Deep Learning Factor Alpha*

Guanhao Feng† — *College of Business*, *City University of Hong Kong*

Nicholas G. Polson ‡ — *Booth School of Business*, *University of Chicago*

Jianeng Xu§ — *Booth School of Business*, *University of Chicago*

Link to the Latest Version

Abstract

Does a factor model exist that explains all anomalies? To address this questions we provide an automated factor search algorithm that generates long-short spread factors within deep learning. This performs a deep search over a multi-layer space of multivariate characteristics used for security sorting with one asset pricing objective: minimizing alphas. Sorting securities on firm characteristics, a common practice in finance, can be viewed as a nonlinear activation function. Our deep factor are trained greedily and we use a time series regression with deep factors over a benchmark models, such as Fama-French. We have designed a train-validation-test study for monthly U.S. equity returns from 1975 to 2017 and 56 published firm characteristics. In an out-of-sample evaluation, deep factor alpha provides a forecasting improvement over a benchmark with factors that offer significant alphas.

Key Words: Characteristic-based Anomalies, Cross-Sectional Returns, Deep Learning, Long-Short Factors, Security Sorting, Mispricing Alpha, Neural Network.

*We appreciate insightful comments from Li Deng and Dacheng Xiu. We are also grateful to helpful comments from seminar and conference participants at R/Finance 2018, and EcoSta 2018. We acknowledge research support from the Unigestion Alternative Risk Premia Research Academy.

†Address: 83 Tat Chee Avenue, Kowloon Tong, Hong Kong. E-mail address: gavin.feng@cityu.edu.hk.

‡Address: 5807 S Woodlawn Avenue, Chicago, IL 60637, USA. E-mail address: ngp@chicagobooth.edu.

§Address: 5807 S Woodlawn Avenue, Chicago, IL 60637, USA. E-mail address: jianeng@uchicago.edu.

One of our central themes is that if assets are priced rationally, variables that are related to average returns, such as size and book-to-market equity, must proxy for sensitivity to common (shared and thus undiversifiable) risk factors in returns.

In such regressions, a well-specified asset-pricing model produces intercepts that are indistinguishable from 0 [Merton (1973)]. The estimated intercepts provide a simple return metric and a formal test of how well different combinations of the common factors capture the cross section of average returns.

— Fama and French (1993)

1 Introduction

Empirical finance researchers have identified many patterns or anomalies in cross-sectional average stock returns[1]. The standard protocol is to build portfolios that sort on firm characteristics (size, earnings/price, book-to-market equity, etc.) and to test its "alpha" by regressing the long-short spread of sorted portfolios over a benchmark, such as CAPM. A significant intercept indicates the long-short spread factor is not spanned by the benchmark and could mean an anomaly. A subsequent test is to check if this factor proxies for some common risk factor and explains the average returns. Fama and French (1992) are pioneers in the field and show different measures of firm characteristics related to the size and value anomalies line up with average returns of sorted portfolios. Fama and French (1993) add Small-minus-Big and High-minus-Low to explain these two anomalies by substantially reducing the average pricing errors.

Our paper provides an automated algorithm that generates characteristic-based factors, and a parsimonious model of returns and average returns that absorbs anomalies and cross-sectional pricing errors. The factor model provides a dimension-reduction formulation that summarizes the time series variation of thousands of securities to a few factors. Characteristic-based long-short spread factors are popular because they reflect compensation for exposure to underlying risk factors and can be tested by the regression intercept alpha as a tradable portfolio. However, many of these characteristics are highly related to each other from the perspective of accounting, trading, or economics. Therefore, how their sorted portfolios have a vast difference due to the minor construction difference in characteristics is unclear.

[1] See Harvey, Liu, and Zhu (2016), Green, Hand, and Zhang (2017) and Hou, Xue, and Zhang (2017)

Given the zoo of factors or characteristics, we attempt to answer a similar question in Fama and French (1996): Does a factor model exist that dissects the existing anomalies? To provide a unified framework for the characteristic-based factor generation, we propose the use of a deep learning algorithm to explore the multi-layer space of multivariate characteristics for security sorting and generate long-short spread factors with a pure optimization perspective: minimizing alphas.

First, the widely used security sorting is a quantile function and can be viewed as a nonlinear activation function within the model training. Second, our algorithm searches for the best transformation and combination of firm characteristics while controlling for a benchmark model, and thus deep factors are not spanned by the benchmark. Third, it is a greedy algorithm that takes the feedback of the loss function through the channel of backward propagation: how to change the sorted characteristics to decrease the average pricing errors further. Forth, when a new characteristic is proposed, we can test the incremental contribution of its optimal long-short spread beyond a benchmark model.

To generate the deep factors, we use the stock universe with the annual largest 3000 firms in the U.S. equity market and 56 published characteristics from Green et al. (2017). In an out-of-sample evaluation, we use the period 1975-2004 to train the model and generate deep factor to predict the period 2005-2017. Our deep factor model outperform the benchmark CAPM, FF3, and FF5 by 12.5%, 1%, and 2.1% in the relative out-of-sample R-squared over CAPM.

The rest of the paper is outlined as follows. We have a brief introduction about the method in section 1.1 and a comprehensive comparison with the related literature in section 1.2. Section 2 adds an introduction to the neural network and how we interpret Fama-French models in deep learning. The details of the Deep Factor Alpha algorithm are discussed in section 3. Section 4 illustrates the empirical study design and the findings for asset pricing testing. Section 5 discusses the automated generation of deep factors and the future application in empirical asset pricing.

1.1 Deep Factor Methodology

In this paper, we employ the classical time series regression approach by regressing excess asset returns on the returns of the generated deep factors F_t, and a benchmark model with tradable

factors G_t, such as CAPM or Fama-French models.

$$R_{i,t} = \alpha_i + \beta_i^\intercal F_t + \gamma_i^\intercal G_t + \epsilon_{i,t} \tag{1}$$

Here, $R_{i,t}$ is the excess return for the testing asset. F_t are deep factors comprising long-short spread factors generated by sorting individual firm returns on the output characteristics from a deep neural network. In our unified framework, F_t are generated while controlling G_t within deep learning. The tradable alphas, cross-sectional pricing errors, are constructed as

$$\widehat{\alpha}_i = \frac{1}{T}\sum_{t=1}^{T}\Big(R_{i,t} - \widehat{\beta}_i^\intercal F_t - \widehat{\gamma}_i^\intercal G_t\Big). \tag{2}$$

Our optimization objective is to minimize the least squares, time series variation across N assets, with an additional penalty on the average pricing errors.

$$\mathcal{L} = \frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\Big(R_{i,t} - \alpha_i - \beta_i^\intercal F_t - \gamma_i^\intercal G_t\Big)^2 + \lambda\frac{1}{N}\sum_{i=1}^{N}\alpha_i^2. \tag{3}$$

The choice of the penalty term mimics the pricing error test[2] in Gibbons, Ross, and Shanken (1989) and measures the average pricing errors. It is derived from the economic constraint from the beta pricing model: the excess asset return can be explained by the risk premia of factors.

$$E(R_{i,t}) = \alpha_i + \beta_i E(F_t) + \gamma_i E(G_t), \tag{4}$$

where $\alpha_i = 0$ for every testing asset i.

