Deep Learning in Asset Pricing[*]

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

February 7, 2019

Abstract

Deep learning provides a framework for characteristics-based factor modeling in empirical asset pricing. We provide a systematic approach for long-short factor generation with a goal to minimize pricing errors in the cross section. Security sorting on firm characteristics provides a nonlinear activation function as part of a deep learning model. Our deep factors are tradable and allow for both nonlinearity and interactions between predictors. For cross-sectional return prediction, we study monthly U.S. equity returns based on lag firm characteristics and macro predictors from 1975 to 2017 with a universe of 3,000 stocks. Finally, with additional deep factors, we find out-of-sample forecast improvements for anomaly-sorted and industry portfolios.

Key Words: Alpha, Cross-Sectional Returns, Deep Learning, Firm Characteristics, Machine Learning, Risk Factors, Security Sorting.

[*]The paper was previously circulated under the title "Deep Learning Factor Alpha". We appreciate insightful comments from Serge Darolles, Li Deng, Jin-Chuan Duan, Shihao Gu, Bryan Kelly, Soohun Kim, Ding Luo, Dacheng Xiu, and Ti Zhou. We are also grateful to helpful comments from seminar and conference participants at SUSTech, ESSEC Business School, City University of Hong Kong, R/Finance, EcoSta, SOFIE Summer School, CQAsia, Wolfe Research, Australian Finance&Banking Conference, New Zealand Finance Meeting. We acknowledge the research award 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`.

1 Introduction

Deep learning, among various machine learning methods (see Gu, Kelly, and Xiu (2018) and Kozak, Nagel, and Santosh (2017)), is shown to provide a powerful framework for cross section and time series stock return prediction. A major goal of empirical asset pricing is understanding cross-sectional and time-series properties of stock returns. From the perspective of ICAPM, a combination of common factors captures the cross section of average returns and the intercept should be zero. Many asset pricing models, however, fail the zero-alpha hypothesis from an empirical viewpoint. We approach the asset pricing problem as a deep learning problem: How does one construct a factor model to minimize the out-of-sample average pricing errors?

On the methodology side, we provide a deep learning framework for the characteristics-based factor model, and we introduce a systematic approach for long-short factor generation. Fama-French factor models are shown to be deep learners with a shallow network. In our framework, asset pricing tests based on alphas and anomalies are examined together with other asset pricing implications. A crucial distinction is that we focus on estimating an entire reduced-form asset pricing model, rather than testing specific factor, or characteristics.

On the empirical side, we show how to generate long-short portfolios by composing deep learner with a benchmark model that minimizes out-of-sample average pricing errors. A traditional "feed-forward" network consisting of an "input layer" of stock characteristics, "hidden layers" of pricing factors and an "output layer" of portfolio returns is used as our architecture. By their very nature, deep learners are nonlinear hidden-factor models. The generation of factors receives feedback from the loss function through backward propagation. This addresses the question of how much can cross-sectional and time series variation be reduced by optimizing over model parameters? This sequential training scheme is different from a traditional protocol, where researchers perform the GRS joint test and stop at the hypothesis rejection, which is why we focus on an optimization problem instead of hypothesis testing.

Our methodology marries current machine learning techniques with asset pricing factor models. Factor modeling provides a dimension-reduction that summarizes the time series variation for thousands of stock returns into a small number of factors. Long-short factors are useful as they reflect compensation for exposure to underlying characteristics and can be evaluated by the intercept alpha relative to a benchmark. Many of these characteristics calculations are highly similar to each other from an accounting, trading, macroeconomics, and behavioral perspective and deep learning provides a way of analyzing how sorted portfolios have an effect on pricing.

Our approach provides an automatic factor generation based on characteristics and a parsimonious model for returns with average returns that minimize alphas. First, we show that security sorting is a nonlinear activation function within the deep layers of our network. Second, rather than sorting on characteristics in a shallow network, deep learning searches for the best nonlinear transformation and interactions in deeper multi-layer networks. Third, one can control for a benchmark model during the search process.

We apply our methodology to the stock universe of the largest 3,000 stocks in the U.S. equity market. Security sorting and long-short factors are generated with the top and bottom 20% of stocks. Both fundamental characteristics, macroeconomic market-timing predictors and their interactions are used to train the network. We perform a variety of empirical exercises that illustrate the use of our procedure in the data. We find a large model (trained with characteristics and macro predictors) outperforms the small model (trained with characteristics), and a shallow network outperforms a deep network. Finally, with additional deep factors, we find improvement for the out-of-sample model forecast to anomaly-sorted and industry portfolios.

The rest of the paper is organized as follows. We compare our method with the relevant literature in section 1.1. Section 2 introduces the method details in asset pricing. Section 3 provides the deep learning architecture. Section 4 illustrates the empirical study design and our findings. Section 5 adds a final discussion and directions for future research.

1.1 Connections with Previous Literature

Our paper builds on several strands of the asset pricing literature. The most related literature are factor models using principal component analysis (PCA) and its generalizations. For example, Kelly et al. (2018) and Kim et al. (2018) use firm characteristics as instruments to model the time-varying coefficients and estimate principal components. Lettau and Pelger (2018) derive properties of RP-PCA that identify factors with small time series variance which are useful in the cross-section. Kozak et al. (2018) show that PCA of anomaly portfolios works as well as a reduced-form factor model in explaining anomaly portfolios. Huang et al. (2018) show that a reduced-rank approach (RRA) outperforms the Fama-French five-factor model in pricing portfolios.

Our deep learning framework differs from PCA in three main ways:

1. PCA relies on a balanced data structure with a fixed number of portfolios or firm returns. Security-sorted long-short factors allow for an unbalanced data structure, where only the top and bottom sorted firm returns are needed.

2. Our deep factors can be created with firm returns to fit a different set of test portfolios, whereas PCA mainly maximizes the variation from the same cross section.

3. PCA can have a poor out-of-sample performance due to fixed PC loadings while our deep factors can be updated by monthly sorting.

The second area is testing return predictability using macro market-timing predictors. The influential work of Welch and Goyal (2007) who examine 14 predictor variables and find little forecasting power in univariate forecasting regressions. Van Binsbergen and Koijen (2010) use a latent-variables approach and find that returns are predictable with R-squared values ranging from 8.2-8.9 percent. Gu et al. (2018) provide a comprehensive empirical investigation of forecasting performance for multiple machine learning algorithms using both characteristics and macro predictors. We build on this literature by adding macro predictors to create sorted portfolios which is new in this literature. Lag predictors for one-month ahead sorting securities create simultaneous pricing factors.

We add to the literature on forecasting stock returns with machine learning. Freyberger et al. (2017) apply adaptive group LASSO for firm characteristics selection and provide evidence of non-linearity. Light et al. (2017) use partial least squares (PLS) to aggregate information on firm characteristics. Han et al. (2018) employ a forecast combination approach and Bianchi et al. (2018) find that machine learning models can forecast bond returns well. One advantage of our approach is that employing security sorting and factor generation is not restricted to an unbalanced panel data structure.

Our approach is closely related to the recent literature on high dimensionality cross-sectional asset pricing models. Harvey et al. (2016) describe multiple testing issues in the zoo of factors that have been produced in the last 40 years. Feng et al. (2019) provide a regularized two-pass cross-sectional regression to tame the factor zoo, and find recursively a small number of factors with incremental contribution. Kozak et al. (2017) use a shrinkage estimator on the SDF coefficients for characteristic-based factors with economic interpretation. Kelly et al. (2018) evaluate the contribution of individual characteristics under a nested-model comparison via R^2 reduction.

The current literature typically evaluates the contribution of new factors relative to some benchmark model through their in-sample risk premium, asset pricing model fit, or alpha significance. Our framework focuses on out-of-sample performance for adding deep factor in an unseen validation sample. We evaluate the deep factor by the model fit measured by the change in average pricing errors. We use the validation sample to select the best model and evaluate the model's forecasting power in another unseen test sample. This train-test-validation empirical design is built into our deep learner.

Finally, we add to the recent development of deep learning in finance and econometrics. LeCun et al. (2015) and Goodfellow et al. (2016) provide comprehensive summaries about how the neural network develops in the modern deep learning, which has attracted tremendous attention in recent years for big data and artificial intelligence. Heaton et al. (2017) introduces deep learning decision models for problems in financial prediction and classification, whereas Polson and Sokolov (2017) provide a Bayesian interpretation of the neural network. Feng et al. (2018) provide a conditional linear forecasting model within a neural network. This continued progress in the deep learning research is promising for both academic research and practical application in finance.

2 Deep Learning Asset Pricing

Section 2.1 demonstrates how a characteristics-based factor model can be reformulated within a deep learning architecture. Fama-French type models are shown to be deep learners and implementation is discussed in Section 2.2. A nonlinear conditional deep factor model is described in Section 2.3. Appendix A provides all the technical details for the deep learning conditional model construction.

