Deep Learning Factor Alpha*

Guanhao Feng† Nicholas G. Polson ‡ Jianeng Xu§

College of Business *Booth School of Business* *Booth School of Business*

City University of Hong Kong *University of Chicago* *University of Chicago*

September 1, 2018

Abstract

Does a factor model exist to absorb all existing anomalies? We provide a deep learning automated solution to generate long-short factors using a high-dimensional firm characteristics. Sorting securities on firm characteristics is a common practice in finance and a nonlinear activation function built into deep learning. Our algorithm performs a nonlinear search and finds the optimal transformation of characteristics used for security sorting, with one asset pricing objective: minimizing alphas. Our deep factors, hidden neurons in the neural network, are trained greedily with the backward propagation feedback from the loss function that considers both time series and cross-sectional variations. Our conditional forecast generalizes a benchmark, such as CAPM, and includes Fama-French type models as special cases. We have designed a train-validation-test study for monthly U.S. equity returns from 1975 to 2017 and 57 published firm characteristics. In an out-of-sample evaluation, the conditional deep factor model shows a forecasting improvement over the benchmark with factors that offer significant alphas. The conclusion is the improvement of insignificant alphas for some anomalies as well as sorted portfolios.

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, Bryan Kelly, and Dacheng Xiu. We are also grateful to helpful comments from seminar and conference participants at R/Finance 2018, EcoSta 2018, SOFIE Summer School 2018. We acknowledge the research grant from Unigestion Alternative Risk Premia Research Academy. Feng acknowledges the ECS grant from Hong Kong Research Grants Council.

†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

Many anomalies are identified in cross-sectional average stock returns[1]. The standard protocol is to build portfolios that are sorted on firm characteristics (market equity, earnings-to-price, book-to-market equity, etc.) and to test the "alpha" by regressing the long-short spread of sorted portfolios over a benchmark model, such as CAPM. A significant intercept indicates the long-short spread is not spanned by the benchmark and could be an anomaly. A subsequent test is if the asset-factor betas help to explain the variation in average returns. Fama and French (1992) show 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.

This paper attempts to provide an automated algorithm that generates characteristic-based factors, and a parsimonious model for returns and average returns that absorbs anomalies. The factor model provides a dimension-reduction formulation that summarizes the time series variation of thousands of securities to a small number of factors. Long-short 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 perspectives of accounting, trading, economics, or psychology as well as its construction. Therefore, how their sorted portfolios have a vast difference due to the minor difference in characteristics themselves is unclear in the literature.

意味着最简单的模型理论，具有最少的假设和变量，但具有最大的解释力。

beta: slope, alpha: vertical intercept

[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 question in Fama and French (1996): Does a factor model exist that dissects existing anomalies? The common approach in the literature is to test the model specification using existing factors. In this paper, we combine top-down and bottom-up approaches to create a unified framework: is it possible to use all characteristics to generate factors that explain existing anomalies? Our second question is, whether a conditional factor model, with conditional asymmetric betas, helps to reduce the pricing errors.

First, security sorting is a quantile function and can be viewed as a nonlinear activation function in a neural network. 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 by construction. Third, it is a greedy algorithm that takes the feedback of the loss function through the channel of backward propagation: how to decrease the pricing errors. Forth, after the factor generation, we can add another neural network structure to generate a conditional factor model to explore conditions like the bull-bear market. We use the activation function ReLU, rectified linear unit, and minimize the price errors in a conditional factor model.

To generate the deep factors, we use the stock universe with the annually largest 3000 firms in the U.S. equity market and 57 published characteristics from Green et al. (2017) and Hou et al. (2017). In an out-of-sample evaluation, we use the period 1975-2017 to train the model and generate deep factor to predict the period 2005-2017. Our conditional deep factor model outperforms the benchmark CAPM, FF3, and FF5 by 45.6%, 32.5%, and 2.2% in the average pricing errors. By considering the cross-section, the substantial over-performance are 48.6%, 35.6%, and 33.6% respectively for lower pricing errors. For these 57 anomalies, there are 33, 28, and 22 of them tested with significant alpha with respect to CAPM, FF3, and FF5. However, by controlling for the conditional deep factor model, there are 8, 13 and 3 anomalies tested with significant alphas.

The rest of the paper outlines 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 provides an introduction to the neural network and the relationship between Fama-French type factor models and deep learning. The details of the deep learning algorithm are discussed in section 3. The extension to a conditional factor model is provided in section 4. Section 5 illustrates the empirical study design and the findings for asset pricing testing. Section 6 summarizes 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 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

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

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

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

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

[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.

Risk-Premium PCA (RP-PCA), can beinterpreted as PCA generalized with a penalty term to account for the pricing error.

problem: how to generate long-short factors to minimize the average pricing errors. Our loss function is the same as the RP-PCA of Lettau and Pelger (2018), who provide a regularized estimation with a penalty accounting for cross-sectional variation for 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 the latent factors in the beta model. We also follow them to provide a list for choice of λ for robustness.

The second extension of our deep learning framework is a conditional factor model. We have a neural network to transform and combine firm characteristics before the security sorting. We can also add a post-sorting neural network to fit a conditional model within a unified framework. We consider a conditional model with Q states, where S_q represents a state.

$$R_{i,t} = \alpha_i + \sum_{q=1}^{Q} \left(\beta_i^{(q)\intercal} F_t + \gamma_i^{(q)\intercal} G_t \right) * \mathrm{I}\{S_q = 1\} + \epsilon_{i,t} \tag{5}$$

Fabozzi and Francis (1977) provide the early evidence of asymmetric beta in the bull and bear market conditions. Our automated algorithm uses the ReLU activation to decompose the model space into K states to minimize the pricing errors. The bull/bear market condition only considers the market factor, while our algorithm considers linear combinations of all factors. A description for our use of ReLU is provided in section 4. If every period has a different state, then the model becomes a time-vary beta time series regression.