By combining the time series model (1) and cross-sectional model (4), we solve an optimization problem: how to generate long-short spread factors to minimizing the average pricing errors. Our loss function design is similar to the RP-PCA of Lettau and Pelger (2018), who provide a regularized estimation with a penalty accounting for cross-sectional variation of average returns. Their regularized PCA is designed to maximize the time series variation while penalizing PCs with small risk premium, while ours uses the time series variation to determine the factor loading and train

[2]It is possible to replace $\sum_{i=1}^{N}\alpha_i^2$ with the weighted average $\alpha^\intercal\Sigma_\alpha^{-1}\alpha$ at the expense of estimating the additional Σ_α.

the latent factors in the beta model. Another difference between two approaches is the trade-off between using statistical factors and characteristic-based factors.

1.2 Related Literature

The most closely related literature is the statistical factor model, such as PCA. The original PCA of Chamberlain and Rothschild (1983) and Connor and Korajczyk (1986, 1988) estimates factors that can best explain the time series variation for a large cross-section of stocks. The recent IPCA of Kelly et al. (2018) uses additional information, namely, firm characteristics, as instruments to estimate PCs and further allows time-varying factor loadings. Lettau and Pelger (2018) derive the statistical properties of RP-PCA that helps to identify factors with small time series variance but useful to the cross-sectional variation. Kozak, Nagel, and Santosh (2018) show PCA of anomaly portfolios works as well as popular reduced-form factor models in explaining the cross section of average returns on anomaly portfolios.

Our method figures out the best-transformed characteristics for security sorting by marrying the time series and cross-sectional variation. Using a purely statistical factor approach results in a few shortcomings. First, PCA uses either individual firms or portfolios, whereas our deep factors are generated using individual firm returns to fit a different set of portfolios. We use individual firms as train assets and Fama-French 25 portfolios as validation assets. Second, PCs generally perform poorly out of sample, whereas deep factors are sorted portfolios that behave similarly to Fama-French factors. Third, PCA lacks the flexibility to estimate latent factors by controlling for a benchmark model, such as CAPM or Fama-French factors. Our method does not create an entirely new factor model but adds additional deep factors on a benchmark model. Fourth, PCA relies on a balanced data structure, where security sorting on characteristics does not require.

The second related area is forecasting stock return via machine learning. Kozak, Nagel, and Santosh (2017) use a shrinkage estimator on the SDF coefficients for characteristic-based factors with economic interpretation. Freyberger, Neuhierl, and Weber (2017) apply the adaptive group LASSO for firm characteristics selection and provide evidence of nonlinearity, whereas Light, Maslov, and Rytchkov (2017) uses partial least squares (PLS) to aggregate information of firm characteristics. Chinco, Clark-Joseph, and Ye (2017) forecast one-minute-ahead returns using the entire cross sec-

tion of lagged returns by LASSO and find news about fundamentals can be useful predictors. For comprehensive empirical investigation of forecasting performance for multiple machine learning algorithms, see Gu, Kelly, and Xiu (2018).

Our method develops an asset pricing model instead of a pure return forecasting machine. First, our goal is not to test or select characteristics for forecasting. We find the best combination and transformation of all characteristics through a multi-layer neural network, rather than throwing away most of them. Second, the empirical literature lacks a clear out-of-sample design for machine learning, because right-hand-side factors are portfolios constructed by (left-hand-side) individual firm returns. We provide a typical machine learning train-validation-test design from factor creation to return forecasting. Third, deep learning is widespread in the investment industry but still lacks the interpretation as a black box. Our deep factors, the hidden neurons in the deep neural network, are tradable strategies with significant alphas.

Second, directly using dynamic firm characteristics in return forecasting is not easy. Some firms do not have historical data, and some disappear in the future. The firm characteristics data can easily go missing at random. One major advantage of using factors is that security sorting is not restricted to the unbalanced panel data structure. Kozak et al. (2017) sort securities on nonlinear and interaction forms of characteristics for factor selection. Other common practices include using a small set of firms or imputation for missing data. Recently, Freyberger et al. (2017) and Kelly et al. (2018) have applied different machine learning methods that employ period-by-period cross-sectional regressions similar to those in Fama and MacBeth (1973).

Another approach is the use of incremental information of characteristics to deal with the high-dimensionality challenge in Cochrane (2011). Harvey et al. (2016) show the multiple testing issues in the zoo of factors produced in the last 40 years. Feng, Giglio, and Xiu (2017) provide a high-dimensional inference method to tame the factor zoo and find a small number of factors with incremental contribution recursively. Kelly et al. (2018) evaluate the contribution of individual characteristics under a nested model comparison by R-squared reduction.

Our framework provides an evaluation of the incremental contribution of a set of characteristics beyond a benchmark factor model. The deep factors can be generated by controlling for a benchmark model within a deep learning architecture. Unlike maximizing the time series predictability,

adding a factor does not necessarily decrease the in-sample cross-sectional average pricing errors. We can test the out-of-sample improvement using the test assets or future returns of validation assets over the benchmark. We can dissect the existing anomalies with a deep factor model. Finally, the deep factors are trading strategies and can be tested if an anomaly over the benchmark exists.

2 Deep Factor Alpha

In this section,we have a brief introduction to deep learning in section 2.1 and illustrate how to implement the Fama-French type models within the deep neural network in section 2.2.

2.1 Deep Learning

Artificial Neural Network is a pattern recognition machine learning method to use a high-dimensional data X to predict Y[3]. The theoretical root of the prediction power for deep learning is established in Kolmogorov (1963). LeCun, Bengio, and Hinton (2015) and Goodfellow, Bengio, Courville, and Bengio (2016) provide comprehensive summaries about how the neural network develops to the modern deep learning, which attracts tremendous attention in recent years for big data and artificial intelligence. The recent development of deep learning in finance and statistics include Heaton, Polson, and Witte (2017) and Polson and Sokolov (2017). The former introduces deep learning decision models for problems in financial prediction and classification, while the latter provides a Bayesian interpretation to the neural network.

In a simple L-layer NN, using $X^{[l-1]}$ from the previous layer, the l-th layer performs a composition of an affine transformation $AX + b$ and a nonlinear activation $f(\cdot)$. Therefore, the network feeds forward like a pipeline and applies operations sequentially

$$X^{[l]} = f^{[l]}\left(A^{[l]}X^{[l-1]} + b^{[l]}\right) \text{ for } l = 1, 2, ..., L.$$

Here, $X^{[0]} = X$ is the input layer, $X^{[L]}$ is the output layer as a predictor, and those intermediary $X^{[l]}$ are hidden layers. Commonly used activation functions $f(\cdot)$ include sigmoidal $1/(1 + \exp(-x))$, $\cosh(x)$, $\tanh(x)$ and rectified linear unit (ReLU): $\max\{x, 0\}$.

[3]Early studies of artificial neural network in computer science and economics include Gallant and White (1988), White (1988), Hornik, Stinchcombe, and White (1989), Poggio and Girosi (1990) and Kuan and White (1994)

A layer $X \in \mathbb{R}^K$ is represented by K neurons with activation functions performed on them. A weight matrix A is represented by a bunch of edges (or arrows) between 2 layers. A bias vector b can be viewed as neurons unconnected with previous layers. Figure (1), a neural network diagram, shows a neural network with 3 hidden layers of size 3, 5 and 4 respectively. There are 8 predictors in the input layer and a scalar Y in the output layer. Each blue circle denotes a neuron and each edge linking 2 neurons is an element of weight matrix A, where red edges are positive weights and green ones are negative. The edge width and opacity are proportional to absolute values.

