pricing models. Kozak et al. (2017)

construct a stochastic discount factor that summarizes the joint explanatory power of many cross-sectional stock return predictors. Freyberger et al. (2017) use the adaptive group LASSO to select characteristics to approximate a nonlinear function for expected returns. Feng et al. (2017) propose a model-selection method to systematically evaluate the contribution to asset pricing of any new factor. Sun (2018) uses a newly developed machine learning tool to regularize a large set of factors, grouping highly correlated factors while shrinking off the useless ones simultaneously. Though these attempts have shown promise and a reduction in the number of significant factors, there appears to be no consensus. This has led to Kozak et al. (2017) conclusion that the quest to summarize the cross-section of stock returns with sparse characteristics-based factor models to be ultimately futile, as there is not enough redundancy among the many return predictors.

2.3. *Econometric methods to deal with model uncertainty and parameter instability*

In addition to addressing high dimensionality, more recent strategies have improved forecasts by addressing model uncertainty and parameter instability concerns that traditional linear regression models ignore. Model uncertainty recognizes that a forecaster does not know the optimal model specification or the corresponding parameter values. Parameter instability recognizes that the model and its parameters may change over time. Rapach and Zhou (2013) describe four strands of forecasting strategies that deliver statistically significant out-of-sample gains: economically motivated model restrictions (Campbell and Thompson (2008)), forecast combination (Rapach et al. (2010)), diffusion indices (Neely et al. (2014)), and regime shifts (Dangl and Halling (2012)). Additionally, parameter instability has been investigated with time-varying coefficients, where Dangl and Halling (2012) find that models with time-varying coefficients dominate models with constant coefficients. They also find that stock return predictions are closely linked to business cycles, where stronger accuracy is achieved during recessions.

2.4. *Machine learning methods suitable for high dimensionality, model uncertainty and parameter instability*

Considering the discussed issues of high dimensionality, model uncertainty, and parameter instability, machine learning techniques provide attractive options. As described by Gu et al.

(2018), first, the task in asset pricing, of trying to understand the cross-section of asset returns or the aggregate market risk premium, is fundamentally about prediction, and machine learning methods are largely specialized for prediction tasks. Second, with its wide range of methods from linear models to regression trees and neural networks, machine learning is explicitly designed to deal with model uncertainty and complex nonlinear functions. Third, in the problem of high dimensionality due to hundreds of predictor variables, machine learning algorithms are well-capable of reducing dimensions and condensing redundant variation among highly correlated predictors.

The best performing machine learning models for stock return prediction have generally been neural networks and regression trees (Gu et al. (2018), Messmer (2017), Cao et al. (2005)), and improving on them, ensemble methods that pool and combine predictions from several different machine learning models (Tsai et al. (2011)).

Gu et al. (2018) conduct a comparative analysis on the performance of various machine learning algorithms in individual stock return prediction. Included are linear regression, generalized linear models with penalization, dimension reduction via principal components regression and partial least squares, regression trees (including boosted trees and random forests), and neural networks. Their analysis includes a large set of individual stocks over 60 years with a set of roughly 100 predictive variables. They find that allowing for nonlinearities substantially improves predictions, where trees and neural nets improve return predictions from a benchmark monthly stock-level R^2 of 0.16% to R^2's between 0.27% and 0.39%. The benchmark is a panel regression of individual stock returns onto the lagged size, book-to-market, and momentum variables. When they run the analysis for bottom-up portfolio-level return forecasts of the S&P 500 index from stock-level forecasts, the monthly R^2 for trees and neural networks is 1.39% and 1.80%, compared to the benchmark of -0.11%. The portfolio level prediction averages out more of the stock-level noise while boosting signal strength.

Messmer (2017) trains deep feedforward neural networks on a set of 68 firm characteristics to predict the U.S. cross-section of stock returns, finding that neural network long-short portfolios can generate attractive risk-adjusted returns compared to a linear benchmark. Cao et al. (2005) find the same result with Chinese stocks, where neural network models outperformed linear models. Tsai et al. (2011) use classifier ensemble methods to predict whether quarterly stock returns are positive or negative and find that classifier ensembles outperform single neural network models, and that the return on investment is better for all tested machine learning models than the buy and hold strategy. For the reasons, further analysis on neural networks and their performance in stock return prediction is valuable and interesting. The goal of this paper is to provide a more rigorous examination of neural networks and analyze the prediction performance of optimized networks and infer information on which predictors are the most important.

3. Data

I follow Green et al.'s (2017) influential paper in using the dataset from their website[1] with 102 firm characteristics and monthly individual stock returns from the time period 1980 to 2018, totaling 38 years, with an average of around 4000 stocks per month. I also add one-month lagged market return as a predictor. The firm data is entirely calculable from CRSP, Compustat, or I/B/E/S data, and begins in 1980 due to most characteristics becoming robustly available only then. Details of the characteristics including sources, definitions, and percentage missing of each characteristic is provided in Table A1 in the appendix.

Green et al. (2017) begin the data creation with all firms with common stock on the NYSE, AMEX, or NASDAQ that have a month-end market value on CRSP and a nonmissing value for common equity in their annual financial statements. Data is integrated across Compustat, I/B/E/S, and CRSP and characteristics are computed and aligned in calendar time. For each month *t*'s return they calculate characteristics as they were at the end of month *t-1*. Also, annual accounting data is assumed to be available at the end of month *t-1* if the firm's fiscal year ended at least six months before the end of month *t-1*, and quarterly accounting data to be available if the fiscal quarter ended at least four months before the end of month *t-1*. Green et al. (2017) also include delisting returns in the monthly stock returns taken from CRSP, delete 20 observations that have a monthly return less than -100%, and set blank values of analyst following (*nanalyst*) to zero. The dependent variable is monthly returns in excess of the risk-free rate (*XRET*). Excess return is calculated as a stock's monthly return minus the risk-free rate. The market return variable is based on the Wilshire 5000 Total Market Full Cap Index and the risk-free rate is the "3-Month Treasury Bill: Secondary Market Rate". Both are retrieved from the economic research data base at the Federal Reserve Bank at St. Louis.

[1] Dataset retrieved using the SAS code on their website: https://sites.google.com/site/jeremiahrgreenacctg/home

4. Methodology

The methodology applied in this paper consists of five steps. First, I split the entire data set into training, validation, and testing data. Second, I find the optimal neural network topologies used in further analysis through random grid searches. Third, I train three optimized neural networks 100 times each to produce 300 neural networks models to control for randomness among different iterations of the model, and compute three linear OLS benchmarks to compare against. Fourth, I generate predictions using the 300 neural networks and 3 OLS benchmarks and analyze average neural network prediction performance statistically and economically. Fifth, I analyze relative variable importance to infer which predictors are most impactful to the predictions.

4.1. *Constructing training, validation and testing data sets*

I split the data set into three subsets: training, validation, and testing sets. The training data set is used for training the model, where the model calculates the optimal parameter coefficients for the predictor variables. The validation data set is used during training (both in random grid search and training the optimized models) to enable "early stopping" regularization reducing overfit. The testing data is a true out-of-sample test of the tuned models, using data that has not been used for model training or validation.

Before creating the dataset splits, the characteristics are first winsorized at the 1st and 99th percentiles of their monthly distributions. Then, I split the data into a training set with 75% of all data (1980-2006) and testing set with 25% of all data (2007-2018). I shuffle the order of the training data set, as shuffling training data is considered best practice for neural networks to make learning more robust and less susceptible to time-specific outliers (Brownlee (2016)). Shuffling the training data breaks up the temporal order and enables the model to better learn information from the entire panel. I standardize the training dataset to zero mean and unit variance and store the standardization parameters (mean and standard deviation) for each characteristic. The reason to standardize the variables is both to deal with missing values, and to make neural network model training faster and reduce the chances of getting stuck in local optima (Sarle (2002)). Then, the testing set is scaled according to the stored parameters of the 75% training set. Scaling the testing dataset according to stored standardization parameters from the training set is done to avoid introducing future information of variable distributions into the testing dataset. After standardizing and scaling the data, I set all missing values to the post-standardized mean of zero as Green et al. (2017).

Lastly, I create the validation dataset with 25% of all data by splitting a portion off from the training set. The validation dataset is thus also in a randomized order. Choosing a shuffled validation dataset instead of a time-period specific set (e.g. training using data from 1980 to 1997, then validation from 1998 to 2006) attempts to strike a reasonable compromise between trying to avoid a "lucky split" and being a representative out-of-sample test. A "lucky split" would mean, for example, that data in 1998 to 2006 could be unrepresentative of the general data, which would push the models to overfit to that period and result in poor true out-of-sample prediction performance later. The randomized validation dataset contains data from different time periods, so is more representative of general data, but is less "out-of-sample", sacrificing some its representativeness as an out-of-sample test. However, none of the validation data is used during training, so it is still data the model has not seen. Also, since the feedforward networks I train do not explicitly learn temporal information from the order of the rows and the data is already in panel form, the order of data is less significant. I consider the potentially better k-folds cross-validation method but decide not to use it due to high computational cost and H2O not supporting cross-validation if recursive model refitting is done (which I do during the true out-of-sample testing).

The result is a shuffled and standardized training dataset with 75% of all data – from which I split a 50% training set and 25% validation set for hyperparameter tuning – and a temporally ordered and scaled testing dataset with 25% of all data.

4.2. *Over-arching function and methods*

We follow Gu et al.'s (2018) description of the over-arching functional form of the models utilizing panel data. In constructing predictive models for stock returns based on lagged information, the primary objective of the models is to minimize the mean squared forecast error (MSFE). Some regularization is imposed on the models through variations of the MSFE objective, in order to avoid overfitting and to improve out-of-sample predictive performance.

In its most general form, an asset's excess returns can be described as:

$$r_{i,t+1} = E_t\left(r_{i,t+1}\right) + \varepsilon_{i,t+1}$$

where

$$E_t\left(r_{i,t+1}\right) = g^*\left(z_{i,t}\right).$$

Stocks are indexed as $i = 1, \dots, N_t$ and months as $t = 1, \dots, T$. The predictive models construct a representation of $E_t(r_{i,t+1})$ as a function of predictor variables that minimizes the out-of-sample MSFE between the predicted return and realized return $r_{i,t+1}$. The predictors are denoted as the P-dimensional vector $z_{i,t}$ (lagged information), and the conditional expected return $g^*(\cdot)$ is assumed to be a flexible function of these predictors. Thus, the function maintains the same form over time and across stocks (doesn't depend on i or t) and doesn't use information from the history prior to t or from individual stocks other than the i^{th} in predictions. This means that the previously described shuffling of training data is acceptable, as the models essentially consider each row of data independent and identically distributed (iid). A limitation of this functional form may be the lack of year-fixed-effects, where for example market capitalization is generally lower in the training data than in the testing data time period. However, annual refitting of the models controls this to an extent.

The neural network models are refitted annually to produce predictions for the next year. This strikes a reasonable compromise between computational expensiveness of refitting every month and simulating a real-life modeling process and follows Gu et al. (2018). For neural networks, model refitting is done using "checkpointing"[2], where an existing model's training process is resumed using only next year's data as an input, resulting in the existing model updating its parameter weights accordingly. The algorithm uses the same validation data for model regularization during each checkpointing iteration.

For robustness of the results, I will include prediction results from different subsets of data related to firm size, such as all stocks and top or bottom 500 based on market capitalization. To evaluate statistical prediction performance, I will use out-of-sample R^2, correlation between predicted and realized returns, and directional accuracy. To evaluate the economic value of predictions, I will perform bottom-up estimates simulating stock index predictions, where I predict the all stocks' and top/bottom 500 largest/smallest stocks' equal-weighted average returns and compare them to their actual equal-weighted average return. This bottom-up approach adds an equity premium prediction aspect to my thesis, demonstrating the degree of equity premium predictability based on individual stock return predictions. Additionally, I will compare the return and volatility of stock portfolios sorted on predicted returns that buys the top decile and shorts the bottom.

[2] See description of the implementation of checkpointing in H2O: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/algo-params/checkpoint.html

4.3. *Benchmark ordinary least squares linear regression*

As a benchmark to the neural network results, I use three different OLS benchmarks, with lagged market return (*mkt_ret_lag*), market capitalization (*mve*), book-to-market (*bm*), and 6-month momentum (*mom6m*) as predictors. OLS-3 is a linear regression on the full training data sample that makes predictions on the entire testing data. OLS-3-R is an annually refit linear regression that annually adds the next year data to its training data, representing the methodologically closest comparison to the neural networks. OLS-FM follows the procedure of Fama and French (2006), in which first coefficient estimates are produced using monthly cross-sectional regressions of firm-level returns on lagged values of predictors (dropping market return as it is the same across firms in a given month), and then those coefficient estimates are used to calculate predicted one month ahead returns using current values of predictors. Notably, OLS-FM models can only utilize information from the previous month and not the entire panel, unlike the previous two models.

4.4. *Neural networks anatomy*

In the computer science or machine learning field, neural networks are arguably the most powerful modeling device, being able to approximate any smooth predictive association. Neural networks were designed as conceptual models of human brain activity, where input signals that arrive in neurons are weighted by dendrites according to their relative importance, processed by the cell body and passed on down the axon to the next neuron through synapses.

The brief description of neural networks follows Krauss et al. (2017). A neural network has an input layer equal to the predictors, one or more hidden layers of nodes that interact and nonlinearly transform the predictor signals, and an output layer that aggregates hidden layers into an outcome prediction. All layers are composed of nodes ($x_i^{(L)}$), the basic units of neural networks. In feedforward networks that I use, each node in a previous layer L is fully connected to all nodes in a subsequent layer $L+1$ via directed edges, each representing a certain weight ($w_i^{(L)}$). Also, each non-output layer of the network has a bias unit $b^{(L)}$, serving as an activation threshold for the nodes in the subsequent layer. As such, each node of layer $L+1$ receives a weighted combination $\alpha^{(L)}$ of the $n^{(L)}$ outputs of the neurons that are connected to it from the previous layer L as input:

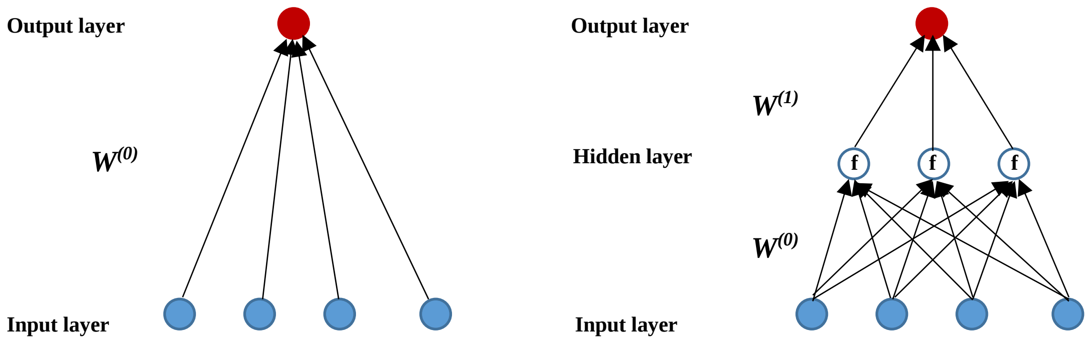
$$\alpha^{(L)} = b^{(L)} + \sum_{i=1}^{n^{(L)}} w_i^{(L)} x_i^{(L)}$$

Using the left example in Figure 0, a neural network with zero hidden layers represents a linear regression model:

$$g(x) = b^{(0)} + \sum_{i=1}^{4} w_i^{(0)} x_i^{(0)}$$

Figure 0. Simple neural network visualizations.