2.1 Minimizing pricing errors

We study an optimization problem to create a deep learner to minimize pricing errors.Let $\widehat{R}_{i,t}$ be a linear portfolio constructed to mimic the portfolio return $R_{i,t}$. All predictors need to be tradable portfolios in the spirit of the time series regression. $\widehat{R}_{i,t}$ is formed as a linear combination of portfolios *without an intercept*. The time series expectation difference, α_i, is the forecast bias or pricing error:

$$\mathrm{E}(R_{i,t} - \widehat{R}_{i,t}) = \alpha_i. \tag{1}$$

Our goal is to generate deep factors, f_t, on a benchmark model g_t, which can be CAPM or Fama-French type models, to form $\widehat{R}_{i,t}$ and minimize the average pricing errors:

$$\begin{aligned} \widehat{R}_{i,t} &= \beta_i^\intercal f_t + \gamma_i^\intercal g_t & (2) \\ R_{i,t} - \widehat{R}_{i,t} &= \alpha_i + \epsilon_{i,t}. & (3) \end{aligned}$$

We use excess portfolio returns for $R_{i,t}$. The zero mean residual, $\epsilon_{i,t}$, measures the time series variation in forecasting error.

The deep factors, f_t, are long-short portfolios constructed by sorting individual firms on lag predictors, for example, firm characteristics z_{t-1} and macro predictors x_{t-1}:

$$\begin{aligned} f_t &= W_{t-1} r_t & (4) \\ W_{t-1} &= \mathrm{H}\Big(z_{t-1} \otimes x_{t-1}\Big). & (5) \end{aligned}$$

Here, r_t is the vector of individual firm returns, and W_{t-1} is the long-short portfolio weight for each firm. $\mathrm{H}(\cdot)$ is a complex function for $z_{t-1} \otimes x_{t-1}$, which includes lag firm characteristics, lag macro predictors, and their interactions. Deep learning constructs $\mathrm{H}(\cdot)$ by a multi-layer multi-neuron network structure. From a forecasting perspective, the long-short factor f_t is built with only information at time $t-1$.

In our framework, f_t is generated while controlling g_t within the deep learning architecture.

The tradable alphas, namely, cross-sectional pricing errors, are constructed as

$$\widehat{\alpha}_i = \frac{1}{T}\sum_{t=1}^{T}\Big(R_{i,t} - \widehat{\beta}_i^{\intercal} f_t - \widehat{\gamma}_i^{\intercal} g_t\Big). \tag{6}$$

Our optimization objective is to minimize the squared sum of pricing errors:

$$\begin{aligned}
\mathcal{L} &= \frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\Big(R_{i,t} - \widehat{R}_{i,t}\Big)^2 \\
&= \frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\Big(R_{i,t} - \beta_i^{\intercal} f_t - \gamma_i^{\intercal} g_t\Big)^2 \\
&= \frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\Big(\epsilon_{i,t} + \alpha_i\Big)^2 \\
&= \underbrace{\frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\epsilon_{i,t}^2}_{\text{time series variation}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N}\alpha_i^2}_{\text{cross-sectional variation}} + \underbrace{\frac{2}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\alpha_i\epsilon_{i,t}}_{\text{estimation error}}.
\end{aligned}$$

The core of our loss function design, $\frac{1}{N}\sum_{i=1}^{N}\alpha_i^2$, is an equally weighted version of the pricing error joint test statistic[1] and measures the average pricing errors. It follows from the economic constraint from the beta-pricing model: The excess asset return can be explained by the risk premia of factors:

$$\mathrm{E}(R_{i,t}) = \alpha_i + \beta_i^{\intercal}\mathrm{E}(f_t) + \gamma_i^{\intercal}\mathrm{E}(g_t), \tag{7}$$

The goal of our deep learning architecture is to minimize all α_i jointly.

Our loss function is similar to the RP-PCA of Lettau and Pelger (2018), who add a penalty to account for cross-sectional variation in average returns. They add the cross-sectional variation to identify those factors with small time series variation, but help price the cross section. They use a tuning parameter to balance the weights between time-series and cross-sectional variation, whereas we use an equally weighted version.

[1]Given the part for time-series variation, replacing with the weighted average version, $\alpha^{\intercal}\Sigma_{\alpha}^{-1}\alpha$, has a relatively small impact on changing the prediction performance.

2.2 Fama-French and Deep Learning

The factor zoo contains many similar firm characteristics used to proxy for the same fundamental information. For example, multiple momentum factors exist: long-term reversal (13-60), short-term reversal (1-1), the Carhart Momentum (2-12), seasonality (1-13), and so forth. Sorting securities on these calculated characteristics might be a trial-and-error experiment, which finds the one with the best in-sample performance but does not guarantee any out-of-sample performance.

Figure 1: Fama-French 5-Factor Model as a Deep Learner

This figure provides a deep learner representation of building Fama-French five factors using firm characteristics to forecast portfolio returns, where the factors are hidden neurons and security sorting is an activation.

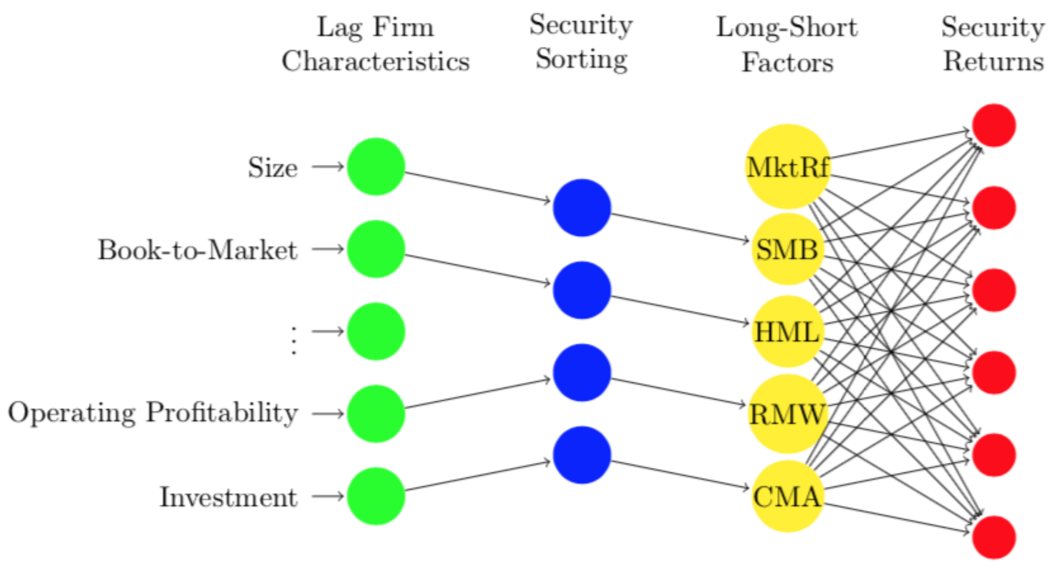


Figure 1 shows a deep learner representation of Fama-French factors modelling firm characteristics to portfolio return forecasts. Researchers typically start with a formula for the calculation of characteristics used for security sorting as in the green circles. At the beginning of the period, they sort individual firms on the lag characteristics to determine the top and bottom stocks as in the purple circles.[2] Then, in the yellow circles, researchers construct factors as long-short portfolios by the top and bottom stock portfolios. Along with the market factor, an augmented factor model is

[2]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 works for the imbalanced panel data structure with missing values in the nature of firm dynamics.

produced to forecast stock returns in the red circles.

The potential multi-layer transformations and combinations, denoted by $H(\cdot)$, of characteristics are determined before the green circles. Here, researchers typically determine the formula for anomalies that help pricing in the cross section. A major drawback of this approach is that the characteristics' usefulness is tested statistically ex post, but the feedback for model fitting is never returned to characteristics construction.

Fama and French (2015) add two additional factors, Robust-minus-Weak and Conservative-minus-Aggressive to form a five-factor model. The five-factor model still fails with the GRS test and the zero-alpha hypothesis remains rejected. Continuously adding factors does not necessarily decrease the regression intercept, namely, the pricing error, even if the factor added has a significant risk premium or risk price. For this reason, we propose changing the hypothesis testing problem into an optimization problem. This new approach to asset pricing uses the flexible deep learning architecture.

2.3 Conditional Deep Learning Models

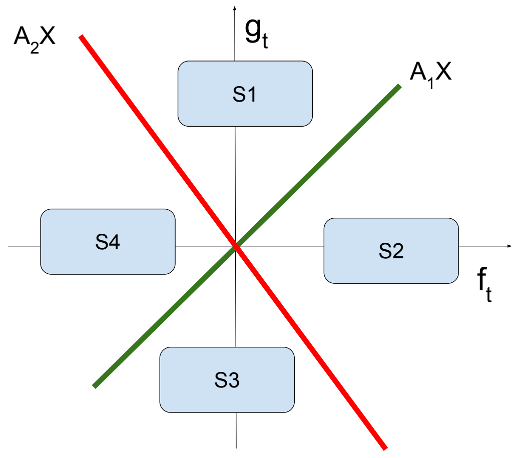
One important extension of our deep learning framework is a conditional linear factor model. We have a neural network to transform and interact firm characteristics before the security sorting. We can also add a second post-sorting neural network to fit a conditional model to explore the nonlinearity. Particularly, we follow Feng et al. (2018) and use the ReLU activation to construct a data-driven conditional linear model. We define a conditional model with Q conditional states, where S_q represents the q-th state,

$$\widehat{R}_{i,t} = \sum_{q=1}^{Q} \left(\beta_i^{(q)\intercal} f_t + \gamma_i^{(q)\intercal} g_t \right) * \mathbb{1}_{S_q}. \tag{8}$$

One motivation for the conditional model is the "up and down" market asymmetry in market beta. We follow a similar definition for the "up and down" market in Fabozzi and Francis (1977), such that the market factor is positive or negative in the same period. This approach is nonlinear and exploits both time-varying market risk premia and time-varying factor loading. To generalize the linear conditions, we use linear combinations of f_t and g_t to determine the "up and down" conditional states for the conditional model.