1.2 Related Literature

The most 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, firm characteristics, as instruments to estimate PCs and further allows time-varying factor betas. 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 anomaly portfolios.

Our method generates the best-transformed characteristics for security sorting by combining the time series and cross-sectional variation. Using a purely statistical factor approach results in a few shortcomings. First, PCA relies on a balanced data structure: portfolios or a limited number of firms. Our deep factors are generated using individual firm returns to fit a different set of portfolios and allow for unbalanced data. 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 incremental deep factors on a benchmark model.

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. 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 to 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 or artificial intelligence is popular in the investment industry but still considered as a black box. Our deep factors, the hidden neurons, 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).

The conditional factor model is a nonlinear representation for factors in a linear formulation, thus the intercept alpha is still tradable. Lettau and Ludvigson (2001) find adding consumption-wealth ratio into the conditional CAPM can account for the difference in returns between sorted portfolios in book-to-market equity, while Lewellen and Nagel (2006) show conditional CAPM fails to explain anomalies like momentum and the value premium. Our conditional factor model uses the linear combination of deep factors and benchmark factors to mimic the bull and bear market conditions. This is an automated optimization to decompose the model space into a small number of states using the time-varying factor risk premia.

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, because deep factors are automatically generated by controlling for a benchmark model. Unlike maximizing the time series predictability, adding a factor or condition 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. Finally, the deep factors are tradable factors and can be tested if it is an anomaly.

2 Deep Learning Factor Alpha

In this section,we have a brief introduction to deep learning in section 2.1 and illustrate how to the Fama-French type model is implemented within a neural network in section 2.2. The perspective of nonlinear conditional model is briefly discussed in section 2.3.

2.1 Deep Learning

Artificial Neural Network is a pattern recognition machine learning method to use a high-dimensional data X to predict Y 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): $\max\{x, 0\}$.

The layer $X \in \mathbb{R}^K$ is represented by K neurons with activation functions performed on them. The weight matrix A is represented by a bunch of edges (or arrows) between 2 layers. The 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.

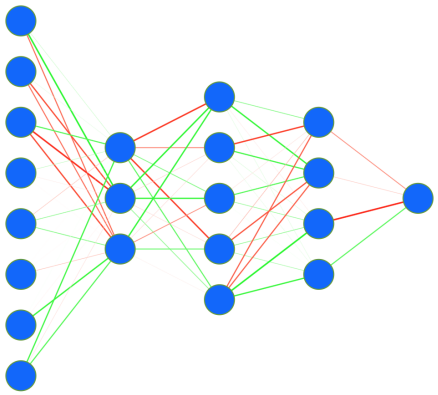

Figure 1: A Neural Network Diagram

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.

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

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

A neural network aims to learn hidden 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 $\hat{Y} = (f_L \circ ... \circ f_1)(X)$. For example, squared error loss $\mathcal{L}(\hat{Y}, Y) := \|Y - \hat{Y}\|_2^2$ is used in regression problems. 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

CMA: the investment factor

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 characteristics, there are other similar measures. 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 returns for certain months. Sorting securities on calculated characteristics might be a trial-and-error experiment to find the one with the best in-sample performance.

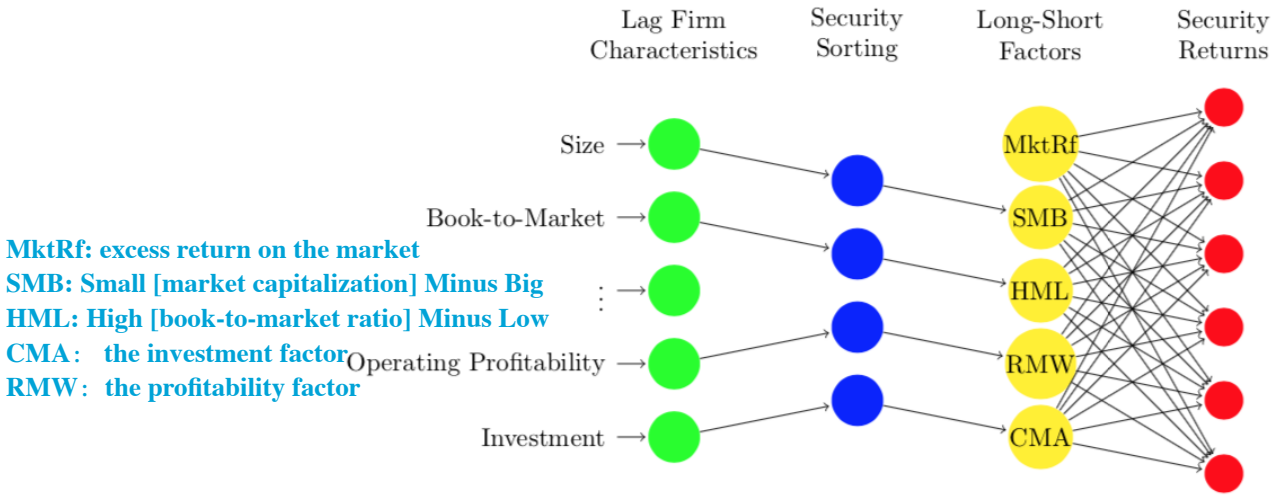

Figure 2: Fama-French 5-factor model as a neural network

In a neural network diagram, 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 factors with long-short portfolio by selecting those firms rank top and bottom in the sorting. The potential multi-layer transformations and combinations of characteristics are determined 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. However, the drawback of this approach is, the characteristics usefulness is tested

statistically, but the feedback for model fitting is never returned to characteristics construction.