The left figure shows a neural network with no hidden layers, essentially a linear regression. The right figure shows a neural network with one hidden layer (3 nodes). The arrows represent the weights (w_i) between signals connecting nodes, and ***W*** represents the *n*-dimensional vector of weights, where *n* is the number of outputs from the previous layer. In the hidden layer, a nonlinear activation function *f* transforms the inputs before passing them on to the output.



What enables neural networks to learn nonlinear relations and interactions between predictors is the connectedness of all predictors to the hidden layer nodes and the activation function that is used to transform the aggregated signal before passing it on. Using the right example in Figure 0, the left-most node in the hidden layer transforms its four inputs into an output as:

$$x_1^{(1)} = f(b^{(0)} + \sum_{i=1}^{4} w_i^{(0)} x_i^{(0)})$$

The output from each node in the hidden layer are then linearly aggregated into the output prediction:

$$g(x) = b^{(1)} + \sum_{k=1}^{3} w_k^{(1)} x_k^{(1)}$$

Applying the same logic to deeper and wider models, the functional form becomes a much more nested function aggregating all weight and bias matrices of each layer and node. This is why neural network models are infamously difficult to interpret, unlike a simple linear regression. Machine learning is implemented by adapting the entire collection of weight matrices in order to minimize the error on the training data. The process by which the weights are adapted is called the training algorithm, discussed in detail in the next section, among other parameters that control the regularization of neural networks that are prone to overfitting.

4.5. *Neural network hyperparameters*

Neural network hyperparameters describe the set of topology and regularization technique parameter choices when initializing a neural network. Hyperparameter value choices are non-trivial and "more of an art than science" (Zhang, Patuwo, and Hu (1998)). Thus, neural networks require rigorous optimization through grid search to enable the researcher to find an optimal set of hyperparameter values that work for the specific data in question. In this section I describe the most important hyperparameters analyzed in this paper and defer the description of my grid search process to the next section. The main neural network characteristic choices are **activation function, network topology, training algorithm, learning rate,** and **loss function**. The main regularization techniques used are **early stopping, dropout,** and **L1/L2 regularization**.

First, the **activation function** transforms a neuron's net input signal into a single output signal to be broadcasted further in the network. There are many potential choices for the activation function (such as sigmoid, tanh, rectifier, etc.) (Lantz (2015)). A simple linear activation function with no hidden layers would essentially be an OLS linear regression. One popular example could be the sigmoid function, $f(x) = \frac{1}{1+e^{-x}}$, where the sum of input signals x, determines an output value in the range of 0 to 1. However, the sigmoid function has fallen out in favor of better performing functions in recent literature, such as the rectifier (Gu et al. (2018)). The R package I use, H2O, offers three popular activation function alternatives: Rectifier, Tanh, and Maxout. The activation function choice is applied to all hidden layers. During hyperparameter tuning, among other variables, I compare the performance of the three activation functions using random grid search, where random combinations of given hyperparameters are used to build different neural network models.

The Rectifier is defined as the positive part of its argument:

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{otherwise} \end{cases}$$

where x is the input to a neuron. It is the most popular activation function as of 2017 (Ramachandran et al. (2017)), and its advantages include 1) computational efficiency through only requiring a max(0,x) function and being capable of outputting true zero values unlike tanh functions, and 2) linear behavior, which makes the network easier to optimize and much less likely to encounter vanishing gradient problems, making the model more stable.

The Tanh is defined as:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Its benefits include being zero-centered, making it easier to model inputs that may have strongly negative or positive value. Finally, the most recently developed Maxout activation function is defined as:

$$f(x) = \max(w_1^T x + b_1, w_2^T x + b_2)$$

and is a generalization of the Rectifier. It enjoys the benefits of the Rectifier while fixing the so-called problem of “dying Rectifiers”, where a “dead” neuron always outputs zero for any input for the rest of the training process. The trade-off for Maxout is the doubling of parameters for each neuron, requiring a higher total number of parameters to be trained.

Second, the **network topology** refers to the chosen structure of number of layers, whether information in the network is allowed to travel backwards, and the number of nodes within each layer. Generally, the larger and more complex networks are capable of identifying more subtle patterns, but for example Gu et al. (2018) show that a network with three layers performs better than one with five layers for stock return prediction with a large set of predictor variables. In my work I use the grid search method that varies the number of layers as well as nodes within each layer, to find the optimal model based on out-of-sample performance. The most commonly used feedforward networks only allow information to flow forwards in a network, and that is the topology I will be using.

Third, the **training algorithm** refers to the process by which the network is trained using input data. Training a neural network means how the network’s connection weights are adjusted to reflect patterns observed from data. Modern training algorithms are variations on a strategy of back-propagating errors, known simply as backpropagation. In its most general form, backpropagation iterates through many cycles (known as epochs) of two processes, forward and backward phases. In the forward phase an output signal is produced based on the current weights in the network. Then, in the backward phase the error of the output signal compared to the true realized value is propagated backwards in the network to adjust the weights between neurons to reduce future errors. There are many different training algorithms to choose from, such as the stochastic gradient descent, conjugate gradient or Newton method (Quesada (n.d.)). I use the parallelized stochastic gradient descent offered by the H2O R-package, as it is widely used and fits well with large datasets with many predictor variables. Parallelization drastically improves training time at the cost of result reproducibility (details about controlling for randomness described in Section 4.7).

Finally, for the **loss function** and **learning rate**, two important hyperparameters of neural networks, I use the default options that the H2O-package offers: mean squared error (MSE) loss function objective and the adaptive learning rate ADADELTA (Zeiler (2012)). The loss function refers to the minimization objective in backpropagation. The MSE loss function widely used in linear regression is also frequently used in continuous variable predictions in neural networks, measuring and minimizing the inconsistency between predicted values and actual values during training. The learning rate in neural networks refers to the amount parameter weights are updated during training: the higher the learning rate, the faster the model learns, but at the cost of arriving on sub-optimal final weights. Smaller learning rates may allow the model to learn a more optimal set of weights but may take significantly longer to train, or never converge. Various adaptive learning rates have been developed that monitor the model performance during training and adjust the rate in response, which results in generally better performance than manually configured rates. There is no consensus on the best adaptive learning rate to use, but ADADELTA is among the most popular (Goodfellow (2016)). Two ADADELTA learning rate hyperparameters, the time decay factor (*rho*) and time smoothing factor (*epsilon*), are tuned through grid search of optimal values.

For regularization techniques, first, **early stopping** is used to evaluate model skill during training, with user-given stopping rounds *n* and stopping tolerance *p*: at regular intervals during training, a value for a chosen scoring metric is calculated, and if this metric hasn't improved by *p* in *n* scoring events, model training is stopped. For example, when a separate validation sample is used, and the stopping metric is R^2, the algorithm regularly calculates the pseudo-out-of-sample R^2 of the model using the validation sample and uses early stopping if R^2 doesn't improve enough. The important consideration is which data to use in scoring the models during training: in-sample, validation sample (where a specific hold-out sample is used), or cross-validation. I use the H2O-package's default stopping metric, deviance (similar to mean-squared-error (MSE)) and choose to use a separate hold-out validation sample. The construction of the validation sample was detailed in Section 4.1.

Second, **dropout** is a regularization technique that generally reduces overfit by randomly dropping out a percentage of nodes in different layers during training. Srinivastava et al. (2014) describe dropout to enable breaking up situations where network layers co-adapt to correct errors from other units, where these co-adaptations do not generalize to unseen data. In my model, different dropout ratios are applied to the input layer and the hidden layers, where common values are around 0.2 for input layers and 0.5 for hidden layers (Srinivastava (2014)). I use the H2O-package default of 0.5 for the hidden layer dropout ratio, but tune the input dropout ratio through grid search.

Third, **L1 and L2 regularization** prevents neural networks from overfitting by keeping the values of weights and biases small. Both techniques add a regularization term to the loss function, resulting in weight values to decrease. L1 regularization penalizes the absolute sum of weights and can reduce weights to zero, whereas L2 penalizes the squared sum of weights and decays weights towards zero. A regularization parameter determines the strength of L1/L2 regularization and its value is optimized through grid search. Common values for L1 and L2 regularization parameters are small, around 1e-4 to 1e-5. I search for optimal L1/L2 shrinkage parameters through grid search.

4.6. *Hyperparameter tuning through random grid search*

In grid search, the user inputs sets of values for hyperparameters to be considered, and neural network models will be built for each combination of the given hyperparameter values. Then, the out-of-sample performance based on the validation data set can be compared, and the best models selected. Since the hyperparameter set I input is wide (288 combinations), a random grid search (following Bergstra and Bengio (2012)) over a set number of maximum models (150) is performed instead of a cartesian search where all possible combinations are iterated through. To make the results more robust to the randomness caused by parallelized stochastic gradient descent, I repeat this random grid search 10 times before making model choice decisions. This is to counter any lucky or unlucky iterations.

For each model, I calculate the R^2 based on the validation data and rank the model from 150 to 1 (150 being the best) for each of the 10 iterations. Then, I separate the results based on the activation function (Rectifier, Maxout, and Tanh). For each hyperparameter value under an activation function, I calculate the average validation R^2 and average rank for models that include that hyperparameter value. Then, for each activation function I choose the hyperparameter values that have the highest average validation R^2, confirmed by the average rank that diminishes effects of outliers.

The hyperparameters I test for are hidden layer topology, ADADELTA learning rate time decay factor (*rho*), ADADELTA learning rate time smoothing factor (*epsilon*), input dropout ratio, L1 parameter, and L2 parameter. I initially test for the following sets of hyperparameter values resulting in 288 different combinations:

- Activation function: Rectifier, Maxout, and Tanh
- Hidden layers: single-layer 64-nodes, two-layer [64, 32] nodes, three-layer [64, 32, 16] nodes (following the geometric pyramid rule of Masters (1993))
- *Rho*: 0.9 and 0.99, where 0.99 is the H2O default
- *Epsilon*: 1e-8 and 1d-9, where 1e-8 is the H2O default
- Input dropout ratio: 0.1 and 0.2, where 0 is the H2O default
- L1: 0 and 1e-4, where 0 is the H2O default
- L2: 0 and 1e-4, where 0 is the H2O default

Based on the results of the above search, I will run a second grid search iteration with incrementally adjusted hyperparameter values to test for improvement. For example, if models with Rho = 0.99 have higher average validation R^2 and rank than models with Rho = 0.9, which means a larger Rho seems better, I will adjust the second iteration to test for a larger Rho: Rho = 0.99 and Rho = 0.999.

4.7. *Controlling for randomness in model training and results through repetition*

H2O uses the HOGWILD! (Niu et al. (2011)) scheme to parallelize stochastic gradient descent computation, which drastically speeds up model training, at the cost of reproducibility. As training times without parallelization were unfeasible for practical reasons, using HOGWILD! was a necessity.

Randomness is also an inherent part of machine learning algorithms and particularly neural networks (Brownlee (2016)). Randomness is introduced through the shuffled order of training data, inherent randomness in the algorithms (such as initializing neural network parameter weights with random values), and randomness in data sampling during training (the model trains on random subsets of data at a time). H2O requires setting a seed before beginning training for its random number generator that affects the above three sources of randomness.

Choosing a single seed and reporting only on results produced by that seed would reveal a limited image of prediction performance.

Thus, I repeat hyperparameter grid search 10 times, making model choice decisions based on average R^{2}s, and repeat the resulting three optimized models 100 times each and analyze the mean and standard deviation of their prediction accuracy measures. This also provides an interesting benchmark range of R^2 values to compare existing literature on and for future research.

4.8. *Measuring statistical prediction performance using out-of-sample R^2, return correlation and directional accuracy*

The main measure of prediction accuracy I use is the out-of-sample R^2. The measure is defined as:

$$R^2_{oos} = 1 - \frac{\sum_{(i,t)}\left(r_{i,t+1}-f_{i,t+1}\right)^2}{\sum_{(i,t)}\left(f_{i,t+1}-\bar{r}_t\right)^2},$$

where $r_{i,t+1}$ is the realized return of stock i at time $t+1$, $f_{i,t+1}$ is the same stock's predicted return, and $\bar{r}_t$ is the historic average return of all stocks in the training sample. Between different subsets of the training sample (e.g. sample that only contains large stocks), the historic average return is calculated separately for each subset. The prediction accuracy is only measured on the testing data set, whose data are not used in model estimation or tuning. In addition, I examine the prediction performance using correlation between predicted the realized returns and percentage of times predicted and realized returns have the same sign (directional accuracy).

4.9. *Measuring economic value of predictions: Bottom-up equity premium predictions and excess returns from machine learning portfolios*

I test the predictive performance of my models in forecasting returns of custom equal-weighted indices of all stocks, largest 500 stocks, and smallest 500 stocks. For this, I construct bottom-up estimates of the index returns using individual stock return predictions of the respective stocks each month. The R^2_{oos} metric for this experiment is calculated as the same:

$$R^2_{oos} = 1 - \frac{\sum_{t=1}^{T}(r_t-f_t)^2}{\sum_{t=1}^{T}(r_t-\bar{r}_t)^2},$$

where $\bar{r}_t$ is the historical average return of the index calculated through the training sample time period. These results can then be compared to both the R^2 metric of individual stock return predictions, and equity premium prediction results like Campbell and Thomson (2008), who estimate the out-of-sample predictability of the S&P 500 index based on several valuation ratios.

Second, I test for the profitability of machine learning long-short portfolios trading all stocks, top 500 largest stocks, and bottom 500 smallest stocks. The portfolios are built by sorting stocks of each respective size category each month based on the individual stock return predictions and buying the top decile stocks and shorting the bottom decile. I then calculate the average monthly returns, volatilities, Sharpe ratios, and the risk-adjusted alphas compared to the Carhart (1997) four-factor model. Trading costs are not considered within the scope of this paper. The results from this test give an idea of the theoretical profitability that could have been realized by investors using the forecasts.

4.10. *Measuring relative predictor variable importance*

To gain insight into relative predictive strength of the predictors, I extract variable importance using the Gedeon (1997) method that H2O has integrated as part of its deep learning functions. The method utilizes the sum of products of normalized weights to evaluate the weight matrices connecting the inputs with the first two hidden layers. I note that evaluating trained neural networks, which are considered "black box" models, is infamously difficult, and many measures of variable importance have been proposed over the years, all riddled with different potential shortcomings (see for example Sarle's (2000) analysis). The Gedeon method is also applied by Krauss et al. (2017), while Gu et al. (2018) use a simpler measure.