Figure 2: Conditional Model Space

This figures illustrates the separation of linear conditions into the factor space. The conditional linear model separates it into 4 states, using linear constraints of the factors.



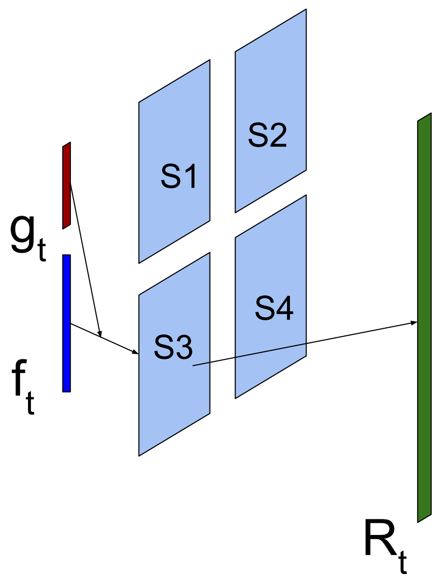
Our model adopts the rectified linear unit (ReLU) activation to decompose the factor space into Q states to minimize the loss function. The "up and down" market condition only considers the market factor, whereas our approach considers multiple linear combinations of all factors. Figure 2 demonstrates how the neural network decomposes the factor space. A ReLU neural network then searches for the optimal linear conditions with a fixed number of states. in Figure 2 with $X = [f_t : g_t]$, we use the conditions $A_1X > 0$ and $A_2X > 0$ to separate the factor space into four states. This approach is equivalent to separating the factor value space using a number of hyperplanes defined by A_1 and A_2 and fitting a linear model in each region. This allows factor loadings to be different in each region. The linear conditions and the number of states are data-driven parameters and hyper-parameters trained in deep learning.

Figure 3 is an illustration of the conditional factor model. We take an example about the "up and down" market. ReLU activation allows the factor loading to differ between "up" ($g_t > 0$) and "down" ($g_t < 0$) market conditions. Then, $X = g_t$ and the parameter A is a scalar 1.

The ReLU activation function is suitable for this purpose because it keeps the positive part of its input. A simple way to obtain the negative part is to feed another ReLU unit with the negative input, $-g_t$. In this way, we separate the input g_t-space into two regions. The two outputs, $ReLU(g)$

Figure 3: Conditional Factor Model

This figures shows how deep factors f_t together with benchmark factors g_t are used to fit the test portfolios R_t in a conditional model. The factor space is separated into 4 states.



and $ReLU(-g)$, are then multiplied by two factor loadings. With a multi-layer ReLU network and multi-dimensional factors $X = [f_t : g_t]$, we generalize this idea and separate the factor space into even more regions. Please refer to Appendix A for further details of the architecture and how we recover the conditional model coefficients from neural network parameters.

3 Deep Learning Factor Models

In this section, we discuss the details of our unified deep learner. Section 3.1 illustrates how a feed-forward neural network is built to construct the deep characteristics. Section 3.2 provides the construction of deep factors via security sorting. Appendix B shows construction details for security sorting, and Appendix C shows optimization details.

Our deep learning framework consists mainly of two parts. The first part takes lagged firm characteristics, macro predictors, and their interactions as input and feeds them into a multi-layer network. It simultaneously performs dimension reduction and nonlinearity extraction. The flexibility of a deep network allows us to search for any possible patterns and summarize them in the deep characteristics as output.

The generation of deep characteristics is then supervised by the second part, which implements the security sorting and long-short factor generation. Then, we combine deep factors with benchmark factors to fit test portfolios and calculate the loss function. A typical training observation indexed by time t includes five types of data:

$$\begin{aligned}
&\{R_{i,t}\}_{i=1}^{N}, \text{ excess returns of } N \text{ test portfolios} \\
&\{r_{j,t}\}_{j=1}^{M}, \text{ excess returns of } M \text{ individual stocks} \\
&\{z_{k,j,t-1} : 1 \le k \le K\}_{j=1}^{M}, \ K \text{ lagged characteristics of } M \text{ firms} \\
&\{x_{e,t-1}\}_{e=1}^{E}, \ E \text{ lagged macro predictors} \\
&\{g_{d,t}\}_{d=1}^{D}, \ D \text{ benchmark factors.}
\end{aligned}$$

We use a matrix notation for $\{R_t, r_t, x_{t-1}, z_{t-1}, g_t\}$ where 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; x_{t-1} is a $E \times 1$ vector; g_t is a $D \times 1$ vector.

3.1 Deep Characteristics

We now show how to design a L-layer neural network with the purpose of generating P deep characteristics. We drop for now the subscript t, bearing in mind that the inputs z and x are lagged variables. The architecture is as follows, for $j = 1, 2, ..., M$:

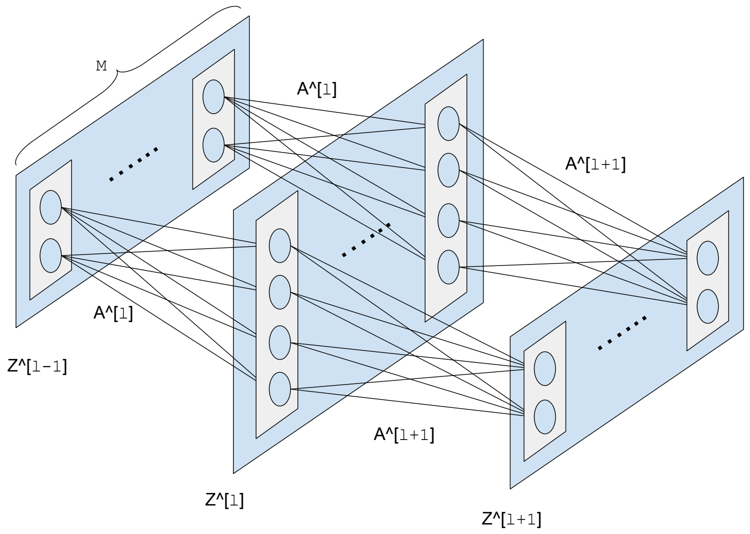
$$\begin{aligned}
&Z_{\cdot,j}^{[l]} = F\Big(A^{[l]} Z_{\cdot,j}^{[l-1]} + b^{[l]}\Big), \text{ for } l = 1, 2, ..., L \\
&Z_{k,j}^{[0]} := [z_{1,j}, ..., z_{K,j}, \ x_1, ..., x_E, \ z_{1,j}x_1, ..., z_{K,j}x_1, \ ..., \ z_{1,j}x_E, ..., z_{K,j}x_E]^{\intercal},
\end{aligned}$$

where $Z_{\cdot,j}^{[l]}$ is the j-th column of a $K_l \times M$ matrix $Z^{[l]}$. We set $K_0 = K + E + KE$ and $K_L = P$. F is the univariate activation function, 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,

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

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

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



The transformations are made column by column with no communication across different firms unless an industry indicator is added. This univariate transformation is perfectly built for the security sorting for different stock universes. Notice that, the input layer for deep characteristics is a linear function for firm characteristics, macro predictors, and their interactions. The multi-layer structure helps to train the parameters for this linear equation. Our deep characteristics are not built for one particular characteristic or macro predictor, but the linear combinations.

With some abuse of notation, we rewrite the architecture for the output Y is our $P \times M$ deep characteristics,

$$\begin{aligned} Y &:= Z^{[L]}, \\ Z^{[l]} &= F\Big(A^{[l]} Z^{[l-1]} + b^{[l]}\Big), \text{ for } l = 1, 2, 3, ..., L \\ Z^{[0]} &:= z \otimes x. \end{aligned}$$

Unlike a standard feed-forward neural network, the l-th layer in our architecture is a neural matrix $Z^{[l]}$. Each row of $Z^{[l]}$ is a $1 \times M$ vector representing the k_l-th intermediary 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. Therefore, the formula for deep characteristics is the same for every firm.

Here K_l denotes the dimension of the l-th layer because the number of columns is fixed as M for all $Z^{[l]}$'s. Figure 4 illustrates how our deep-learning network operates by showing a sample architecture from the $(l-1)$-th to the $(l+1)$-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. By contrast, $Z^{[0]} := z \otimes x$ in our deep network goes through multiple layers of affine transformations and nonlinear activations, and ends up with a low-dimensional deep characteristic Y. Here, the layer sizes $\{K_l\}_{l=1}^{L}$, and the number of layers L are architecture parameters chosen by model designers.