2.3 Conditional Model in Deep Learning

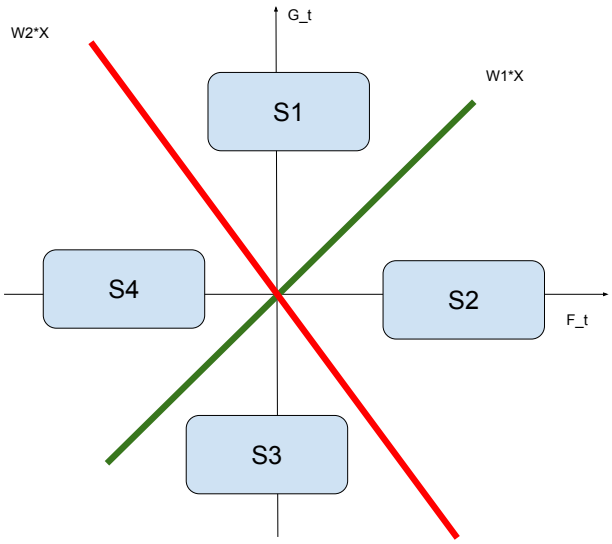

Figure 3: Conditional Model Space

Figure (3) shows the nature of non-linearity in a conditional model. Neural network is designed to capture the non-linearity, while linear factor models are useful because the linear combination is a tradable portfolio. A conditional linear factor model is under-appreciated for its built-in non-linearity as a portfolio. We use a ReLU neural network that searches the optimal bull-bear market conditions for a fixed number of states. If $X = [F_t, G_t]$, then we use the conditions $W_1 * X > 0$ and $W_2 * X > 0$ to separate four states for the factor model. We use the same set of factors for each states, but their factor betas are different in different states.

In short, we want to illustrate that the common approach to create characteristic-based factors for decades is part of a deep neural network. With an informed 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 search bull-bear market conditions to improve the model fitting.

3 Model Training

In this section, we discuss technical details for the unified deep learning model training, including multi-layer characteristics transformation and combination, tradable factor construction, factor model estimation, cross-sectional return forecast, and the loss function minimization.

3.1 Model Notation

With traditional neural network as building blocks, we are now ready to design the architecture for our deep learning factor alpha. Our deep learning framework mainly consists of 3 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 non-linearity 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 third part is optional. Given the augmented factor set, we are ready to price target assets. This part improves the linear factor model by separating the factor value space into a number of regions via a ReLU network. In each region, the augmented factor set is used to price target assets via a linear model. We allow different coefficients for the same factor in different regions. Specifically, the construction of deep factors is via 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:

$$\begin{cases} \{R_{i,t}\}_{i=1}^{N}, & \text{as excess returns of } N \text{ target assets} \\ \{r_{j,t}\}_{j=1}^{M}, & \text{as excess returns of } M \text{ individual firms} \\ \{Z_{k,j,t-1} : 1 \leq k \leq K\}_{j=1}^{M}, & \text{as } K \text{ lagged characteristics of } M \text{ firm} \\ \{G_{d,t}\}_{d=1}^{D}, & \text{as } D \text{ benchmark factors} \end{cases} \tag{6}$$

For simple notations, we formulate them in a 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:

$$X_{\cdot,j}^{[l]} = f^{[l]}\left(A^{[l]} X_{\cdot,j}^{[l-1]} + b^{[l]}\right), \text{ for } l = 1, 2, 3, ..., L \text{ and } j = 1, 2, ..., M \tag{7}$$

$$Z := X^{[0]}. \tag{8}$$

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

$$Y := X^{[L]}, \tag{9}$$

$$X^{[l]} = f^{[l]}\left(A^{[l]} X^{[l-1]} + b^{[l]}\right), \text{ for } l = 1, 2, 3, ..., L \tag{10}$$

$$Z := X^{[0]}. \tag{11}$$

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 (4) 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 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.

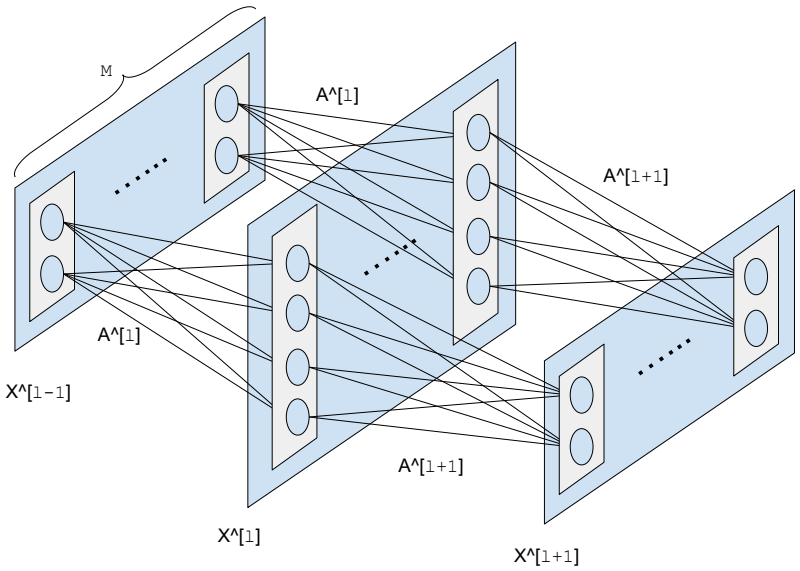

Figure 4: Deep network of $X^{[l-1]} \to X^{[l]} \to X^{[l+1]}$.

The deep learning network forwards from $X^{[l-1]}$ to $X^{[l+1]}$. $K_{l-1} = K_{l+1} = 2$, $K_l = 4$. The lines connecting 2 layers represent affine transformation and the circles represent activation function.

