



Hopfield networks for asset allocation

Carlo Nicolini
Ipazia SpA
Italy
c.nicolini@ipazia.com

Bruno Lepri
Fondazione Bruno Kessler
Italy
Ipazia SpA
Italy
lepri@fbk.eu

Monisha Gopalan
Ipazia SpA
Italy
m.gopalan@ipazia.com

Jacopo Staiano
University of Trento
Italy
jacopo.staiano@unitn.it

Abstract

We present the first application of modern Hopfield networks to the problem of portfolio optimization. We performed an extensive study based on combinatorial purged cross-validation over several datasets and compared our results to both traditional and deep-learning-based methods for portfolio selection. Compared to state-of-the-art deep-learning methods such as Long-Short Term Memory networks and Transformers, we find that the proposed approach performs on par or better, while providing faster training times and better stability. Our results show that Modern Hopfield Networks represent a promising approach to portfolio optimization, allowing for an efficient, scalable, and robust solution for asset allocation, risk management, and dynamic rebalancing.

CCS Concepts

• **Applied computing** → **Economics; Information systems; • Computing methodologies** → **Neural networks; Cross-validation; Neural networks; Cross-validation; Machine learning; • Theory of computation** → **Mathematical optimization; Mathematical optimization; • Information systems** → **Information systems applications; • Machine learning** → **Machine learning approaches; • Mathematics of computing** → **Probability and statistics; • Probability and statistics** → **Statistical paradigms; • Statistical paradigms** → **Time series analysis;**

Keywords

deep learning, asset allocation, transformers, hopfield networks

ACM Reference Format:

Carlo Nicolini, Monisha Gopalan, Bruno Lepri, and Jacopo Staiano. 2024. Hopfield networks for asset allocation. In *5th ACM International Conference on AI in Finance (ICAIF '24)*, November 14–17, 2024, Brooklyn, NY, USA. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3677052.3698605>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICAIF '24, November 14–17, 2024, Brooklyn, NY, USA

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1081-0/24/11
<https://doi.org/10.1145/3677052.3698605>

1 Introduction

Starting from the seminal work of Markowitz [21], financial literature has seen multiple attempts to tackle some of the fundamental problems of portfolio optimization, like the estimation of covariance matrix and asset returns. Within the standard portfolio optimization theory, a significant challenge is posed by the accurate estimation of asset returns: misestimations can lead to sub-optimal asset allocations, thus increasing the risk of financial losses and undermining the efficiency of the investment strategy. While some authors adopt theoretically sound approaches (the ideas from [8] or [22, 23] to cite a few), several tend to be guided by practice and intuition, such as the bootstrap-based resampled efficient frontier by Michaud and Michaud [24].

Recently, the work of Zhang et al. [35] sparked a wave of interest towards the application of deep learning to portfolio optimization: the authors formulated the portfolio optimization problem within an end-to-end approach based on maximizing the strategy's Sharpe ratio by means of a Long-Short-Term-Memory (LSTM) network [11] to learn asset returns' temporal patterns. The memory cells of the LSTM layer are good at modeling asset returns as they are able to generalize quite well to the test set, resulting in high out-of-sample Sharpe ratio.

A more complex architecture was proposed in the *Portfolio Transformer* by Kisiel and Gorse [16]. Their model is largely inspired by the standard Transformer architecture proposed by Vaswani et al. [33]: a 4-layer encoder is connected to a 4-layer decoder in a very similar way to the original structure proposed for machine translation tasks. Differently from the standard Transformer, where positional encoding is adopted to inject temporal dependencies to the self-attention layer, the authors of the Portfolio Transformer embed asset returns through a Time2Vec layer [14], in order to learn frequency and phase parameters within positional embeddings.

Inspired by these two works, and guided by a physical intuition [18], we adopt a neural architecture based on Modern Hopfield Networks [26], here abbreviated as MHN. This recently proposed architecture is the continuous counterpart of traditional Hopfield networks, with an exponential activation function leading to a much larger repertoire of stored patterns [6, 17, 19]. The expressive power of MHNs, together with their fully connected, recurrent architecture [13], makes them suitable for tasks where time-series are provided in input.

At the core of our approach is the mitigation of the adverse impacts of estimation errors inherent in the Markowitz method [2, 28], while simultaneously preserving the high-capacity storage and retrieval capabilities for intricate patterns, a distinctive feature of MHNs [26] that is reflected in better estimation of future returns. By leveraging their high storage capacity, MHNs can assimilate a broader spectrum of market conditions and anomalies, leading to more robust and accurate return estimates in adverse regimes.

In this work, we empirically test this hypothesis via an experimental setup including two MHN-based architectures: the first design simply replaces the LSTM layer in [35] with an Hopfield Pooling operator, internally guided by a dense associative memory mechanism; in the second approach, we use a Transformer-like architecture but with an Hopfield layer replacing the multi-head self-attention operator.

Our results reveal that a simple MHN often surpasses more intricate architectures such as LSTMs, achieving superior performance within significantly shorter training times, and thus underscoring the efficacy of neural architectures within the domain of portfolio optimization.

In the following we describe both models in detail, present the benchmark results and finally discuss implications and possible problems of our proposed approaches.

2 Background and methods

2.1 Notation and metrics

In this study, we analyze time series of *daily* asset prices for the portfolio optimisation task. We denote the time series of asset prices as a $T \times N$ matrix \mathbf{P} with elements $P_{t,i}$ denoting asset i at time t . Similarly, asset returns are denoted by a $(T - 1) \times N$ matrix \mathbf{R} with elements $R_{t,i}$. We compute asset returns as linear returns $r_{ti} = (P_{t,i} - P_{t-1,i})/P_{t-1,i}$.