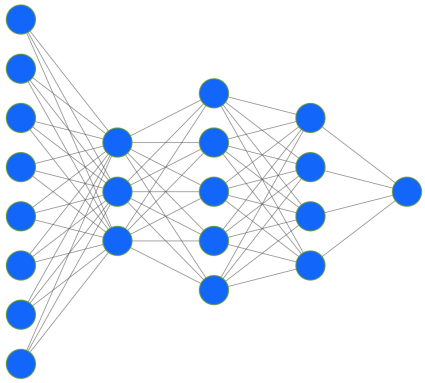

Figure 1: A Neural Network Diagram

As L, the depth of network increases, we get a more complicated composition of functions

$$\left(f_L \circ f_{L-1} \circ f_{L-2} \circ ... \circ f_1\right)(X), \quad \text{where } f_l(x) := f^{[l]}\left(A^{[l]}x + b^{[l]}\right)$$

A neural network aims to learn patterns in X to forecast Y. The search of patterns is supervised by a target response Y. Suppose the multi-layer architecture and $f(\cdot)$ are given, the choice of model parameters A and b should leads to a good approximation of $Y = Y(X)$. The performance is then evaluated by a loss function $\mathcal{L}(\hat{Y}, Y)$ where $\hat{Y} = \left(f_L \circ ... \circ f_1\right)(X)$. For example, squared error loss $\mathcal{L}(\hat{Y}, Y) := \|Y - \hat{Y}\|_2^2$ is used in regression problems. Like most other machine learning methods, the model training of deep learning involves solving an optimization problem. Given N training

data points $\{X_i, Y_i\}_{i=1}^N$, we find the layer architecture and model parameters to minimize

$$\mathcal{L}\left(\hat{Y}(X), Y\right) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(\hat{Y}(X_i), Y_i\right).$$

2.2 Fama-French model in Deep Learning

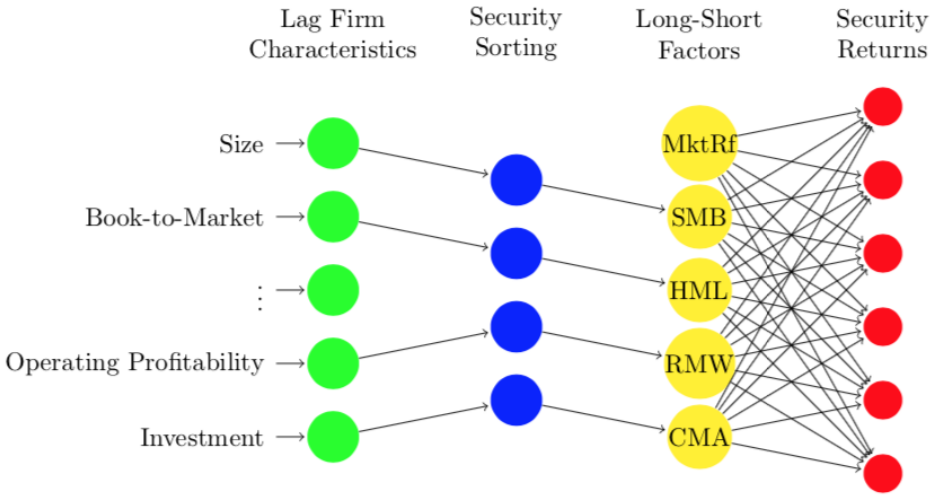

Figure 2: Fama-French 5-factor model as a neural network

Fama and French (2015, 2016) continue to add RMW and CMW to form a five-factor model, which carry high risk premia and are shown to dissect many anomalies beyond FF3. However, in the existing zoo of factors, there are similar measures to size, book-to-market, profitability, and investment. The economic theory is silent about how to calculate the firm characteristics with extensive fundamental information. For example, there are multiple momentum factors: long-term reversal (13-60), short-term reversal (1-1), the Carhart Momentum (2-12), and so forth. All of the similar momentum characteristics are a sum of past firm returns. The behavioral economics or systematic risk does not tell which past months and how many to use. Sorting securities on calculated characteristics might be a trial-and-error experiment to proxy for the common risk factor.

In a deep learning perspective, Figure (2) shows how to build Fama-French factors from the firm characteristics to return forecasts. Researchers find out the best formula for some firm characteristics used for security sorting. At the beginning of the period, they sort individual firms on

their lag characteristics to determine the top and bottom value-weighted portfolios. If a firm does not exist or has missing characteristics in some periods, it is not included in the security sorting for those periods. Therefore, security sorting, the quantile activation function, works perfectly for the imbalanced data structure with missing values for the nature of firm dynamics.

Researchers construct the factors with long-short portfolio spreads by selecting those firms rank top and bottom in the sorting. The factors are expected to produce positive average returns or risk premia. If the factors have explanation power to the cross-sectional returns at the same period, it could be a proxy for a common risk factor. The potential multi-layer transformations and combinations of firm characteristics happen before the green circles. The next activation for security sorting is the quantile function in the blue circles. Finally, we create the factor model with other factors, such as the excess market return.

In this paper, we argue the approach that researchers have been using to create characteristic-based factors for decades is part of a deep neural network. With a correct loss function for the supervised learning, it is possible to exploit the modern computation to automate the best calculation of firm characteristics. The gain is to unified the factor creation from the characteristic calculation, to security sorting, to an augmented factor model with a benchmark, to minimizing the pricing errors. The greedy algorithm of a deep neural network can be useful to create such characteristics useful to dissect the pricing errors in the loss function.

3 Model Training

In this section, we discuss the details of model training: including the multi-layer characteristics transformation, the factor construction, and the unified model to minimize the loss function.

3.1 Model Notation

With traditional neural network as building blocks, we are now ready to design the architecture for our deep factor alpha. Our deep learning framework mainly consists of 2 parts. The first part takes lagged firm characteristics as input and feeds them into a deep multi-layer network. It simultaneously performs dimension reduction and nonlinearity extraction. The flexibility of deep network allows us to search any possible pattern in firm characteristic space and summarize them

in the deep characteristics as output. The generation of deep characteristics is then supervised by the second part, which implements augmented Fama-French factor model. The second part uses deep characteristics, and individual firm returns to construct deep factors and combine them with the inputted ones, i.e., existing benchmark factors. The augmented factor set is used to price target assets via a linear model. Specifically, the construction of deep factors is via bivariate sorting, and the final step is illustrated in Equation 1. In the end, the whole model training procedure aims to improve deep factors and minimize pricing errors defined in Equation 3.

For our deep learning framework, a typical training observation indexed by time t includes 4 types of data:

$$\begin{cases} \{R_{i,t}\}_{i=1}^{N}, & \text{as excess returns of } N \text{ target assests} \\ \{r_{j,t}\}_{j=1}^{M}, & \text{as excess returns of } M \text{ individual firms} \\ \{Z_{k,j,t-1} : 1 \leq k \leq K\}_{j=1}^{M}, & \text{as } K \text{ lagged characteristics of } M \text{ firm} \\ \{G_{d,t}\}_{d=1}^{D}, & \text{as } D \text{ benchmark factors} \end{cases} \tag{5}$$

For notational simplicity we write them in matrix form $\{R_t, r_t, Z_{t-1}, G_t\}$. Here R_t is a $N \times 1$ vector; r_t is a $M \times 1$ vector; Z_{t-1} is a $K \times M$ matrix; G_t is a $D \times 1$ vector.