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 sorting and then an augmented factor model for asset pricing. The

architecture after L-th layer is as follows:

$$\hat{R} := X^{[L+4]} = h^{[4]}\left(X^{[L+3]}, G\right) \tag{12}$$

$$F := X^{[L+3]} = h^{[3]}\left(X^{[L+2]}, r\right) \tag{13}$$

$$W := X^{[L+2]} = h^{[2]}\left(X^{[L+1]}, V\right) \tag{14}$$

$$U := X^{[L+1]} = h^{[1]}\left(X^{[L]}\right) \tag{15}$$

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 a linear transformation of its 2 arguments, and the parameters are denoted as $\beta \in \mathbb{R}^{N \times P}$ and $\gamma \in \mathbb{R}^{N \times D}$.

$$h^{[4]}(F, G) = [\beta\ \gamma] \begin{bmatrix} F \\ G \end{bmatrix}. \tag{16}$$

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

$$h^{[3]}(W, r) = Wr. \tag{17}$$

The key procedures are $h^{[2]}$ and $h^{[1]}$. They essentially performs 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 sorted portfolios. Then $h^{[2]}$ outputs the "difference in averages" of the 4 sorted weights, using V as a vector of candidate weights. Finally the generated F are long-short portfolios.

3.3.1 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

$$h^{[1]}(y) = \begin{bmatrix} \mathbb{1}\{y_1 \geq q_\nu(y)\} \\ \vdots \\ \mathbb{1}\{y_j \geq q_\nu(y)\} \\ \vdots \\ \mathbb{1}\{y_M \geq q_\nu(y)\} \end{bmatrix} + \begin{bmatrix} \mathbb{1}\{y_1 \geq q_\tau(y)\} \\ \vdots \\ \mathbb{1}\{y_j \geq q_\tau(y)\} \\ \vdots \\ \mathbb{1}\{y_M \geq q_\tau(y)\} \end{bmatrix} - \mathbf{1}_M \tag{18}$$

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 $\nu = 0.1, \tau = 0.9$ 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

$$\left[h^{[1]}(y)\right]_{(j)} = \begin{cases} -1 & \text{if } \frac{j}{M} < \nu \\ 0 & \text{if } \nu \leq \frac{j}{M} < \tau \\ 1 & \text{if } \frac{j}{M} \geq \tau \end{cases} \tag{19}$$

To better understand the procedure of $h^{[1]}$, 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.

Finally, $h^{[2]}$ calculates portfolios weights given the market equity V and the result from $h^{[1]}$, x_1.

$$h^{[2]}(x_1, V) = \left[\frac{(x_1)_+ \odot V}{((x_1)_+ \odot V)'\mathbf{1}_M}\right] - \left[\frac{(x_1)_- \odot V}{((x_1)_- \odot V)'\mathbf{1}_M}\right] \tag{20}$$

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

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	$\beta F + \gamma G$	(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 learning factor alpha architecture

The deep learning network feeds forwards from bottom to top. The initial input is firm characteristics. For each layer, it take the output from the immediate lower layer as its input as well as the additional input, if exists. We summarize the output data dimension, operation and the parameters needed to be trained for each layer.

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

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

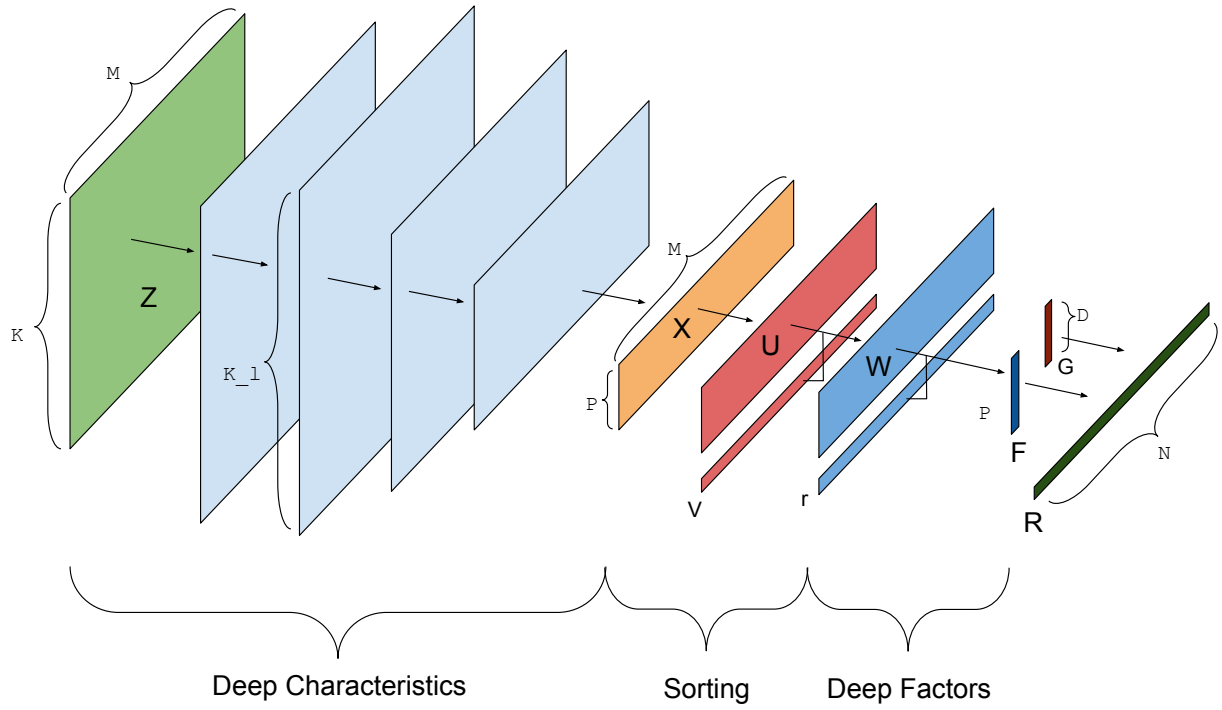

Figure 5: Deep learning factor alpha architecture

The firms characteristics Z are transformed to deep characteristics X via deep network. Then we sort X and combine the result U with market equity V to generate factor weight W. The deep factors F and benchmark factors G together are used to price the target asset return R.

where ddd

$$\hat{R}_{it} = \beta_i^\intercal F_t + \gamma_i^\intercal G_t$$
$$\alpha_i = \frac{1}{T}\sum_{i=1}^{T}\left(R_{it} - \hat{R}_{it}\right)$$