Asset weights in the portfolio are denoted by a vector \mathbf{w} with shape $(1, N)$ and elements $0 \leq w_i \leq 1$. For long-only portfolios asset weights are all positive and sum to 1. One could also deal with short portfolios, having positions with the only constraint of summing to 1, thus allowing negative weights. In this work, however we only discuss long portfolios.

The portfolio return series $R_p(t) = \mathbf{w}^T \mathbf{R}(t)$ allows for the computation of various performance metrics, such as the average return, Sharpe ratio, and Sortino ratio.

The Sharpe ratio [29] evaluates risk-adjusted return by comparing the average portfolio return $\bar{R}_p = \langle R_p \rangle$ to its standard deviation (or volatility) $\sigma_p = \sqrt{\langle (\bar{R}_p - R)^2 \rangle}$, relative to the risk-free rate R_f . Meanwhile, the Sortino ratio [27, 30] focuses on downside risk, using downside deviation $\sigma_d = \sqrt{\langle \min(\bar{R}_p - R, 0)^2 \rangle}$ to assess volatility based solely on negative returns, providing a nuanced view of portfolio performance under adverse conditions. These metrics are essential tools in modern portfolio analysis, as highlighted in Chen's comprehensive framework [4].

We also reported the Average Drawdown [3] (Avg. DD), computed as the average of all drawdown series over the portfolio history. It provides an average measure of the peak-to-trough decline in asset value over multiple drawdown periods, offering insights into the consistency and severity of losses experienced by an investment.

2.2 Time2Vec embeddings

The input of our models are the asset returns, with a first layer operating a transformation of returns from the time domain to the frequency domain. Specifically, we employ Time2Vec [14] to incorporate trainable embeddings into the model parameters. Each univariate time series representing the i -th asset returns $r_i(t)$ undergoes a transformation onto a $K+1$ dimensional vector \mathbf{v}_i through learned frequencies and phase shifts $\omega_k \phi_k$, hence capturing periodic rhythms within $r_i(t)$ through a periodic activation function \mathcal{F} (like sine or cosine):

$$v_i^k(t) = \begin{cases} \omega_k t + \phi_k & \text{if } k = 0 \\ \mathcal{F}(\omega_k t + \phi_k) & \text{if } 1 \leq k \leq K \end{cases} \quad (1)$$

Working with multivariate time series, we embed each asset time series consisting of T observations within each batch and concatenate the embeddings.

2.3 Hopfield Networks

Modern Hopfield Networks [26], like their traditional counterparts [12] are densely connected Recurrent Neural Networks [31]. Differently from their traditional counterpart, they can work with continuous input, store an exponentially large number of patterns and are able to retrieve patterns with a single update [26]. To be precise, let $\mathbf{X} \in \mathbb{R}^{N \times d}$ be a matrix of N data samples $\mathbf{x}_1, \dots, \mathbf{x}_n$ (also called *memory patterns* in Hopfield networks literature), each one with dimension d . Memory patterns \mathbf{X} , such as asset returns or their corresponding *Time2Vec* embeddings, are forwarded through the Hopfield network. The backpropagation step iteratively adjusts the network's internal state pattern $\mathbf{q}_t \in \mathbb{R}^d$ converging towards stored patterns \mathbf{X} or their combinations. This process has an associated energy function described by Equation 2, first described by Ramsauer et al. [26]:

$$E = -\beta^{-1} \log \left(\sum_{l=1}^N \exp(\beta(\mathbf{X}^T \mathbf{q}_l)) \right) + \frac{1}{2} \mathbf{q}^T \mathbf{q} + \beta^{-1} \log N + \frac{1}{2} M^2. \quad (2)$$

The last term in the equation is the regularization term where M is the largest norm of all stored patterns: $M = \max_i \|\mathbf{x}_i\|$. The coefficient β has its foundation in statistical physics and is often considered to act as an inverse temperature. In Equation 2 it acts as a regularization term, controlling the balance between the free-energy term (first addend) and the internal energy (second addend).

Interestingly, as pointed out by Ramsauer et al. [26] the update rule for internal states \mathbf{q} defines exactly the attention mechanism used within the Transformer model: direct minimization of the energy-based model 2 yields an update rule

$$\mathbf{q}_{t+1} = \text{softmax}(\beta \mathbf{X} \mathbf{q}_t) \quad (3)$$

In this sense, Modern Hopfield networks can be seen as the theoretical background behind the Transformers' attention mechanism.

2.4 Model architecture

We adopt the energy model operationally described by Equations 2 and 3 within two effective network architectures: Pooling [7] and Transformer [33]. The first architecture, termed the Hopfield Pooling (Figure 1, left panel), resembles the foundational architecture

proposed by Zhang et al. [35], albeit with the substitution of the LSTM component by a Hopfield Pooling layer.

The second architecture, denoted as the Hopfield Encoder (Figure 1, right panel), draws inspiration from the Transformer’s encoder, wherein the multi-head self-attention mechanism is substituted by a Hopfield layer.

Hopfield Pooling. We build over the idea of [35], by replacing the original Long Short-Term Memory (LSTM) network [11] with an Hopfield Pooling layer. The Hopfield Pooling layer is a neural operator mapping $[B, T, N]$ tensors into $[B, N]$ arrays, hence operating a pooling operation on the temporal dimension. The recursive properties of the Hopfield layer are used with a hidden neuron dimension of 2048, thus allowing to store long and complex patterns. More specifically, this large associative memory provided by the Hopfield Pooling layer allows to capture and recall complex patterns during down-sampling and feature extraction, hence propagating rich information to the downstream layers.