5. Analysis and results

5.1. *Hyperparameter grid search results*

The first grid search reported in Table 1 reveals a consistent picture of the best-performing parameter values underlined in the table. Across the three models (Rectifier, Maxout, Tanh), the same hyperparameter values seem to be the best fit. Overall, the Rectifier models performed the best in the validation sample with an average R^2 of 0.61%, followed by Maxout (0.36%)

and Tanh (0.30%). I note that, on average, the grid search R^2 results should appear more positive than in the true out-of-sample test, as the validation sample is a shuffled split from the training sample containing information from the past (as explained in Section 4.1).

Considering the standard deviations reported in brackets, the most unstable model turns out to be Maxout, with an overall average standard deviation of 0.62%. However, it seems the instability is concentrated in models with certain hyperparameter values, such as more than one hidden layer (standard deviations of 0.73% and 0.74%), input dropout of 0.2 (0.73%), L1 of 1.0E-4 (0.80%), so the instability can be avoided through the optimized parameters. The Rectifier models seem equally unstable across parameter values, exhibiting standard deviations of around the average 0.49% against its 0.61% mean R^2. The Tanh models display similar behavior, although with lower R^2 and standard deviations overall.

Table 1. Hyperparameter grid search results comparing average validation sample R^2 (%) across parameter values.

The values reported are average R^2 (%) based on the validation data for models including the hyperparameter value defined in the left-most column. Rectifier, Maxout, and Tanh models are compared that differ by their activation function. The random grid search trains 150 models out of 288 possibilities (3 x 3 x 2 x 2 x 2 x 2 x 2), and grid search is repeated 10 times, producing a maximum of 1500 models. 95 models failed in training, and I filter out an additional 18 models with in-sample R^2 less than -10%, resulting in a total sample size of 1387 models.

	Rectifier	Maxout	Tanh
Overall	0.61 (0.49)	0.36 (0.62)	0.30 (0.18)
Hidden layers and units per layer			
[64]	0.89 (0.42)	0.50 (0.23)	0.37 (0.17)
[64, 32]	0.49 (0.42)	0.36 (0.73)	0.34 (0.13)
[64, 32, 16]	0.34 (0.47)	0.19 (0.74)	0.19 (0.19)
Input dropout			
0.1	0.65 (0.52)	0.43 (0.46)	0.31 (0.19)
0.2	0.58 (0.45)	0.29 (0.73)	0.29 (0.18)
L1			
0	0.71 (0.37)	0.55 (0.26)	0.33 (0.16)
1.00E-04	0.48 (0.60)	0.14 (0.80)	0.26 (0.20)
L2			
0	0.61 (0.46)	0.35 (0.68)	0.29 (0.18)
1.00E-04	0.62 (0.52)	0.36 (0.56)	0.31 (0.18)
Rho			
0.99	0.84 (0.55)	0.54 (0.49)	0.38 (0.19)
0.9	0.38 (0.26)	0.17 (0.68)	0.22 (0.14)
Epsilon			
1.00E-08	0.56 (0.55)	0.22 (0.80)	0.24 (0.17)
1.00E-09	0.67 (0.40)	0.49 (0.32)	0.36 (0.18)

I conduct two iterations of the entire grid search process, where I set each grid to produce 150 models out of the 288 possible combinations, and each iteration to produce 10 grids. The first iteration created 1405 models and the second 1500. The numbers are less than the theoretical 1500 (150 models each grid with 10 grids), because some models become unstable during training fail to complete. I additionally filter away models with in-sample R^2 less than -10%, as these can be considered computational failures, and filtering based on in-sample R^2 is acceptable prior to engaging in the pseudo out-of-sample validation test. The hyperparameter values tested in the second iterations were adjusted based on the results from the first iteration to further optimize hyperparameter choices. The resulting Table A2 for the second grid search iteration is reported in the Appendix. Overall, changes in activation function, hidden layer topology, and rho had the most significant effects on model performance, while L1, L2 and input dropout were less impactful with models generally working better with smaller regularization parameter values. Based on the second grid search of Table A2, the best-performing models on average chosen for additional performance-testing are shown in Table 2.

Table 2. Optimized neural network model topologies based on hyperparameter grid search.

"Activation" refers to the activation function of the model, "Hidden" refers to the number of hidden layers and number of nodes within each layer separated by a comma, and "Input dropout" refers to the percentage of inputs randomly dropped out to improve generalization. L1 and L2 refers to the L1- or L2-regularization parameter strength, where a higher value equals more respective regularization. Epsilon is the time smoothing factor and Rho the time decay factor of the ADADELTA adaptive learning rate algorithm used.

Activation	Hidden	Input dropout	L1	L2	Epsilon	Rho
Rectifier	[64]	0	1e-5	1e-5	1e-9	0.999
Maxout	[64, 32]	0	0	1e-5	1e-10	0.999
Tanh	[64, 32]	0	0	1e-5	1e-10	0.999

5.2. *Main results: Out-of-sample R^2 compared to linear benchmarks*

First, I examine the true out-of-sample monthly prediction performance of the three optimized models measured by R^2 as defined in Section 4.8. As explained in Section 4.7 about the randomness of separate iterations of the same model, the figures are reported as means accompanied by their standard deviations to give a better picture of average model performance. Shown in Table 3, the highest out-of-sample statistical prediction performance is achieved by Maxout models, with an average out-of-sample R^2 of 0.80%, followed by Tanh models with R^2 of 0.39% and Rectifiers with 0.17%. The values are in line with existing literature, such as Gu et al. (2018) who find a monthly out-of-sample R^2 of 0.39% on their best-performing rectifier neural network model trained on an extended version of my dataset. The highest individual R^2 of all models was 1.07% produced by a Rectifier model, followed by 1.03% by a Maxout model.

Table 3. Stock-level monthly prediction performance measured by average out-of-sample R^2 (%).

The table reports the mean out-of-sample R^2 (%) and the standard deviations of R^2 results (in brackets) from repeating model training and prediction 100 times for the three neural network models (Rectifier, Maxout, Tanh), and compares them to the OLS benchmark using lagged market return, firm size, book-to-market, and momentum (OLS-3), OLS-3 with annual refitting (OLS-3-R), and Fama and French (2006) OLS (OLS-FM) that uses coefficient estimates from monthly cross-sectional regressions on OLS-3 predictors except market return to predict next month returns. Results are also reported for subsamples that include only the top 500 (Big) and bottom 500 (Small) stocks by market value (models built on full data, predictions subsampled).

	All	Big	Small
Maxout	0.80 (0.12)	0.54 (0.18)	1.36 (0.13)
Tanh	0.39 (0.09)	0.06 (0.27)	1.01 (0.06)
Rectifier	0.17 (0.42)	-1.47 (0.86)	0.75 (0.54)
OLS-3	0.73	-0.02	0.55
OLS-3-R	0.64	-0.46	0.54
OLS-FM	-7.15	-33.53	-1.96

In addition, I analyze prediction performance in the largest and smallest stocks by market value. This is done by calculating the R^2 using returns of only the top 500 and bottom 500 stocks by market value each month, subset from existing predicted returns produced by models trained on the full data. Table 3 shows that neural network predictions are generally better in small stocks and worse in large stocks: all R^2 in the “All” category are less than those in “Small” and greater than those in “Big”. For the linear benchmarks, overall predictions on all stocks performed the best, followed by small stock predictions. This confirms there are differences in the return predictability of small versus large stocks and indicates that the neural networks fit more towards small stocks’ rather than large stocks’ return behavior, which seems reasonable as a majority of stocks in the U.S. market data belongs to small stocks.

The high small stock performance is in line with existing research that find return predictability to be the strongest among stocks with the highest levels of arbitrage frictions or high trading costs (Lesmond, Schill, and Zhou (2004), Hou and Moskowitz (2005), Chordia, Roll, and Subrahmanyam (2008), Li and Zhang (2010), and Lam and Wei (2011)). My findings also confirm Green et al.'s (2017) observation that anomalies are mostly present in microcap stocks and not robustly so in all stocks. The result contradicts those of Gu et al. (2018), who find that predictions by neural networks are the most accurate in the top 1000 stocks (0.72% at best), then the bottom 1000 stocks (0.46% at best), then all stocks (0.39% at best). The differences are most likely to stem from data and algorithm differences. Gu et al. train their models on data from 1957 to 1986, include more macroeconomic predictors, and conduct out-of-sample testing on data from 1987 to 2016. A deeper investigation into the reasons behind the result differences is beyond the scope of this paper.

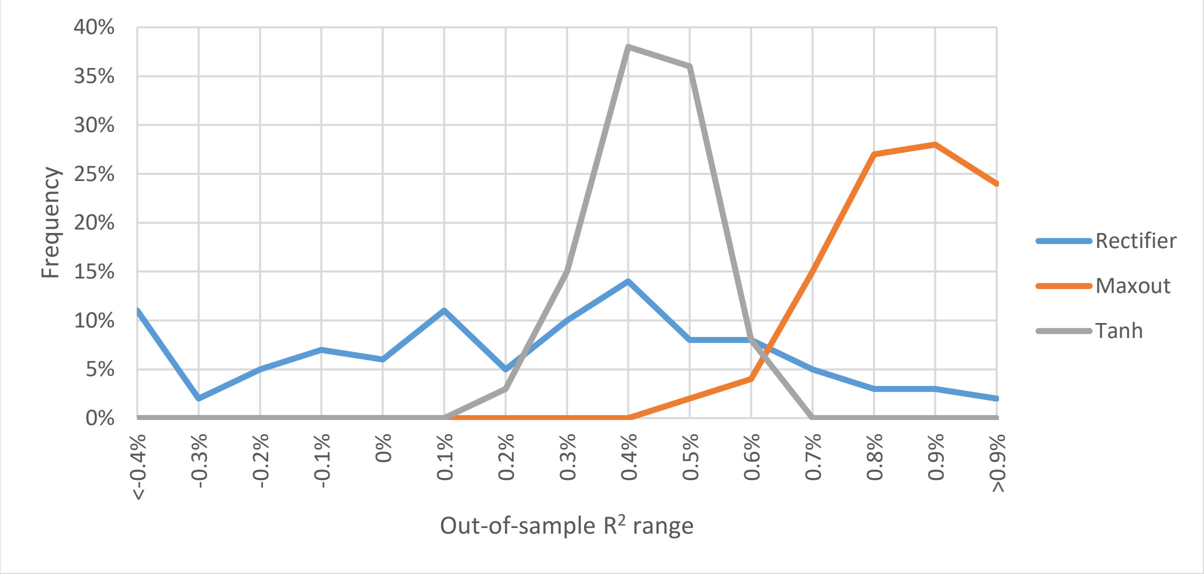
Comparing the results to the three linear regression benchmarks, neural networks do not universally outperform the linear benchmarks. The worst-performing OLS-FM, which cannot use lagged market return as a predictor in its monthly cross-sectional regressions, underperforms all other models significantly, with an OOS R^2 of -7.15%. The other two, OLS-3 and OLS-3-R, which both include lagged market return as a predictor in their panel regressions, outperform Tanh and Rectifier models on average. Both the neural networks and the OLS regressions heavily use the lagged market return as the most important predictor, shown more evidently in Section 5.8 about variable importances. So, a linear model using a longer training sample without annual refitting seems to generalize well into the bull-market period of 2009-2018. The linear benchmarks also perform worse in big stocks. For small stocks, the results are worse than for all stocks, which could imply that the models fit more towards average-sized stocks, and that neural networks are better able to learn the anomalies in small stocks.

5.3. *Standard deviations of out-of-sample R^2 and correlation between in-sample and out-of-sample performance*

The standard deviations vary between the models and the stock size categories. Overall, the Rectifier models seem to be the most unstable, with the highest standard deviations across size categories. For example, the Rectifier has a standard deviation of 0.42% for all stocks, against a mean of 0.17%, whereas the Maxout has a standard deviation of 0.12% against a mean of 0.80%. Figure 1 presents a visualization of the variance in results, plotting the percentage frequency distribution of out-of-sample (OOS) R^2 results. The graph shows the high variance of Rectifier model results, with R^2 values almost uniformly spread across the entire range, and the relatively more stable performance of the Maxout and Tanh models. Tanh models in particular center densely around the 0.4% mean, with 38% of R^2 results between 0.3% and 0.4%. The variance is further visualized in the scatterplot of Figure 2.

Figure 1. Frequency distribution of out-of-sample R^2s of three neural network models iterated 100 times.

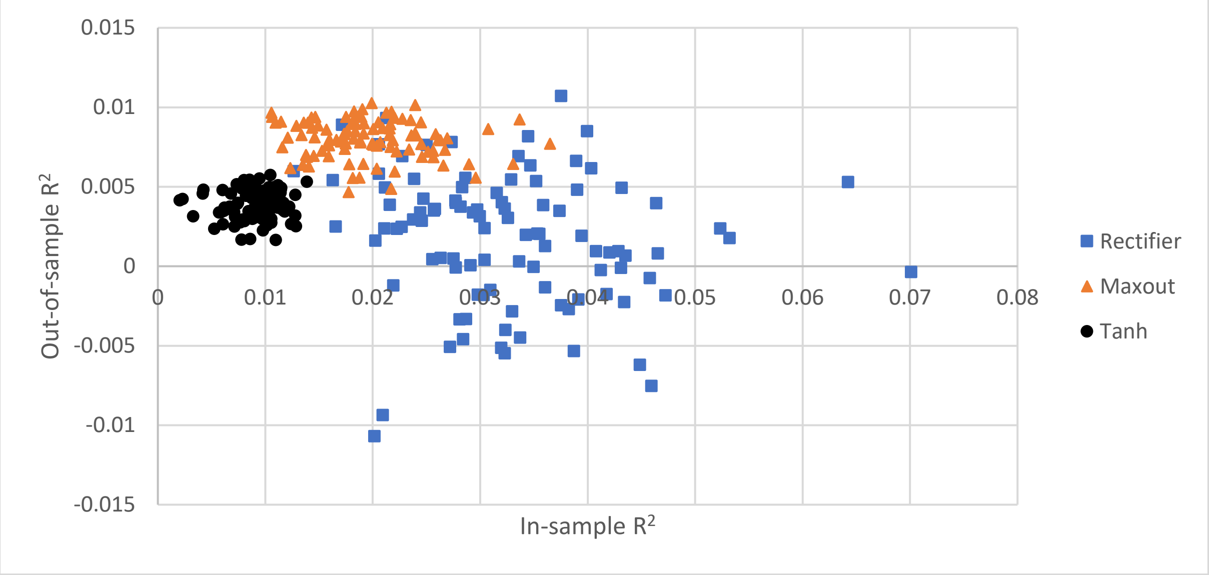
Three neural network models are used to predict monthly excess stock returns. Each model is repeated 100 times with its prediction performance measured by out-of-sample R^2 recorded. The graph shows the frequency distribution of the resulting 100 R^2s for each model, distinguished by the three different lines. The x-axis describes the R^2 value ranges, and the y-axis the percentage of results in that value range.