3.2 Deep Factors

With the deep characteristics generated from the first part, our deep framework continues with the construction of deep factors via sorting and then an augmented factor model for asset pricing. The architecture after the L-th layer is as follows:

$$\begin{aligned}
\hat{R} &:= Z^{[L+4]} = h^{[4]}\left(Z^{[L+3]}, g\right) \\
f &:= Z^{[L+3]} = h^{[3]}\left(Z^{[L+2]}, r\right) \\
W &:= Z^{[L+2]} = h^{[2]}\left(Z^{[L+1]}, v\right) \\
u &:= Z^{[L+1]} = h^{[1]}\left(Y\right).
\end{aligned}$$

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

We now describe these four operators in detail. $h^{[4]} : \mathbb{R}^P \times \mathbb{R}^D \to \mathbb{R}^N$ is a linear transformation of its two arguments, and the parameters are denoted as $\beta \in \mathbb{R}^{N\times P}$ and $\gamma \in \mathbb{R}^{N\times D}$:

$$h^{[4]}(f, g) = [\beta \; \gamma] \begin{bmatrix} f \\ g \end{bmatrix}. \tag{9}$$

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:

$$h^{[3]}(W, r) = Wr. \tag{10}$$

The key procedures are $h^{[1]}$ and $h^{[2]}$ essentially perform sorting, portfolio formation, and weight normalization. $h^{[1]} : \mathbb{R}^{M} \to \{-1, 0, 1\}^{M}$ is the univariate security sorting function, which indicates the memberships of individual firms in long and short portfolios. When the argument is a matrix, it broadcasts to all rows. Regarding $h^{[2]} : \{-1, 0, 1\}^{M} \times \mathbb{R}^{M} \to \mathbb{R}^{M}$, it defines how we combine the sorting results of deep characteristics to calculate long-short portfolio weights. Suppose the arguments of $h^{[2]}$ are (u, v). Here, u is produced by $h^{[1]}$, whereas v provides the market equity values. Similarly, when u is a matrix of P rows, it performs in a row-by-row style with the same v.

3.3 A Multi-Layer Architecture

The function H maps the lag predictors to the portfolio long-short weights,

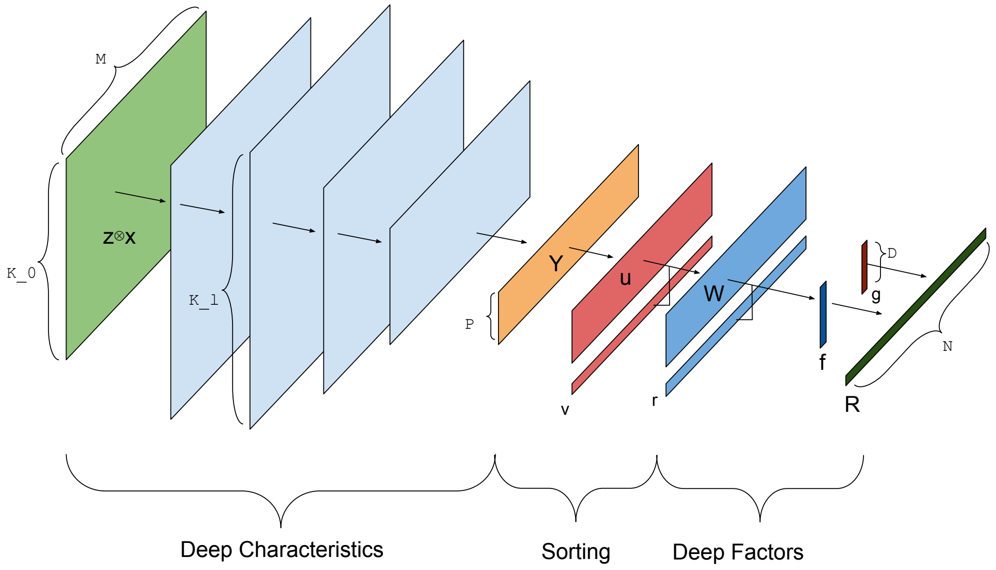
$$W_{t-1} = H\Big(z_{t-1} \otimes x_{t-1}\Big),$$

is essentially a composite given by $H(z \otimes x) = h^{[2]} \circ h^{[1]} \circ F^{[L]} \circ \cdots \circ F^{[1]}(z \otimes x)$. This multi-layer structure is the key idea of interpreting the security sorting as an activation function within a deep learner.

Fixing L, $\{K_l\}_{l=1}^{L}$, and τ, which are architecture parameters, our loss function is the mean

Figure 5: Deep Learning Network Architecture

This figures provides a visualization of deep learning architecture. The firm's characteristics z, macro predictors x, and their interactions are transformed to deep characteristics Y via the deep network. Then, we sort Y 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 asset return R.



squared prediction error regularized by mean squared pricing error

$$\mathcal{L}(A,b,\beta,\gamma) := \frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\left(R_{i,t}-\widehat{R}_{i,t}\right)^2 \tag{11}$$

where $\widehat{R}_{i,t} = \beta_i^{\intercal} f_t + \gamma_i^{\intercal} g_t$ and $\beta = [\beta_1, \beta_2, ..., \beta_N]^{\intercal}$, $\gamma = [\gamma_1, \gamma_2, ..., \gamma_N]^{\intercal}$. To train the deep network is then equivalent to obtaining a joint estimation of $(A, b) := \left\{A^{[l]}, b^{[l]}\right\}_{l=1}^{L}$ and (β, γ). The corresponding estimates are

$$(\hat{A},\hat{b},\hat{\beta},\hat{\gamma}) = \arg\min\left\{\frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\left(R_{i,t}-\widehat{R}_{i,t}\right)^2\right\}.$$

Empirically, we set layer size K_l for the l-th layer as 2^{8-l} and the number of layers $1 \le L \le 5$. For example, a four-layer network is $128 - 64 - 32 - 16$. We summarize the above deep learning framework in Table (1) and Figure (5). Optimization details are provided in Appendix C.

Table 1: Deep-Learning Factor Alpha Architecture

This table provides a summary of the output data dimension, operation, and the parameters needed to be trained for each layer. The deep learning network feeds forward from the bottom to the top. The initial input is firm characteristics, macro predictors, and their interactions. For each layer, the network takes the output from the immediate lower layer as its input as well as the additional input, if exists. Here, $F^{[l]}(Z) := F(A^{[l]}Z + b^{[l]})$.

	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	
Long-short portfolio weight	$P \times M$	W	v	$h^{[2]}(u, v)$	
Univariate Sorting	$P \times M$	u		$h^{[1]}(Y)$	
Deep Characteristics	$K_L \times M$	Y		$F^{[L]}\left(Z^{[L-1]}\right)$	$(A^{[L]}, b^{[L]})$
	$\vdots$	$\vdots$		$\vdots$	$\vdots$
	$K_l \times M$	$Z^{[l]}$		$F^{[l]}\left(Z^{[l-1]}\right)$	$(A^{[l]}, b^{[l]})$
	$\vdots$	$\vdots$		$\vdots$	$\vdots$
Firm Characteristics and Macro Predictors	$K_0 \times M$	$Z^{[0]}$	z, x	$Z^{[0]} = z \otimes x$	

4 Empirical Results

Section 4.1 describes the data. Section 4.2 provides the time series forecasting performance of our deep factor model. Section 4.3 shows the results of how the deep factor model dissects multiple anomalies and its out-of-sample asset pricing performance.

4.1 Data

A critical distinction in the application of deep learning in empirical asset pricing are the right-hand-side factors are constructed by individual stock returns, whereas the loss function is evaluated by a set of left-hand-side test portfolios. The train-validation-test sample is from January 1975 to December 2017. We follow the Fama-French factor in the construction for individual stock filtering but only use the largest 3,000 firms for lag market equity.[3] For the predictors, we use 15 firm characteristics for z_{t-1} and 10 macroeconomic market-timing predictors for x_{t-1} to train the deep learner.

[3]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 for more than one year. We exclude observations with negative book equity or negative lag market equity.

The 15 firm characteristics follow Green et al. (2017) who study the cross-sectional return prediction using firm characteristics. The chosen firm characteristics include all main categories: market equity, market beta, book-to-market ratio, dividend yield, earning-price ratio, asset growth, leverage, operating profitability, return on equity, illiquidity, bid-ask spread, idiosyncratic volatility, return volatility, 1-month momentum, and 12-month momentum.

The 10 macroeconomic predictors are from Welch and Goyal (2007), who study the time-series return prediction of S&P 500 using market-timing predictors. The chosen macro predictors are the treasury-bill rate, inflation, long-term yield, term spread, default yield spread, net equity expansion, and aggregate characteristics for S&P 500, such as stock variance, book-to-market ratio, earning-price ratio, and dividend yield.

For the test portfolios, we train the deep learner with three different datasets: 5×5 size and book-to-market sorted portfolios, 49 industry portfolios, and the factor zoo of Feng et al. (2019) with 147 tradable factors. Given the loss function involved with the test portfolios, the trained deep factors and the deep learning architectures could be entirely different. Our three cases show the forecasting power of deep factors in each scenario: The sorted portfolios have a stable factor loading, whereas the tradable factors have the unsolved alphas.