3.2 Deep Characteristics

In this section we show how to design a L-layer deep network, of which the purpose is to generate P deep characteristics. We drop for now the subscript t, bearing in mind that the input Z at time t denotes the firm characteristics in time $t-1$. The architecture is as follows:

$$X_{\cdot,j}^{[l]} = f^{[l]}\left(A^{[l]} X_{\cdot,j}^{[l-1]} + b^{[l]}\right), \text{ for } l = 1, 2, 3, ..., L \text{ and } j = 1, 2, ..., M \tag{6}$$

$$Z := X^{[0]}. \tag{7}$$

where $X_{\cdot,j}^{[l]}$ is the j-th column of a $K_l \times M$ matrix $X^{[l]}$. We set $K_0 = K$ and $K_L = P$. $f^{[l]}$ is the univariate activation function in the l-th layer, broadcasting to every element of a matrix. The

parameters to be trained in this part are deep learning weights A's and biases b's, namely

$$\left\{(A^{[l]}, b^{[l]}) : A^{[l]} \in \mathbb{R}^{K_l \times K_{l-1}}, b^{[l]} \in \mathbb{R}^{K_l}\right\}_{l=1}^{L}.$$

We point out that here the transformations are made column by column. With a little abuse of notation, we will rewrite the architecture as

$$Y := X^{[L]}, \tag{8}$$

$$X^{[l]} = f^{[l]}\left(A^{[l]} X^{[l-1]} + b^{[l]}\right), \text{ for } l = 1, 2, 3, ..., L \tag{9}$$

$$Z := X^{[0]}. \tag{10}$$

where the output Y is our $P \times M$ deep characteristics.

Unlike standard feed-forward neural network, the l-th layer in our architecture is a neural matrix $X^{[l]}$. Each row of $X^{[l]}$ is a $1 \times M$ vector representing the k_l-th "intermediate characteristics" for M firms, $k_l = 1, 2, ..., K_l$. We explicitly make all the columns (firms) share the same parameters $A^{[l]}$ and $b^{[l]}$, whose dimensions are independent of M. We use K_l to denote the dimension of l-th layer since the number of columns are fixed as M for all $X^{[l]}$'s. Figure (3) illustrates how our deep learning network forwards by showing an example architecture from $(l-1)$-th layer to $(l+1)$-th layer, where $K_{l-1} = K_{l+1} = 2$ and $K_l = 4$. The Fama-French approach simply drops all hidden layers and uses $Y := Z$ for bivariate sorting in the latter part. In contrast, Z in our deep network goes through multiple layers of affine transformations and nonlinear activations, and ends up with a low dimensional deep characteristics Y. Here the layer sizes $\{K_l\}_{l=1}^{L}$ and the number of layers L are tuning parameters.

3.3 Deep Factors

With a $P \times 1$ vector generated from the first part, Y, our deep framework continues with the construction of deep factors via bivariate sorting and then an augmented factor model for asset

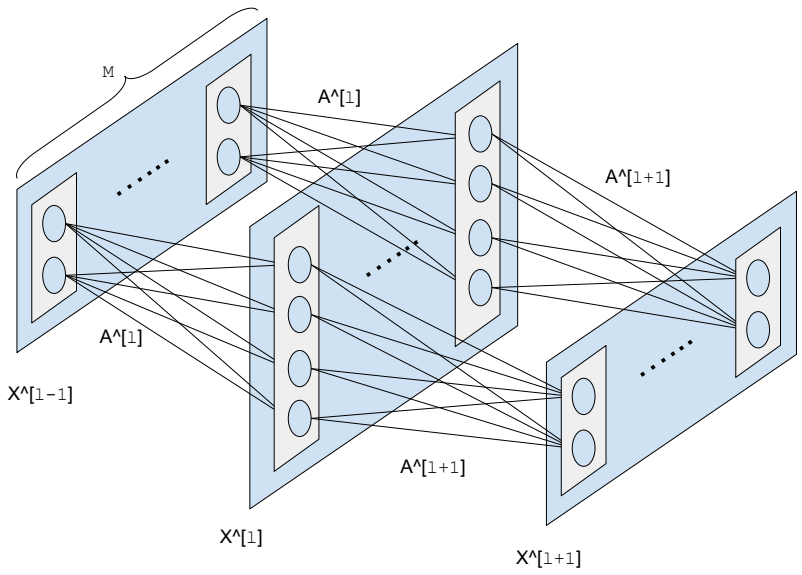

Figure 3: Deep network of $X^{[l-1]} \to X^{[l]} \to X^{[l+1]}$. $K_{l-1} = K_{l+1} = 2$, $K_l = 4$.

pricing. The architecture after L-th layer is as follows:

$$\hat{R} := X^{[L+4]} = h^{[4]}\left(X^{[L+3]}, G\right) \tag{11}$$

$$F := X^{[L+3]} = h^{[3]}\left(X^{[L+2]}, r\right) \tag{12}$$

$$W := X^{[L+2]} = h^{[2]}\left(X^{[L+1]}, V\right) \tag{13}$$

$$U := X^{[L+1]} = h^{[1]}\left(X^{[L]}\right) \tag{14}$$

Here $h^{[1]}, h^{[2]}, h^{[3]}, h^{[4]}$ are no longer univariate activation functions. Instead, each of them is an operator specially defined in order to conduct important transformations. Also note that $h^{[2]}, h^{[3]}, h^{[4]}$ all take 2 arguments, one from the previous layer and another from additional input. V is a $M \times 1$ vector of lagged market equity value.

We now describe these 4 operators in details. $h^{[4]} : \mathbb{R}^P \times \mathbb{R}^D \to \mathbb{R}^N$ is an affine transformation of its 2 arguments, and the parameters are denoted as $\alpha \in \mathbb{R}^N$, $\beta \in \mathbb{R}^{N \times P}$ and $\gamma \in \mathbb{R}^{N \times D}$.

$$h^{[4]}(F, G) = \alpha + [\beta\ \gamma] \begin{bmatrix} F \\ G \end{bmatrix}. \tag{15}$$

Therefore $h^{[4]}$ is related to the augmented factor model.

$h^{[3]} : \mathbb{R}^{P\times M} \times \mathbb{R}^M \to \mathbb{R}^P$ defines how we construct deep factors as tradable value-weighted portfolios. Once given the portfolio weights W and individual firm returns r, it is simply a matrix production

$$h^{[3]}(W, r) = Wr. \tag{16}$$

The key procedures are $h^{[2]}$ and $h^{[1]}$. They essentially performs bivariate sorting, portfolio selection and weight normalization. $h^{[1]}$ is also implicitly incorporated in $h^{[2]}$, as a preliminary step. Regarding $h^{[2]} : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}^M$, it defines how we combine the sorting results of deep characteristics with market equity to produce long-short portfolio weights. When its first argument is a matrix of P rows, it performs P separated operations (i.e. row by row) with the same second argument and aggregates the results in a matrix of the same size. Suppose the arguments of $h^{[2]}$ are 2 vectors, x_1 and V. We will see later that $h^{[2]}$ first combines x_1 and $h^{[1]}(V)$ to generate 4 indicator vectors (each element is either 0 or 1). These 4 vectors actually indicate the memberships of individual firms in 4 bivariate-sorted portfolios. Then $h^{[2]}$ outputs the "difference in averages" of the 4 bivariate-sorted weights, using V as a vector of candidate weights. Finally the generated F are long-short portfolios.

3.3.1 Bivariate Sorting

Let's first define $h^{[1]} : \mathbb{R}^M \to \mathbb{R}^M$, which is the univariate sorting operating on a vector. Again, when its argument is a matrix, $h^{[1]}$ performs univariate sorting row by row and aggregates the outputs in a matrix.

