Deep Learning for Asset Pricing Using Characteristics-Sorted Factor Models

Manish Agarwal, Mehdi Zouiten, Goutam Tulsiyan

Introduction

- We attempt to fit a 'risk model' for equity stocks to explain the returns of the cross-section.
- Unlike prediction problems that focus on forecasting future returns, this study aims to find latent factors that explain the contemporaneous returns of the cross-section.
- This is crucial in finance for risk control and return attribution.
- Traditionally, asset pricing models decompose excess asset returns on factor risk premiums using linear models.

$$\boldsymbol{r}_{i,t} = \beta_{i,t-1}^{T} \boldsymbol{f}_{d,t} + \boldsymbol{\eta}_{i,t}$$

- The presence of non-linear effects suggests that neural networks, with their natural ability to approximate non-linear functions, could be a good candidate for modeling these complexities.
- We use benchmark factors and estimate new deep characteristic factors to expand the explainability of cross-sectional variance, while simultaneously estimating the non-linear betas as a function of input characteristics $\boldsymbol{z}_{i,t-1}$.

$$\boldsymbol{r}_{i,t} = \beta_d(\boldsymbol{z}_{i,t-1})^T \boldsymbol{f}_{d,t} + \beta_b(\boldsymbol{z}_{i,t-1})^T \boldsymbol{f}_{b,t} + \boldsymbol{\epsilon}_{i,t}$$

Related Work

Recent literature highlights the increasing use of deep learning in asset pricing, particularly for nonlinear feature extraction. Studies by [1], [2], and [3] demonstrate deep learning's ability to generate latent factors with greater explanatory power than traditional PCA-based models. This project builds on these approaches by integrating characteristics-sorted factors within a deep learning framework that optimizes economic fitness through latent factor generation.

Dataset and Data Pre-processing

We utilized historical U.S. stock data from FinancialModelingPrep, covering the period from 1990 to 2020. Our methodology involved constructing a universe of the 1,000 most liquid stocks to ensure significant contributions to market movements while excluding stocks with lower liquidity data.

The raw dataset included price, volume, and fundamental data, from which we developed $\sim$ 15 technical and $\sim$ 35 fundamental factors:

- **Returns**: Daily stock returns using were aggregated to compute monthly returns.
- **Technical Factors**: Derived from daily price and volume data like momentum, long-term reversal, and short-term reversal.
- **Fundamental Factors**: Extracted from quarterly filings such as standardized unexpected earnings, dividend yield, and return on equity.
- **Benchmark Returns**: We include Fama-French 3 factors model [4] as factors to improve upon.

We carefully handled missing values with zero imputation and excluded firms lacking fundamental data during those periods. After constructing the data, we applied monthly cross-sectional standardization, normalizing values to the range of [-1, 1] for model stability and feature comparability.

References

[1] Guanhao Feng, Jingyu He, Nicholas Polson, and Jianeng Xu. "Deep Learning in Characteristics-Sorted Factor Models." *Journal of Financial and Quantitative Analysis*, 2023.

[2] Shihao Gu, Bryan Kelly, and Dacheng Xiu. "Autoencoder Asset Pricing Models." *The Journal of Finance*, 2021.

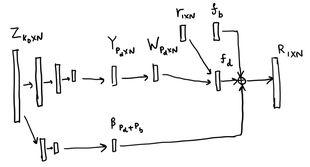
[3] Bryan Kelly, Seth Pruitt, and Yinan Su. "Instrumented Principal Component Analysis." *Journal of Financial Economics*, 2019.

[4] Fama, E. F., & French, K. R. Available at: `https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html`

Methodology

We implement a deep learning framework to enhance cross-sectional asset pricing by generating latent risk factors from these characteristics while minimizing pricing errors.

We generate $P_d = 2$ deep factors $\boldsymbol{f}_{d,t}$, in addition to a $P_b = 3$-factor benchmark $\boldsymbol{f}_{b,t}$, which is Fama-French-3 factor model, to price individual stock returns jointly by estimating return predictor $\hat{r}_{i,t}$ as a linear combination of these factors without an intercept. The fitting error measures the cross-sectional and time-series variation. Here, $H(\cdot)$ is a neural network to find latent characteristics, while $G(\cdot)$ is a neural network to estimate the betas. Both are trained simultaneously.



$$\begin{aligned}
\hat{r}_{i,t} &= \beta_d(\boldsymbol{z}_{i,t-1})^T \boldsymbol{f}_{d,t} + \beta_b(\boldsymbol{z}_{i,t-1})^T \boldsymbol{f}_{b,t} \\
e_{i,t} &= r_{i,t} - \hat{r}_{i,t} \\
\boldsymbol{f}_{d,t} &= \boldsymbol{W}_{t-1} \boldsymbol{r}_t \\
\boldsymbol{W}_{t-1} &= H(\boldsymbol{Z}_{t-1}) \\
\beta(\boldsymbol{z}_{i,t-1}) &= G(\boldsymbol{z}_{i,t-1})
\end{aligned} \tag{1}$$