Hopfield Encoder. Here, we maintain the encoder part of the Transformer, but replacing the multi-head self-attention mechanism with a Hopfield layer. Following the Time2Vec embedding process, tensors denoted as X with a shape of $K + 1$ are forwarded into the encoder. The encoder comprises of replicated modules of a designated encoder block function denoted as f . The function f , as described in Equation 4, consists of a Hopfield layer, succeeded by a skip connection and layer normalization as per the formulation proposed by Ba et al. [1]. Subsequently, the output undergoes processing by a multilayer perceptron (MLP) featuring the GELU activation function proposed by Hendrycks and Gimpel [10]. This MLP output is then subjected to an additional skip connection, as outlined by He et al. [9], followed by another layer normalization step.

In progressing from block b to block $b + 1$, each data tensor X undergoes two transformations as described in Equation 4:

$$\begin{aligned} X' &\leftarrow \text{LayerNorm}(\text{HopfieldLayer}(X) + X) \\ X_{b+1} &\leftarrow \text{LayerNorm}(\text{MLP}(X') + X') \end{aligned} \quad (4)$$

By stacking multiple layers of the encoder, the model progressively captures increasingly intricate representations of the structural properties inherent in the time series data. The hidden size dimension of the MLP is set to four times the model embedding size. For computational efficiency and to match the architecture by [16], we opted for an embedding dimension of $D = 128$ and implemented a configuration comprising four layers in the encoder.

Output layer. For both the pooling and the encoder models, the output tensor X' at the last block is fed through a final dense layer (*Linear* in Figure 1) reshaping the D -sized input onto N shaped logits. Portfolio asset weights then result from the softmax operation applied on the output logits. In the simplest settings, the output layer consist of a softmax that normalizes the weights to unit-sum, followed by the calculation of the loss function. Alternatively short portfolios are possible, by taking the output s_i from the last layer (*logits*) s_{it} of the model, considering the sign and normalizing through a softmax function over the assets [34].

Our methods make it possible to implement any differentiable portfolio metric in terms of a loss function. We experimented both

with maximization of strategy’s Sharpe ratio \bar{R}_P/σ_P and minimization of strategy volatility σ_P .

2.5 Models training

Training data are forwarded to the network in mini-batches of size $B = 32$, with each batch $b \in [1, B]$ containing observations from the training set with rows $[b + t, \dots, b + t + t_l]$ where $t_l = 128$ days is the look-back window. This batching procedure is illustrated in the left panel of Figure 2. We apply the same batch size both for training and validation data. This data batching approach facilitates efficient learning from the time series sequential data.

At inference-time, portfolio weights are obtained as the average of the predicted weights over all input batches. This marks a significant difference with the work of Zhang et al. [35], whereas they implicitly assumed daily rebalancing within the (walk-forward) backtesting.

We have utilized AdamW for model training, an adaptive momentum strategy [15, 20] that works best in terms of out-of-sample results’ quality. We have limited the number of epochs to 500, however we have found that over a multitude of datasets, typically the optimizer finds a good solution reaching good generalization around epoch 10-20. Unsurprisingly, we have seen that overtraining the model, even in conditions of better validation loss, tends to produce bad test-time allocations when evaluated with a fair cross-validation technique.

For model training we have utilized an AWS g5.2xlarge with 8 CPUs and a single NVidia A10 GPU equipped with 24GB of VRAM. The Hopfield Encoder has the highest utilization of GPU resources and training on 20 years of daily data for almost 200 epochs takes around ten minutes.

2.6 Cross validation and statistical testing

The combinatorial purged cross-validation (CPCV) method introduced by De Prado [5] was employed to assess the performance of the portfolio allocation strategy. The CPCV method tries to overcome the overfitting challenges faced by traditional k-fold and walk forward cross validation by generating multiple paths of combinations of training and test data. Moreover, in order to make our comparison more fair, purging was applied to remove any data leakage between the training and test sets in addition to embargoing to remove any serial correlation.

To minimize the probability of false discoveries, we have used 10 training folds and 8 test folds, resulting in 36 back-test testing paths and 45 training combinations. The average training set size is ≈ 3.5 years, while the test size is ≈ 2 years. For both the purging and embargoing periods we have set 21 days, corresponding approximately to one business month.

Within each cross-validation training fold we further subdivided the training set into two distinct consecutive sets in proportions of 80% training and 20% validation, in order to keep track of the loss function and trigger early stopping. We have used the `skfolio`¹ implementation of combinatorial purged cross-validation.

Figure 2 depicts the cross-validation setup we used.