Let y be a $M \times 1$ vector representing some deep characteristic, i.e. a row of Y, or the market equity value V. We define $h^{[1]}(y)$ as

$$h^{[1]}(y) = \begin{bmatrix} \mathbb{1}\{y_1 \geq q_\nu(y)\} \\ \vdots \\ \mathbb{1}\{y_j \geq q_\nu(y)\} \\ \vdots \\ \mathbb{1}\{y_M \geq q_\nu(y)\} \end{bmatrix} + \begin{bmatrix} \mathbb{1}\{y_1 \geq q_\tau(y)\} \\ \vdots \\ \mathbb{1}\{y_j \geq q_\tau(y)\} \\ \vdots \\ \mathbb{1}\{y_M \geq q_\tau(y)\} \end{bmatrix} - \mathbf{1}_M \tag{17}$$

where $\mathbb{1}$ is the indicator function and $\mathbf{1}_M$ is a $M \times 1$ vector of ones. q_ν and q_τ are lower $\nu-$ and $\tau-$ quantiles respectively, with $\nu + \tau = 1$ and $0 < \nu \leq \tau$. For example, we choose $\nu = 0.2, \tau = 0.8$ for deep characteristics Y and $\nu = \tau = 0.5$ for market equity value V.

It is clear that each coordinate of $h^{[1]}(y)$ takes value from $\{-1, 0, 1\}$, depending on the rank of $y_1, y_2, ...y_M$. In other words, assume $y_{(1)} \leq y_{(2)} \leq ... \leq y_{(M)}$, then

$$\left[h^{[1]}(y)\right]_{(j)} = \begin{cases} -1 & \text{if } \frac{j}{M} < \nu \\ 0 & \text{if } \nu \leq \frac{j}{M} < \tau \\ 1 & \text{if } \frac{j}{M} \geq \tau \end{cases} \tag{18}$$

Finally, $h^{[2]}(x_1, V)$ combines the sorting results of deep characteristics with the information of market equity. Define $x_2 := h^{[1]}(V)$. Then those 4 indicator vectors mentioned above are:

$$\begin{cases} \Delta^{Bt} &= (x_1)_+ \odot (x_2)_+ \\ \Delta^{Bb} &= (x_1)_- \odot (x_2)_+ \\ \Delta^{St} &= (x_1)_+ \odot (x_2)_- \\ \Delta^{Sb} &= (x_1)_- \odot (x_2)_- \end{cases} \tag{19}$$

where $(x)_+ := \max\{x, 0\}$ and $(x)_- := \max\{-x, 0\}$ and "$\odot$" denotes the element-wise production. Then

$$h^{[2]}(x_1, V) = \frac{1}{2}\left[\frac{\Delta^{Bt} \odot V}{(\Delta^{Bt} \odot V)'\mathbf{1}_M} + \frac{\Delta^{St} \odot V}{(\Delta^{St} \odot V)'\mathbf{1}_M}\right] - \frac{1}{2}\left[\frac{\Delta^{Bb} \odot V}{(\Delta^{Bb} \odot V)'\mathbf{1}_M} + \frac{\Delta^{st} \odot V}{(\Delta^{sb} \odot V)'\mathbf{1}_M}\right] \tag{20}$$

where the portfolio weights $\Delta \odot V$ are normalized by $1/(\Delta \odot V)'\mathbf{1}_M$ so that the sum $h^{[2]'}\mathbf{1}_M = 0$.

To better understand the procedure of $h^{[1]}$ and $h^{[2]}$, first imagine dividing the firm universe into 3 parts using cut-off values $q_\nu(y)$ and $q_\tau(y)$. This division is with respect to some deep characteristic y and $x_1 = h^{[1]}(y)$, each coordinate representing a firm. The proportions of firms in each part are ν, $\tau - \nu$ and $1 - \tau$, respectively. We set $x_1 = 1$ for top firms, $x_1 = 0$ for middle firms and $x_1 = -1$ for bottom firms. Second, divide the firm universe into 2 halves using the median firm market equity value. With $x_2 = h^{[1]}(V)$, $x_2 = 1$ if a firm is big and $x_2 = -1$ if it's small. $\Delta_j = 1$ if and only if the

j-th firm is selected in the corresponding portfolio. If we replace V with $\mathbf{1}_M$, then $h^{[2]}$ gives weights for equally-weighted portfolios. Figure (4) illustrates our bivariate sorting.

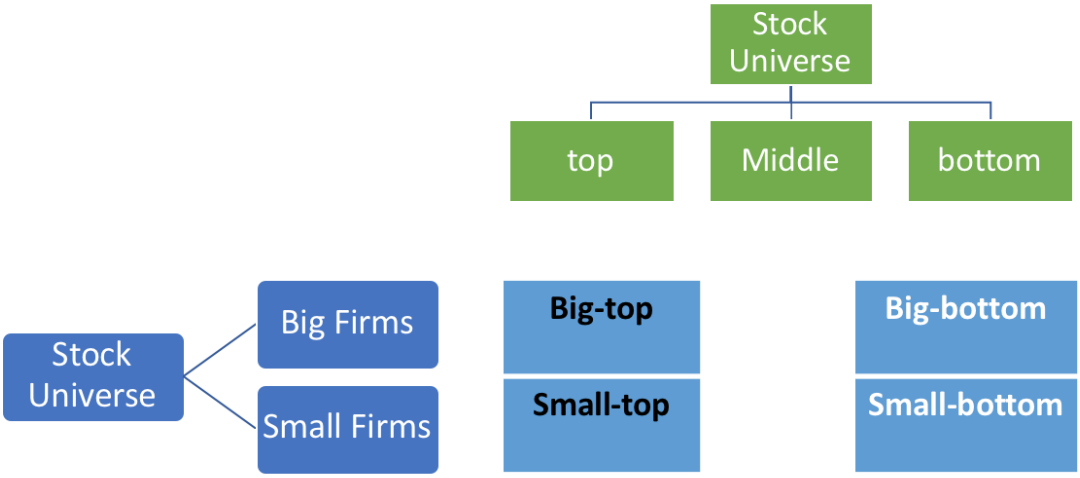

Figure 4: Bivariate Sorted Portfolios

3.4 Minimize Mispricing Alphas

We summarize the above deep learning framework in Table (1) and Figure (5):

	Dimension	Output	Additional Input	Operation	Parameters
Asset return	$N \times 1$	$\hat{R}$	G	$\alpha + \beta F + \gamma G$	(α, β, γ)
Deep factor	$P \times 1$	F	r	Wr	
Portfolio weight	$P \times M$	W	V	$h^{[2]}(U, V)$	
Sorting	$P \times M$	U		$h^{[1]}(Y)$	
Deep characteristic	$K_L \times M$	Y		$f^{[L]}\left(A^{[L]} X^{[L-1]} + b^{[L]}\right)$	$(A^{[L]}, b^{[L]})$
	$\vdots$	$\vdots$		$\vdots$	$\vdots$
	$K_l \times M$	$X^{[l]}$		$f^{[l]}\left(A^{[l]} X^{[l-1]} + b^{[l]}\right)$	$(A^{[l]}, b^{[l]})$
	$\vdots$	$\vdots$		$\vdots$	$\vdots$
Firm characteristic	$K \times M$	$X^{[0]}$	Z	$X^{[0]} = Z$	

Table 1: Deep factor alpha architecture

Fixing L, $\{K_l\}_{l=1}^{L}$ and (ν, τ), our loss function is the mean squared prediction error regularized