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 (β, γ). We use the notation $F_t^{A,b}$ to indicate the dependence of F_t on the parameters. The corresponding estimates are

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

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 $D \sim$ 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

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

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

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

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

4 Conditional Factor Model

The complete deep learning model training is discussed in section 3. In this section, we illustrate the extension for conditional factor model. We show how to create different bull-bear states by the neurons of ReLU activation, and how to covert it back to a conditional factor model. Figure (**??**)

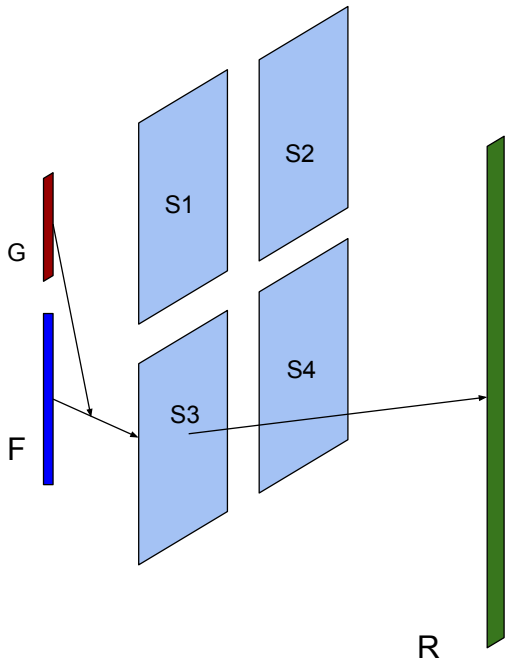

Figure 6: Conditional factor model

Deep factors F together with benchmark factors G are to price the target asset return R. The conditional linear model split the factor space into 4 states: S_1, S_2, S_3 and S_4. Each state has a different set of factor coefficients.

4.1 Bull-Bear States from ReLU

Given the constructed deep factors F and the existing factor G, we use simple linear models to price the target assets, $R_{i,t} = \beta_i^\intercal F_t + \gamma_i^\intercal G_t$ for $i = 1, 2, ..., N$. Here the coefficients β_i and γ_i for the $i-$th asset are fixed throughout the time. To introduce more flexibility and improve the pricing performance of the linear factor models, we consider further augmenting them with conditions. Those factor coefficients are allowed to be different depending on whether some condition is satisfied. We assume conditions to be linear, in the form of $Ax > 0$. This is equivalent to separating the factor

value space using a number of hyperplanes defined by A and fitting a linear model in each region. For example, when G is market excess return, we can allow its coefficient γ to differ between bull market ($G > 0$) and bear market ($G < 0$).

ReLU activation function is suitable for this purpose as it keeps the positive part of its input, for example the 1-dimensional G. To get the negative part, the simple way is to feed the same ReLU neuron with the negative input, $-G$. In this way, we separate the input G-space into 2 regions. The 2 outputs, $ReLU(G)$ and $ReLU(-G)$, are then with 2 different γ.

With a multi-layer ReLU network, we generalize this idea and separate the factor space into even more regions. The architecture of the conditional factor model is as follows:

$$
\begin{aligned}
R_{i,t} = \beta_i^\intercal F_t^{ReLU} &= \sum_{q=1}^{Q} \left(\beta^{(q)\intercal}_i F_t + \gamma^{(q)\intercal}_i G_t \right) * \mathrm{I}\{S_q = 1\} \\
F^{ReLU} &:= [F^{[L'],+}; F^{[L'],-}], \\
F^{[l],+} &= ReLU\left(\tilde{A}^{[l]} F^{[l-1],+} \right),\ F^{[l],-} = ReLU\left(\tilde{A}^{[l]} F^{[l-1],-} \right),\ \text{for } l = 1, 2, 3, ..., L' \\
F^{[0],+} &:= [F; G],\ F^{[0],-} := [-F; -G].
\end{aligned}
$$

where S_q represents a state.

4.2 Conditional Model Coefficient

Since ReLU function is piece-wise linear, we can unwrap F^{ReLU} and write them as simple conditional linear functions of $[F; G]$. Furthermore, given the β for F^{ReLU}, we can also recover the actual β for those original deep factor F and G.

For illustration, let's consider an 1-layer ReLU network where the output F^{ReLU} is of dimen-

sion P_1. The $2 * P_1$ outputs are

$$ReLU(\tilde{A}_1^\intercal F^{[0]}), ReLU(-\tilde{A}_1^\intercal F^{[0]}),$$
$$ReLU(\tilde{A}_2^\intercal F^{[0]}), ReLU(-\tilde{A}_2^\intercal F^{[0]}),$$
$$...,$$
$$ReLU(\tilde{A}_{P_1}^\intercal F^{[0]}), ReLU(-\tilde{A}_{P_1}^\intercal F^{[0]})$$

where $\tilde{A}^{[1]} = [\tilde{A}_1, \tilde{A}_2, ..., \tilde{A}_{P_1}]^\intercal$.

We have P_1 pairs of ReLU units here. For each row, say the p-th, one of the two units will outputs the absolute value of $\tilde{A}_p^\intercal F^{[0]}$ and the other outputs 0. We will see that these vectors, $\tilde{A}_1, \tilde{A}_2, ..., \tilde{A}_{P_1}$ are the basic components to define factor coefficients. Note that

$$\begin{aligned}
\beta^\intercal F^{ReLU} &:= \beta_+^\intercal F^{[1],+} + \beta_-^\intercal F^{[1],-} \\
&= \sum_{p=1}^{P_1} \beta_{p,+} ReLU(\tilde{A}_p^\intercal F^{[0]}) + \beta_{p,-} ReLU(-\tilde{A}_p^\intercal F^{[0]}) \\
&= \left(\sum_{p=1}^{P_1} A_p^+ \mathrm{I}\{\tilde{A}_p^\intercal F^{[0]} > 0\} + A_p^- \mathrm{I}\{\tilde{A}_p^\intercal F^{[0]} < 0\} \right) F^{[0]} \\
&:= \sum_{q=1}^{Q} \left(\beta^{(q)\intercal} F_t + \gamma^{(q)\intercal} G_t \right) * \mathrm{I}\{S_q = 1\}
\end{aligned}$$