¹<https://github.com/skfolio/skfolio>

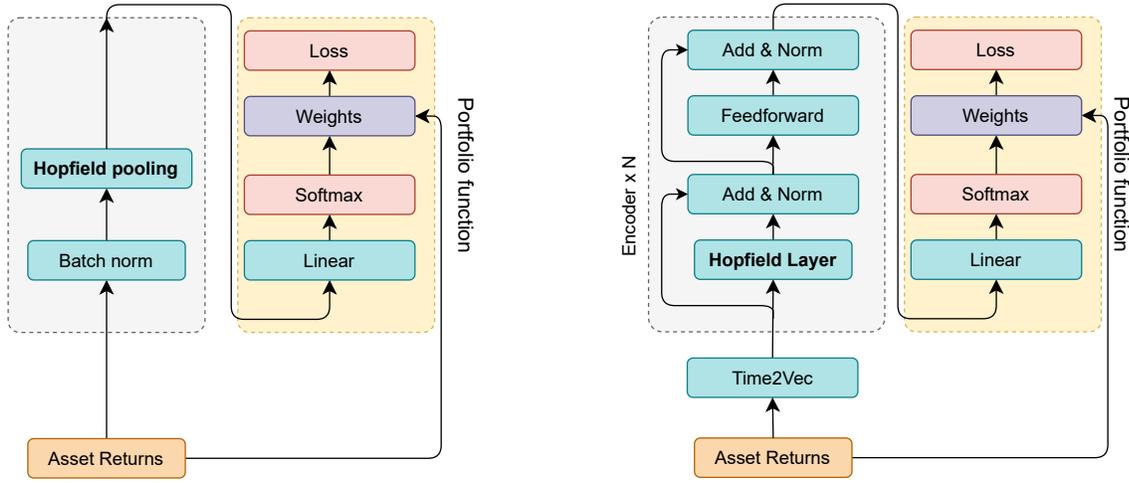


Figure 1: Left: The Hopfield Pooling model (HOP-POOL), where a simple pooling operator replaces an LSTM layer. Right: In our Hopfield Encoder model (HOP-TRA) we stacked the encoder block four times (dashed gray blocks denoted by Encoder $\times N$).

Transaction costs. Importantly, in order to focus on specific behaviours of our methods, in the following Results Section we have ignored transaction costs and management fees.

2.7 Datasets

We compare our method against traditional baselines over four different datasets representing different sectors (stocks, commodities, ETFs) hence having different returns' properties. All the datasets cover a span of almost 20 years.

The first dataset (named ETF in the text) consists of the four ETFs used by Zhang et al. [35], although over a different temporal interval. The period analyzed spans from February 6th 2006 to December 29th 2023. The four assets are *iShares Core U.S. Aggregate Bond ETF* (AGG), *CBOE Volatility Index* (VIX), *Invesco DB Commodity Index Tracking Fund* (DBC) and *Vanguard Total Stock Market Index Fund* (VTI).

As the second dataset (named STOCKS in the text) we have obtained the spot prices of 66 securities from the Nasdaq index, from January 2nd 2003 to December 30th 2022. We have selected this subset of 66 stocks as it was the longest subsection of Nasdaq data with no missing data. All the data have been obtained from Yahoo Finance.

As the intermediate size dataset (named FAMA-FRENCH in the text), we considered the daily returns from 10 industry portfolios starting from January 1st 1973 until November 30th 2023. Data were obtained from the Fama-French data library.²

Finally, we have run experiments on commodities future prices downloaded from Yahoo finance,³ from 2003 to 2023. The dataset named COMMODITIES included 19 commodities from gold to gasoline and soybean. All the datasets tickers list is provided in the Supplementary Materials Section.

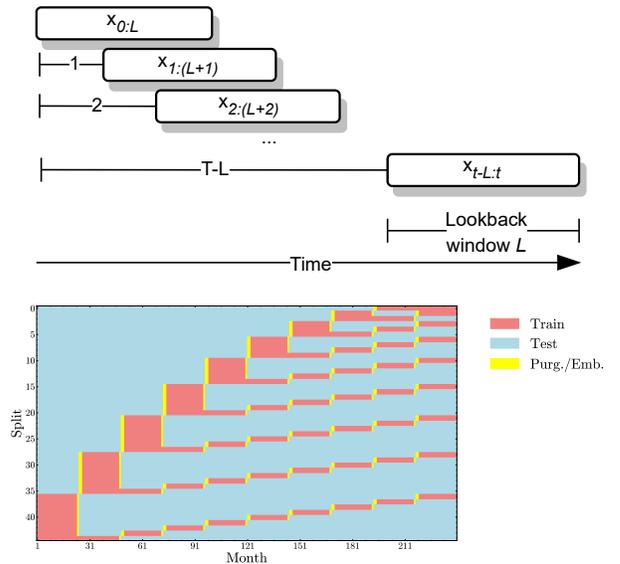


Figure 2: Left: Data batching procedure. For each training and validation set, B batches are created of shape L, N for each batch, where L is the look-back window and each batch element is shifted by one observation from the previous, until the final element is reached. Right: Schematic representation of training and testing folds in combinatorial purged cross-validation. For visual clarity we aggregated folds over 20 years of data at monthly granularity.

2.8 Benchmarks

We have compared the results of our two Hopfield-based models to the traditional convex optimization methods designed to look for portfolios with the least risk. First and foremost, the Markowitz

²https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/10_Industry_Portfolios_daily_CSV.zip

³<https://finance.yahoo.com/>

Dataset	#assets	Start	End	Coverage
ETF	4	2006-02-06	2023-12-29	≈ 15 years
FAMAFRENCH	10	1973-01-02	2023-11-30	≈ 50 years
STOCKS	66	2003-01-02	2022-12-30	≈ 20 years
COMMODITIES	19	2003-01-02	2023-12-29	≈ 20 years

Table 1: Properties of the datasets used through this work.

Mean-Variance portfolio [21] where we have looked at the minimum volatility solution (MVO). Second, we have selected the excellent Hierarchical Risk Parity method by De Prado [5], indicated as HRP. The entire data without any batching was used for these convex optimisation methods. To make the comparison fair with another deep learning method, we have replicated the algorithm by Zhang et al. [35] using the LSTM network. However, we used only the asset returns as input, instead of concatenating asset prices and returns, to ensure consistency in the input data for all the methods. Finally, we evaluated the Hopfield Pooling method (HOP-POOL) and the Hopfield Transformer method (HOP-TRA).

