Convolutional Network for Portfolio Optimization

Xiyao Fu, Yuao Peng, Shun Wang

A project paper presented for ORIE 5370

Cornell University

05/11/2024

Abstract

In this paper, we introduce a convolutional neural network (CNN) model for portfolio optimization. Unlike traditional approaches that require forecasting expected returns, our CNN model directly maximizes the portfolio's Sharpe Ratio by updating asset weights through gradient descent. We construct portfolios using Exchange-Traded Funds (ETFs) representing key asset classes: stocks, bonds, commodities, and volatility, which exhibit strong correlations. Comparing our model with a baseline asset allocation assigning equal weight to each asset class, we find that the CNN model significantly improves portfolio performance over the testing period. Our findings underscore the potential of deep learning techniques in enhancing portfolio optimization strategies.

1 Introduction

1.1 Portfolio Optimization

Portfolio optimization is a crucial aspect of modern finance, serving as a foundational element in contemporary quantitative finance. It entails selecting the most advantageous asset combination within a defined set of portfolios, guided by particular objectives. These objectives typically emphasize the maximization of expected returns while concurrently mitigating various costs, such as financial risks, thereby presenting a multi-objective optimization endeavor.

Markowitz (1952), the Nobel laureate in economics, introduced modern portfolio theory and outlined the Markowitz model. Within this framework, efficient portfolios are characterized by abilities to maximize expected returns while controlling the risk. Markowitz's model lays the foundation for subsequent research endeavors in portfolio optimization.

1.2 Deep Learning

Deep learning, characterized by its reliance on neural networks, involves the integration of multiple layers within the network in machine learning methodologies. Commonly employed techniques in deep learning include supervised, semi-supervised, or unsupervised approaches LeCun et al. (2015).

Various architectures within the domain of deep learning, including deep neural networks, deep belief networks, recurrent neural networks, convolutional neural networks, and transformers, have been extensively employed across diverse fields such as computer vision, speech recognition, natural language processing, and finance, among others. These applications have demonstrated results that are comparable to, and occasionally exceed, human expert performance.

1.3 Deep Learning in Portfolio Optimization

Portfolio optimization, involving critical tasks such as security selection and risk management, presents challenges for finance professionals. Dealing with astronomical data with complex and non-linear interactions, often lacking clear specifications, is often necessary. Deep learning techniques offer a promising direction for effectively addressing these challenges. These methodologies are able to uncover hidden patterns and relationships which can significantly enhance the effectiveness of portfolio optimization strategies. Consequently, integrating deep learning methodologies into portfolio optimization frameworks holds great potential for improving decision-making processes in finance Heaton et al. (2016)

In this study, inspired by the work of Zhang et al. (2020), we employ a convolutional neural network (CNN) to optimize the portfolio's Sharpe Ratio. This methodology first entails aggregating individual features of various assets to form a single observation. Subsequently, the CNN is utilized to extract relevant information, enabling the derivation of optimal asset allocations directly aimed at maximizing the Sharpe Ratio.

2 Literature Review

In this section, we review several popular portfolio optimization techniques. One notable theory is the Modern Portfolio Theory (MPT), also known as mean-variance analysis, as discussed previously. This mathematical framework aims to construct a portfolio of assets that maximizes expected returns for a given level of risk Markowitz (1952). MPT formalizes the concept of diversification in investing, demonstrating that holding a variety of financial assets can lower the overall portfolio risk compared to owning only one type.

Another significant method in asset allocation is the Black-Litterman model, developed in the early 1990s by Fischer Black and Robert Litterman. This model addresses a key weakness of MPT, which assumes that asset returns follow a Gaussian distribution, implying that investors make decisions based solely on expected returns and variance. However, this assumption is often unrealistic in practice. Instead, the Black-Litterman model introduces an equilibrium assumption: future asset performance mirrors past performance. It then incorporates investors' views on future asset performance into the allocation process. Consequently, the Black-Litterman model provides insights that reflect investors' perspectives while also considering market sentiment.