where $A_p^+ = \beta_{p,+}\tilde{A}_p$ and $A_p^- = -\beta_{p,-}\tilde{A}_p$. Therefore, the conditional model assigns a scaled sum of $\tilde{A}_1, \tilde{A}_2, ..., \tilde{A}_{P_1}$ as the factor coefficients to $[F; G]$. The scale choice, between $\beta_{p,+}$ and $-\beta_{p,-}$, depends on whether the corresponding condition defined by $\tilde{A}_p$ is satisfied. Equivalently, the $F^{[0]}$-space is separated into 2^{P_1} regions by the P_1 hyperplanes and each regions has a different factor coefficient for $F^{[0]}$.

5 Empirical Findings

5.1 Data Information

The characteristics availability is from 1975 January to 2017 December. To calculate some lag 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 57 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 (A) in the Appendix. To evaluate the quality of the characteristics, we create the monthly-sorted factors using the top 10% and bottom 10% of firms. Using all 43 years of data, there are 33, 28, and 22 out of 57 factors tested with significant alphas by controlling CAPM, Fama-French three and five factors. During the 30-year training period 1975-2004, there are 30, 22, and 15 of them tested with significant alpha respectively.

5.2 Train-Validation-Test Design

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 (7). First, we use individual firm returns as train assets to create long-short factors because we need the firm characteristics for sorting. Second, we use a large combination of 202 sorted portfolios[3] as validation assets to calculate the training loss. In addition, we also provide a case with 57 anomalies as the validation asset. Third, we use another set of portfolios, 49 industry portfolios, as test asset to perform the model evaluation.

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 varia-

[3]We follow Giglio and Xiu (2017) to include 202 portfolios for the validation asset: 25 portfolios on size and book-to-market ratio, 17 industry portfolios, 25 portfolios on operating profitability and investment, 25 portfolios on size and variance, 35 portfolios on size and net issuance, 25 portfolios on size and accruals, 25 portfolios on size and beta, and 25 portfolio on size and momentum.

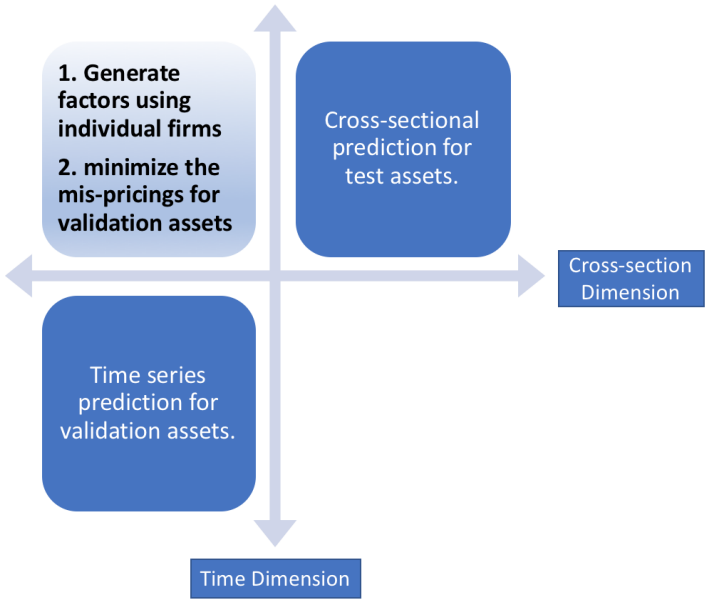

Figure 7: Train-Validation-Test asset design

tion. As in Figure (7), for time series forecast of future returns, it is a natural out-of-sample approach using the fixed factor and betas from the train sample. For cross-sectional evaluation of test assets, using factors created with train and validation assets is also an out-of-sample approach.

5.3 Out-of-Sample Forecasts

In Table (2), we separate the train and test periods as 1974-2004 and 2005-2017. We use the train data to train and select the conditional deep factor model, then report its out-of-sample improvements on the test data. The estimated factor betas are fixed in forecasting the future cross-section. The below measures are used to report the empirical results.

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

Adding more factors to a model does not necessarily decrease the cross-sectional RMSE or increase this relative 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, we can only compare the relative R-squared values in either the train and test data, but we can compare the cross-sectional RMSE in both data.

For time series forecasting, we find a substantial out-of-sample improvement for adding deep factors to CAPM for different validation assets Portfolio 202 or Anomaly 57. Adding deep factors to Fama-French three and five factors has also seen a large improvement for Anomaly 57. It is also noted in Lewellen et al. (2010) that Fama-French factors have great explanatory power to common sorted portfolios but not to existing anomalies. Using our deep factors, the time series forecast for other anomalies is large even by only controlling for CAPM.

Portfolio 202						
Model	TS IS RMSE	TS IS R2	TS OS RMSE	TS OS R2	CS OS RMSE	CS OS R2
CAPM	0.379%	0.000%	0.221%	0.000%	0.251%	0.000%
FF3	0.228%	39.721%	0.196%	11.253%	0.267%	-6.749%
FF5	0.192%	49.211%	0.156%	29.295%	0.357%	-42.523%
DL(3,3,1,tanh) + CAPM	0.106%	71.934%	0.191%	13.786%	0.147%	41.206%
DL(3,5,5,tanh) + FF3	0.112%	70.389%	0.187%	15.392%	0.194%	22.618%
DL(5,5,5,relu) + FF5	0.144%	62.039%	0.159%	27.986%	0.281%	-12.199%
Anomaly 57						
Model	TS IS RMSE	TS IS R2	TS OS RMSE	TS OS R2	CS OS RMSE	CS OS R2
CAPM	0.711%	0.000%	0.463%	0.000%	0.251%	0.000%
FF3	0.490%	30.994%	0.369%	20.262%	0.267%	-6.749%
FF5	0.353%	50.260%	0.229%	50.627%	0.357%	-42.523%
DL(5,5,1,tanh) + CAPM	0.135%	81.046%	0.252%	45.658%	0.129%	48.395%
DL(3,5,25,relu) + FF3	0.178%	74.885%	0.249%	46.129%	0.172%	31.346%
DL(5,5,5,tanh) + FF5	0.215%	69.761%	0.224%	51.654%	0.237%	5.587%