3 Results

We have run several portfolio optimization methods using combinatorial purged cross validation. Each backtesting path resulted in different test returns' series, possibly with small intersections among the testing paths, which we tried to minimize setting both embargo and purging. Over all the backtesting paths we have computed the average and standard deviation of several metrics, specifically the annualized average return, annualized Sharpe ratio, annualized Sortino ratio, and average drawdown. Moreover, with all the backtesting paths we were able to conduct accurate statistical analyses on portfolio metrics to assess the significance of our claims. Results are shown in Table 2 and Figure 3.

Moreover, in order to provide a better assessment of the statistical relevance of our analyses we have run a Tukey HSD test [32], a method for comparing the means of multiple groups that highlights which pairs are significantly different from each other. We have run the Tukey test independently over all the four datasets presented in this work. In Table 3 we indicate with Compact Letters Display notation [25] the similarity groups of methods' Sharpe ratios among the different datasets.

To summarize the findings from the statistical analyses of Table 3 we can conclude that in the ETF, STOCKS and FAMAFRENCH datasets the three deep learning methods perform similarly. However the Hopfield Encoder performs better than both LSTM and Hopfield Pooling in the COMMODITIES dataset.

Additionally we visually explored the distribution of backtesting Sharpe ratios through a Box plot in Figure 4.

The results, taken over all the four datasets are very encouraging, although some weakness of the proposed methods is evidenced. On the smallest dataset (ETF), we reach high Sharpe ratio consistently, with the statistical tests indicating that our proposed method is not statistically different from the LSTM, but presents statistically relevant higher Sharpe ratios with respect to the two other traditional methods MVO and HRP. The superiority of the end-to-end deep learning approaches (LSTM, HOP-POOL, HOP-TRA) are also clear

in the STOCKS dataset, again with statistically significant higher Sharpe ratios between the deep learning methods against the traditional ones. In this case however all the deep learning methods perform similarly.

Another kind of results are highlighted for the COMMODITIES and FAMAFRENCH datasets. Here all the deep learning methods fail in comparison to traditional approaches. We believe that this result stems from a poor generalization effect in return series where multiple regimes are present. Indeed, differently from the STOCKS and ETF datasets where a growth trend is evident, the COMMODITIES and FAMAFRENCH are much more variable. Hence, taking an average of almost two years of predicted weights over highly variable regimes may hinder the beneficial effects observed with other methods.

Equally weighted portfolio. In the aforementioned analysis, we only reported the cross-validation metrics for optimization based methods with their uncertainty computed as the standard deviation of each metric over the backtesting paths. We omitted the metrics uncertainty for the equally weighted portfolio as each metric had the same value over all backtesting paths, hence zero standard deviation. We also note that the equally weighted portfolio allocation has serious problems in the datasets analyzed. First, in the ETF dataset, allocating 25% to the VIX index was deemed unfavorable due to its association with excessive volatility, despite yielding a commendable final total return. In the STOCKS dataset, comprising 66 assets, adhering to an equally weighted portfolio approach would lead to impractical transaction costs during rigorous back-testing procedures. The same applies to the FAMAFRENCH and COMMODITIES datasets. Nonetheless we report the equally weighted backtesting results in a separate chart for completeness in the Supplementary Materials section.

4 Discussion

In this section we discuss some of the observations resulting from the experiments, together with possible interpretations and comments.

Hopfield pooling networks work like LSTMs. Our work shows that modern Hopfield networks can reliably be used also in the time series domain, specifically in non-standard tasks like the one of portfolio optimization. What we have found is that a LSTM network can be replaced with an Hopfield Pooling network, even without the need of a Time2Vec embedding. Our experiments show that most of the times the results from the LSTM and HOP-POOL models are similar, but training HOP-POOL is faster, given the lower number of parameters.

A Hopfield network can replace the multihead self attention layer. In the proposed Hopfield Encoder architecture, we have observed that the Hopfield layer can replace the classical multi-head self-attention layer used in the Transformer block. In our experiments we have not run any hyper-optimization on the multitude of parameters of the encoder architecture, like number of layers, number of heads within each Hopfield layer or internal embedding dimension, instead we have chosen the same parameters as in the original implementation by Ramsauer et al. [26]. However, as shown in the previous section, results are encouraging and while we see slightly