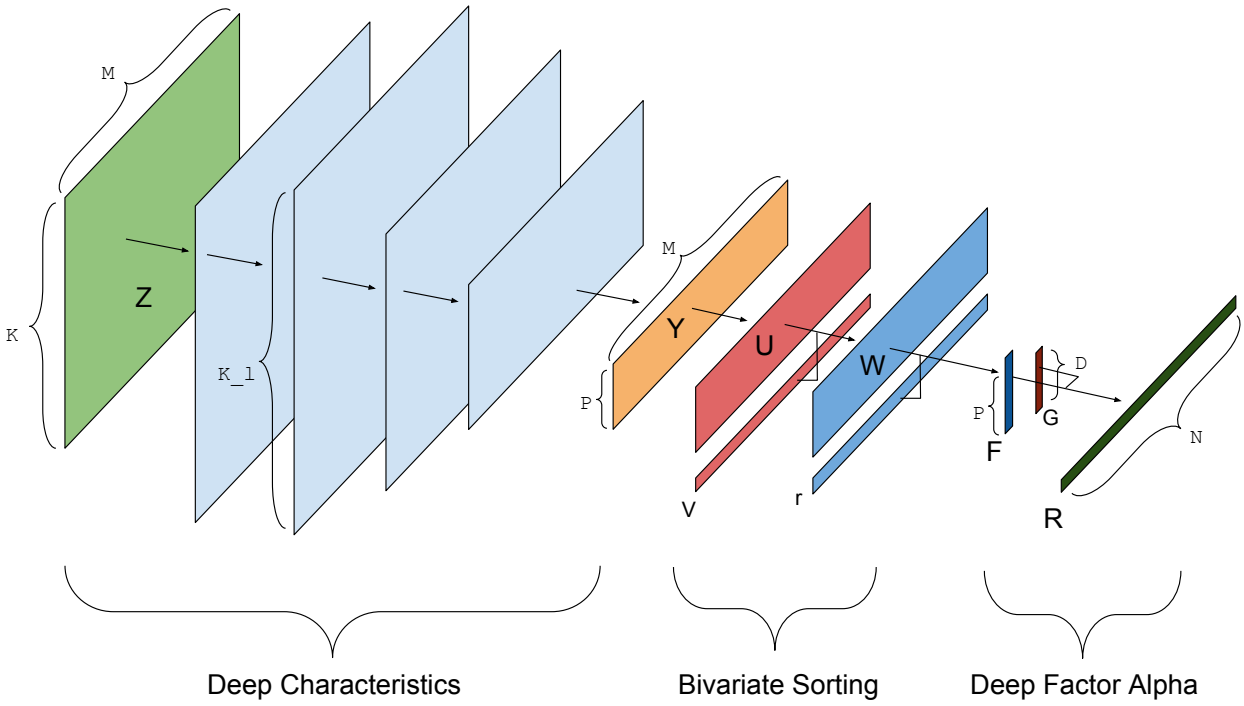

Figure 5: Deep factor alpha architecture

by mean squared mispricing error

$$\mathcal{L}(A,b,\alpha,\beta,\gamma) := \frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\left(R_{it}-\hat{R}_{it}\right)^2 + \lambda\frac{1}{N}\sum_{i=1}^{N}\alpha_i^2 \tag{21}$$

where

$$\hat{R}_{it} = \alpha_i + \beta_i^\intercal F_t + \gamma_i^\intercal G_t$$

and λ controls the amount of regularization. $\beta = [\beta_1, \beta_2, ..., \beta_N]^\intercal, \gamma = [\gamma_1, \gamma_2, ..., \gamma_N]^\intercal$. To train the deep network is then equivalent to get a joint estimation of $(A,b) := \left\{A^{[l]}, b^{[l]}\right\}_{l=1}^{L}$ and (α,β,γ). We use the notation $F_t^{A,b}$ to indicate the dependence of F_t on the parameters. The corresponding estimates are

$$(\hat{A},\hat{b},\hat{\alpha},\hat{\beta},\hat{\gamma}) = \arg\min_{A,b}\left\{\frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\left(R_{it}-\hat{R}_{it}\right)^2 + \lambda\frac{1}{N}\sum_{i=1}^{N}\alpha_i^2\right\}$$

Dropout is used to improve the estimation. Here the input space of $X^{[l-1]}$ is replaced by $D \odot X^{[l-1]}$

for $l = 1, 2, ..., L$, where $D \sim \text{Bernoulli}(p)$ is a matrix of randomly assigned Bernoulli variables. This acts as a ridge penalty. See Hinton and Salakhutdinov (2006), Polson and Sokolov (2017). As opposed to sparsity, the network architecture averages small models using dropout.

Although being highly nonlinear and non-convex, the structure of deep learning model makes its loss function differentiable with respect to its parameters. The first-order derivative information is directly available by carefully applying backward chain rule. TensorFlow library performs automatic derivative calculation for practitioners. This allows us to train the model using stochastic gradient descent (SGD), see Robbins and Monro (1951) and Kiefer and Wolfowitz (1952). Let the superscript (t) to denote the t-th iterate, SGD updates the parameters by

$$\begin{bmatrix} \hat{A}^{(t+1)} \\ \hat{b}^{(t+1)} \\ \hat{\alpha}^{(t+1)} \\ \hat{\beta}^{(t+1)} \\ \hat{\gamma}^{(t+1)} \end{bmatrix} \longleftarrow \begin{bmatrix} \hat{A}^{(t)} \\ \hat{b}^{(t)} \\ \hat{\alpha}^{(t)} \\ \hat{\beta}^{(t)} \\ \hat{\gamma}^{(t)} \end{bmatrix} - \eta^{(t+1)} \nabla \mathcal{L}^{(t)} \tag{22}$$

until convergence, where η is the step size and the gradient is evaluated at $(\hat{A}^{(t)}, \hat{b}^{(t)}, \hat{\alpha}^{(t)}, \hat{\beta}^{(t)}, \hat{\gamma}^{(t)})$. At each iterate, the loss $\mathcal{L}^{(t)}$ only involves a random subset of data, $\mathcal{B} \subset \{1, 2, ..., T\}$, called mini-batch,

$$\mathcal{L}^{(t)}(A, b, \alpha, \beta, \gamma) = \frac{1}{N|\mathcal{B}|} \sum_{t \in \mathcal{B}} \sum_{i=1}^{N} \left(R_{it} - \hat{R}_{it} \right)^2 + \frac{\lambda}{N} \sum_{i=1}^{N} \alpha_i^2 \tag{23}$$

where $|\mathcal{B}| \ll T$.

3.5 Model Selection

We may consider augmenting our deep characteristics Y by the raw firm characteristics Z, namely replacing Y with $\tilde{Y} = [Y, Z]$ for factor construction. The resulting $P + K$ deep factors are then selected by adaptive group lasso method. The penalty corresponds to the j-th factor is

$$\lambda_j \|\beta_{\cdot,j}\|_2 \propto \frac{1}{|a_j|} \|\beta_{\cdot,j}\|_2$$

where $j = 1, 2, ...P+K$ and a_j is the intercept of the regression model $F_{j,t} \sim G_t$ where $F_{j,t}$ is the j-th element of F_t. When $|a_j| \to 0$, the constructed F_j is useless because it's nearly a linear combination of the existing G. Then the parameter $\lambda_j \propto \frac{1}{|a_j|} \to \infty$ and we expect $\beta_{\cdot,j}$ to be penalized to 0. This leads to a sparse factor model where the number of selected factors is less than $K + P$.