Table 2: Out-of-sample Forecasting

This table provides time series and cross-sectional forecast comparison. The first two columns are in-sample evaluation for train data, and the middle two are out-of-sample evaluation for test data. The last two columns are cross-sectional out-of-sample evaluation for the test asset: 49 industry portfolios. The benchmark model are CAPM, FF3, and FF5. The deep learning models are selected by cross-validation. The 3 numbers in the bracket correspond to number of factors, number of ReLU neurons and λ in the loss function; The last one is the activation function for deep characteristics generation.

For cross-sectional forecasting, we try to evaluate the model out-of-sample performance for 49 industry portfolios, which are different from the validation assets. We can see, Fama-French three and five factors even underperform CAPM. Adding more factors does not help to explain the cross-section and reduce the pricing errors. However, adding our deep factors is extremely useful to reduce the pricing error (RMSE) and increase the cross-sectional model fitting (R^2). The in-sample

evaluation on validation assets are provided in the first two columns, where we find consistent performance for deep factors in 57 anomalies and 49 industry portfolios.

5.4 Dissecting Anomalies

We follow the study of Fama and French (1996) to see the pricing performance of our to existing anomalies. We use our train deep learning model to dissect those 57 long-short factors in Table (A). The goal is to compare the evaluation performance for anomalies by different factor models. We simply compare the economical and statistical significance for different alpha estimates of 57 factors in Table (3).

For these 57 anomalies, there are 33, 28, and 22 of them tested with significant alpha with respect to CAPM, FF3, and FF5. However, by controlling for the conditional deep factor model, there are 8, 13 and 3 anomalies tested with significant alphas.

Table 3: Alpha Significance of Anomaly 57

This table provides the t-statistics of the alpha intercept test from regressing the 57 anomalies on different factor models correspondingly. The first row shows the number of significant alphas. The first column are variable names, whose descriptions are given in Table (A).

Variable	CAPM	FF3	FF5	DL + CAPM	DL + FF3	DL + FF5
Significance Count	33	28	22	8	13	3
mve0	0.34	-1.11	0.11	-0.42	-0.30	-0.36
beta	-3.23	-3.13	-0.62	-1.89	-1.94	-1.30
chmom	-0.13	-0.02	0.18	-0.59	1.21	-1.69
idiovol	4.31	5.16	2.31	0.78	2.43	0.02
indmom	0.45	0.60	-0.24	0.02	-0.67	-0.36
mom1m	2.27	2.65	2.36	1.44	1.36	0.92
mom6m	3.33	3.90	3.31	1.48	1.17	1.77
mom12m	3.72	4.57	3.77	2.13	1.77	0.55
pricedelay	1.57	1.79	1.96	0.39	2.12	1.93
absacc	-1.86	-0.93	0.87	-1.14	-0.60	0.61
acc	2.43	2.01	3.15	1.31	1.52	-0.02
age	-1.81	-0.91	3.20	2.66	0.64	2.73
agr	4.78	3.41	1.80	0.10	1.48	-0.59
bm	2.64	-0.41	-1.59	-0.61	-0.71	0.17
bm_ia	1.23	-0.66	0.77	-0.75	0.89	0.71
cashdebt	2.15	2.77	0.95	1.91	1.64	1.53
cashpr	4.04	2.37	3.00	1.04	1.99	0.91

Variable	CAPM	FF3	FF5	DL + CAPM	DL + FF3	DL + FF5
cfp	1.29	0.87	-1.41	0.96	0.36	0.16
chpmia	0.89	1.05	0.52	-0.88	0.34	-0.84
currat	2.67	2.27	-0.57	0.23	1.28	-0.88
depr	-1.85	-0.66	0.99	-0.13	0.49	0.82
dy	-2.29	-1.12	0.46	-1.32	-0.95	-0.65
egr	4.78	3.39	2.14	-0.47	1.46	1.45
ep	1.92	1.52	-0.30	0.16	0.34	0.10
gma	0.62	2.17	1.10	-0.65	0.46	0.63
herf	-1.02	-1.37	-2.25	-0.99	-0.41	-0.47
hire	2.91	1.39	-1.24	0.27	0.23	-1.33
invest	5.40	4.39	3.90	2.74	3.70	1.95
lev	3.00	0.59	-2.45	0.74	0.66	-1.36
lgr	4.02	2.80	1.20	0.52	1.49	0.11
mve_ia	0.92	0.15	0.60	2.12	0.35	0.66
operprof	3.18	3.52	0.00	0.99	1.66	-0.42
pchcapx_ia	1.39	1.30	-1.03	0.06	0.43	-0.82
pchcurrat	2.11	2.25	1.71	-1.19	0.38	0.63
pchdepr	1.52	1.80	2.60	1.91	2.32	2.10
pchgm_pchsale	1.86	2.68	2.17	1.14	1.82	-0.16
pchquick	0.90	0.92	0.43	-0.93	0.63	-0.07
pchsale_pchrect	0.11	-0.47	-0.82	0.15	-1.93	-0.09
pchsale_pchxsga	2.10	1.27	0.79	-1.66	0.39	0.59
pctacc	1.22	0.67	-0.21	0.94	-0.36	-1.06
ps	3.97	4.44	2.72	0.52	3.04	0.70
quick	2.78	2.07	-1.12	-0.55	0.87	-1.21
roic	3.06	3.79	1.35	0.49	2.40	0.43
salecash	3.01	1.67	-2.15	-1.58	-0.34	-1.81
saleinv	3.27	3.13	1.50	0.46	2.06	-1.09
salerec	4.69	4.11	2.41	-0.15	2.55	0.30
sgr	2.36	0.58	-0.99	-0.18	0.01	0.35
sp	3.85	1.78	-1.44	-0.36	1.02	-1.73
tang	1.64	0.27	-2.64	-0.38	0.77	-1.60
tb	1.14	1.66	1.43	1.57	1.38	0.69
baspread	3.65	4.18	1.11	2.61	2.30	0.46
maxret	3.35	3.48	0.46	0.99	1.39	-0.40
retvol	4.55	5.11	2.38	2.77	2.82	1.15