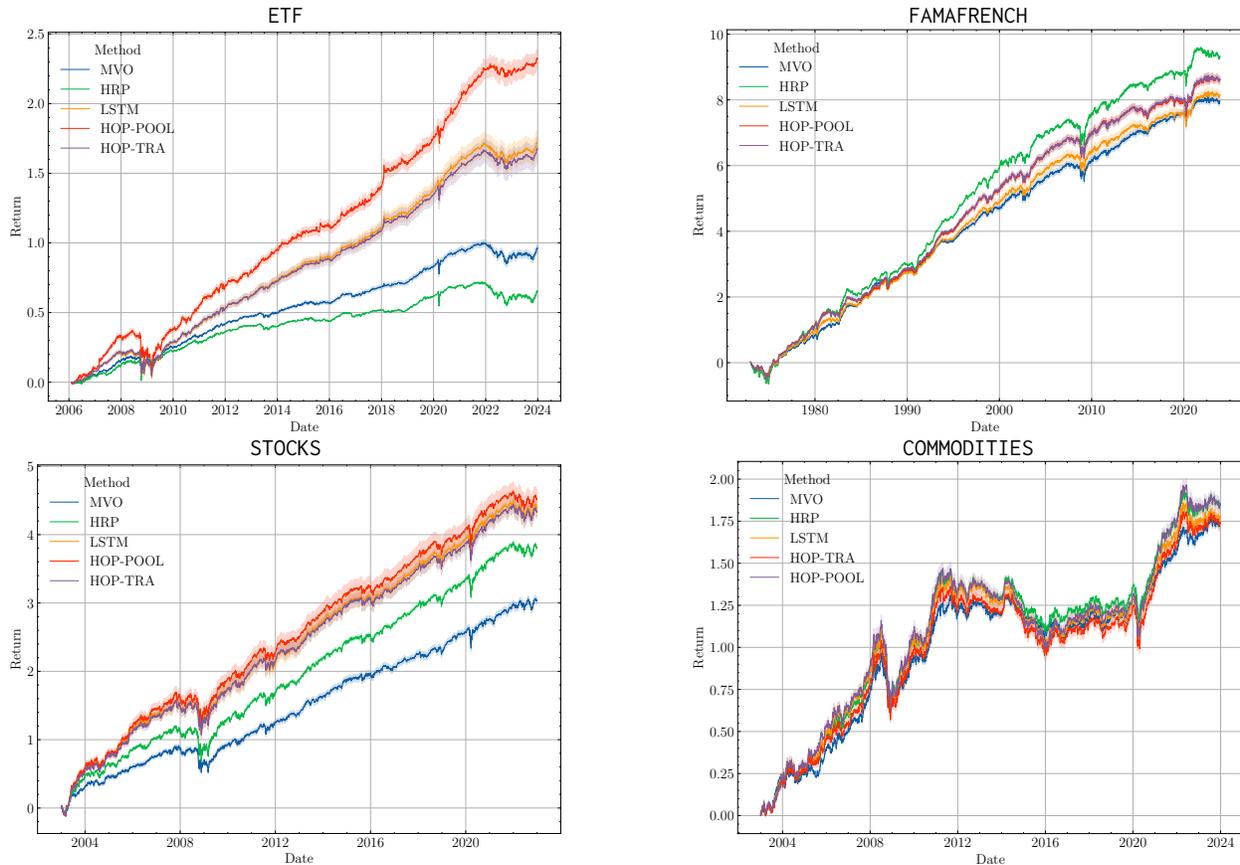


Figure 3: Long only portfolios. Cumulative returns over combinatorial purged cross-validation for the four datasets.

lower Sharpe ratio, this happens at both better average and maximum drawdown. Moreover in the hardest dataset (COMMODITIES) we have statistically higher Sharpe ratio for the Hopfield Encoder with respect to both LSTM and Hopfield Pooling. In future, we may perform a large hyper-parameters search in order to evaluate how the results are influenced by an optimal selection of parameters. However, we are confident that an accurate hyperparameters' search could yield even better results in downstream tasks.

Combinatorial purged cross-validation and statistical analysis. This research marks one of the few attempts to employ combinatorial purged cross-validation extensively in testing various backtesting paths, particularly with the application of deep-learning models. Using this method offers the benefit of providing a more reliable assessment of a portfolio optimization technique across numerous historical scenarios and helps reduce the likelihood of false discoveries. Consequently, we strongly advise against using the standard walk-forward cross-validation in finance, as it tends to test a single path with a pronounced over-fitting effect. This approach may compromise the accuracy and robustness of the results. Statistically accurate analyses on an even larger number of backtesting paths can provide a better view on the future properties of all allocation methods.

5 Conclusions

In the context of portfolio optimization, our research aims to harness the capabilities of deep learning to enhance the decision-making process of asset allocation.

The application of deep learning in this context aligns with the broader trend in the financial industry to integrate advanced technologies for more sophisticated portfolio strategies. Our exploration of the applicability of modern Hopfield networks in the realm of portfolio optimization signifies a departure from conventional approaches. This renewed interest is driven by the potential advantages that Hopfield networks may offer in handling the inherent complexities of optimization tasks, providing an alternative avenue for practitioners to explore. The two methods proposed in this work - Hopfield Pooling and Hopfield Encoder, demonstrates better or comparable performance against other traditional and deep-learning based methods in most of the tasks analyzed.

In this study, our focus has been solely on analyzing returns data. However, the deep architectures we have introduced can be effectively applied to model unconventional data sources, provided they are appropriately processed. Examples of alternative data sources include Environmental, Social, and Governance (ESG) factors or macroeconomic sentiment indicators. Traditional methods based on convex optimization encounter challenges when incorporating