The correlation between in-sample R^2 and out-of-sample R^2 is -28.9% for all results, -15.8% for Rectifier model results, -12.6% for Maxout results, and 8.1% for Tanh results. From Figure 2 we see that the overall negative correlation is driven by the strong in-sample performance of some rectifier models and their relatively poor out-of-sample performance, as well as the strong out-of-sample performance of the Maxout models. The Rectifier's negative correlation is mainly driven by the models with very strong in-sample but poor out-of-sample performance, suggesting Rectifiers more easily overfit for the noisy data being unable to generalize out-of-sample. The Maxout and Tanh models are more stable and produce denser scatters on the graph. The scatterplot also shows the average outperformance of Maxout compared to the other two models, with its scatter almost entirely above Tanh results, and on average above the high-variance Rectifier results.

Figure 2. Scatterplot of in-sample versus out-of-sample R^2 for the three optimized models.

The x-axis is the in-sample R^2, and y-axis the out-of-sample R^2 from the selected model performance test where 100 of each model are trained and tested on out-of-sample data.



5.4. *Rolling past 12-month out-of-sample R^2 over time*

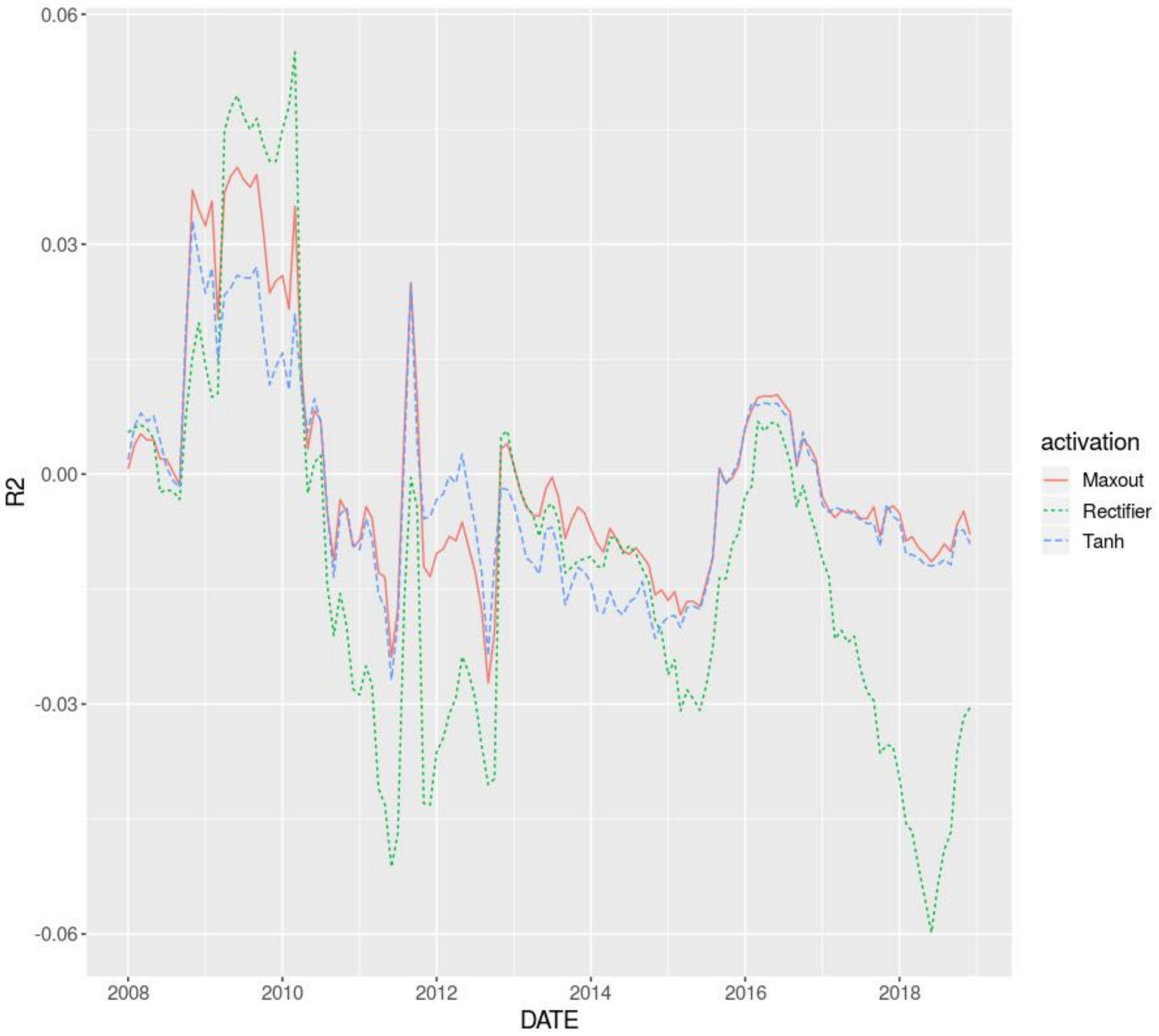
Figure 3 plots the rolling 12-month out-of-sample R^2 over the testing period (2007-2018). The graph provides information on the temporal development of the R^2 results. Overall, the models follow the same trends with peaks and troughs at roughly the same time. Maxout and Tanh models align very closely, while Rectifiers are generally lower. It seems that the accuracy of predictions measured by R^2 varies greatly over time, peaking during 2009-2010 and bottoming during 2011-2013. At a surface level analysis, there seem to be indications of greater prediction accuracies during times of higher volatility as measured by the CBOE Volatility Index (VIX). For example, during the R^2 peaks of 2009-2010, 2011-2012, 2015-2016 and end of 2018, the VIX also displays spikes in its level[3] roughly during these times. The findings are somewhat surprising, considering the models' reliance on the market return predictor (discussed in Section 5.8), and the understanding in literature that, for example, momentum strategies perform poorly during high market volatility (Wang and Xu (2015)). One might expect models to perform the best during low volatility periods when stocks follow a similar trend, but the temporal results of Figure 3 reveal that the neural networks perform particularly well during market turbulence. One explanation could lie in the out-of-sample R^2 measure itself, where

[3] See VIX levels for example at Yahoo Finance for the time period 2007-2018.

during turbulent times the historical average becomes a worse predictor, and the neural network predictions are able to better predict differences. During calmer periods the historical average serves as a solid predictor, and neural networks may introduce unnecessary variation in predictions. There may be many other reasons why out-of-sample R^2 varies over time, but I leave a deeper investigation into the temporal development of R^2 to future research.

Figure 3. Rolling past 12-month out-of-sample R^2 over the testing period.

The figure plots the rolling past 12-month out-of-sample R^2 over the testing period 2007-2018, with the first value being January 2008 and its out-of-sample R^2 based on the past 12-months of predictions. The three different lines represent the three neural networks, Maxout, Rectifier, and Tanh. The x-axis represents the date, and the y-axis the R^2.



5.5. *Correlation between predicted and realized returns and directional accuracy*

To give more perspective to the statistical performance of the models, Table 4 reports the average correlation between predicted and realized returns, and Table 5 the average directional accuracy for the same three stock size groupings (all, big, small) as earlier. Surprisingly, the correlation results differ from the R^2 results, where Rectifier performs the best with an average correlation of 10.08%, followed by Maxout (9.66%) and Tanh (6.97%), despite Rectifier performing the worst when measured by R^2. Both Rectifier and Maxout outperform all linear benchmarks, whereas the Tanh models underperform. The underperformance of big stock predictions compared to small stock and all stock predictions persists in neural networks. With big stocks, the significantly negative R^2 results by Rectifiers persists, where the average correlation is only 5.58%. Both Tanh and Rectifier underperform linear benchmarks in big stocks (except OLS-FM). For small stocks, neural networks have generally higher return correlations, suggesting the models are better able to learn predictive information in the smallest stocks, outperforming the linear benchmarks.

Table 4. Stock-level monthly prediction performance measured by average correlation (%) between predicted and realized returns.

The table reports the mean correlation between predicted and realized returns and the standard deviations (in brackets) of the correlation results from repeating model training and prediction 100 times for the three neural network models (Rectifier, Maxout, Tanh), and compares them to the OLS benchmark using lagged market return, firm size, book-to-market, and momentum (OLS-3), OLS-3 with annual refitting (OLS-3-R), and Fama and French (2006) OLS (OLS-FM) that uses coefficient estimates from monthly cross-sectional regressions on OLS-3 predictors except market return to predict next month returns. Results are also reported for subsamples that include only the top 500 (Big) and bottom 500 (Small) stocks by market value (models built on full data, predictions subsampled).

	All	Big	Small
Maxout	9.66 (0.26)	8.19 (0.64)	10.30 (0.35)
Tanh	6.97 (0.30)	6.81 (0.41)	8.20 (0.18)
Rectifier	10.08 (0.55)	5.58 (1.10)	10.39 (0.82)
OLS-3	8.34	8.09	8.47
OLS-3-R	7.80	7.15	8.14
OLS-FM	7.26	4.87	7.78

In directional accuracy, neural network models come out with a slight edge over the linear benchmarks, shown in Table 5. In all stocks, where 50.75% of realized returns were positive, all models outperform a constant positive guess, with Maxout having the highest average performance at 52.96% (maximum single result was 53.46%). Surprisingly, OLS-FM outperforms the other two OLS models here, with a directional accuracy of 51.50%. This might be explained by cross-sectional monthly regressions being able to identify cross-sectional winners and losers better than the other two panel-regression OLS models, which weigh their market return variable much more. An additional explanation could be sizeable outlier realized returns that are in the same direction as predicted returns. These would cause poor statistical prediction accuracy (measured by R^2 and correlation) but good directional accuracy. In OLS-FM's case, the cross-sectional regression coefficients are understandably unable to fit for potential outlier realized returns, but the cross-sectional differences in predictors still seem to effectively capture differences between winners and losers, resulting in good directional accuracy.

In big stocks, 55.73% of realized returns were positive, providing a harder baseline to beat. None of the models exceed this baseline, with Maxout coming closest at 55.13%. Thus, the relative underperformance of big stock predictions persists, as the models are trained on data skewed towards small stocks. In small stocks, only 44.01% of realized returns were positive, so although the directional accuracy results are lower overall in small stocks, they beat the baseline much more convincingly in this size category, with OLS-FM delivering the highest accuracy at 52.51%, and Rectifier the best average for neural networks at 51.32% (maximum single result was 53.12% by a Rectifier model).

5.6. *Market premium perspective: Index return prediction accuracy*

To gain insight into the extent of predictability of the market premium, I examine return predictions for three equal-weighted portfolios, essentially constructed from the same data used for firm size categories earlier in this paper: 1) a market index portfolio constituting all tradeable stocks each month, 2) a portfolio consisting of the 500 largest stocks each month by market capitalization, and 3) a portfolio consisting of the 500 smallest stocks each month by market capitalization. The portfolio return predictions are constructed bottom-up, from individual stock predictions of the stocks forming the portfolio each month. Table 6 reports the results.

Table 5. Stock-level monthly prediction performance measured by average directional accuracy (%).

The table reports the mean directional accuracy and the standard deviations (in brackets) of the directional accuracy results from repeating model training and prediction 100 times for the three neural network models (Rectifier, Maxout, Tanh), and compares them to the three linear ordinary least squares (OLS) benchmarks. Directional accuracy is calculated as the percentage of predicted returns having the same sign as realized returns. In the entire testing data, big stocks subsample, and small stocks subsample, 50.75%, 55.73%, and 44.01% of realized returns were greater than zero, respectively (reported in brackets after size category titles). OLS-3 is a linear regression using lagged market return, firm size, book-to-market, and momentum as predictors. OLS-3-R is the same as OLS-3, adding annual refitting. OLS-FM uses coefficient estimates from monthly cross-sectional regressions on OLS-3 predictors except market return to predict next month returns. Results are also reported for subsamples that include only the top 500 (Big) and bottom 500 (Small) stocks by market value (models built on full data, predictions subsampled).

	All (50.75)	Big (55.73)	Small (44.01)
Maxout	52.96 (0.21)	55.13 (0.70)	50.61 (0.76)
Tanh	51.96 (0.47)	52.88 (1.58)	50.55 (0.81)
Rectifier	52.68 (0.33)	54.68 (1.30)	51.32 (0.77)
OLS-3	50.85	51.67	48.45
OLS-3-R	50.84	51.76	48.60
OLS-FM	51.50	51.54	52.51

From Table 6, I observe that R^2 results increase dramatically across the board. Particularly in All stocks and Small stocks, the model predictions of the respective portfolio returns clearly outperform their respective historical average return-based predictions. Notably, the historical average return was around 0.85% for all stocks, 0.73% for big stocks, and 2.5% for small stocks: the large and highly volatile nature of small-stock returns causes the historical average to be a particularly poor predictor for the Small portfolio, which amplifies the R^2 results in their category. However, for All stocks and Big stocks, the historical average serves as a fine benchmark, and choosing a zero-return prediction like Gu et al. (2018) would artificially increase the results in those two size categories. The overall trend is consistent with results of earlier Table 3, where small-stock predictions outperform big-stock and all-stock, and neural networks barely outperform their linear benchmarks. Out of the neural networks, surprisingly, Rectifiers performs the best, with an average R^2 of 7.12% in all stocks, compared to Maxout with 6.29%, which contradicts Table 3 results where Rectifiers were performing the worst on average. This indicates that averaging the return predictions at a portfolio-level seems to alleviate effects of noisy individual predictions and distill predictive information about the portfolio as a whole effectively.

Table 6. Portfolio-level monthly prediction performance measured by out-of-sample R^2 and directional accuracy (%).

The table reports the out-of-sample R^2 (R^2_{OOS}) and percentage of predicted returns that have the same sign as realized returns (Directional accuracy) for three neural network models (Maxout, Tanh, Rectifier) and three OLS benchmarks, when predicting monthly returns of three equal-weighted portfolios: "All" is a market index constituting all tradeable stocks on the market each month, "Big" constitutes the 500 largest stocks each month by market capitalization, and "Small" constitutes the 500 smallest stocks each month by market capitalization. In the Directional accuracy panel, the numbers within brackets after the size category subtitles are the percentage of realized portfolio returns that are positive – the baseline against which the results should be compared.