We also follow Table 2 of Fama and French (2016) to evaluate four sets of anomaly sorted portfolios correspondingly: 25 portfolios on size and accruals, 25 portfolios on size and market beta, 25 portfolios on size and variance, and 25 portfolios on size and residual variance. The goal is to evaluate the model specification using the average pricing errors.

In Table (4), we repeat the cross-sectional out-of-sample evaluation using each 25 sorted portfolio above. The finding is consistently positive: adding deep factors help reducing the pricing errors (RMSE) and increasing the cross-sectional model fitting (R^2). Fama-French three and five factors only do well for 25 portfolios size and beta. However, simply adding deep factors to CAPM can increase the relative R^2 by 70% in average.

Portfolio 202								
	ME_beta25		ME_ac25		ME_var25		ME_resvar25	
Method	RMSE	R2	RMSE	R2	RMSE	R2	RMSE	R2
CAPM	0.365%	0.000%	0.255%	0.000%	0.534%	0.000%	0.537%	0.000%
FF3	0.123%	66.214%	0.163%	36.244%	0.337%	36.851%	0.348%	35.253%
FF5	0.110%	69.956%	0.175%	31.278%	0.233%	56.459%	0.228%	57.601%
DL(3,3,1,tanh) + CAPM	0.130%	64.495%	0.075%	70.631%	0.166%	68.996%	0.167%	68.891%
DL(3,5,5,tanh) + FF3	0.099%	72.880%	0.103%	59.695%	0.197%	63.224%	0.198%	63.094%
DL(5,5,5,relu) + FF5	0.109%	70.165%	0.129%	49.307%	0.152%	71.616%	0.149%	72.294%
Anomaly 57								
	ME_beta25		ME_ac25		ME_var25		ME_resvar25	
Method	RMSE	R2	RMSE	R2	RMSE	R2	RMSE	R2
CAPM	0.365%	0.000%	0.255%	0.000%	0.534%	0.000%	0.537%	0.000%
FF3	0.123%	66.214%	0.163%	36.244%	0.337%	36.851%	0.348%	35.253%
FF5	0.110%	69.956%	0.175%	31.278%	0.233%	56.459%	0.228%	57.601%
DL(5,5,1,tanh) + CAPM	0.109%	70.249%	0.080%	68.789%	0.136%	74.465%	0.135%	74.931%
DL(3,5,25,relu) + FF3	0.106%	70.898%	0.086%	66.257%	0.160%	69.983%	0.165%	69.297%
DL(5,5,5,tanh) + FF5	0.090%	75.304%	0.101%	60.366%	0.149%	72.103%	0.146%	72.907%

Table 4: Dissecting Anomaly sorted Portfolios

This table provides the cross-sectional out-of-sample evaluation for four different anomaly sorted portfolios used in Fama and French (2016). The benchmark model are CAPM, FF3, and FF5. The deep learning models are selected by cross-validation in Table (2). The 3 numbers in the bracket correspond to number of factors, number of ReLU neurons and λ in the loss function; The last one is the activation function for deep characteristics generation.

6 Conclusion

In this paper, we propose a deep learning factor alpha model to establish a deep learning representation for the characteristics-based factors and the conditional factor model. We try to address an asset pricing question: whether many existing anomalies can be dissected.

Our method includes two building blocks. First, we generate hidden deep factors using firm fundamentals through a multi-layer neural network. Second, we incorporate multiple state conditions to explore the bull-bear asymmetric factor exposures. We show a couple of encouraging results about the development of deep learning in asset pricing research. Any naive application of deep learning can quickly fail. The key is to adopt the recent deep learning methods within those workhorse empirical models in this field.

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. We also study a second connection between the linear asset pricing portfolio and the non-linear neural network. We provide an automated search of the bull-bear state conditions for the asymmetric factor exposures. We study an asset pricing optimization problem: minimizing the average pricing errors. This conditional factor model gives the tremendous advantage to the nonlinear model space.

Finally, 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-stationarity. A simple application of deep learning does not exist with the short-history of financial data and the imbalanced data structure. The nonlinear security sorting and the conditional model could be starting points for many future applications.

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.

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.

Fabozzi, F. J. and J. C. Francis (1977). Stability tests for alphas and betas over bull and bear market conditions. *The Journal of Finance 32*(4), 1093–1099.

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.

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.

Giglio, S. and D. Xiu (2017). Asset pricing with omitted factors. Technical report, Yale University.

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.

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.

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

Lettau, M. and S. Ludvigson (2001). Resurrecting the (c) capm: A cross-sectional test when risk premia are time-varying. *Journal of Political Economy 109*(6), 1238–1287.

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

Lewellen, J. and S. Nagel (2006). The conditional capm does not explain asset-pricing anomalies. *Journal of financial economics 82*(2), 289–314.

Lewellen, J., S. Nagel, and J. Shanken (2010). A skeptical appraisal of asset pricing tests. *Journal of Financial economics 96*(2), 175–194.

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.

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.

Appendix A Characteristics and Anomaly Summary

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	Characteristics 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%
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%

Variable	Characteristics Description	Average Return	Sharpe Ratio
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%