Numerical Simulations in Financial Mathematics Applications: Asset Pricing Modeling with Deep Learning

Hikaru Belzer, Dr. Bedřich Sousedík
Department of Mathematics and Statistics
University of Maryland, Baltimore County

Background

- **Arbitrage:** A strategy in which a trader simultaneously buys and sells the same asset in different markets, taking advantage of its different prices to ensure a risk-free profit.

- **No-Arbitrage Pricing Theory:** In an efficient market, there is no opportunity for riskless profit. If an asset were mispriced (creating an arbitrage opportunity), traders would immediately exploit it, pushing the price back to equilibrium.

- **Stochastic Discount Factor (SDF):** A function that accounts for the stochastic (random) nature of economic outcomes to "discount" future asset payoffs based on both time and risk, in order to determine the asset's present value and fair price.

$$M_{t+1} = 1 - \omega_t^\top R_{t+1}^e$$

M_{t+1}	$=$ SDF	$R_{t+1,i}^e = R_{t+1,i} - R_{t+1}^f$	$=$ excess returns
N	$=$ number of assets	R_{t+1}^f	$=$ risk-free rate
ω_t	$=$ vector of asset weights in portfolio	$\omega_t^\top R_{t+1}^e$	$=$ dot product of vectors

- **No-Arbitrage Assumption:** The existence of an SDF, $M_{t+1} > 0$, such that, for any return in excess of the risk-free rate, it holds

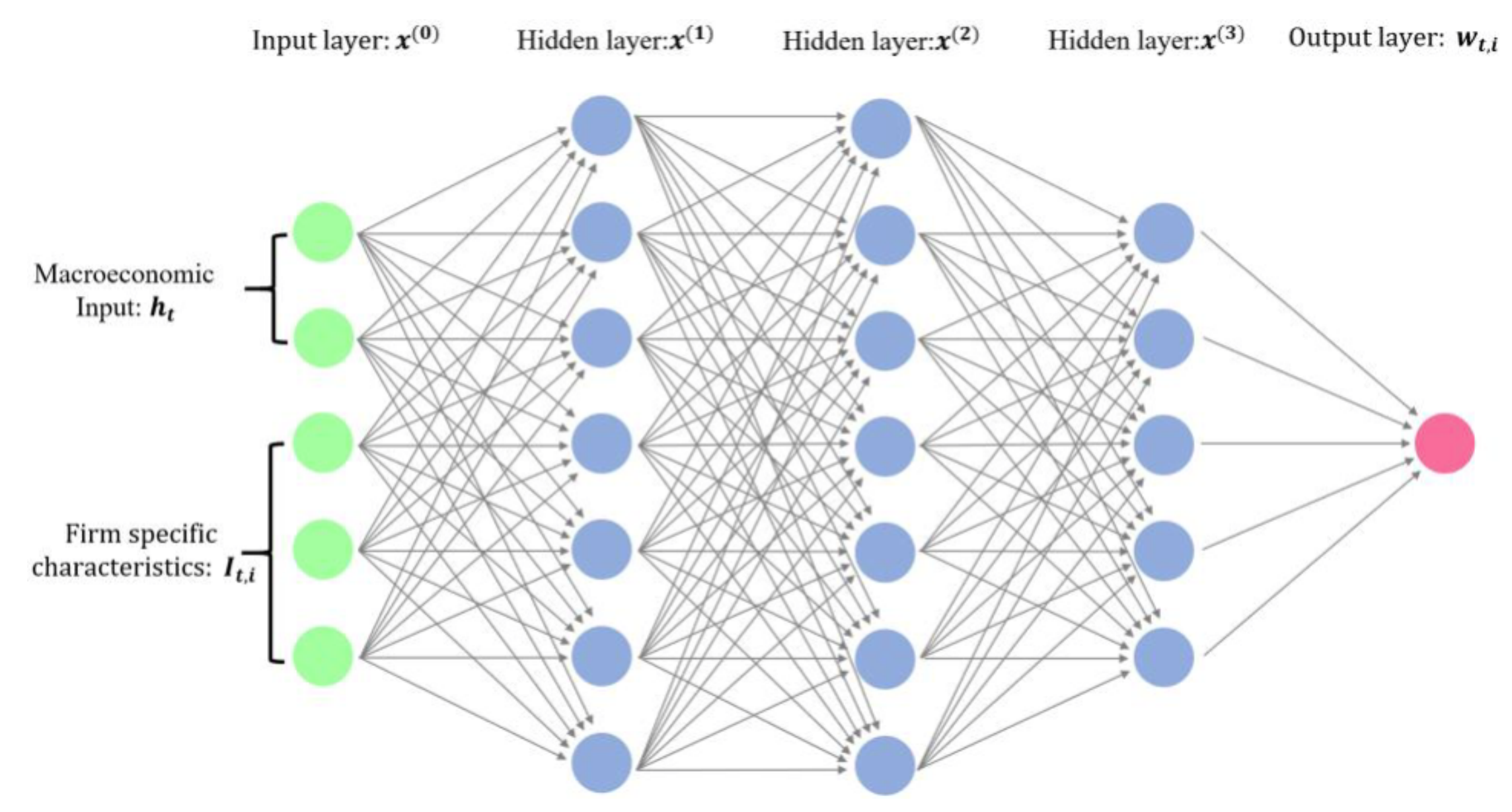
$$\mathbb{E}_t[M_{t+1} R_{t+1,i}^e] = 0$$