4.2 Out-of-Sample Forecasting

We use the below measures to report the empirical results. We use the cross-sectional pricing errors to calculate the RMSE and follow Campbell and Thompson (2007) to build a similar out-of-sample R^2 for the pricing errors. RMSE = $\sqrt{\frac{1}{N}\sum_{i=1}^{N}\alpha_i^2}$ is the standard deviation for the pricing error. $R^2 = 1 - \text{RMSE}_M^2/\text{RMSE}_{Avg}^2$ is the relative performance of the Model (M) versus the historical average (Avg). For the empirical evidence of out-of-sample forecast, the historical average is the strongest benchmark.

We use the sample 1975-2002 to train models with different layers (1-5) and a different number of factors (1-5). Then, we use the sample 2003-2010 as a validation sample to select the best model. The estimated factor loadings, $\widehat{\beta}$ and $\widehat{\gamma}$, are kept fixed when selecting the models from the validation sample. Then, we re-train the selected model with the additional sample 2003-2010 to get the updated $\widehat{\beta}$ and $\widehat{\gamma}$. The goal is to see the performance of this model in the test sample 2011-2017.

For conditional models, we also try a different number of conditions (1-8) and use the validation for model selection. Notice that, for the model selection of validation sample, if the unconditional model frequently outperforms the conditional ones, then we use the unconditional model.

The result for out-of-sample evaluation is in Table 2. We have six sets of results: different predictors versus different test portfolios. We have a small set of 15 characteristics, as well as a large set of characteristics, 10 macro predictors, and their interactions. We use three different test portfolios: 5$\times$5 size and book-to-market sorted portfolios, industry portfolios, and the factor zoo. For each set of results, we compare three benchmark models g_t (CAPM, Fama-French three factors, and FF4 with the Carhart Momentum) by adding our f_t in unconditional and conditional versions.

Our main conclusion is that a large model (characteristics $\otimes$ macro predictors) is better than the small model (characteristics). For the right panel of Table 2, we find consistent improvement by adding the deep factors in unconditional and conditional versions. The in-sample and validation improvement are significant, whereas the test sample improvement is marginal but mostly positive. We can find the same trend in the left panel of Table 2. The worst case for model fitness is the case of the factor zoo, where adding deep factors finds overfitting. In line of Lewellen et al. (2010), we also find the best case for model fitness is for 5$\times$5 size and book-to-market sorted portfolios, which has a strong factor structure.

We also find that when controlling for FF3 and FF4, there is no consistent positive improvement for the conditional model in either the validation or test samples. This conclusion is in line with Cochrane (2009), such that a conditional model is equivalent to adding more factors. A multi-factor model is sufficient to capture the variation of a conditional model. However, simply adding deep factors into CAPM, the out-of-sample R^2 increases 12%, on average, if we consider combining the validation and test samples, 2003-2017. A third conclusion is related to the model selection; namely, a shallow model frequently outperforms a deep model (with more layers, factors, or conditions), and this finding is consistent with the finding of Gu et al. (2018). Our selected models are mostly trained with two or three layers and two or three factors.

4.3 Dissecting Anomalies

We assess the pricing performance of our deep factor model for the existing anomalies. We fix the deep learners trained in Table 2 and evaluate those 147 tradable factors in Feng et al. (2019). The goal is to compare the evaluation performance for anomalies by different factor models. We examine the out-of-sample statistical significance for different alpha estimates of 147 factors in the test period 2011-2017.

For i-th anomaly, we count its alpha as significant with respect to a specific model if the t-statistic of $\{R_{i,t} - \widehat{R}_{i,t}\}_{t=1}^{T}$ has absolute value greater than 1.96, namely,

$$\left|\frac{\sqrt{T}\bar{e}_i}{\hat{\sigma}(e_i)}\right| > 1.96$$

where $e_{i,t} := R_{i,t} - \widehat{R}_{i,t}$, $\bar{e}_i = \frac{1}{T}\sum_{t=1}^{T} e_{i,t}$ and $\hat{\sigma}(e_i) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(e_{i,t} - \bar{e}_i)^2}$.

For these 147 anomalies in Table 3, 39, 26, and 23 of them are tested with significant alpha to CAPM, FF3, and FF4 in the test sample anomalies. By controlling for the additional deep factors, we find a slight decrease for 21, 22 and 24 significances using the small model (characteristics). However, the big model (characteristics $\otimes$ macro predictors) does not outperform the small model, and the conditional model even finds more significances. One possible reason is that a more complex model causes higher estimation errors when using such a short history, as in the test sample.

Table 4 follows Table 2 of Fama and French (2016) to evaluate four different sets of anomaly-sorted portfolios correspondingly: 5×5 portfolios on size and accruals, 5×5 portfolios on size and market beta, 5×5 portfolios on size and variance, and 5×5 portfolios on size and residual variance. We fix the deep learners trained in Table 2 and evaluate these "hold-out" portfolios. The goal is to evaluate the model out-of-sample fit using the average pricing errors, because we know factor models have various performance to different test assets. Therefore, both the validation and test version are out-of-sample evaluations, with the difference being that the latter models are updated with a few more years of data.

Table 4 repeats the evaluation using each 5×5 sorted portfolio above. For the upper panel, the findings are consistently positive: Adding deep factors helps increase the out-of-sample R^2 in both validation and test samples. Many R^2 are negative for Size-Beta, Size-Variance, and Size-Residual

Variance sorted portfolios because the factor models underperform the historical average in this test period. The Fama-French factors work well for Size-Accrual sorted portfolios, where adding the deep factors leads to only a marginal improvement. Simply adding deep factors into CAPM helps beat the historical average benchmark and make the R^2 mostly positive.

For the lower panel with large models (characteristics $\otimes$ macro predictors), the findings are mixed. For using Size-Book-to-Market sorted portfolios and industry portfolios, we find adding deep factors mostly helps increase the out-of-sample R^2 in both validation and test samples, but the improvements are not as substantial as in the upper panel with small models (characteristics). If we use the factor zoo as test portfolios, the comparisons are mixed. On average, simply adding deep factors to CAPM can increase the relative R^2 by 30% on average.

5 Discussion

Our goal is to provide a deep learning framework for empirical asset pricing based on characteristics. We offer a deep learner representation for the Fama-French style models: building from lag firm characteristics, feeding forward the nonlinear predictor interactions via the security sorting, and forming long-short tradable factors. Given a benchmark model g_t, we generate additional deep factors f_t using lag firm characteristics and macroeconomic predictors, with an asset pricing objective of "minimizing the pricing errors". Our method is the first one that unifies all procedures of the characteristics-based asset pricing models with a clear optimization objective.

Our procedure builds on the connection between security sorting on lag characteristics to a quantile activation function within a unified deep learner. The multi-layer multi-neuron deep learner helps search for the best nonlinear predictor interactions for security sorting by controlling for a benchmark model. Our method is comparable to different versions of principal component analysis in asset pricing but has some advantages over them, such as allowing for an imbalanced dataset and a natural out-of-sample research design.

The recent computation breakthroughs in artificial intelligence have enabled numerous potential automatic data-driven applications in many areas, including finance and econometrics. Asset pricing has a low signal-to-noise ratio and a steady pattern of non-stationarity, which implements deep learning challenging with the short history of financial data and unbalanced data panel. In our experience, the key is to adopt these data-driven methods within those workhorse empirical models. The cross-sectional asset pricing model has a relatively high signal-to-noise ratio and stable factor loadings. The out-of-sample improvement over Fama-French factors is significant, but we believe our deep learner representation provides useful insights to both financial economists and computer scientists.

Finally, training a conditional linear model of deep learning is one convenient representation of exploiting nonlinear interaction in an interpretable linear framework. However, the stochastic approximation is a fast and flexible way to train a complicated model, but the fitted model can be completely different using different random seeds. Feng et al. (2019) use multiple random seeds to form a robust inference for testing a factor. To ease interpretation of the deep factor, we provide a relatively reliable deep learner via ensemble learning, where the details are listed in Appendix C. Future research is required to appreciate the applications of deep learning in finance fully.

References

Bianchi, D., M. Büchner, and A. Tamoni (2018). Bond risk premia with machine learning. Technical report, University of Warwick.

Campbell, J. Y. and S. B. Thompson (2007). Predicting excess stock returns out of sample: Can anything beat the historical average? *The Review of Financial Studies 21*(4), 1509–1531.

Cochrane, J. H. (2009). *Asset pricing: Revised edition*. Princeton university press.

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 (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.

Feng, G., S. Giglio, and D. Xiu (2019). Taming the factor zoo: A test of new factors. Technical report, National Bureau of Economic Research.

Feng, G., J. He, and N. Polson (2018). Deep learning for predicting asset returns. 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.

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.

Han, Y., A. He, D. Rapach, and G. Zhou (2018). What firm characteristics drive us stock returns? Technical report, Washington University in St. Louis.

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.

Huang, D., J. Li, and G. Zhou (2018). Shrinking factor dimension: A reduced-rank approach. Technical report, Washington University in St. Louis.

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.