The loss function is:

$$\mathcal{L}(r, \hat{r}) = \frac{1}{NT} \sum_{t=1}^{T} \sum_{i=1}^{N} (r_{i,t} - \hat{r}_{i,t})^2 + \lambda_1 \frac{1}{T} \sum_{t=1}^{T} \left(\frac{1}{N} \sum_{i=1}^{N} (r_{i,t} - \hat{r}_{i,t}) \right)^2 + \lambda_2 \sum_{l=1}^{L-1} |A_{i,j}^{[l]}| + \lambda_3 \sum_{l=1}^{L-1} B_{i,j}^{[l]2}.$$

- The first term in the loss minimizes the model produced returns against real returns.
- The second term forces average value of cross-section alpha to 0 (as required by a cross-sectional factor).
- The motive of the L_1 loss in the third term is to reduce the size of the characteristic weight matrix to control the effect of outliers.
- The fourth term gives L_2 regularization to the benchmark and deep factor betas.

We use batch gradient descent as the training strategy.

Evaluation Metric

Following [1], we define total R^2 as:

$$\text{Total } R^2 = 1 - \frac{\sum_{i=1}^{N} \sum_{t=1}^{T} (r_{i,t} - \hat{r}_{i,t})^2}{\sum_{i=1}^{N} \sum_{t=1}^{T} (r_{i,t} - \text{MktRF}_t)^2}$$

The numerator represents the sum of squared prediction errors, while the denominator reflects the sum of squared excess returns over market returns, serving as a normalizer. A higher total R^2 indicates better predictive performance of the factor model. We also use cumulative plots of factor returns to qualitatively assess the results.

Parameters chosen

- a deep characteristic neural network with layers [32, 16, 8, 2],
- benchmark beta layers [8, 4, 3],
- and deep beta layers [8, 4, 2].
- There are a total of 3227 parameters, so it is relatively a smaller network.

We used a dropout of 0.25 for the deep characteristic layers, with a batch size of 75, `tanh` activation, the Adam optimizer, and a learning rate of 0.01 in our final configuration. The regularization parameters were set to $\lambda_1 = 1 \times 10^{-6}$, $\lambda_2 = 1 \times 10^{-6}$, and $\lambda_3 = 1 \times 10^{-7}$. We used early stopping with a patience of 3 as our final model.

Results

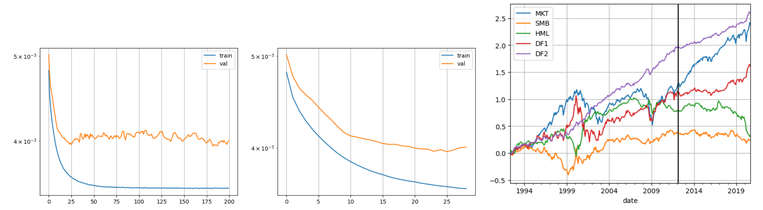


Figure 1. 1. Training and Validation Loss, 2. Train and validation loss with Early Stopping 2. Benchmark and deep characteristics factor returns.

- **Test metric**: Test $R^2 \approx 0.15$. Benchmark $R^2 \approx 0.07$, doubling the variance explained!
- **Deep characteristics**: Each of the five curves shows the cumulative performance from a unit investment in a factor portfolio. The first factor, DF1 (red), captures technology industry movements, while the second factor, DF2 (violet), exhibits smooth performance resembling the mean reversion and momentum factors recognized in the market.
- **Feature importance**: We back-attributed using sensitivity of the latent factors to input characteristics. At the final epoch, only Leverage and Book Value Per Share (BVPS) have meaningful impact on learning. Importance of the (j)-th input feature is the average gradient of the (p)-th deep characteristic (Y) with respect to the (j)-th input feature (z) for the (i)-th stock at time (t).

$$Importance_Y(z_j) = \frac{1}{NTP} \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{p=1}^{P} \left| \frac{\partial Y_{i,p,t}}{\partial z_{i,j,t}} \right|$$

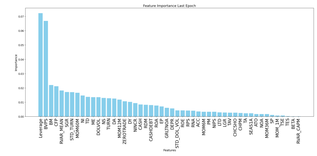


Figure 2. Feature importance as measured using gradient of deep characteristic to input feature at last epoch

- **Different losses**: Our experiments indicated that the Huber generalizes better than L1 and L2 losses.

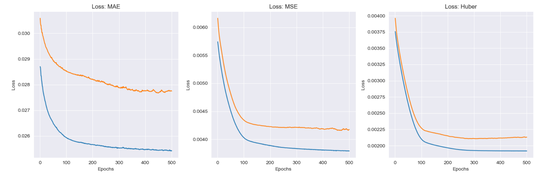


Figure 3. Training and Validation Loss for Different Loss Functions

Future Work

- Incorporate the Huber loss function to enhance generalization.
- Utilize the model's output to construct optimized portfolios, assessing whether latent factors enhance predictability on the test set with a focus on Sharpe ratios and hedging effectiveness.