		R^2_{OOS} (%)	
	All	Big	Small
Maxout	6.29	2.57	17.87
Tanh	4.74	1.57	13.77
Rectifier	7.12	0.29	20.81
OLS-3	6.26	0.31	16.48
OLS-3-R	5.41	-1.23	16.41
OLS-FM	-52.32	-114.49	-12.88
		Directional accuracy (%)	
	All (59.7)	Big (62.5)	Small (50.0)
Maxout	60.49	62.12	55.47
Tanh	58.07	57.64	55.68
Rectifier	60.55	61.29	56.70
OLS-3	56.94	55.56	55.56
OLS-3-R	56.94	56.25	54.17
OLS-FM	56.94	56.25	57.64

Taking a closer look at prediction performance in the All-stocks category, Rectifiers and Maxout models, with average R^{2}s of 6.29% and 7.12%, narrowly outperform the best linear benchmark, OLS-3 with an R^2 of 6.26%. This is consistent with Table 3 results where OLS-3 came close to the best neural network results. However, the Directional accuracy panel reveals a new picture, where OLS benchmarks underperform the neural networks: in All stocks, they fail to even exceed the 59.7% baseline (percentage of portfolio realized returns that were positive). All three linear benchmarks happen to have a directional accuracy of 56.94%, predicting 82/144 monthly returns of the All-stock portfolio correctly, whereas Maxout and Rectifiers achieve a directional accuracy of 60.49% and 60.55%, respectively.

In the Big-stocks category, overall performance is much lower, consistent with Table 3, where predictions are clearly weaker in the biggest stocks. Maxout performs the best with an R^2 of 2.57%, with Rectifiers now underperforming with an R^2 of 0.29%. The poor performance of predicting a portfolio of the 500 largest stocks each month is surprising, given the importance of the lagged market return predictor. However, this is also encouraging as indications of cross-sectional predictive power even in small stocks with overall higher idiosyncratic risks. Directional accuracy mirrors earlier results, where none of the models exceed the baseline (62.5%) and neural networks outperform OLS models. The R^2 results are in line with existing literature such as Gu et al. (2018), who find an R^2 of 1.8% for neural network predictions on the S&P 500 index portfolio. The results also show surprisingly powerful out-of-sample performance compared to traditional literature, such as Welch and Goyal (2008) who fail to produce positive out-of-sample R^2 using macroeconomic predictors to predict the market return, or Kelly and Pruitt (2013) who find monthly out-of-sample R^2's of around 1% for the market index delivered by PLS. It seems neural networks are able to capture significant stock-level predictability that increases even further when averaged together in portfolio return predictions.

In Small stocks, I observe abnormally high R^2 results – puzzling considering the few benchmark values from comparable literature mentioned earlier. Rectifier models achieve the highest average R^2s of 20.81%, followed by Maxouts with 17.87%. The Rectifier models' outperformance despite low R^2 in Table 3 could be explained by the models generating highly varying individual predictions, but when averaged in a portfolio, the overall predictive result becomes less noisy and grants a much better prediction than the historical average of the portfolio. Again, I note that microstocks have very volatile returns and using their historical returns (around 2.5%) may serve as a poor predictor, as it ends up inflating R^2 results by around 5 percentage points compared to using a zero-return prediction as Gu et al. (2018). The opposite would be true for All and Big stocks, where a zero-return prediction amplifies R^2 results by a few percentage points, which is why to keep the methodology consistent across size categories, I keep the conventional historical mean as the benchmark for R^2 calculations. The linear benchmarks also perform almost on par with the neural networks, with OLS-3 having R^2 of 16.48%. This is encouraging since if there were some model or data construction error particularly in the market return variable, the predictions should be better in large stocks instead, as they follow the market movements more closely. The downside is that stronger predictability in microstocks has less economic significance as utilizing the information is more expensive in illiquid microstocks for investors. Directional accuracy follows the trend in Table 5, where all models exceed the baseline (50%) and surprisingly OLS-FM has the highest accuracy (57.64%), considering its very poor R^2.

Finally, as a robustness test, I run autoregressive regressions (AR(1)) predicting the market return using one-month lagged market returns (Wilshire 5000 Total Market Full Cap

Index) and calculate its out-of-sample R^2. The resulting monthly R^2 is 1.15% for the AR(1) model, which, compared to the R^2 of 6.26% of OLS-3 or 7.12% of Rectifiers, shows that combining firm characteristics with the market return in a panel data form increases return prediction accuracy at the market return level.

5.7. *Neural network long-short portfolio results*

To examine the practical economic significance of neural network forecasts, I build long-short portfolios on the three stock size subsamples (All, Big, Small), where each month out-of-sample predicted returns are used to sort the stocks in the sample, and the top decile is bought while the bottom decile sold short. In addition, I regress the monthly returns against the Fama-French three-factor model (Fama and French (1996)) augmented with the momentum factor of Carhart (1997), to analyze the exposure of returns to common sources of systematic risk. Same as for all neural network results of this paper, all calculations are done for the 100 different repeated model iterations of each of the three neural network models, so the reported measures are averages. The results are reported in Table 7.

Overall, the neural networks now unquestionably beat the OLS benchmarks in average monthly returns and Sharpe ratio, with advantages stemming from both higher returns and lower standard deviations. The neural network models also exhibit high levels of alpha, which are highly statistically significant. Performance trends observed in earlier tables across size categories still roughly hold, with big stocks underperforming small stocks. This confirms observations in the anomalies literature, where anomalies are concentrated in small and microcap stocks and are less robustly present in large stocks (Green et al. (2017)). Table 7 is the clearest demonstration of the predictive power of the neural networks over the linear benchmarks, where earlier statistical measures did not necessarily show clear winners. The results suggest neural networks are able to learn cross-panel return predictive signals to identify stock picks producing returns significantly above the market averages. However, I note that trading costs are not taken into account, so the actual realizable returns are most likely much lower than stated due to high portfolio turnover.

Table 7. Monthly profitability of long-short top-bottom decile portfolios using neural network predictions.

This table reports the average monthly return (Ret), standard deviation of returns (Std), Sharpe ratio (SR), and alpha with its t-stat (α (*t-stat*)), based on Fama-French three-factor model augmented with momentum, of a long-short top-bottom decile portfolio sorted on predicted returns. The values are in percentages except for SR and t-stats. The results are reported for three size categories, where "All stocks" includes all stocks, "Big stocks" only the 500 largest stocks each month, and "Small stocks" the 500 smallest stocks each month by market capitalization. The values reported for the three neural network models (Maxout, Tanh, Rectifier) are averages of 100 separately-trained models each, compared to the singular results of their three OLS benchmarks.

		All stocks		
	Ret	Std	SR	α (*t-stat*)
Maxout	1.32	1.84	2.49	1.39 (9.80)
Tanh	0.84	1.58	1.85	0.92 (7.48)
Rectifier	1.51	1.99	2.62	1.55 (10.46)
OLS-3	-0.09	2.28	-0.14	0.04 (0.26)
OLS-3-R	-0.13	2.22	-0.20	0.01 (0.08)
OLS-FM	0.17	2.89	0.21	0.16 (0.64)
		Big stocks		
	Ret	Std	SR	α (*t-stat*)
Maxout	0.43	1.57	0.96	0.52 (4.52)
Tanh	0.28	1.45	0.66	0.35 (3.12)
Rectifier	0.33	1.52	0.76	0.37 (3.13)
OLS-3	-0.25	1.90	-0.45	-0.18 (-2.23)
OLS-3-R	-0.27	1.90	-0.50	-0.21 (-2.31)
OLS-FM	0.15	2.64	0.19	0.18 (0.84)
		Small stocks		
	Ret	Std	SR	α (*t-stat*)
Maxout	3.03	4.41	2.39	3.02 (8.40)
Tanh	2.11	3.59	2.04	2.12 (7.15)
Rectifier	3.33	4.87	2.38	3.36 (8.47)
OLS-3	0.85	3.68	0.80	0.97 (3.16)
OLS-3-R	0.81	3.70	0.76	0.91 (2.95)
OLS-FM	-0.11	4.85	-0.08	-0.30 (-0.75)

In All stocks, Rectifiers perform the best with average monthly average returns of 1.51% with an average annualized Sharpe ratio of 2.62, followed by Maxouts with average returns of 1.32% and a Sharpe ratio of 2.49. The highest individual neural network result was by a Rectifier model iteration that achieved average returns of 1.67% with a Sharpe ratio of 2.88. Over the test period (2007-2018), the comparable Wilshire 5000 Total Market Full Cap Index returned 0.7% (3.9% standard deviation), with an annualized Sharpe ratio of 0.60. The OLS benchmarks underperform, with only OLS-FM achieving slightly positive returns, and all three having higher standard deviations than the neural networks. The highly significant Maxout and Rectifier model average alphas of 1.39% and 1.55 (t-stats of 9.80 and 10.46), respectively, indicate there is a large portion of returns unexplained by the three Fama-French factors and momentum, and the neural networks are able to produce sizeable abnormal excess returns. Compared to literature, Gu et al. (2018) find higher monthly average returns of 3.19% with 4.77% standard deviation, with similar Sharpe ratios 2.32, using the three-layered neural networks and the same long-short portfolio. They also find higher, significant Fama-French 5-factor (+ momentum) alphas of 2.98%. Notably, the size of the alpha is similar to the average return (alpha of 2.98% versus returns of 3.19%) – a finding my results echo (e.g. alphas of 1.55% versus returns of 1.51%).

In Big stocks, overall performance decreases, but neural networks still outperform their linear benchmarks. Maxout models perform the best, with average monthly returns of 0.43% and Sharpe ratios of 0.96, compared to a maximum of 0.15% return and Sharpe of 0.19 by the OLS-FM. Over the test period (2007-2018), the comparable S&P 500 Index returned on average 0.5% each month (4.2% standard deviation), with an annualized Sharpe ratio of 0.40. Thus, most of the Sharpe ratio improvement by the neural networks stems from lower standard deviations. The results suggest that neural networks are able to consistently identify some winners and losers despite market fluctuations, resulting in a less volatile portfolio than the market. Krauss et al. (2017) find drastically larger returns of 0.33% per day (roughly 7% per month) on a portfolio long-shorting the top/bottom 10 stocks ranked daily by predicted probability to outperform the market, using neural networks trained on lagged returns of S&P 500 index constituents. They find comparable-sized annualized Sharpe ratios of 2.44. However, after transaction costs the average daily returns and annual Sharpe ratio decrease to 0.13% and 0.55, respectively.

In Small stocks, overall performance increases with neural networks maintaining a decisive margin over the OLS benchmarks. The Rectifiers perform the best, with average monthly returns of 3.33% and Sharpe ratios of 2.38, compared to the maximum of 0.85% return and Sharpe of 0.80 by the OLS-3. Although the average monthly returns achieved by the neural networks in small stocks are double those in all stocks, the standard deviations also end up higher, resulting in slightly lower Sharpe ratios.

Figure 4 plots the average monthly returns of the neural network portfolios against the out-of-sample R^2s for each of the 300 separate model iterations. The plot would look almost identical when plotting Sharpe ratios instead of returns. The scatterplot visually displays the somewhat curious return outperformance of the Rectifier models despite their volatile and lower-on-average out-of-sample R^2, where the Rectifier points place consistently higher than the other models on the y-axis, but have a much wider scatter on the x-axis. On the other hand, Maxout and Tanh models display a more linear result, where higher out-of-sample R^2 results in higher return performance, both with relatively dense scatters.

Figure 4. Scatterplot of average monthly returns against out-of-sample R^2s of neural network portfolios.

The figure plots the average monthly returns (y-axis) achieved by neural network long-short top-bottom decile portfolios against the out-of-sample R^2s (x-axis) of the models. Each model is repeated 100 times, resulting in the 300 points plotted.

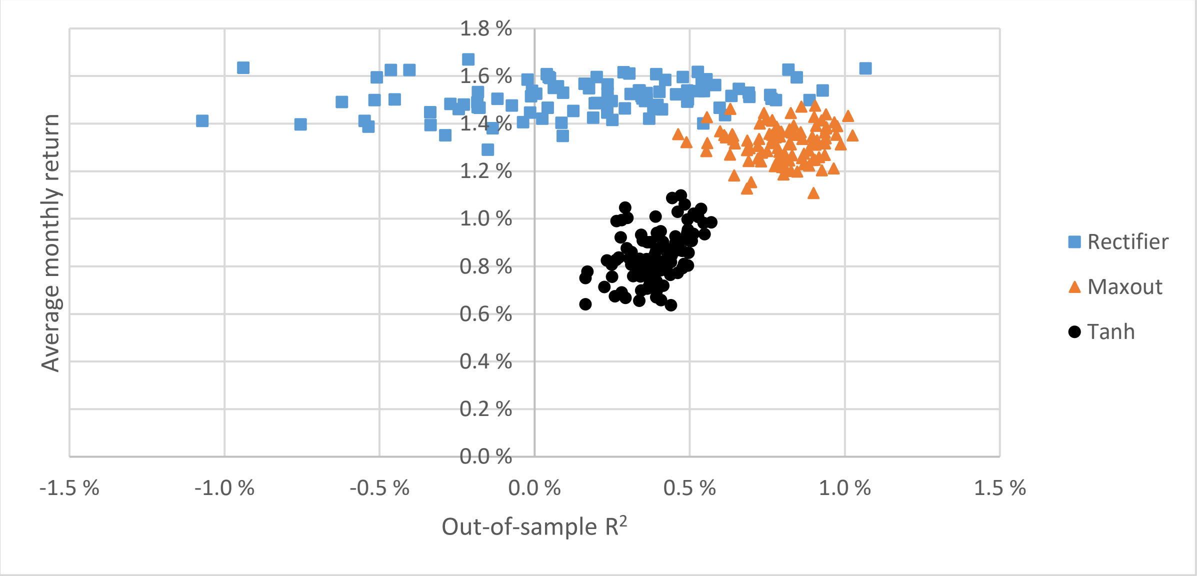
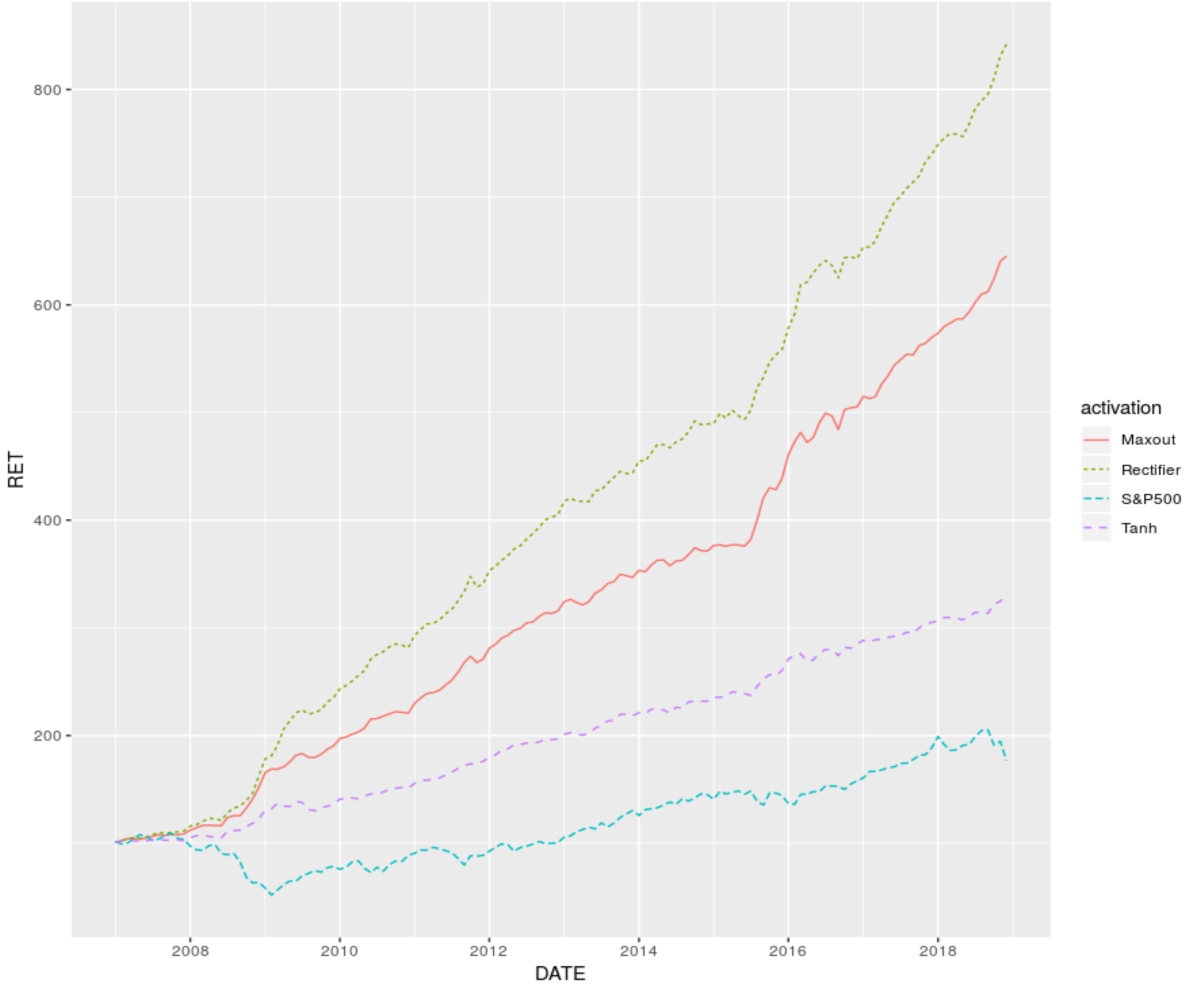