The adaptive group lasso method along with the multivariate regression $F_t \sim G_t$ can be easily incorporated into SGD. We modify the loss function as

$$\tilde{\mathcal{L}} := \mathcal{L} + \frac{\lambda_2}{(K+P)T} \sum_{t=1}^{T} \sum_{j=1}^{K+P} \left(F_{j,t} - a_j - c_j^\intercal G_t \right)^2 + \frac{\lambda_3}{K+P} \sum_{j=1}^{K+P} \frac{1}{|a_j|} \|\beta_{\cdot,j}\|_2. \tag{24}$$

4 Empirical Findings

4.1 Data Information

The characteristics availability is from 1975 January to 2017 December. To calculate some characteristics, we use past data before 1975. We only include stocks for companies listed on three main exchanges in the United States: NYSE, AMEX, or NASDAQ. We use those observations for firms with a CRSP share code of 10 or 11. We only include observations for firms listed more than one year. We exclude observations with negative book equity or negative lag market equity.

For the zoo of characteristics, we use 56 continuous variables surveyed in Green et al. (2017), which include size, book-to-market, profitability, and investment. The description of these characteristics are listed in Table (3) in the appendix. To evaluate the quality of the characteristics, we follow Fama and French (1993) and create the monthly bivariate-sorted factors using the top 20% and bottom 20% of firms. Using all 43 years of data, there are 42, 40, and 34 out of 56 factors tested with significant alphas. During the 30-year training period 1975-2004, there are 42, 40, and 30 of them tested with significant alpha respectively.

4.2 Model Selection and Forecasting

The unique feature of factor model is using portfolios to "explain" portfolios. To perform a valid machine learning forecast, we provide a train-validation-test design to build the out-of-sample forecasting in Figure (6). First, we use individual firm returns as train assets to create long-short

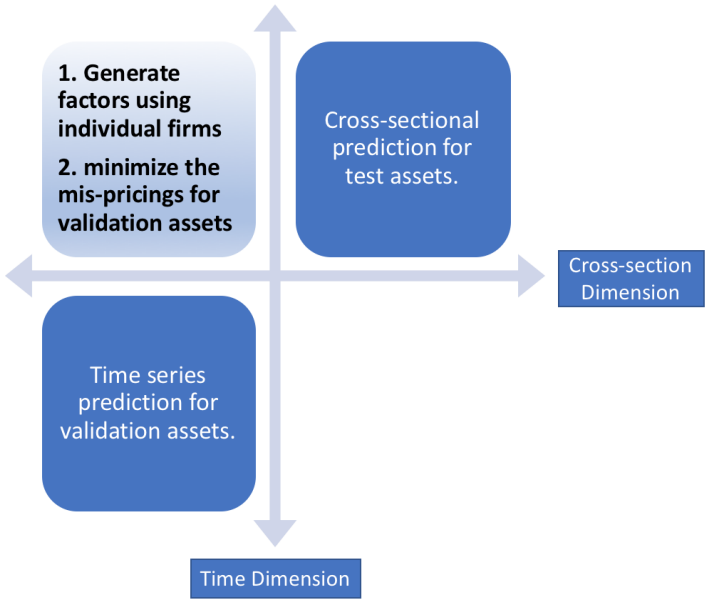

Figure 6: Train-Validation-Test asset design

spread factors. Second, we use 25 size and book-to-market sorted portfolios as validation assets to calculate the training loss. Within deep learning, we run a regression using 25 portfolios on the augmented factor model. Third, we use another set of portfolios (25 size and profitability, 25 size and investment, and 30 industries) as test asset to perform the augmented factor model selection.

Data-driven model selection is an essential issue in asset pricing. We apply such an out-of-sample validation design to determine the number of deep factors, the number of layers for the neural network, and the tuning parameter λ that balances the time series and cross-sectional variation. For time series forecast of validation assets, using the test assets to determine the model is a pure out-of-sample approach. For cross-sectional evaluation of test assets, using validation assets to assess the factor creation is also a pure out-of-sample approach.

In Table (2), we separate the train and test periods as 1974-2004 and 2005-2017. We also provide a resampling design to address the time-varying model data reality. For every four months, we randomly label three in the train data and one in the test data. We use the train data to train and select the deep factor model, then report its out-of-sample improvements on the test data. The estimated factor loadings are fixed in forecasting the future cross-section. The below measures are used to report the empirical results.

Table 2: Out-of-sample Forecasting

	RMSE-train	R2-train	RMSE-test	R2-test
		Past-Future		
CAPM + DF(L=3 P=4)	0.360%	27.99%	0.201%	12.47%
FF3 + DF(L=4 P=2)	0.171%	83.82%	0.181%	28.78%
FF5 + DF(L=3 P=2)	0.173%	83.36%	0.156%	47.50%
CAPM	0.425%	0.00%	0.215%	0.00%
FF(3)	0.170%	83.92%	0.182%	27.77%
FF(5)	0.179%	82.23%	0.159%	45.37%
		Resampling		
CAPM + DF(L=2 P=7)	0.269%	7.06%	0.427%	17.39%
FF3 + DF(L=3 P=6)	0.151%	70.67%	0.205%	81.00%
FF5 + DF(L=2 P=1)	0.130%	78.29%	0.192%	83.25%
CAPM	0.279%	0.00%	0.470%	0.00%
FF(3)	0.153%	69.90%	0.216%	78.88%
FF(5)	0.133%	77.41%	0.208%	80.47%

- RMSE = $\sqrt{\frac{1}{N}\sum_{i=1}^{N}\hat{\alpha}_i^2}$ is the standard deviation for the pricing error.
- $R^2 = 1 - \text{RMSE}^2_{M_1}/\text{RMSE}^2_{M_2}$ is the relative performance of Model 1 over Model 2.

Adding more factors to a model does not necessarily decrease RMSE or increase R-squared. FF5 has a slightly lower in-sample R-squared but a significantly higher out-of-sample one than FF3. The reported R-squared in Table (2) are relative performance over CAPM. Therefore, the R-squared values are not comparable between the train and test data.

Finally, we find a substantial improvement for the CAPM-DL model, which includes four deep factors from a 3-layer neural network. The gain for FF3-DL and FF5-DL models are all positive with more than 1% improvement, both of which are added two more deep factors. In the resampling design, we also see substantial improvement for deep factor model over the benchmark. The CAPM-DL has 17% improvement, while both FF3-DL and FF5-DL have more than 2% improvement.

5 Conclusion

In this paper, we propose a deep factor alpha model to establish a deep learning representation for the characteristics-based factor model to answer an asset pricing problem: dissecting anoma-

lies. Our procedure builds an underappreciated connection between security sorting on calculated characteristics to a quantile activation function within a unified deep neural network. The greedy algorithm helps to search for the best transformation of firm fundamentals for security sorting by controlling for a benchmark model. The main difference between our approach and various statistical factor methods is that ours does not create an entirely new factor model but adds additional deep factors on a benchmark model.

The recent computation breakthrough of artificial intelligence has enabled numerous potential automatic data-driven applications to many areas, including finance and investment. However, asset pricing is such a field that has a low signal-to-noise ratio and a stable pattern of non-stationary. A simple application of deep learning does not exist with the short-history of financial data and the imbalanced data structure. In this paper, we show encouraging results about the development of deep learning in asset pricing research. The key is to adapt the recent deep learning methods within those workhorse empirical models in this field.

References

Chamberlain, G. and M. Rothschild (1983). Arbitrage, factor structure, and mean-variance analysis on large asset markets. *Econometrica: Journal of the Econometric Society*, 1281–1304.

Chinco, A. M., A. D. Clark-Joseph, and M. Ye (2017). Sparse signals in the cross-section of returns. Technical report, National Bureau of Economic Research.

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

Connor, G. and R. A. Korajczyk (1986). Performance measurement with the arbitrage pricing theory: A new framework for analysis. *Journal of Financial Economics 15*(3), 373–394.