- **Loss Function:** Minimizes pricing errors and enforces the no-arbitrage condition. An adversarial network chooses $\hat{g}(I_t, I_{t,i})$ to emphasize assets with the largest mispricings. The model optimizes by minimizing the worst pricing errors. (Smaller residuals $\implies$ Better fit)

$$L(\omega \mid \hat{g}, I_t, I_{t,i}) = \frac{1}{N} \sum_{i=1}^{N} \frac{T_i}{T} \left\| \frac{1}{T_i} \sum_{t \in T_i} M_{t+1} R_{t+1,i}^e \hat{g}(I_t, I_{t,i}) \right\|^2$$

L	$=$ loss function	I_t	$=$ macro feature at time t
T	$=$ total number of time periods	$I_{t,i}$	$=$ asset-specific feature for asset i
T_i	$=$ total number of time periods for asset i	$\sum_{i=1}^{N}$	$=$ sum over all assets
$\hat{g}(I_t, I_{t,i})$	$=$ selects difficult-to-price assets	$\sum_{t \in T_i}$	$=$ sum over all time periods for asset i

- **Neural Network:** Estimates the SDF by learning optimal portfolio weights. This image shows three hidden layers, but this can vary. The optimal model uses two.



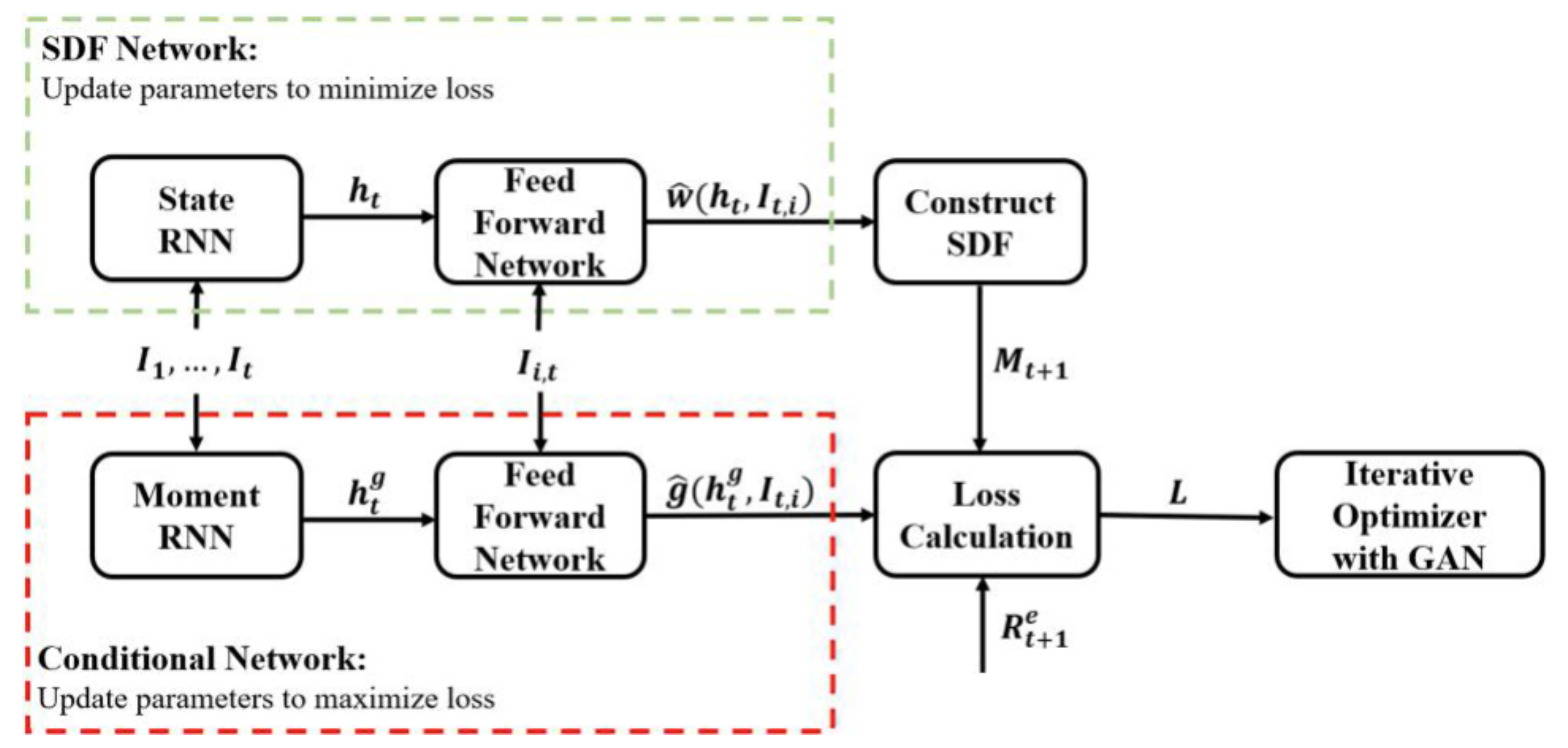
?

- **Question:** Why do different assets have different expected returns?
- **Answer:** According to No-Arbitrage Pricing Theory, expected returns differ because assets have different exposure to the SDF. We estimate the SDF that explains all stock returns from the conditional moment constraints implied by no arbitrage.