Figure 5 plots the time series of cumulative returns of the neural network portfolios compared to the S&P 500 index, starting from value 100. The graph shows that the neural network portfolios are able to perform consistently well across the whole time period, and particularly well during market turbulence. For example, during 2008-2009, the S&P 500 decreases significantly, while all three machine learning portfolios gain at a fast pace. Also, during mid 2015 to 2016, the S&P 500 faces some turbulence, but Maxout and Rectifier models only accelerate their gains. The findings support the earlier discovery from Figure 3 that R^2 performance seems to be better during times of higher market volatility measured by the VIX: returns gained from the predictions also seem to improve during market turbulence. This is in line with Dangl and Halling (2012) who find evidence of stronger S&P 500 return predictability during recessions.

Figure 5. Cumulative returns on neural network portfolios compared to the S&P 500 index (2007-2018).

The figure plots the development of the value of the three neural network portfolios (Maxout, Rectifier, Tanh) compared to the S&P 500 index, with 100 as their starting value, from 2007 to 2018. The effects of trading costs are not included.



5.8. *Variable importance*

Using the Gedeon (1997) method to extract information on relative predictor strength, I calculate the average of average variable importance for the three neural network models separately for their 100 iterations. The results are reported in Figure 6, where relative variable importance is measured on a standardized scale from 0 to 1. I note that, as pointed out by Sarle (2000), particularly drawing conclusions from evaluating relative importances of the same size from the Gedeon (1997) method is problematic, as relative importance of the same size through the method does not necessarily mean the variables are equally important. Thus, I focus my analysis on variables that have a significant importance difference.

Common to the three models, lagged market return is the single most important predictor, with all except Maxout having an average importance of 1 for the predictor across the 100 model iterations each. For example, when training Maxout models with a dataset excluding lagged market return, the average out-of-sample R^2 for 80 model iterations was -0.17%, return correlation 3.44%, and directional accuracy 50.50%, far lower than values reported in earlier tables. One explanation for the dominance of the market return is the structure of the panel data set: market return is the same for all stocks in monthly cross-sections, thus providing powerful information on the general level of the market last month. Also, since market return is the only macroeconomic predictor in my data set, it is reasonable that the models learn most of the systematic variation in stock prices through the variable. In essence, it is likely that the market return is used to predict the overall level of return, while the other firm variables are used to predict cross-sectional differences. Arguably, then, macroeconomic variables in conjunction with firm variables should be very important in generating individual stock return predictions.

Examining Figure 6 one model at a time, Rectifiers have nine variables with relative importance above 0.4 aside from market return: Four of the nine are related to momentum: 6-month stock momentum (*mom6m*), 1-month stock momentum (*mom1m*), industry momentum (*indmom*), and change in 6-month momentum (*chmom*). Three of the nine are related to liquidity: volatility of dollar trading volume (*std_dolvol*), zero trading days (*zerotrade*), and dollar trading volume (*dolvol*). One is related to analysts (number of analysts covering the stock (*nanalyst*), and the last is the market capitalization of the firm. The last 10 variables of the top 10 fall below relative importance of 0.4, and are more similar in scale, so inferences are less reliable in that range. Nevertheless, I observe that of the last 10, one is momentum-related (*mom12m*), three are liquidity related (*turn, std_turn, ill*), three are risk measures (*retvol*, *idiovol*, *beta*), two are firm fundamentals (*cash, lev*), and one is analyst forecasts (*fgr5yr*). Based on the top 10 variables, and supported by the bottom 10, it seems that Rectifiers focus on momentum and liquidity-based measures to generate forecasts.

The findings are very similar to those of Gu et al. (2018) and Messmer (2017), who both use Rectifier neural networks, and find momentum, liquidity and risk-based measures to be the most prominent. Also, Green et al. (2017) find 12 multivariately identified independent characteristics that provide information about average stock returns, out of which 7 are found in the top 20 of Rectifiers: cash, one-month momentum, change in 6-month momentum, return volatility (*retvol*), share turnover (*turn*), volatility of share turnover (*std_turn*), and zero trading days. The reliance on momentum and liquidity signals may explain the outperformance of Rectifiers in portfolio returns, where momentum is likely a key variable in identifying price trends, and especially in small stocks, where liquidity is likely to signal more about future returns. Notably, the inclusion of market capitalization (*mve*) in the top 10 would suggest Rectifiers learn and apply differences between big and small stocks.

For Maxout models, nine variables (aside from market return) have variable importance over 0.4, although much less distinctively than the Rectifier top 10. Two of the nine are risk measures: idiosyncratic return volatility (*idiovol*), return volatility (*retvol*). Four of the nine are firm fundamentals: cash, cash flow volatility (*stdcf*), % change in sales - % change in SG&A (*pchsale_pchxsga*), % change in gross margin - % change in sales (*pchgm_pchsale*). Three of the nine are related to market participants and analyst forecasts: earnings announcement returns (*ear*), dispersion in analyst forecasted EPS (*disp*), and change in forecasted EPS (*chfeps*). The bottom 10 are very similar in size compared to each other, so inferences are less robust, but we note anyhow that seven of the ten are firm fundamentals (*chtx, ms, hire, gma, pchsale_pchinvt, sgr, rdsale*), two of the ten are risk measures (*maxret, betasq*), and one is a valuation ratio (*roaq*). Based on the top 10, and supported by the bottom 10, it seems that Maxout models focus particularly on firm fundamentals and risk measures to generate forecasts – a very different picture from the momentum/liquidity-focused Rectifiers.

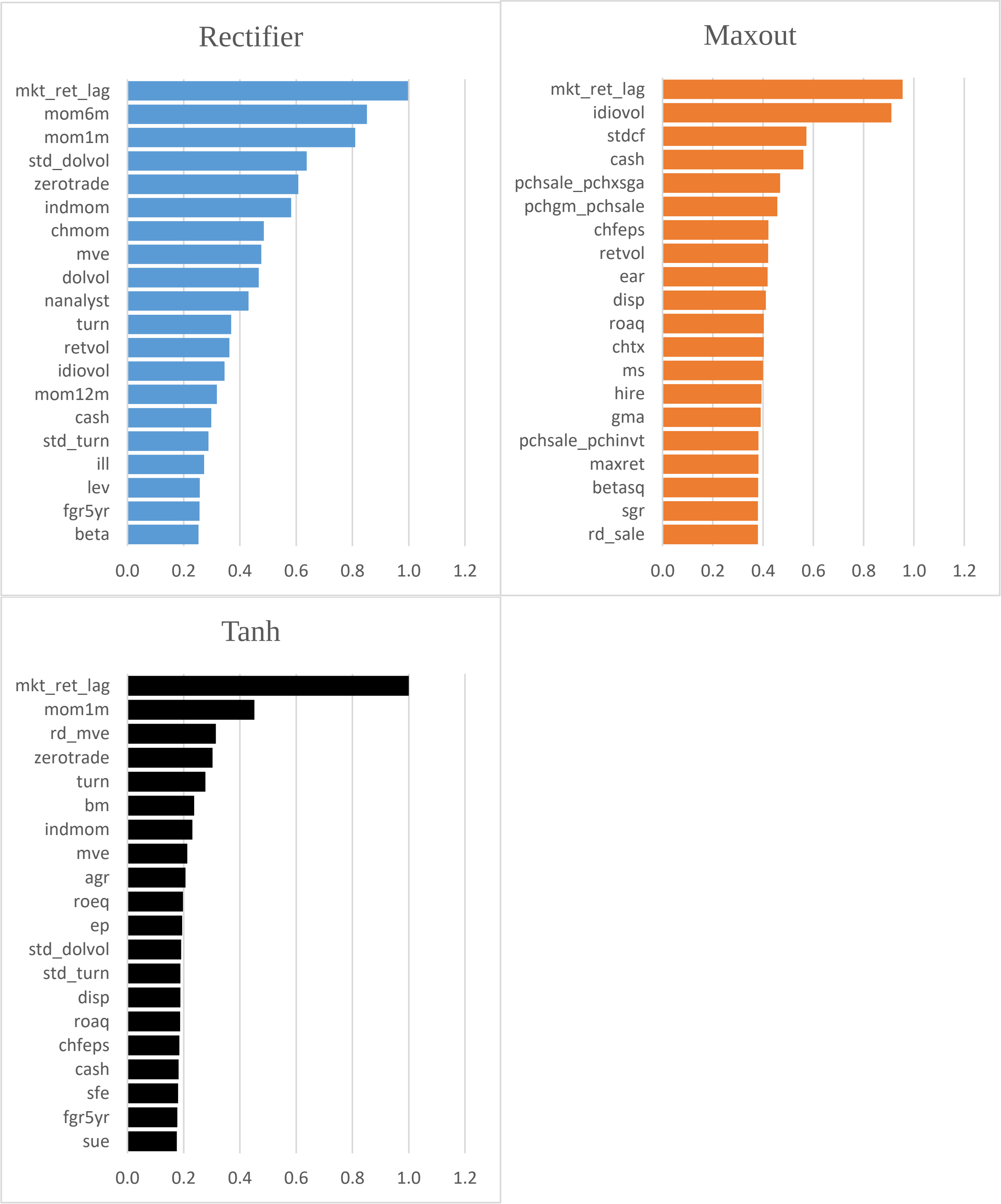
Interestingly, Maxout models seem to align more closely with the univariately significant independent characteristics that Green et al. (2017) find, instead of the multivariately significant characteristics such as Rectifiers. 10 of the 12 univariately significant characteristics are firm fundamental signals. This suggests Maxout models focus more on long-term predictors compared to the short-term focused Rectifiers. Overall, Maxouts use all predictors more, with the smallest relative importance being 0.2, compared to 0.08 for Rectifiers and 0.06 for Tanhs. This suggests that Maxout models are able to take advantage of more diverse predictive information, which may explain their outperformance in out-of-sample R^2, being less prone to overfit on a few types of predictors. It is also interesting that Maxout models do not have momentum variables at all in the top 20, which contradicts findings of Gu et al. (2018). The first momentum variable appears at rank 21, the 3-year momentum, followed by 1-year momentum at rank 23. The momentum variable importances in Maxout models are also in reverse order compared to the other two models, where Maxouts prefer longer time-frame momentum and other models short-term signals.

Lastly, Tanh models focus on a very narrow range of variables, with only one variable (1-month momentum) with relative importance above 0.4 in addition to market return. The other 18 in the top 20 have importances of roughly 0.2. Two of the 18 are fundamental signals: asset growth (*agr*) and cash. Six of the 18 are valuation ratios: book-to-market (*bm*), market cap (*mve*), return on equity (*roeq*), R&D to market cap (*rd_mve*), earnings to price (*ep*), return on assets (*roaq*). Four of the 18 are related to liquidity: zero trading days (*zerotrade*), share turnover (*turn*), volatility of dollar trading volume (*std_dolvol*) and volatility of share turnover (*std_turn*). Five of the 18 are analyst forecasts: dispersion of analyst forecasted EPS (*disp*), change in forecasted EPS (*chfeps*), scaled analyst earnings forecast (*sfe*), forecasted 5-year growth (*fgr5yr*), and unexpected quarterly earnings (*sue*). One of 18 is related to momentum: industry momentum (*indmom*). There is no clear category the 18 variables in Tanh models focus on, with the most important being valuation ratios, liquidity, and analyst forecasts. It is a much more mixed picture than the previous two. The underperformance of the Tanh models compared to the other two neural networks may be explained by its likelihood to overfit due to only using 1-month momentum in addition to market return. It seems the Tanh models are unable to fully learn predictive information from a more diverse set of predictors like the Maxout models.

Taken as a whole, it seems that macroeconomic variables can be powerful predictors of the systematic variation in individual stock returns through panel data, when used in conjunction with firm characteristics, shown by the high relative importance of the market return variable across the three neural networks. Interestingly, depending on the activation function chosen for the neural network, the models use different types of data for predictions: Rectifiers lean towards market trends with momentum and liquidity variables, Maxout towards firm fundamentals and risk measures, and Tanh lies somewhere in between. My findings confirm those of Gu et al. (2018), who also find momentum and liquidity variables to be the most important with neural networks using Rectifier activation functions. My findings also echo some of Green et al.'s (2017) findings, where many of the variables they identify to be significant providers of independent information on stock returns are also found in the top lists of my neural network predictors. The variable importance findings for Maxout models presents a challenge to the prevalent usage of Rectifiers as activation functions among researchers applying neural networks to finance. The findings are the first of its kind to my knowledge and emphasize the significant impact the choice of activation function can have on the entire model and how it learns. For the anomalies literature, I confirm the importance of momentum and liquidity variables, but also many firm fundamentals that Maxout models were able to widely use to successfully predict out-of-sample returns.

Figure 6. Variable importance by model (top 20 variables).

The figures display the top 20 most important variables in each model.