Connor, G. and R. A. Korajczyk (1988). Risk and return in an equilibrium apt: Application of a new test methodology. *Journal of Financial Economics 21*(2), 255–289.

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

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

Fama, E. F. and K. R. French (1996). Multifactor explanations of asset pricing anomalies. *The journal of finance 51*(1), 55–84.

Fama, E. F. and K. R. French (2015). A five-factor asset pricing model. *Journal of Financial Economics 116*(1), 1 – 22.

Fama, E. F. and K. R. French (2016). Dissecting anomalies with a five-factor model. *The Review of Financial Studies 29*(1), 69–103.

Fama, E. F. and J. D. MacBeth (1973). Risk, return, and equilibrium: Empirical tests. *Journal of Political Economy 81*(3), 607–636.

Feng, G., S. Giglio, and D. Xiu (2017). Taming the factor zoo. Technical report, City University of Hong Kong.

Freyberger, J., A. Neuhierl, and M. Weber (2017). Dissecting characteristics nonparametrically. Technical report, National Bureau of Economic Research.

Gallant, A. R. and H. White (1988). *A unified theory of estimation and inference for nonlinear dynamic models*. Blackwell.

Gibbons, M. R., S. A. Ross, and J. Shanken (1989). A test of the efficiency of a given portfolio. *Econometrica: Journal of the Econometric Society*, 1121–1152.

Goodfellow, I., Y. Bengio, A. Courville, and Y. Bengio (2016). *Deep learning*, Volume 1. MIT press Cambridge.

Green, J., J. R. Hand, and X. F. Zhang (2017). The characteristics that provide independent information about average us monthly stock returns. *The Review of Financial Studies 30*(12), 4389–4436.

Gu, S., B. T. Kelly, and D. Xiu (2018). Empirical asset pricing via machine learning. Technical report, The University of Chicago.

Harvey, C. R., Y. Liu, and H. Zhu (2016). ... and the cross-section of expected returns. *The Review of Financial Studies 29*(1), 5–68.

Heaton, J., N. Polson, and J. H. Witte (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry 33*(1), 3–12.

Hinton, G. E. and R. R. Salakhutdinov (2006). Reducing the dimensionality of data with neural networks. *science 313*(5786), 504–507.

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural Networks 2*(5), 359–366.

Hou, K., C. Xue, and L. Zhang (2017). Replicating anomalies. Technical report, National Bureau of Economic Research.

Kelly, B., S. Pruitt, and Y. Su (2018). Characteristics are covariances: A unified model of risk and return. Technical report, National Bureau of Economic Research.

Kiefer, J. and J. Wolfowitz (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 462–466.

Kolmogorov, A. N. (1963). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *American Mathematical Society Translation 28*(2), 55–59.

Kozak, S., S. Nagel, and S. Santosh (2017). Shrinking the cross section. Technical report, University of Michigan.

Kozak, S., S. Nagel, and S. Santosh (2018). Interpreting factor models. *The Journal of Finance 73*(3), 1183–1223.

Kuan, C.-M. and H. White (1994). Artificial neural networks: an econometric perspective. *Econometric Reviews 13*(1), 1–91.

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *Nature 521*(7553), 436.

Lettau, M. and M. Pelger (2018). Estimating latent asset-pricing factors. Technical report, National Bureau of Economic Research.

Light, N., D. Maslov, and O. Rytchkov (2017). Aggregation of information about the cross section of stock returns: A latent variable approach. *The Review of Financial Studies 30*(4), 1339–1381.

Poggio, T. and F. Girosi (1990). Networks for approximation and learning. *Proceedings of the IEEE 78*(9), 1481–1497.

Polson, N. and V. Sokolov (2017). Deep learning: A bayesian perspective. *Bayesian Analysis 12*(4), 1275–1304.

Robbins, H. and S. Monro (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 400–407.

White, H. (1988). Economic prediction using neural networks: The case of ibm daily stock returns. Technical report, University of California, San Diego.

6 Appendix

Table 3: Characteristics and Anomaly Summary

Variable	Chacteristics Description	Average Return	Sharpe Ratio
mve0	Market equity	0.23%	20.29%
beta	Market Beta	0.06%	2.75%
chmom	Change in 6-month momentum	0.02%	2.04%
idiovol	Idiosyncratic return volatility	0.32%	15.83%
indmom	Industry momentum	0.35%	28.81%
mom1m	1-month momentum	0.17%	14.98%
mom6m	6-month momentum	0.66%	45.94%
mom12m	12-month momentum	0.83%	55.35%
pricedelay	Price delay	0.10%	19.61%
absacc	Absolute accruals	0.11%	14.99%
acc	Working capital accruals	0.31%	52.89%
age	years since first Compustat coverage	0.22%	22.16%
agr	Asset growth	0.51%	73.60%
bm	Book-to-market	0.49%	44.22%
bm_ia	Industry-adjusted book to market	0.27%	49.51%
cashdebt	Cash flow to debt	0.27%	32.20%
cashpr	Cash productivity	0.41%	46.23%
cfp	Cash flow to price ratio	0.51%	42.55%
cfp_ia	Industry-adjusted cash flow to price ratio	0.33%	65.81%
chcsho	Change in shares outstanding	0.47%	70.23%
chempia	Industry-adjusted change in employees	0.18%	42.94%
chinv	Change in inventory	0.31%	60.21%
chpmia	Industry-adjusted change in profit margin	0.01%	1.81%
currat	Current ratio	0.08%	7.52%
depr	Depreciation / PP&E	0.22%	23.08%
dy	Dividend to price	0.00%	0.16%
egr	Growth in common shareholder equity	0.37%	55.41%
ep	Earnings to price	0.56%	45.12%
gma	Gross profitability	0.23%	28.70%
herf	Industry Concentration	0.02%	2.35%

The characteristics are calculated based on the SAS program of Green et al. (2017). The factors are monthly sorted long-short spreads using the data from 1975 to 2017.

Variable	Chacteristics Description	Average Return	Sharpe Ratio
hire	Employee growth rate	0.31%	44.80%
invest	Capital expenditures and inventory	0.49%	73.90%
lev	Leverage	0.39%	28.85%
lgr	Growth in long-term debt	0.34%	66.79%
mve_ia	Industry-adjusted size	0.24%	22.26%
operprof	Operating profitability	0.38%	39.90%
pchcapx_ia	Industry adjusted % change in capital expenditures	0.15%	31.60%
pchcurrat	% change in current ratio	0.01%	2.68%
pchdepr	% change in depreciation	0.15%	44.44%
pchgm_pchsale	% change in gross margin - % change in sales	0.18%	40.74%
pchquick	% change in quick ratio	0.03%	10.63%
pchsale_pchrect	% change in sales - % change in A/R	0.09%	26.38%
pctacc	Percent accruals	0.24%	50.57%
ps	Financial statements score	0.21%	47.56%
quick	Quick ratio	0.09%	8.35%
roic	Return on invested capital	0.42%	43.02%
salecash	Sales to cash	0.20%	20.48%
saleinv	Sales to inventory	0.18%	33.80%
salerec	Sales to receivables	0.27%	42.77%
sgr	Sales growth	0.29%	40.45%
sp	Sales to price	0.68%	57.18%
tang	Debt capacity/firm tangibility	0.04%	3.56%
tb	Tax income to book income	0.32%	55.98%
baspread	Bid-ask spread	0.56%	27.84%
maxret	Maximum daily return	0.35%	22.65%
retvol	Return volatility	0.46%	24.70%