Kim, S., R. A. Korajczyk, and A. Neuhierl (2018). Arbitrage portfolios in large panels. Technical report, Georgia Institute of Technology.

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

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.

Van Binsbergen, J. H. and R. S. Koijen (2010). Predictive regressions: A present-value approach. *The Journal of Finance 65*(4), 1439–1471.

Welch, I. and A. Goyal (2007). A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies 21*(4), 1455–1508.

Table 2: Out-of-Sample Forecast

This table shows the out-of-sample R^2 for different models that are trained with different lag predictors and test portfolios. The in-sample (INS) period is 1975-2002, the validation (VAL) sample is 2003-2010, and the test sample is 2011-2017. The model is selected using the validation sample and is updated with the validation sample when forecasting the test sample. The out-of-sample R^2 for VLD and Test are the relative performance over the historical average.

	Characteristics				Characteristics $\otimes$ Macro Predictors			
Test Portfolio	Model	INS R^2	VLD R^2	Test R^2	Model	INS R^2	VLD R^2	Test R^2
FF 5×5	CAPM	0.812	-1.817	0.380	CAPM	0.812	-1.817	0.380
	CAPM+DL	0.837	-0.965	0.185	CAPM+DL	0.809	-1.370	0.421
	CAPM + DL + Cond	0.951	-0.640	0.756	CAPM + DL + Cond	0.948	-0.236	0.336
	FF3	0.966	0.089	0.76	FF3	0.966	0.089	0.76
	FF3 + DL	0.966	0.166	0.766	FF3 + DL	0.966	0.204	0.733
	FF3+ DL + Cond	0.966	0.166	0.766	FF3+ DL + Cond	0.966	0.204	0.733
	FF4	0.974	0.086	0.783	FF4	0.974	0.086	0.783
	FF4 + DL	0.973	0.113	0.765	FF4 + DL	0.974	0.147	0.783
	FF4 + DL + Cond	0.957	0.173	0.227	FF4 + DL + Cond	0.974	0.147	0.783
Industry	CAPM	0.873	0.155	0.132	CAPM	0.873	0.155	0.132
	CAPM+DL	0.863	0.237	0.092	CAPM+DL	0.873	0.203	0.135
	CAPM + DL + Cond	0.863	0.237	0.092	CAPM + DL + Cond	0.873	0.203	0.135
	FF3	0.834	0.301	0.222	FF3	0.834	0.301	0.222
	FF3 + DL	0.830	0.352	0.226	FF3 + DL	0.838	0.395	0.231
	FF3+ DL + Cond	0.830	0.352	0.226	FF3+ DL + Cond	0.838	0.395	0.231
	FF4	0.844	0.299	0.204	FF4	0.844	0.299	0.204
	FF4 + DL	0.839	0.337	0.169	FF4 + DL	0.847	0.342	0.213
	FF4 + DL + Cond	0.839	0.337	0.169	FF4 + DL + Cond	0.847	0.342	0.213
Factor Zoo	CAPM	-0.617	0.408	-1.073	CAPM	-0.617	0.408	-1.073
	CAPM+DL	-1.028	0.517	-0.815	CAPM+DL	-0.678	0.453	-1.187
	CAPM + DL + Cond	-1.028	0.517	-0.815	CAPM + DL + Cond	-0.678	0.453	-1.187
	FF3	0.057	0.682	-0.411	FF3	0.057	0.682	-0.411
	FF3 + DL	0.052	0.695	-0.468	FF3 + DL	0.041	0.705	-0.398
	FF3+ DL + Cond	0.052	0.695	-0.468	FF3+ DL + Cond	0.041	0.705	-0.398
	FF4	0.402	0.683	-0.260	FF4	0.402	0.683	-0.260
	FF4 + DL	0.413	0.700	-0.323	FF4 + DL	0.402	0.700	-0.309
	FF4 + DL + Cond	0.413	0.700	-0.323	FF4 + DL + Cond	0.402	0.700	-0.309

Table 3: Dissecting Anomalies

This table provides the number of significant alphas for those 147 factors in Feng, Giglio, and Xiu (2019). The results are provided for different models that are trained with different lag predictors and test portfolios. The in-sample (INS) period is 1975-2002, the validation (VAL) sample is 2003-2010, and the test sample is 2011-2017. The model is selected using the validation sample and is updated with the validation sample when forecasting the test sample.

	Characteristics				Characteristics $\otimes$ Macro Predictors			
Test Portfolio	Model	INS Sig.	VLD Sig.	Test Sig.	Model	INS Sig.	VLD Sig.	Test Sig.
FF 5×5	CAPM	82	7	39	CAPM	82	7	39
	CAPM+DL	78	4	21	CAPM+DL	81	6	33
	CAPM + DL + Cond	17	12	18	CAPM + DL + Cond	26	18	25
	FF3	73	8	26	FF3	73	8	26
	FF3 + DL	74	10	22	FF3 + DL	71	7	26
	FF3+ DL + Cond	74	10	22	FF3+ DL + Cond	71	7	26
	FF4	54	7	23	FF4	54	7	23
	FF4 + DL	54	7	24	FF4 + DL	55	6	26
	FF4 + DL + Cond	104	41	47	FF4 + DL + Cond	55	6	26
Industry	CAPM	82	7	39	CAPM	82	7	39
	CAPM+DL	87	4	33	CAPM+DL	82	6	38
	CAPM + DL + Cond	87	4	33	CAPM + DL + Cond	82	6	38
	FF3	73	8	26	FF3	73	8	26
	FF3 + DL	73	8	23	FF3 + DL	69	7	28
	FF3+ DL + Cond	73	8	23	FF3+ DL + Cond	69	7	28
	FF4	54	7	23	FF4	54	7	23
	FF4 + DL	51	7	25	FF4 + DL	53	4	22
	FF4 + DL + Cond	51	7	25	FF4 + DL + Cond	53	4	22
Factor Zoo	CAPM	82	7	39	CAPM	82	7	39
	CAPM+DL	93	3	28	CAPM+DL	85	5	39
	CAPM + DL + Cond	93	3	28	CAPM + DL + Cond	85	5	39
	FF3	73	8	26	FF3	73	8	26
	FF3 + DL	73	7	29	FF3 + DL	72	6	23
	FF3+ DL + Cond	73	7	29	FF3+ DL + Cond	72	6	23
	FF4	54	7	23	FF4	54	7	23
	FF4 + DL	54	6	23	FF4 + DL	56	6	29
	FF4 + DL + Cond	54	6	23	FF4 + DL + Cond	56	6	29

Table 4: Dissecting Sorted Portfolios

This table provides the model out-of-sample evaluation for different sorted portfolios used in Fama and French (2016), which are unseen by the model training. The rest of the empirical research design is the same as in Table 2.

		Size-Beta		Size-Accrual		Size-Variance		Size-Residual Variance	
Test Portfolio	Model	VLD R^2	Test R^2	VLD R^2	Test R^2	VLD R^2	Test R^2	VLD R^2	Test R^2
Characteristics									
FF 5×5	CAPM	-0.998	-0.931	-1.246	0.044	0.004	-2.713	-0.082	-2.566
	CAPM+DL	-0.388	-0.490	-0.677	0.245	0.225	-2.016	0.146	-1.817
	FF3	0.666	-0.277	0.485	0.632	0.802	-1.423	0.764	-1.255
	FF3 + DL	0.732	-0.236	0.586	0.628	0.827	-1.274	0.793	-1.117
	FF4	0.698	-0.083	0.519	0.656	0.794	-0.977	0.757	-0.858
	FF4 + DL	0.711	-0.031	0.581	0.653	0.794	-0.948	0.756	-0.805
Industry	CAPM	-0.998	-0.931	-1.246	0.044	0.004	-2.713	-0.082	-2.566
	CAPM+DL	-0.907	-0.751	-1.366	0.206	0.163	-2.350	0.104	-2.182
	FF3	0.666	-0.277	0.485	0.632	0.802	-1.423	0.764	-1.255
	FF3 + DL	0.687	-0.209	0.531	0.639	0.803	-1.273	0.764	-1.120
	FF4	0.698	-0.083	0.519	0.656	0.794	-0.977	0.757	-0.858
	FF4 + DL	0.615	-0.029	0.430	0.647	0.744	-0.810	0.682	-0.689
Factor Zoo	CAPM	-0.344	-0.316	-0.096	0.376	0.223	-1.769	0.172	-1.556
	CAPM+DL	-0.373	-0.020	-0.245	0.547	0.201	-1.121	0.159	-0.902
	FF3	0.763	0.140	0.737	0.746	0.864	-0.826	0.836	-0.629
	FF3 + DL	0.774	0.132	0.743	0.747	0.871	-0.935	0.848	-0.734
	FF4	0.792	0.277	0.755	0.763	0.861	-0.481	0.834	-0.334
	FF4 + DL	0.817	0.267	0.764	0.756	0.892	-0.536	0.872	-0.379
Characteristics $\otimes$ Macro Predictors									
FF 5×5	CAPM	-0.998	-0.931	-1.246	0.044	0.004	-2.713	-0.082	-2.566
	CAPM+DL	-0.584	-0.899	-0.870	0.111	0.188	-2.576	0.117	-2.424
	FF3	0.666	-0.277	0.485	0.632	0.802	-1.423	0.764	-1.255
	FF3 + DL	0.722	-0.263	0.470	0.631	0.843	-1.469	0.816	-1.335
	FF4	0.698	-0.083	0.519	0.656	0.794	-0.977	0.757	-0.858
	FF4 + DL	0.748	-0.100	0.545	0.649	0.822	-1.056	0.799	-0.941
Industry	CAPM	-0.998	-0.931	-1.246	0.044	0.004	-2.713	-0.082	-2.566
	CAPM+DL	-1.012	-0.915	-1.334	0.160	0.031	-2.837	-0.051	-2.657
	FF3	0.666	-0.277	0.485	0.632	0.802	-1.423	0.764	-1.255
	FF3 + DL	0.726	-0.279	0.541	0.609	0.845	-1.461	0.830	-1.303
	FF4	0.698	-0.083	0.519	0.656	0.794	-0.977	0.757	-0.858
	FF4 + DL	0.726	-0.038	0.534	0.650	0.856	-0.902	0.836	-0.768
Factor Zoo	CAPM	-0.344	-0.316	-0.096	0.376	0.223	-1.769	0.172	-1.556
	CAPM+DL	-0.553	-0.393	-0.247	0.284	0.166	-1.990	0.123	-1.772
	FF3	0.763	0.140	0.737	0.746	0.864	-0.826	0.836	-0.629
	FF3 + DL	0.720	0.171	0.714	0.733	0.858	-0.840	0.837	-0.634
	FF4	0.792	0.277	0.755	0.763	0.861	-0.481	0.834	-0.334
	FF4 + DL	0.808	0.243	0.750	0.758	0.909	-0.596	0.901	-0.445