6. Conclusion

The analysis of neural network predictions contributes to the anomalies, return predictability, and machine learning literatures. For the anomalies literature, the analysis on variable importances confirms particularly the significance of momentum (e.g. 1-month momentum) and liquidity (e.g. standard deviation of dollar trading volume) variables, but also firm fundamentals (e.g. cash or standard deviation of cash flow) and risk measures (e.g. idiosyncratic volatility). Two neural network models provide contrasting pictures of which variables are the most important, with the popular Rectifier models focusing on momentum and liquidity, and Maxout models on firm fundamentals and risk measures. My findings are in line with Messmer (2017) and Gu et al. (2018), who also find momentum and liquidity variables to be the most important for Rectifier neural networks. Many of the top variables from both models also match with predictors Green et al. (2017) identify using conventional linear methods as informative of cross-sectional returns. Contradicting previous findings that posit only a very narrow set of predictors contain significant return predictive information, the Maxout models in this paper appear to use the entire predictor set of 103 predictors more evenly than other models. This suggests that depending on the model, a high-dimensional predictor set can still be valuable. Since neural networks that allow complex nonlinearities between inputs can find varying interpretations of predictor importance, traditional linear methods may be limited to certain views as well.

This study finds notable predictability of individual stock returns both measured by out-of-sample R^2 and profitability of a long-short decile portfolio. The results are supported by analysis on the correlation between predicted and realized returns as well as the directional accuracy of the predictions. The best neural networks outperformed linear benchmarks, but some more unstable neural networks underperformed in terms of R^2. However, all neural networks outperformed OLS models in portfolio profitability on long-short top-bottom decile portfolios sorted on predicted returns. Statistical performance and economic value of predictions are found to be higher during times of market turbulence, aligning with findings of Dangl and Halling (2012). A large portion of the predictability stems from the only macroeconomic variable available in the data, the lagged market return. This suggests individual stock return predictability benefits greatly from macroeconomic variables that are more explanatory of the systematic portion of return variation. This predictability is highlighted when aggregating individual stock returns into average monthly market return predictions, where abnormally large out-of-sample R^2s were documented. The results of this paper that compare predictability between all stocks, top 500 largest stocks, and bottom 500 smallest stocks confirm earlier findings in literature (e.g. Chordia, Roll, and Subrahmanyam (2008) or Green et al. (2017)) that return predictability and anomalies are stronger in illiquid small stocks, and sometimes completely absent from large stocks.

The profitability results found for the neural network long-short portfolios should be interpreted more as a theoretical economic perspective to the value of neural network predictions rather than actual realizable profits, as transaction costs are not taken into account. For example, for Rectifier models that lean on momentum and liquidity predictors, their trading activity is likely to be similar to momentum strategies. Portfolio turnover is likely to be high, as in the All stocks portfolio the top and bottom deciles together contain on average almost 800 stocks each month, ranging from large caps to microcaps. Lesmond, Schill, and Zhou (2004) show that theoretical momentum trading strategy profits all but disappear after accounting for transaction costs, and that stocks that generate large momentum returns are precisely those stocks with high trading costs, such as small stocks. As my results show that predictability and profitability is strongest among small stocks, it is likely that a large proportion of profits is generated from smaller, more illiquid stocks. Thus, realizable profits after trading costs is significantly lower than the theoretical profits reported.

For the machine learning literature, this paper contributes by providing an overview of how varying neural network topologies perform in learning from highly noisy data, and the effects of randomness. Based on the results, neural network topology plays a key role in the stability, general level of predictions, as well as how the model learns from predictors. First, the results of hyperparameter tuning through grid search show that optimizing hyperparameters has a sizeable impact on the model performance. Second, the analysis on variable importance findings reveals that Maxout models are better at using the entire predictor set rather than narrowly focus on few, like the Rectifier and Tanh models. Models using the popular Rectifier activation function were highly unstable in our dataset, similarly observed by Messmer (2017), producing highly varied predictions on different iterations. Deeper than one-layer topologies effectively did not work for Rectifiers, as most deeper iterations failed in training during random grid search. While the prediction performance of Rectifiers had high variance measured by out-of-sample R^2, the spread was much smaller with the other two activation functions studied, Maxout and Tanh, showing denser scatters around mean results. On the other hand, despite the Rectifiers' unstable statistical prediction performance, they still outperformed all other models in profitability measures – a somewhat surprising finding. Even Rectifiers with very poor out-of-sample R^2 could achieve better profitability results than the other models. Combining the inherent randomness of neural network iterations and the difficulty of evaluating prediction performance based on statistical measures such as out-of-sample R^2, I caution future research on drawing conclusions from naïve model compositions without rigorous hyperparameter optimization, sufficient re-iteration to investigate variance, and critical evaluation of a diverse set of performance measures. Specifically, the choice of activation function seems to be particularly influential on not only performance, but the entire way the model learns from the data.

7. References

Avramov, D. and Chordia, T. (2006). Predicting stock returns. *Journal of Financial Economics, 82*, 387-415.

Banz, R. W. (1981). The relationship between return and market value of common stocks. *Journal of Financial Economics, 9*(1), 3-18.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*(2), 281-305.

Bossaerts, P. and Hillion, P. (1999). Implementing statistical criteria to select return forecasting models: What do we learn? *Review of Financial Studies, 12*(2), 405-428.

Brownlee, J. (2016). Embrace randomness in machine learning. Retrieved from: https://machinelearningmastery.com/randomness-in-machine-learning/

Campbell, J. and Thompson, S. (2008). Predicting excess stock returns out of sample: Can anything beat the historical average? *Review of Financial Studies, 21*(4), 1509-1531.

Carhart, M. M. (1997). On Persistence in mutual fund performance. *Journal of Finance, 52*(1), 57-82.

Cao, Q., Leggio, K. B., and Schniederjans, M. J. (2005). A comparison between Fama and French's model and artificial neural networks in predicting the Chinese stock market. *Computers and Operations Research, 35*(10), 2499-2512.

Chen, Y-M., Lin, Y-C., Tsai, C-F., and Yen, D. (2011). Predicting stock returns by classifier ensembles. *Applied Soft Computing, 11,* 2452-2459.

Chordia, T., Roll, R., and Subrahmanyam, A. (2008). Liquidity and market efficiency. *Journal of Financial Economics, 87,* 249-268.

Chordia, T., Goyal, A., and Saretto, A. (2018). P-Hacking: Evidence from two million trading strategies. *Swiss Finance Institute Research Paper No. 17-37*.

Cochrane, J. H. (2011). Presidential address: Discount rates. *Journal of Finance, 66*(4), 1047-1108.

Dangl, T. and Halling, M. (2012). Predictive regressions with time-varying coefficients. *Journal of Financial Economics, 106,* 157-181.

Fama, E. F., and French, K. R. (1988). Dividend yields and expected stock returns. *Journal of Financial Economics*, *22*(1), 3-25.

Fama, E. F., and French, K. R. (1992). The cross-section of expected stock returns. *Journal of Finance, 47*(2), 427-465

Fama, E. F., and French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics, 33*(1), 3-56.

Fama, E.f., and French, K. R. (1996). Multifactor explanations of asset pricing anomalies. *Journal of Finance, 51*(1), 55-84.

Fama, E. F., and French, K. R. (2006). Profitability, investment and average returns. *Journal of Financial Economics, 82*, 491-518.

Feng, G., Giglio, S., and Xiu, D. (2017). Taming the factor zoo. *Chicago Booth Research Paper No. 17-04*.

Ferreira, M. A., and Santa-Clara, P. (2011). Forecasting stock market returns: The sum of the parts is more than the whole. *Journal of Financial Economics*, *100*(3), 514-537

Freyberger, J., Neuhierl, A., and Weber, M. (2017). Dissecting characteristics nonparametrically. *NBER Working paper.*

Gedeon, T. D. (1997). Data mining of inputs: Analysing magnitude and functional measures. *International Journal of Neural Systems, 8*(2), 209-218.

Giglio, S., and Xiu, D. (2017). Inference on risk premia in the presence of omitted factors. *NBER Working paper.*

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. *MIT press.*

Green, J., Hand, J. RM, and Zhang, X. F (2013). The supraview of return predictive signals. *Review of Accounting Studies, 18,* 692-730.

Green, J., Hand, J., and Zhang, X. F. (2017). The characteristics that provide independent information about average US monthly stock returns. *Review of Financial Studies, 30,* 4389-4436

Goyal, A., and Welch, I. (2008). A comprehensive look at the empirical performance of equity premium prediction. *Review of Financial Studies, 21,* 1455-1508.

Gu, S., Kelly, B. T., and Xiu, D. (2018). Empirical asset pricing via machine learning. *Chicago Booth Research Paper*.

Henkel, S. J., Martin, J. S., and Nardari, F. (2011). Time-varying short-horizon predictability. *Journal of Financial Economics, 99*(3), 560-580.

Hou, K., and Moskowitz, T. J. (2005). Market frictions, price delay, and the cross section of expected returns. *Review of Financial Studies, 18,* 981-1020.

Hou, K., Xue, C., and Zhang, L. (2017). Replicating anomalies. *NBER Working Paper 23394*.

Jegadeesh, N. and Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *Journal of Finance, 48*(1), 65-91.

Jordan, S. J., Vivian, A. J., and Wohar, M. E. (2014). Forecasting returns: New European evidence. *Journal of Empirical Finance, 26,* 76-95.

Kelly, B., and Pruitt, S. (2013). Market expectations in the cross-section of present values. *Journal of Finance, 68*(5), 1721-1756.

Kelly, B., Pruitt, S., and Su, Y. (2018). Characteristics are covariances: A unified model of risk and return. *SSRN Working paper.*

Kozak, S., Nagel, S., and Santosh, S. (2017). Shrinking the cross section. *NBER Working paper.*

Krauss, C., Do, X. A., and Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research, 259*(2), 689-702.

Lam, F. Y. E. C., and Wei, K. C. J. (2011). Limits-to-arbitrage, investment frictions, and the asset growth anomaly. *Journal of Financial Economics, 102,* 127-149.

Lantz, B. (2015). *Machine Learning with R.* Packt Publishing Ltd

Lesmond, D. A., Schill, M. J., and Zhou, C. (2004). The illusory nature of momentum profits. *Journal of Financial Economics, 71,* 349-380.

Li, D., and Zhang, L. (2010). Does q-theory with investment frictions explain anomalies in the cross section of returns? *Journal of Financial Economics, 98*, 297-314.

Lintner, J. (1965). Security prices, risk, and maximal gains from diversification. *Journal of Finance, 20*(4), 587-615.

Masters, T. (1993). Practical Neural Network Recipes in C++. *Academic Press*.

McLean, R. D., and Pontiff, J. (2016). Does academic research destroy stock return predictability? *Journal of Finance, 71*(1), 5-32.

Messmer, M. (2017). Deep learning and the cross-section of expected returns. *University of St. Gallen SSRN Working paper*.

Neely, C. J., Rapach, D., and Zhou, G. (2014). Forecasting the equity risk premium: The role of technical indicators. *Management Science, 60*(7), 1617-1859.

Niu, F., Recht, B., Re, C., and Wright, S. J. (2011). HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information processing Systems*, pp. 693-701, *arXiv:1106.5730*

Pesaran, M. H. and Timmermann, A. (1995). Predictability of stock returns: Robustness and economic significance. *Journal of Finance, 50*(4), 1201-1228.

Quesada, A. (n.d.). *5 algorithms to train a neural network* [Web blog post]. Retrieved from https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv: 170.05941*, *Cornell University*.

Rapach, D., Strauss, J., and Zhou, G. (2010). Out-of-sample equity premium prediction: Combination forecasts and links to the real economy. *Review of Financial Studies, 23*(2), 821-862.

Rapach, D. and Zhou, G. (2013). Chapter 6 – Forecasting stock returns. *Handbook of Economic Forecasting, Volume 2 Part A,* 328-383.

Rosenberg, B., Reid, K., and Lanstein, R. (1985). Persuasive evidence of market inefficiency. *Journal of Portfolio Management*, *11*, 9-17.

Sarle, W. S. (2000). How to measure importance of inputs? SAS Institute, retrieved from: ftp://ftp.sas.com/pub/neural/importance.html

Sarle, W. S. (2002). *Comp.ai.neural-nets FAQ*. Retrieved from http://www.faqs.org/faqs/ai-faq/neural-nets/part2/

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *Journal of Finance, 19*(3), 425-442.

Sun, C. (2018). Regularising the factor zoo with OWL. *Working Paper*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I, and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15, 1929-1958.*

Tsai, C. F., Lin, Y. C., Yen, D. C., and Chen, Y. M. (2011). Predicting stock returns by classifier ensembles. *Applied Soft Computing, 11*(2), 2452-2459.

Wang, Q. K. and Xu, J. (2015). Market volatility and momentum. *Journal of Empirical Finance, 30,* 79-91.

Welch, I. and Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. *Review of Financial Studies*, *21*(4), 1455-1508.

Xu, Y. (2004). Small levels of predictability and large economic gains. *Journal of Empirical Finance*, *11*, 247-275.

Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, *14*(1), 35-62.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. *arXiv: 1212.5701, New York University*.

Zhou, G. (2010). How much stock return predictability can we expect from an asset pricing model? *Economics Letters*, *108*, 184-186.

Zygmunt, Z. (2016). *What is better: gradient-boosted trees, or a random forest?* Retrieved from http://fastml.com/what-is-better-gradient-boosted-trees-or-random-forest/

8. Appendix

Table A1. Information about the predictor variables used in predicting stock returns.

The dataset consists of 2,075,872 rows of data and a total of 102 firm characteristic columns and one lagged market return column. The firm characteristic data is retrieved using the SAS code from Jeremiah Green's website, and the lagged market return is calculated based on the Wilshire 5000 Total Market Full Cap Index retrieved from the economic research data base at the Federal Reserve Bank at St. Louis.