Methodology

We use a general nonlinear asset pricing model created by Chen, Pelger, and Zhu, with deep neural networks. We utilize monthly U.S. stock data from 1967 to 2016, 178 macro-economic time series and 46 firm-specific characteristics as the input datasets.

- **Generative Adversarial Network (GAN) Model Architecture:** An adversarial network is a model that uses two competing networks that optimize in opposing directions. Here, it is the SDF Network and the Conditional Network. The conditional network selects asset weights that maximize pricing errors to identify the most mispriced assets, forcing the SDF Network to minimize the worst-case errors.



  - **SDF Network:** Uses a State Recurrent Neural Network (SRNN), which processes past feature information $I_1, \dots, I_t$ to generate a hidden state h_t. A Feed Forward Network (FFN) then uses h_t along with asset-specific information $I_{t,i}$ to produce SDF weights $\hat{w}(h_t, I_{t,i})$. The constructed SDF is then applied to asset returns.
  - **Conditional Network:** Learns an adversarial weighting function that maximizes pricing errors, identifying worst-case mispricings. The Moment RNN processes past feature information $I_1, \dots, I_t$ to produce a hidden state h_t^g. An FFN takes h_t^g and asset-specific information $I_{t,i}$ to compute weights $\hat{g}(h_t^g, I_{t,i})$.