3 Methodology

3.1 Data

In line with the work of Zhang et al. (2020), we use the Vanguard Total Stock Market Index Fund ETF (VTI), iShares Core US Aggregate Bond ETF (AGG), Invesco DB Commodity Index Tracking Fund (DBC), and ProShares VIX Short-Term Futures ETF (VIXY) as the assets in our portfolio. We chose these ETF funds because they represent the major asset classes in the market: stocks, bonds, commodities, and volatility. For each ETF, we choose the daily adjusted closing price and the return as the features. The data is collected from Yahoo Finance and covers the period from January 1, 2009, to December 31, 2023.

3.2 Neural Network Architecture

We decide to use a convolutional neural network (CNN) for portfolio optimization. Traditionally, CNN is used mostly used in image recognition tasks, because it is able to capture spatial patterns in the data. In our case, the assets and their features can be seen as a 2D matrix, and we make the assumption that there exists some spatial patterns in the data that can be captured by the CNN. For example, prices and returns may have some patterns that cannot be detected by traditional time series models but can be identified by CNN.

The architecture of the neural network is as follows:

1. Input layer: a single input observation $\mathbf{x}_t$ is represented as a 3D tensor with shape $50 \times 4 \times 2$. The first dimension 50 is the loop-back window size. We make the assumption that the trading decision can be affected by up to 50 days of historical movements. This dimension serves as the equivalent of the "channel" dimension in image data. The second dimension 4 is the

number of assets in the portfolio. The third dimension 2 is the number of features for each asset.

2. Convolutional layer I: a 2D convolutional layer with 16 filters of size 2×2 with padding size 2. The activation function is ReLU.

3. Max-pooling layer I: a 2D max-pooling layer with pool size 2×2.

4. Convolutional layer II: a 2D convolutional layer with 16 filters of size 2×2 with padding size 2. The activation function is ReLU.

5. Max-pooling layer II: a 2D max-pooling layer with pool size 2×2.

6. Convolutional layer III: a 2D convolutional layer with 16 filters of size 2×2 with padding size 1. The activation function is ReLU.

7. Max-pooling layer III: a 2D max-pooling layer with pool size 2×2.

8. Flatten layer: a layer that flattens the output of the previous layer into a 1D tensor.

9. Dense layer I: a fully connected layer with 64 neurons. The activation function is ReLU.

10. Dense layer II: a fully connected with 4 neurons. The activation function is ReLU.

11. Output layer: a softmax layer that maps the input to a 4-dimensional vector. The output of the softmax layer is a probability distribution over the assets in the portfolio. We treat the output as the portfolio weights. In doing so, we are implicitly assuming that the portfolio is long-only since the weights output by the softmax layer are non-negative.

We visualize the architecture of the neural network in Figure 1.

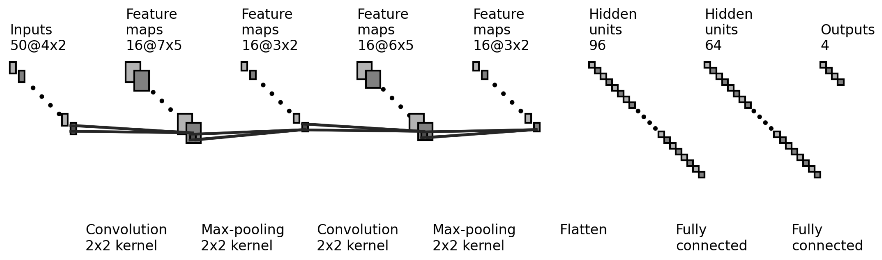

Figure 1: Neural network architecture for portfolio optimization

3.3 Loss Function

The loss function we use is the negative Sharpe ratio of the portfolio. The sharp ratio is defined as

$$\text{Sharpe ratio} = \frac{\mathbb{E}[R_p] - R_f}{\sigma_p},$$

Here R_p is the return of the portfolio, R_f is the risk-free rate, and σ_p is the standard deviation of the portfolio return. The negative Sharpe ratio is used as the loss function because we want to maximize the Sharpe ratio. We make the assumption that the risk-free rate is 0. This will not affect the optimization process, because the risk-free rate is a constant and does not depend on the portfolio weights.