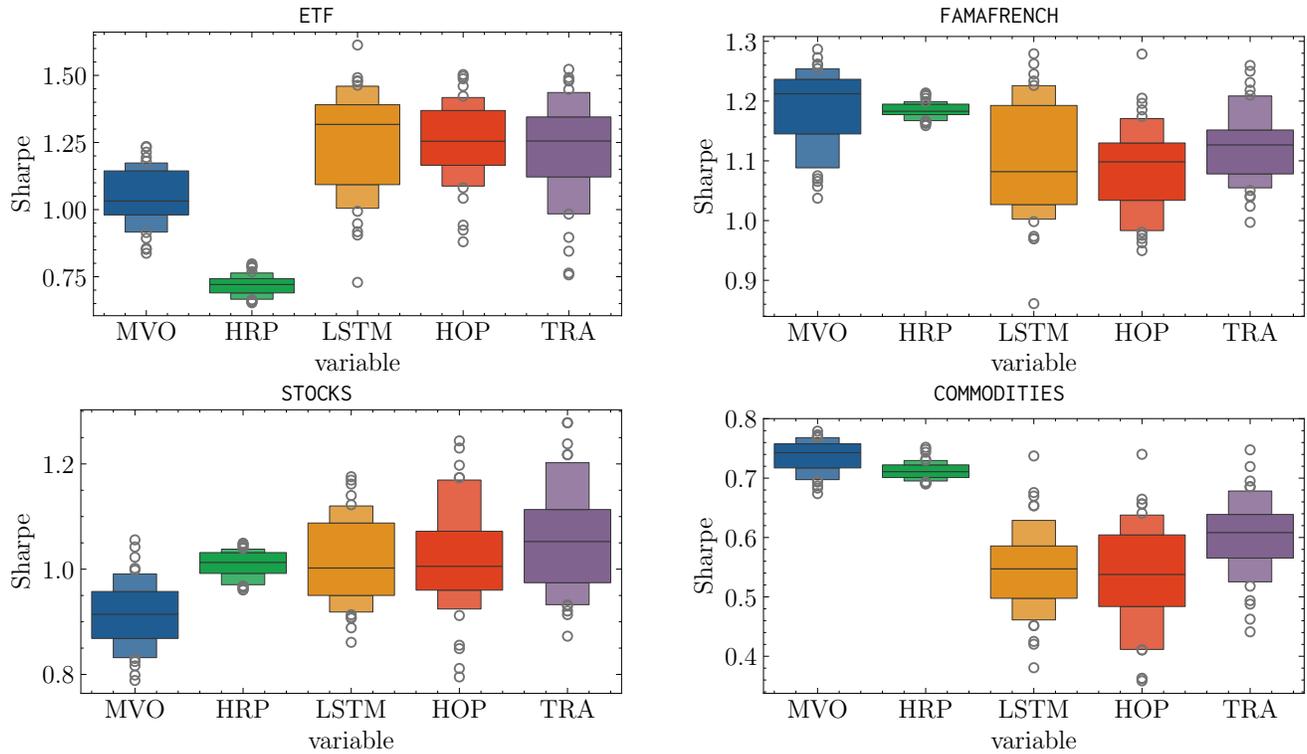


Figure 4: Distribution of Sharpe ratios over combinatorial purged cross-validation for the four datasets in the long-only case.

	EW	MVO	HRP	LSTM	HOP-POOL	HOP-TRA
ETF						
Mean	0.227	0.054 ± 0.007	0.036 ± 0.002	0.096 ± 0.028	0.130 ± 0.022	0.094 ± 0.029
Sharpe	0.827	1.047 ± 0.112	0.719 ± 0.041	1.248 ± 0.205	1.252 ± 0.163	1.219 ± 0.203
Sortino	1.404	1.443 ± 0.180	0.964 ± 0.055	1.847 ± 0.318	1.919 ± 0.247	1.799 ± 0.331
Avg. DD	0.070	0.016 ± 0.001	0.019 ± 0.001	0.020 ± 0.006	0.026 ± 0.009	0.020 ± 0.008
FAMA FRENCH						
Mean	0.188	0.156 ± 0.008	0.183 ± 0.003	0.160 ± 0.014	0.169 ± 0.011	0.169 ± 0.010
Sharpe	1.157	1.190 ± 0.068	1.184 ± 0.014	1.099 ± 0.098	1.087 ± 0.077	1.122 ± 0.064
Sortino	1.557	1.620 ± 0.085	1.595 ± 0.017	1.501 ± 0.126	1.481 ± 0.099	1.526 ± 0.082
Avg. DD	0.078	0.045 ± 0.003	0.073 ± 0.003	0.057 ± 0.015	0.062 ± 0.014	0.058 ± 0.009
STOCKS						
Mean	0.218	0.152 ± 0.013	0.191 ± 0.008	0.220 ± 0.035	0.227 ± 0.039	0.217 ± 0.027
Sharpe	1.019	0.915 ± 0.067	1.010 ± 0.027	1.014 ± 0.087	1.018 ± 0.109	1.059 ± 0.107
Sortino	1.407	1.275 ± 0.093	1.400 ± 0.036	1.420 ± 0.125	1.424 ± 0.154	1.474 ± 0.151
Avg. DD	0.049	0.035 ± 0.005	0.039 ± 0.002	0.050 ± 0.009	0.053 ± 0.011	0.045 ± 0.007
COMMODITIES						
Mean	0.098	0.083 ± 0.003	0.088 ± 0.003	0.084 ± 0.012	0.083 ± 0.009	0.088 ± 0.008
Sharpe	0.658	0.736 ± 0.029	0.713 ± 0.016	0.545 ± 0.076	0.600 ± 0.069	0.536 ± 0.096
Sortino	0.911	1.009 ± 0.042	0.978 ± 0.022	0.755 ± 0.105	0.747 ± 0.129	0.827 ± 0.095
Avg. DD	0.155	0.084 ± 0.013	0.115 ± 0.014	0.155 ± 0.054	0.139 ± 0.036	0.181 ± 0.075

Table 2: Results of combinatorial purged cross validation. HOP-POOL and HOP-TRA have very similar performance in terms of Sharpe ratio.

Table 3: Compact Letters Display notation for the Sharpe ratio after Tukey HSD test on long-only results. Each column indicates to what statistically similar group the specific method belongs. The full Tukey pairwise test is shown in the Supplementary Materials Section.

	ETF	STOCKS	FAMAFRENCH	COMMODITIES
MVO	c	b	b	b
HRP	b	a	b	b
LSTM	a	a	a	c
HOP-POOL	a	a	a	c
HOP-TRA	a	a	a	a

such diverse data sources. In contrast, deep models only necessitate minor architectural adjustments to accommodate them.