Loss Calculation and Iterative Optimizer: The loss function L quantifies pricing errors. The SDF Network minimizes L for accurate pricing, while the Conditional Network maximizes L to identify worst-case mispricings.

Model Metrics

- **(Unconditional) Sharpe Ratio (SR):** Measures the risk-adjusted return of the SDF portfolio by comparing its expected return to its volatility (standard deviation).

$$\text{SR} = \frac{\mathbb{E}[\text{portfolio return}]}{\sqrt{\text{Var}[\text{portfolio return}]}} \qquad (\text{Higher value} \implies \text{More efficient model})$$

- **Explained Variation:** Measures the proportion of the total variation in excess returns that is explained by the SDF. (Higher value $\implies$ Better performance)

$$EV = 1 - \frac{\left(\frac{1}{T}\sum_{t=1}^{T} \frac{1}{N_t} \sum_{i=1}^{N_t} (\epsilon_{t+1,i})^2\right)}{\left(\frac{1}{T}\sum_{t=1}^{T} \frac{1}{N_t} \sum_{i=1}^{N_t} (R_{t+1,i}^e)^2\right)}$$

ϵ $=$ pricing error for asset i at time $t+1$

- **(Weighted) Cross-Sectional Mean:** Measures how well the model explains the cross-sectional variation in asset returns across different time periods. (Higher value $\implies$ Stronger explanatory power)

$$XS - R^2 = 1 - \frac{\frac{1}{N}\sum_{i=1}^{N} \frac{T_i}{T} \left(\frac{1}{T_i}\sum_{t \in T_i} \epsilon_{t+1,i}\right)^2}{\frac{1}{N}\sum_{i=1}^{N} \frac{T_i}{T} \left(\frac{1}{T_i}\sum_{t \in T_i} R_{t+1,i}\right)^2}$$

Results

- **Training Checkpoints and Constructing Decile Portfolios:** We ran a Python training script 18 times, generating 18 checkpoint trials.

Train SDF Model: Train a GAN-based model to estimate the SDF. Compute SDF-implied weights, normalize SDF factors, and save them for portfolio construction later.