In our case, we implement the Sharpe ratio as per the following steps:

1. We calculate the daily returns of the portfolio $r_1, ..., r_T$.

2. The annualized expected return is calculated as

$$\bar{r} = (1 + r_1) \cdots (1 + r_T)^{\frac{252}{T}} - 1.$$

Here we make use of the fact that there are 252 trading days in a year.

3. The annualized standard deviation is calculated as

$$\sigma = \sqrt{252}\sqrt{\frac{1}{T}\sum_{i=1}^{T}(r_i - \bar{r}_{\text{daily}})^2}.$$

   where $\bar{r}_{\text{daily}} = \frac{1}{T}\sum_{i=1}^{T} r_i$.

4. The Sharpe ratio is calculated as

$$\text{Sharpe ratio} = \frac{\bar{r}}{\sigma}$$

   and the loss function is the negative of the Sharpe ratio.

3.4 Training

To ensure the stability of the training process, we first normalize the input data. We scale each feature by a factor of the maximum value of the feature. This is done to ensure that the input data is in the range $[0, 1]$. The reason why we do not use the more common min-max scaling is that the minimum value of the training data may be larger than some data points in the test set, and the normalization parameters need to be shared between the training and test set, which will cause the test set to have negative price data after min-max normalization. This is not desirable because the price of an asset cannot be negative. Therefore, we simply divide the data by the maximum value of the feature and this works well in practice.

The neural network is trained using the Adam optimizer (Kingma and Ba, 2017), which is a variant of stochastic gradient descent with adaptive learning rates and momentum. The learning rate is set to 0.01 and we enable l_2 regularization with a regularization parameter of 0.001 to avoid overfitting issues. The batch size is set to 32 and the number of epochs is set to 100. We use the batched negative Sharpe ratio as the loss function to train the neural network. The training is done using the PyTorch library in Python.

For the inference step, we use the trained neural network to predict the portfolio weights for the test set on a rolling basis. We use a loop-back window of 50 days to predict the portfolio weights for the next day. The portfolio weights are then used to calculate the portfolio return for the next day. The process is repeated for the entire test set.

4 Experimental Setup

4.1 Data Splits

To ensure the robustness and generalization of our deep learning model, we divided our dataset into three distinct sets: training, validation, and testing. We use data from 2009-01-01 to 2016-12-31 (8 years, 53% of data) for model training, and data from 2017-01-01 to 2022-12-31 (6 years,40% of data) for validation. This period includes the recovery phase following the 2008 financial crisis and several subsequent years of economic growth and market volatility. It also covers various economic events, such as the European debt crisis (2010-2012) and the Covid-19 period (2019-2021).

To compare our model's performance over equal weight allocation, we use data from 2023-01-01 to 2023-12-31 for testing(6.7% of data). We aim to evaluate the model's performance on the most recent data.

4.2 Computational Resources

To train and test our deep learning model, we utilized the following computational resources:

- **Hardware Specifications**: Intel Core i7-9700K (CPU), NVIDIA GeForce

RTX 2080 (GPU), 32 GB RAM.

- **Software Environment**: Ubuntu 20.04 LTS (Operating System), PyTorch 1.8.1 (Deep Learning Framework), Python 3.8.5 (Python Version).
- **Running Time**: The training process took approximately 3 minutes to complete 100 epochs. The data loading process, evaluation and comparison script took negligible time.

5 Results

5.1 Sharpe Ratio Comparison

Our performance of the model is evaluated by the comparison of Sharpe ratios between the CNN model and the baseline method. We test model on the daily adjusted close price of the 4 ETF we chose in date range 2023-01-01 to 2023-12-31. Table 1 presents the results. The baseline algorithm used in this study assigns equal weights to each ticker in the portfolio. This approach is commonly known as equal-weighting and serves as a straightforward benchmark. The rationale behind using this baseline is to provide a simple, non-optimized comparison for evaluating the effectiveness of more complex models.