Appendix A Conditional Deep Learning

This section includes the details of a conditional factor model that is introduced in section 2.3. The architecture of the conditional factor model is as follows:

$$
\begin{aligned}
R_{i,t} &= \beta_i^\intercal f_t^{ReLU} = \sum_{q=1}^{Q} \left(\beta_i^{(q)\intercal} f_t + \gamma_i^{(q)\intercal} g_t\right) * \mathbb{1}_{S_q} \\
f^{ReLU} &:= [f^{[L'],+}; f^{[L'],-}], \\
f^{[l],+} &= ReLU\left(\tilde{A}^{[l]} f^{[l-1],+}\right),\ f^{[l],-} = ReLU\left(\tilde{A}^{[l]} f^{[l-1],-}\right),\ \text{for } l = 1, 2, 3, ..., L \\
f^{[0],+} &:= [f; g],\ f^{[0],-} := [-f; -g],
\end{aligned}
$$

where S_q represents a state.

Because the ReLU function is piecewise 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 factors f and g.

For illustration, consider a one-layer ReLU network in which the output f^{ReLU} is of dimension $2 * P$. The $2 * P$ outputs are, with $\tilde{A}^{[1]} = [\tilde{A}_1, \tilde{A}_2, ..., \tilde{A}_P]^\intercal$,

$$
\begin{gathered}
ReLU(\tilde{A}_1^\intercal f^{[0],+}), ReLU(-\tilde{A}_1^\intercal f^{[0],+}), \\
ReLU(\tilde{A}_2^\intercal f^{[0],+}), ReLU(-\tilde{A}_2^\intercal f^{[0],+}), \\
..., \\
ReLU(\tilde{A}_P^\intercal f^{[0],+}), ReLU(-\tilde{A}_P^\intercal f^{[0],+})
\end{gathered}
$$

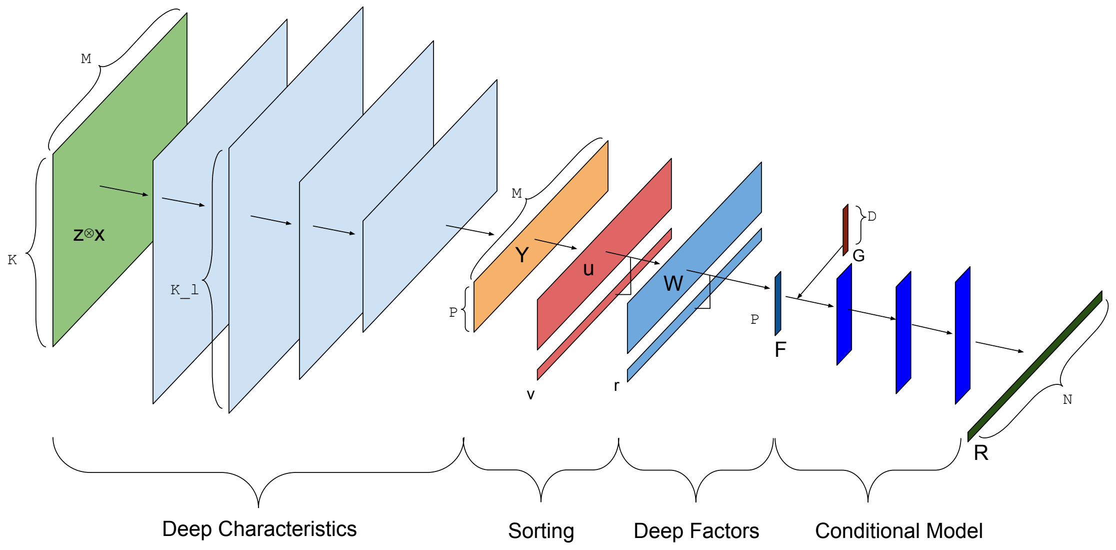
We have P pairs of ReLU units here. For each row, say the p-th, one of the two units outputs the absolute value of $\tilde{A}_p^\intercal [f; g]$ and the other outputs 0. We see these vectors, $\tilde{A}_1, \tilde{A}_2, ..., \tilde{A}_P$, are the basic components to define factor coefficients. Note

$$
\begin{aligned}
\beta^\intercal f_t^{ReLU} &:= \beta_+^\intercal f_t^{[1],+} + \beta_-^\intercal f_t^{[1],-} \\
&= \sum_{p=1}^{P} \beta_{p,+} ReLU(\tilde{A}_p^\intercal f_t^{[0],+}) + \beta_{p,-} ReLU(-\tilde{A}_p^\intercal f_t^{[0],+}) \\
&= \left(\sum_{p=1}^{P} A_p^+ \mathbb{1}\{\tilde{A}_p^\intercal [f_t; g_t] > 0\} + A_p^- \mathbb{1}\{\tilde{A}_p^\intercal [f_t; g_t] < 0\} \right) [f_t; g_t] \\
&:= \sum_{q=1}^{Q} \left(\beta^{(q)\intercal} f_t + \gamma^{(q)\intercal} g_t \right) * \mathbb{1}\{S_q = 1\},
\end{aligned}
$$

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 $\tilde{A}_1, \tilde{A}_2, ..., \tilde{A}_P$ 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 regions by the P hyperplanes and each regions has a different factor coefficient for $[f; g]$. Figure 6 provides an additional layer for conditional model over Figure 5, where dark blue bars represents ReLU layers.

Figure 6: Conditional model network architecutre

This figure provides an illustration of the network architecture for conditional model. After deep factor construction, f and g go through additional layers of ReLU activations to price the test portfolio returns R.



Appendix B Security Sorting

This section defines the functions $h^{[1]}$ and $h^{[2]}$ mentioned in section 3.2. $h^{[1]}$ operates as a univariate security sorting and $h^{[2]}$ gives the long-short portfolio weights.

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, that is, a row of Y, or the market equity value v. We define $h^{[1]}(y)$ as

$$h^{[1]}(y) = \begin{bmatrix} \mathbb{1}\{y_1 \geq q_{1-\tau}(y)\} \\ \vdots \\ \mathbb{1}\{y_j \geq q_{1-\tau}(y)\} \\ \vdots \\ \mathbb{1}\{y_M \geq q_{1-\tau}(y)\} \end{bmatrix} + \begin{bmatrix} \mathbb{1}\{y_1 \geq q_{1-\tau}(y)\} \\ \vdots \\ \mathbb{1}\{y_j \geq q_{1-\tau}(y)\} \\ \vdots \\ \mathbb{1}\{y_M \geq q_{1-\tau}(y)\} \end{bmatrix} - \mathbb{1}_M, \tag{12}$$

where $\mathbb{1}$ is indicator function and $\mathbb{1}_M$ is a $M \times 1$ vector of ones. $q_{1-\tau}$ and q_τ are lower $(1-\tau)$ and τ quantiles, respectively, with $0.5 \leq \tau < 1$. For example, we choose $\tau = 0.8$ for deep characteristics Y in practice.