$\downarrow$

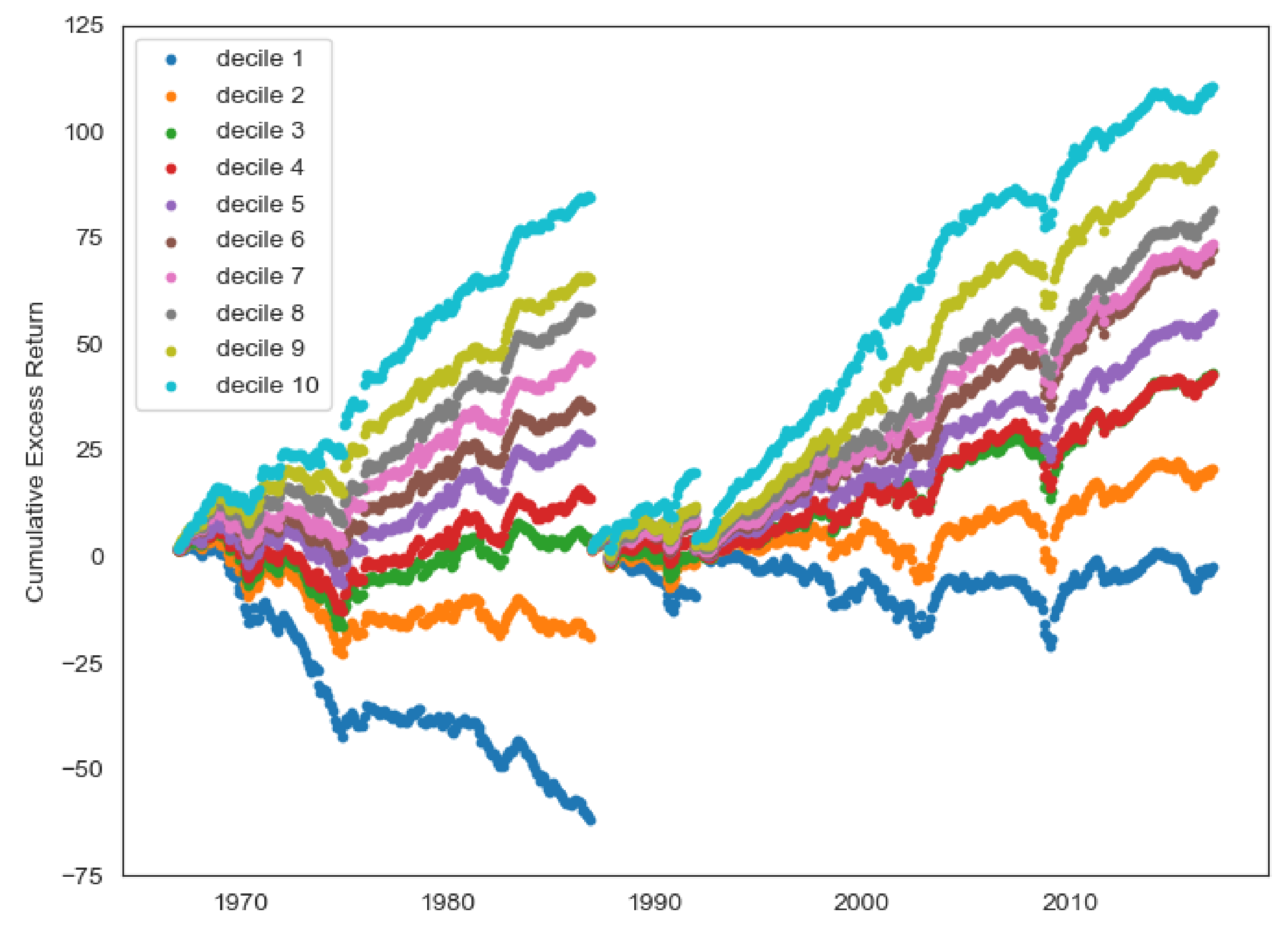
Construct Decile Portfolios: Sort assets into deciles based on predicted returns, calculate portfolio returns for each decile, and plot the performance of the decile portfolios.

- **Selecting the Best Checkpoints:** We selected two checkpoints with the highest test Sharpe ratios. Each checkpoint includes a train, validation, and test Sharpe ratio.

	Train	Validation	Test
Checkpoint 1	4.136	1.272	0.840
Checkpoint 2	5.174	1.104	0.760

- **Final Metrics for the GAN Model**

	Train	Validation	Test
SDF Portfolio Sharpe Ratio	3.10	1.29	0.92
Explained Variation	0.15	0.06	0.06
(Weighted) $XS - R^2$	0.11	0.00	0.19



The plot shows the cumulative excess returns of 10 decile portfolios, calculated relative to the risk-free rate. Portfolio returns for each decile are calculated as the weighted average return of the assets in that decile.

Conclusion: Deep learning appears to be a promising tool that can provide insight into the underlying uncertainties associated with the portfolio optimization process. It can efficiently process large datasets to identify patterns and assist with producing portfolios with higher excess returns as compared to the risk-free rate.

Acknowledgments: This work was funded through an Undergraduate Research Award from the UMBC Office of Undergraduate Education.

References

[1] Bodie, Z., Kane, A., & Marcus, A. (2012). *Essentials of investments*. McGraw-Hill Education.

[2] Chen, L., Pelger, M., & Zhu, J. (2024). Deep learning in asset pricing. *Management Science, 70*(2), 714-750.

[3] Schellhorn, H., & Kong, T. (2024). *Machine learning for asset management and pricing*. Society for Industrial and Applied Mathematics.