Model	**Annualized Sharpe Ratio**
CNN Model	2.307
Baseline Model	-1.626

Table 1: Comparison of Sharpe Ratios

The Sharpe ratio of our model is significantly higher than the baseline model. CNN model has higher risk-adjusted return for data in 2023, highlighting its potential for effective portfolio optimization. This result aligns with the findings of Zhang et al. (2020)

5.2 Discussion of Results

5.2.1 Learning Curve Analysis

In Figure 2, the learning curve illustrates the training and validation loss over 100 epochs.

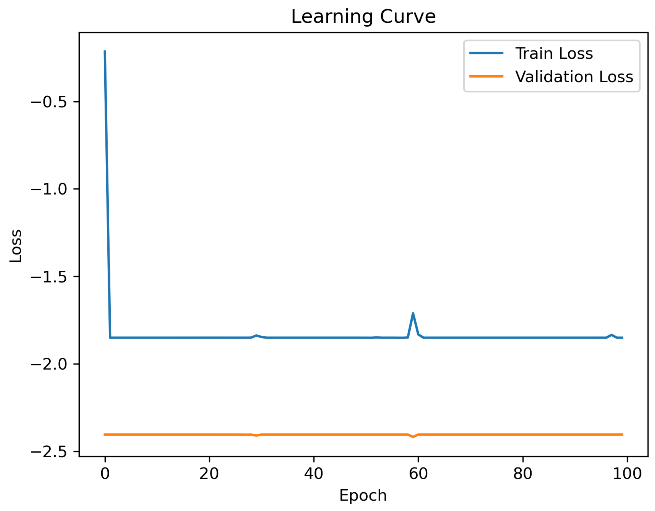

Figure 2: Learning Curve

The training loss decreases significantly in the initial epochs.Both training and validation losses stabilize around -2 and -2.5, respectively, after the initial drop. This early plateau suggests that the model reaches its optimal performance quickly but also indicates that it may have limited capacity or the training process could be improved.

The smooth and converging behavior of the loss curves suggests that the chosen CNN architecture and hyperparameters are appropriate for the task at hand. The absence of significant fluctuations or divergence in the validation loss indicates that the model is stable and not sensitive to small variations in the training data.

The small gap between training and validation losses suggests decent generalization, yet the persistent gap hints a slight overfitting. The final high loss values at convergence suggest potential underfitting.

There is a spike in the both training and validation loss around epoch 60.This instability could be due to high learning rates causing parameter oscillations, overfitting at specific points, or random fluctuations inherent to the stochastic nature of training processes.

5.2.2 Strengths

Unlike the static equal-weighted approach of the baseline model, the CNN model dynamically changes portfolio weights in response to market conditions. This dynamic adjustment results in a better balance of returns and risks, reducing possible losses during periods of severe market volatility.

The CNN architecture is particularly well-suited for modeling the correlations between the chosen ETFs as it can learn hierarchical features from the 2D input matrix. By stacking multiple convolutional layers, the network can potentially identify patterns such as the flight-to-safety dynamic where investors shift from stocks (VTI) to bonds (AGG) during market downturns Ilmanen (2013). Visualizing the learned filters of the CNN could provide further insight into these asset class relationships.

5.2.3 Limitations

One limitation of deep learning models is their ”black box” nature, making it difficult to interpret how they arrive at predictions. Future work could apply techniques like Layer-wise Relevance Propagation Binder et al. (2016) or Integrated Gradients Sundararajan et al. (2017) to obtain insight into which input features the CNN is prioritizing when making allocation decisions. Visualizing the learned filters could also shed light on what asset relationships and temporal dynamics the model is capturing Yoo et al. (2019).