Clearly, each coordinate of $h^{[1]}(y)$ takes value from $\{-1, 0, 1\}$, resulting from sorting $y_1, y_2, ...y_M$. In other words, assume $\{(1), (2), ..., (M)\}$ is a permutation of $\{1, 2, ..., M\}$ and $y_{(1)} \leq y_{(2)} \leq ... \leq y_{(M)}$; then

$$\left[h^{[1]}(y)\right]_{(j)} = \begin{cases} -1 & \text{if } \frac{j}{M} < 1-\tau \\ 0 & \text{if } 1-\tau \leq \frac{j}{M} < \tau \\ 1 & \text{if } \frac{j}{M} \geq \tau. \end{cases} \tag{13}$$

To better understand the procedure of $h^{[1]}$, imagine dividing the firm universe into three parts using cut-off values $q_{1-\tau}(y)$ and $q_\tau(y)$. This division is with respect to some deep characteristic y, and each coordinate corresponds to a specific firm. The proportion of firms in each part is $1-\tau$, $2\tau - 1$, and $1-\tau$, respectively. We set $u = 1$ for top firms, $u = 0$ for middle firms, and $u = -1$ for bottom firms.

Finally, $h^{[2]}$ calculates portfolio weights given the market equity v and the result from $h^{[1]}$, u:

$$h^{[2]}(u,v) = \left[\frac{u_+ \odot v}{(u_+ \odot v)'\mathbb{1}_M}\right] - \left[\frac{u_- \odot v}{(u_- \odot v)'\mathbb{1}_M}\right], \tag{14}$$

where $u_+ := \max\{u, 0\}$, $u_- := \max\{-u, 0\}$, and "$\odot$" denotes the element-wise production.

Appendix C Optimization Details

This section shows how we minimize our loss function to train the deep learner. The techniques includes stochastic gradient descent (SGD), dropout, and ensemble learning.

Although being highly nonlinear and non-convex, the structure of the deep learner makes its loss function differentiable with respect to its parameters. The first-order derivative information is directly available by carefully applying the backward-chain rule. TensorFlow library performs automatic derivative calculation for practitioners, allowing us to train the model using SGD.[4] Let the superscript (t) denote the t-th iterate. SGD updates the parameters by

$$\begin{bmatrix} \hat{A}^{(t+1)} \\ \hat{b}^{(t+1)} \\ \hat{\beta}^{(t+1)} \\ \hat{\gamma}^{(t+1)} \end{bmatrix} \longleftarrow \begin{bmatrix} \hat{A}^{(t)} \\ \hat{b}^{(t)} \\ \hat{\beta}^{(t)} \\ \hat{\gamma}^{(t)} \end{bmatrix} - \eta^{(t+1)} \nabla \mathcal{L}^{(t)} \tag{15}$$

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

$$\mathcal{L}^{(t)}(A, b, \beta, \gamma) = \frac{1}{N|\mathcal{B}|} \sum_{t \in \mathcal{B}} \sum_{i=1}^{N} \left(R_{i,t} - \widehat{R}_{i,t}\right)^2, \tag{16}$$

where $|\mathcal{B}| < T$, and in practice we set $|\mathcal{B}| = 120$; namely, we use a batch of 120 months for training. We set the number of epochs (roughly the number of times SGD explores the whole training set) to be 200, because the loss no longer decreases significantly.

Dropout is used to improve the estimation. Here, the input space of $Z^{[l-1]}$ is replaced by $D \odot Z^{[l-1]}$ for $l = 1, 2, ..., L$, where $D \sim \text{Bernoulli}(p)$ is a matrix of randomly assigned Bernoulli

[4]See Robbins and Monro (1951), Kiefer and Wolfowitz (1952).

variables. This procedure acts as a ridge regularization.[5] As opposed to sparsity, the network architecture averages small models using dropout.

To stabilize the training of the factor model, we further consider using an ensemble of Ω predictions, $\widehat{R}^{(1)}, \widehat{R}^{(2)}, ..., \widehat{R}^{(\Omega)}$, based on the same f and g, but each with a different set of coefficients, $(\beta^{(1)}, \gamma^{(1)}), (\beta^{(2)}, \gamma^{(2)}), ..., (\beta^{(\Omega)}, \gamma^{(\Omega)})$. The final prediction is calculated by averaging over all ensembles. For $t = 1, 2..., T$ and $i = 1, 2, ..., N$,

$$\widehat{R}_{i,t} = \frac{1}{\Omega} \sum_{\omega=1}^{\Omega} \widehat{R}_{i,t}^{(\omega)}.$$

In practice, we use $\Omega = 100$. Figure 7 shows how our deep network extends with this ensemble of coefficients and predictions. This ensemble estimation is equivalent to adding an additional "ensemble prediction" layer in our original architecture.

Figure 7: Ensemble Learning for Estimating Factor Loadings

This figure provides an illustration of ensemble learning, which stabilizes the trained factor loadings by fitting the linear factor model 100 times.

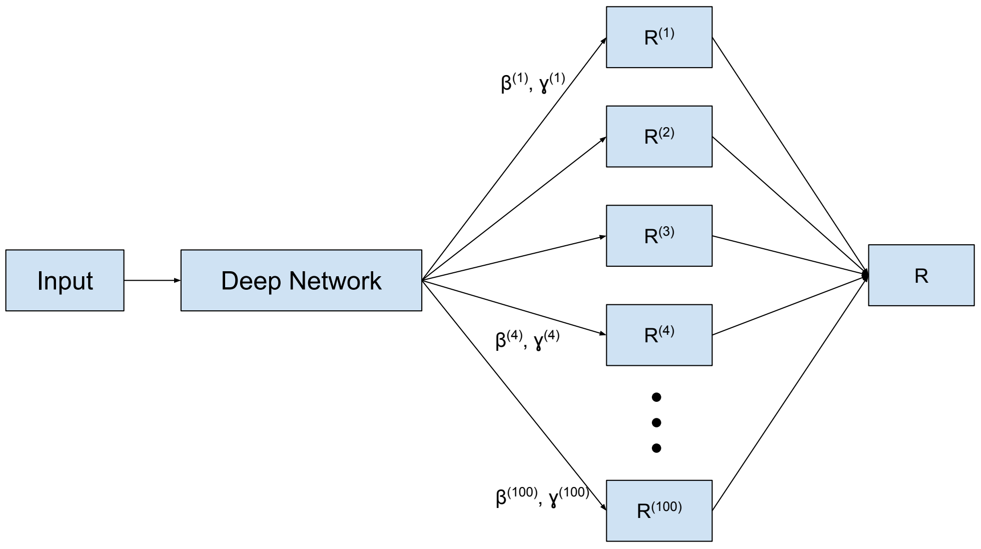
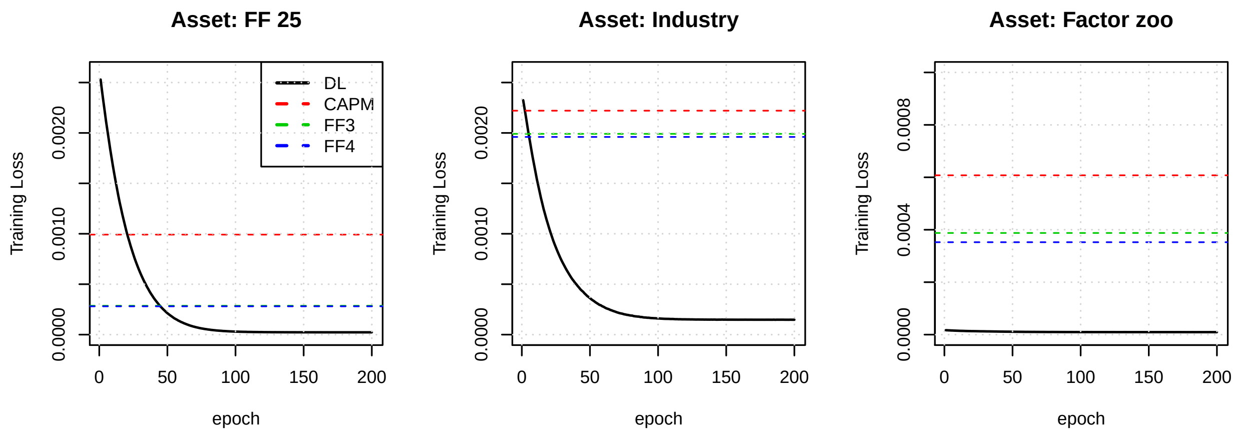


Figure 8 gives an example of the loss function decreasing during training. We use firm characteristics and macro predictors as the input, controlling for Fama-French three factors, to price three

[5]See Hinton and Salakhutdinov (2006), Polson and Sokolov (2017).

Figure 8: Loss Function of the Training Data vs. # Epochs

This figure provides an example of the loss function with respect to the training data decreasing as SGD iterates. We use firm characteristics and macro predictors as the inputs and Fama French three factors as our benchmark g_t.



kinds of test portfolios. The red dashed line is the loss of CAPM model; the green one is the Fama-French three-factors model; the blue one is FF3 + Momentum. They are all estimated using least squares, and thus the corresponding losses stay the same. In all three cases (test portfolios are 5×5 size and book-to-market sorted portfolios, 49 industry portfolios, and the factor zoo, respectively), the loss functions with respect to the training data decrease as SGD goes on and finally converge at levels below the benchmark models (in the first plot, FF3 and FF4 almost have the same loss; in the third plot, the deep learning model beats all three benchmarks right after the first epoch). Therefore, the deep learning models always give better in-sample fits than those benchmarks.