It is essential to note that the methods outlined in this work do not claim to be a definitive solution for portfolio optimization tasks; rather, they represent an additional tool to consider in addressing complex challenges posed to asset managers. Indeed, the primary value proposition of our work lies in the exploration of how deep-learning based methods, specifically Hopfield networks, can augment traditional decision-making processes, offering an alternative perspective.

As a future research we would like to include the ability to add specific constraints into the model optimization. We believe that the ability to handle complex constraints like cardinality constraints, or minimum risk budget could help the quantitative analyst to better align the raw algorithm results with the markets views imposed by the portfolio managers. In this study, our primary focus was on cross-validation metrics. However, we did not delve into exploring how fine-tuning hyper-parameters could potentially improve allocations, nor did we examine the impact of assets turnover and management fees on the final strategy. These areas warrant further investigation, which we leave to subsequent studies.

Broader Impact

This study highlights new areas where modern Hopfield networks can be beneficial, particularly in finance, potentially opening to the possibility of including new kind of data apart from the traditional asset returns, such as NLP embeddings or generally temporal trends and market sentiments. We demonstrate how these networks can enhance portfolio optimization, providing a valuable tool for financial practitioners.

Our findings show that Hopfield networks can improve the efficiency and robustness of portfolio management. They represent a powerful addition to the existing optimization techniques, helping professionals handle complex financial data more effectively. Overall, this research expands the applications of Hopfield networks and offers practical insights for their use in financial optimization.

References

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
- [2] Stephen Boyd, Kasper Johansson, Ronald Kahn, Philipp Schiele, and Thomas Schmelzer. 2024. Markowitz Portfolio Construction at Seventy. *arXiv preprint arXiv:2401.05080* (2024).
- [3] Alexei Chekhlov, Stanislav Uryasev, and Michael Zabarankin. 2004. Portfolio optimization with drawdown constraints. In *Supply chain and finance*. World Scientific, 209–228.
- [4] James Ming Chen and James Ming Chen. 2016. *Modern portfolio theory*. Springer.
- [5] Marcos Lopez De Prado. 2018. *Advances in financial machine learning*. John Wiley & Sons.
- [6] Mete Demircigil, Judith Heusel, Matthias Löwe, Sven Uppgang, and Franck Vermet. 2017. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics* 168 (2017), 288–299.
- [7] Hossein Gholamalizadeh and Hossein Khosravi. 2020. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485* (2020).
- [8] Guangliang He and Robert Litterman. 2002. The intuition behind Black-Litterman model portfolios. Available at SSRN 334304 (2002).
- [9] Kaiping He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
- [10] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).
- [11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [12] John J Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* 79, 8 (1982), 2554–2558.
- [13] John J Hopfield. 2007. Hopfield network. *Scholarpedia* 2, 5 (2007), 1977.
- [14] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. 2019. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321* (2019).
- [15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [16] Damian Kisiel and Denise Gorse. 2022. Portfolio transformer for attention-based asset allocation. In *International Conference on Artificial Intelligence and Soft Computing*. Springer, 61–71.
- [17] Dmitry Krotov. 2023. A new frontier for Hopfield networks. *Nature Reviews Physics* 5, 7 (2023), 366–367.
- [18] Dmitry Krotov and John Hopfield. 2020. Large associative memory problem in neurobiology and machine learning. *arXiv preprint arXiv:2008.06996* (2020).
- [19] Dmitry Krotov and John J Hopfield. 2016. Dense associative memory for pattern recognition. *Advances in neural information processing systems* 29 (2016).
- [20] Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam. (2018).
- [21] Harry Markowitz. 1952. Portfolio selection. *The Journal of Finance* 7, 1 (1952), 77–91.
- [22] Attilio Meucci. 2010. Fully flexible views: Theory and practice. *arXiv preprint arXiv:1012.2848* (2010).
- [23] Attilio Meucci. 2011. Robust Bayesian Allocation. Available at SSRN 681553 (2011).
- [24] Richard O Michaud and Robert Michaud. 2007. Estimation error and portfolio optimization: a resampling solution. Available at SSRN 2658657 (2007).
- [25] Hans-Peter Piepho. 2004. An algorithm for a letter-based representation of all-pairwise comparisons. *Journal of Computational and Graphical Statistics* 13, 2 (2004), 456–466.
- [26] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. 2020. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217* (2020).
- [27] Thomas N Rollinger and Scott T Hoffman. 2013. Sortino: a ‘sharper’ ratio. *Chicago, Illinois: Red Rock Capital* (2013).
- [28] Stephen C Sexauer and Laurence B Siegel. 2024. Harry Markowitz and the philosopher’s stone. *Financial Analysts Journal* 80, 1 (2024), 1–11.
- [29] William F Sharpe. 1966. Mutual fund performance. *The Journal of business* 39, 1 (1966), 119–138.
- [30] Frank A Sortino and Lee N Price. 1994. Performance measurement in a downside risk framework. *the Journal of Investing* 3, 3 (1994), 59–64.
- [31] Ilya Sutskever. 2013. *Training recurrent neural networks*. University of Toronto Toronto, ON, Canada.
- [32] John W Tukey. 1949. Comparing individual means in the analysis of variance. *Biometrics* (1949), 99–114.
- [33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [34] Chao Zhang, Zihao Zhang, Mihai Cucuringu, and Stefan Zohren. 2021. A Universal End-to-End Approach to Portfolio Optimization via Deep Learning. <https://api.semanticscholar.org/CorpusID:244270204>
- [35] Zihao Zhang, Stefan Zohren, and Stephen Roberts. 2020. Deep learning for portfolio optimization. *The Journal of Financial Data Science* (2020).