Acronym	Author	Date, Journal	Definition	% missing
absacc	Bandyopadhyay, Huang, and Wirjanto	2010, WP	Absolute value of acc	13.3
acc	Sloan	1996, TAR	Annual income before extraordinary items (*ib*) minus operating cash flows (*oancf*) divided by average total assets (*at*); if *oancf* is missing then set to change in *act* – change in *che* – change in *lct* + change in *dlc* + change in *txp-dp*	13.3
aeavol	Lerman, Livnat, and Mendenhall	2008, WP	Average daily trading volume (*vol*) for 3 days around earnings announcement minus average daily volume for 1-month ending 2 weeks before earnings announcement divided by1-month average daily volume. Earnings announcement day from Compustat quarterly (*rdq*)	10.8
age	Jiang, Lee, and Zhang	2005, RAS	Number of years since first Compustat coverage	0
agr	Cooper, Gulen, and Schill	2008, JF	Annual percentage change in total assets (*at*)	6.6
baspread	Amihud and Mendelson	1989, JF	Monthly average of daily bid-ask spread divided by average of daily spread	0
beta	Fama and Macbeth	1973, JPE	Estimated market beta from weekly returns and equal weighted market returns for 3 years ending month *t-1* with at least 52 weeks of returns	1.0
betasq	Fama and Macbeth	1973, JPE	Market beta squared	1.0
bm	Rosenberg, Reid, and Lanstein	1985, JPM	Book value of equity (*cep*) divided by end of fiscal year-end market capitalization	0
bm_ia	Asness, Porter, and Stevens	2000, WP	Industry-adjusted book-to-market ratio	0
cash	Palazzo	2012, JFE	Cash and cash equivalents divided by average total assets	10.7
cashdebt	Ou and Penman	1989, JAE	Earnings before depreciation and extraordinary items (*ib*+*dp*) divided by avg. total liabilities (*lt*)	3.5
cashpr	Chandrashekar and Rao	2009, WP	Fiscal year-end market capitalization plus long-term debt (*dltt*) minus total assets (*at*) divided by cash and equivalents (*che*)	1.0
cfp	Desai, Rajgopal, and Venkatachalam	2004, TAR	Operating cash flows divided by fiscal-year-end market capitalization	7.7
cfp_ia	Asness, Porter and Stevens	2000, WP	Industry-adjusted *cfp*	7.7
chatoia	Soliman	2008, TAR	2-digit SIC.fiscal-year mean-adjusted change in sales (*sale*) divided by average total assets (*at*)	13.9

chcsho	Pontiff and Woodgate	2008, JF	Annual percent change in shares outstanding (*csho*)	6.6
chempia	Asness, Porter, and Stevens	1994, WP	Industry-adjusted change in number of employees	6.8
chfeps	Hawkins, Chamberlin, and Daniel	1984, FAJ	Mean analyst forecast in month prior to fiscal period end date from I/B/E/S summary file minus same mean forecast for prior fiscal period using annual earnings forecasts	46.4
chinv	Thomas and Zhang	2002, RAS	Change in inventory (*inv*) scaled by average total assets (*at*)	9.0
chmom	Gettleman and Marks	2006, WP	Cumulative returns from months *t-6* to *t-1* minus months *t-12* to *t-7*	7.1
chanalyst	Scherbina	2008, RF	Change in *nanalyst* from month *t-3* to month *t*	23.1
chpmia	Soliman	2008, TAR	2-digit SIC-fiscal-year mean adjusted change in income before extraordinary items (*ib*) divided by sales (*sale*)	8.1
chtx	Thomas and Zhang	2011, JAR	Percent change in total taxes (*txtq*) from quarter *t-4* to t	11.9
cinvest	Titman, Wei, and Xie	2004, JFQA	Change over one quarter in net PP&E (*ppentq*) divided by sales (*saleq*) – average of this variable for prior 3 quarters; if *saleq* = 0, then scale by 0.01	12.2
convind	Valta	2016, JFQA	An indicator equal to 1 if company has convertible debt obligations	0
currat	Ou and Penman	1989, JAE	Current assets / current liabilities	3.3
depr	Holthausen and Larcker	1992, JAE	Depreciation divided by PP&E	4.3
disp	Diether, Malloy, and Scherbina	2002, JF	Standard deviation of analyst forecasts in month prior to fiscal period end date divided by the absolute value of the mean forecast; if *meanest* = 0, then scalar set to 1. Forecast data from I/B/E/S summary files	55.4
divi	Michaely, Thaler, and Womac	1995, JF	An indicator variable equal to 1 if company pays dividends but did not in prior year	6.6
divo	Michaely, Thaler, and Womack	1995, JF	An indicator variable equal to 1 if company does not pay dividend but did in prior year	6.6
dolvol	Chordia, Subrahmanyam, and Anshuman	2001, JFE	Natural log of trading volume times price per share from month *t-2*	3.6
dy	Litzenberger and Ramaswamy	1982, JF	Total dividends (*dvt*) divided by market capitalization at fiscal year-end	0.3
ear	Kishore et al.	2008, WP	Sum of daily returns in three days around earnings announcement. Earnings announcement from Compustat quarterly file (*rdq*)	10.2
egr	Richardson et al.	2005, JAE	Annual percent change in book value of equity (*ceq*)	6.6
ep	Basu	1977, JF	Annual income before extraordinary items (*ib*) divided by end of fiscal year market cap	0
fgr5yr	Bauman and Dowen	1988, FAJ	Most recently available analyst forecasted 5-year growth	60.2
gma	Novy-Marx	2013, JFE	Revenues (*revt*) minus cost of goods sold (*cogs*) divided by lagged total assets (*at*)	6.8

grcapx	Anderson and Garcia-Feijoo	2006, JF	Percent change in capex from year *t-2* to year *t*	16.1
grltnoa	Fairfiel, Whisenant, and Yohn	2003, TAR	Growth in long-term net operating assets	29.6
herf	Hou and Robinson	2006, JF	2-digit SIC-fiscal-year sales concentration (sum of squared percent of sales in industry for each company)	0
hire	Bazdresch, Belo, and Lin	2014, JPE	Percent change in number of employees (*emp*)	6.8
idiovol	Ali, Hwang, and Trombley	2003, JFE	Standard deviation of residuals of weekly returns on weekly equal weighted market returns for 3 years prior to month end	1.0
ill	Amihud	2002, JFM	Average of daily (absolute return / dollar volume)	3.0
indmom	Moskowitz and Grinblatt	1999, JF	Equal-weighted average industry 12-month returns	0
invest	Chen and Zhang	2010, JF	Annual change in gross property, plant, and equipment (*ppegt*) + annual change in inventories (*invt*) all scaled by lagged total assets (*at*)	9.7
IPO	Loughran and Ritter	1995, JF	An indicator variable equal to 1 if first year available on CRSP monthly stock file	0
lev	Bhandari	1988, JF	Total liabilities (*lt*) divided by fiscal year-end market capitalization	0.3
lgr	Richardson et al.	2005, JAE	Annual percent change in total liabilities (*lt*)	6.9
maxret	Bali, Cakici, and Whitelaw	2011, JFE	Maximum daily return from returns during calendar month *t-1*	0
mom12m	Jegadeesh	1990, JF	11-month cumulative returns ending one month before month end	7.1
mom1m	Jegadeesh and Titman	1993, JF	1-month cumulative return	0
mom36m	Jegadeesh and Titman	1993, JF	Cumulative returns from months *t-36* to *t-13*	21.8
mom6m	Jegadeesh and Titman	1993, JF	5-month cumulative returns ending one month before month end	2.9
ms	Mohanram	2005, RAS	Sum of 8 indicator variables for fundamental performance	10.1
mve	Banz	1981, JFE	Natural log of market capitalization at end of month *t-1*	0
mve_ia	Asness, Porter, and Stevens	2000, WP	2-digit SIC industry-adjusted fiscal year-end market capitalization	0
nanalyst	Elgers, Lo, and Pfeiffer	2001, TAR	Number of analyst forecasts from most recently available I/B/E/S summary files in month prior to month of portfolio formation. *nanalyst* set to zero if not covered in I/B/E/S summary file	21.8
nincr	Barth, Elliott, and Finn	1999, JAR	Number of consecutive quarters (up to eight quarters) with an increase in earnings (*ibq*) over same quarter in the prior year	10.1
operprof	Fama and French	2015, JFE	Revenue minus cost of goods sold minus SG&A expense minutes interest expense divided by lagged common shareholders' equity	6.8
orgcap	Eisfeldt and Papanikolaou	2013, JFE	Capitalized SG&A expenses	26.5
pchcapx_ia	Abarbanell and Bushee	1998, TAR	2-digit SIC-fiscal-year mean-adjusted percent change in capital expenditures (*capx*)	9.2
pchcurrat	Ou and Penman	1989, JAE	Percent change in *currat*	10.0
pchdepr	Holthausen and Larcker	1992, JAE	Percent change in *depr*	11.0

pchgm_pchsale	Abarnell and Bushee	1998, TAR	Percent change in gross margin (*sale-cogs*) minus percent change in sales (*sale*)	7.9
pchquick	Ou and Penman	1989, JAE	Percent change in *quick*	10.6
pchsale_pchinvt	Abarbanell and Bushee	1998, TAR	Annual percent change in sales (*sale*) minus annual percent change in receivables (*rect*)	26.5
pchsale_pchrect	Abarbanell and Bushee	1998, TAR	Annual percent change in sales (*sale*) minus annual percent change in receivables (*rect*)	10.5
pchsale_pchxsga	Abarbanell and Bushee	1998, TAR	Annual percent change in sales (*sale*) minus annual percent change in SG&A (*xsga*)	22.4
pchsaleinv	Ou and Penman	1989, JAE	Percent change in *saleinv*	27.4
pctacc	Hafzalla, Lundholm, and Van Winkle	2011, TAR	Same as *acc* except that the numerator is divided by the absolute value of *ib*; if *ib* = 0 then *ib* set to 0.01 for denominator	13.3
pricedelay	Hou & Moskowitz	2005, RFS	The proportion of variation in weekly returns for 36 months ending in month *t* explained by 4 lags of weekly market returns incremental to contemporaneous market return	1.0
ps	Piotroski	2000, JAR	Sum of 9 indicator variables to form fundamental health score	6.6
quick	Ou and Penman	1989, JAE	(current assets – inventory) / current liabilities	3.8
rd	Eberhart, Maxwell, and Siddique	2004, JF	An indicator variable equal to 1 if R&D expense as a percentage of total assets has an increase greater than 5%	6.6
rd_mve	Guo, Lev, and Shi	2006, JBFA	R&D expense divided by end-of-fiscal-year market capitalization	51.4
rd_sale	Guo, Lev, and Shi	2006, JBFA	R&D expense divided by sales (*xrd*/*sale*)	52.3
realestate	Tuzel	2010, RFS	Buildings and capitalized leases divided by gross PP&E	57.7
retvol	Ang et al.	2006, JF	Standard deviation of daily returns from month *t-1*	0
roaq	Balakrishnan, Bartov, and Faurel	2010, JAE	Income before extraordinary items (*ibq*) divided by one quarter lagged total assets *(atq)*	10.3
roavol	Francis et al.	2004, TAR	Standard deviation for 16 quarters of income before extraordinary items (*ibq*) divided by average total assets (*atq*)	23.7
roeq	Hou, Xue, and Zhang	2015, RFS	Earnings before extraordinary items divided by lagged common shareholders' equity	10.3
roic	Brown and Rowe	2007, WP	Annual earnings before interest and taxes (*ebit*) minus non-operating income (*nopi*) divided by non-cash enterprise value (*ceq* + *lt* – *che*)	4.1
rsup	Kama	2009, JBFA	Sales from quarter t minus sales from quarter *t-4* (*saleq*) divided by fiscal-quarter-end market capitalization (*cshoq* * *prccq*)	10.8
salecash	Ou and Penman	1989, JAE	Annual sales divided by cash and cash equivalents	0.8
saleinv	Ou and Penman	1989, JAE	Annual sales divided by total inventory	21.6
salerec	Ou and Penman	1989, JAE	Annual sales divided by accounts receivable	3.6
secured	Valta	2016, JFQA	Total liability scaled secured debt	41.2
securedind	Valta	2016, JFQA	An indicator equal to 1 if company has secured debt obligations	0
sfe	Elgers, Lo, and Pfeiffer	2001, TAR	Analysts mean annual earnings forecast for nearest upcoming fiscal year from most recent month available prior to month of portfolio formation from I/B/E/S summary files scaled by price per share at fiscal quarter end	46.9

sgr	Lakonishok, Shleifer, and Vishny	1994, JF	Annual percent change in sales (*sale*)	7.9
sin	Hong & Kacperczyk	2009, JFE	An indicator variable equal to 1 if a company's primary industry classification is in smoke or tobacco, beer or alcohol, or gaming	0
sp	Barbee, Mukherji, and Raines	1996, FAJ	Annual revenue (*sale*) divided by fiscal year-end market capitalization	0.3
std_dolvol	Chordia, Subrahmanyam, and Anshuman	2001, JFE	Monthly standard deviation of daily dollar trading volume	3.2
std_turn	Chordia, Subrahmanyam, and Anshuman	2001, JFE	Monthly standard deviation of daily share turnover	2.9
stdacc	Bandyopadhyay, Huang, and Wirjanto	2010, WP	Standard deviation for 16 quarters of accruals (*acc* measured with quarterly Compustat) scaled by sales; if *saleq* = 0, then scale by 0.01	36.5
stdcf	Huang	2009, JEF	Standard deviation for 16 quarters of cash flows divided by sales (*saleq*); if *saleq* = 0, then scale by 0.01. Cash flows defined as *ibq* minus quarterly accruals	36.5
sue	Rendelman, Jones, and Latane	1982, JFE	Unexpected quarterly earnings divided by fiscal-quarter-end market cap. Unexpected earnings is I/B/E/S actual earnings minus media forecasted earnings if available, else it is the seasonally differenced quarterly earnings before extraordinary items from Compustat quarterly file	10.7
tang	Almeida and Campello	2007, RFS	Cash holdings + 0.715 * receivables + 0.547 * inventory + 0.535 * PPE/total assets	4.1
tb	Lev and Nissim	2004, TAR	Tax income, calculated from current tax expense divided by maximum federal tax rate, divided by income before extraordinary items	11.9
turn	Data, Naik, and Radcliffe	1998, JFM	Average monthly trading volume for most recent 3 months scaled by number of shares outstanding current month	3.5
zerotrade	Liu	2006, JFE	Turnover weighted number of zero trading days for most recent month	3.0

Table A2. Second hyperparameter grid search results comparing average validation sample R^2 (%) across parameter values.

The values reported are average R^2 (%) based on the validation data for models including the hyperparameter value defined in the left-most column. Rectifier, Maxout, and Tanh models are compared that differ by their activation function. The random grid search trains 150 models out of 288 possibilities (3 x 3 x 2 x 2 x 2 x 2 x 2), and grid search is repeated 10 times, producing a total of 1500 models.

	Rectifier	Maxout	Tanh
Overall	1.75 (0.57)	1.27 (0.31)	0.71 (0.12)
Hidden layers and units per layer			
[64]	2.00 (0.63)	1.20 (0.24)	0.69 (0.12)
[64, 32]	1.49 (0.34)	1.35 (0.34)	0.73 (0.12)
Input dropout			
0	1.79 (0.56)	1.31 (0.31)	0.72 (0.12)
0.1	1.71 (0.57)	1.24 (0.30)	0.70 (0.12)
L1			
0	1.73 (0.61)	1.28 (0.33)	0.71 (0.13)
1.00E-05	1.78 (0.52)	1.27 (0.28)	0.71 (0.12)
L2			
1.00E-04	1.75 (0.51)	1.24 (0.28)	0.71 (0.12)
1.00E-05	1.76 (0.63)	1.30 (0.32)	0.72 (0.12)
Rho			
0.99	1.30 (0.17)	1.01 (0.09)	0.65 (0.07)
0.999	2.21 (0.45)	1.53 (0.21)	0.78 (0.13)
Epsilon			
1.00E-09	1.76 (0.54)	1.22 (0.27)	0.67 (0.12)
1.00E-10	1.75 (0.60)	1.32 (0.33)	0.76 (0.10)