While our chosen hyperparameters (e.g., lookback window, regularization strength) were effective ETFs, it's important to note that optimal settings may vary for different ETF combinations. Future work could employ Bayesian optimization techniques Snoek et al. (2012) to automatically tune hyperparameters for specific ETF baskets. This would help ensure the CNN is well-calibrated for the dynamics of the selected assets.

In our experiments, we rebalanced the CNN-optimized ETF portfolio on a daily basis. However, in practice, the optimal rebalancing frequency depends on factors like transaction costs and market volatility Jaconetti et al. (2010). High-frequency rebalancing could erode returns due to ETF bid-ask spreads and commissions. Future research could incorporate transaction cost modeling directly into the CNN's loss function to find the best trade-off between rebalancing frequency and portfolio turnover.

6 Conclusion

6.1 Paper Summary

In this study, we employ the convolutional neural network (CNN) to directly maximize the portfolio Sharpe Ratio. Unlike traditional portfolio optimization methods that require forecasting expected returns, our neural network architecture facilitates the direct optimization of the Sharpe Ratio by iteratively updating model parameters through gradient descent.

The comparative analysis reveals that the CNN methodology improves the portfolio Sharpe Ratio compared to our proposed baseline model, which assigns equal weights.

6.2 Future Improvements

6.2.1 Different Asset Classes

In order to reproduce the outcomes outlined by Z. Zhang et al. (2020), we adopted an identical set of ETF asset classes, comprising VTI, AGG, DBC, and VIXY, for our portfolio optimization. However, to evaluate the generalizability of our findings and investigate the potential for maximizing Sharpe Ratio across a broader realm of assets, it is necessary to expand our analysis to encompass additional indices. By integrating a diverse array of asset classes, our objective is to evaluate the efficacy and performance of our model across various financial market scenarios and ascertain its capability to consistently optimize portfolio across diverse asset types.

6.2.2 Different Deep Learning Architectures

Our model has focused on maximizing the Sharpe Ratio through the utilization of a convolutional neural network. To enhance the efficiency of portfolio optimization, we can incorporate additional deep learning architectures such as Long Short-Term Memory (LSTM). By training the dataset with LSTM, we can explore the portfolio performance generated by different approaches, thus broadening our understanding of optimal portfolio construction.

6.2.3 Different Performance Metrics

Moreover, extending our analysis beyond the Sharpe Ratio allows for a deeper comprehension of portfolio performance. As proposed by Z. Zhang et al. (2020), our methodology's flexibility enables us to maximize alternative performance metrics such as the Sortino Ratio or the degree of portfolio diversification, provided that these objective functions are differentiable. This flexibility enhances our ability to tailor portfolio optimization strategies to specific investment objectives and market conditions.

References

Binder, A., Montavon, G., Lapuschkin, S., M"uller, K.-R., & Samek, W. (2016). Layer-wise relevance propagation for neural networks with local renormalization layers. *International Conference on Artificial Neural Networks*, 63–71.

Heaton, J., Polson, N., & Witte, J. (2016). Deep learning for finance: Deep portfolios: J. b. heaton, n. g. polson and j. h. witte. *Applied Stochastic Models in Business and Industry*, *33*.

Ilmanen, A. (2013). The stock-bond correlation. *The Journal of Fixed Income*, *22*(4), 8–19.

Jaconetti, C. M., Kinniry, F. M., & Zilbering, Y. (2010). Best practices for portfolio rebalancing. *Available at SSRN 1609819*.

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–44.

Markowitz, H. (1952). Portfolio selection*. *The Journal of Finance*, *7*(1), 77–91.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 2951–2959.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*.

Yoo, K., Cao, Y., Yang, Y., Fischer, F., Chien, A., Das, S., & Lu, Y. (2019). Visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics*, *25*(1), 215–224.

Zhang, Z., Zohren, S., & Roberts, S. (2020). Deep learning for portfolio optimization. *The Journal of Financial Data Science*, *2*(4), 8–20.