

A Fully Analog Pipeline for Portfolio Optimization

James S. Cummins* and Natalia G. Berloff

*Department of Applied Mathematics and Theoretical Physics,
University of Cambridge, Wilberforce Road, Cambridge CB3 0WA, United Kingdom*

(Dated: November 12, 2024)

Portfolio optimization is a ubiquitous problem in financial mathematics that relies on accurate estimates of covariance matrices for asset returns. However, estimates of pairwise covariance could be better and calculating time-sensitive optimal portfolios is energy-intensive for digital computers. We present an energy-efficient, fast, and fully analog pipeline for solving portfolio optimization problems that overcomes these limitations. The analog paradigm leverages the fundamental principles of physics to recover accurate optimal portfolios in a two-step process. Firstly, we utilize equilibrium propagation, an analog alternative to backpropagation, to train linear autoencoder neural networks to calculate low-rank covariance matrices. Then, analog continuous Hopfield networks output the minimum variance portfolio for a given desired expected return. The entire efficient frontier may then be recovered, and an optimal portfolio selected based on risk appetite.

I. INTRODUCTION

Portfolio optimization involves creating an investment portfolio that balances risk and return. The objective is to allocate assets optimally to maximize expected returns while minimizing risk. Naturally, this problem is of great interest to financial organizations and is pivotal in risk management. However, the problem, formulated by Markowitz's mean-variance model [1], must be solved in practice. Namely, it is well known that estimates of pairwise covariance between assets are notoriously poor [2]. A large financial company may have hundreds of thousands of assets n covering equities, bonds, derivatives, and more, but with only a small sample of observations over the desired timescale. The samples tend to include significant amounts of noise, distorting the underlying relationships between the assets. The symmetric covariance matrix has $n(n+1)/2$ total unique terms: $n(n-1)/2$ pairwise correlations and n variances. Hence, the number of unique terms behaves as $\mathcal{O}(n^2)$, which leads to significant potential for an ill-conditioned covariance matrix [3]. To overcome this issue, factor models were introduced that vastly reduce the dimensionality, and thus the number of numerical estimates required [4]. Factor methods produce low-rank covariance matrices that retain only the largest eigenvalues and discard small eigenvalues associated with noise. Despite this development, the computation of optimal portfolios remains energy-intensive as the efficient frontier is mapped out in n dimensions. In high-frequency trading, this becomes a time-sensitive computation as assets are purchased and sold on microsecond timescales, and portfolios must be regularly rebalanced not to exceed risk appetites. Much attention has been focused on portfolio optimization in the high-frequency domain [5–7], including the use of evolutionary algorithms to update efficient frontiers [8]. By using such fundamental principles as minimizing entropy, energy,

and dissipation [9], or, perhaps, incorporating quantum phenomena like superposition and entanglement [10], we can advance and surpass the classical computations of these problems. At the forefront of this drive to alternate architectures is the integration of analog, physics-based algorithms and hardware, which involve translating complex optimization problems into universal spin Hamiltonians [11–13]. Indeed, the mean-variance portfolio optimization framework can be encoded into a Hamiltonian's coupling strengths with the physical system recovering the Hamiltonian's ground state, which corresponds to the optimal portfolio solution [14, 15]. Efficient mapping from the original problem description to spin Hamiltonian enables the problem to remain manageable despite increasing complexity [16].

In Section II, we introduce the mean-variance optimization framework for calculating optimal portfolios. Then, in Section III, we show that analog continuous Hopfield networks can solve portfolio optimization problems by evolving to the minimum of an energy function that encodes the problem parameters. In Section IV, we address the issues of estimating pairwise covariance by introducing the low-rank approximation that relies on a low-dimensional latent variable representation. In Section V, we show that calculating such a representation can be done using linear autoencoder neural networks, and in Section VI how these networks can be trained on analog hardware using equilibrium propagation. Section VII brings everything together, starting with raw data observations and working through the entire analog pipeline.

II. MEAN-VARIANCE OPTIMIZATION

We define μ_i as the expected return of asset i , and $[\Sigma]_{ij} = \sigma_{ij} = \text{Cov}(i, j)$ as the covariance between assets i and j . The decision variables are w_i , the proportion of the total investment in asset i . For a universe of securities with n assets, the Markowitz mean-variance portfolio

* correspondence address: jsc95@cam.ac.uk

optimization problem is

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} \\ \text{s.t.} \quad & \boldsymbol{\mu}^T \mathbf{w} = R, \\ & \mathbf{1}^T \mathbf{w} = 1, \\ & 0 \leq w_i \leq 1, \end{aligned} \quad (1)$$

for $i = 1, \dots, n$, and the condition $w_i \geq 0$ prohibits shorting [14]. The variance $\mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}$ quantifies the portfolio risk for positive semidefinite matrix $\boldsymbol{\Sigma}$, while R is the desired expected return of the portfolio. $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are not known a priori and must be estimated from historical data. The efficient frontier is calculated by solving (1) for various R . The efficient frontier is the set of portfolios that minimize the risk for a given R . We illustrate a frontier in Fig. (1) for a toy model with $n = 2$ assets. It was recently suggested that portfolio optimization problems could be solved on analog spatial-photonics Ising machines for equal-weighted portfolios, that is, $w_i \in \{0, 1/q\}$ for q selected assets [15]. We go beyond this constraint by utilizing analog Hopfield networks and consider the quadratic continuous optimization problem (1). In Section III, we aim to recover the optimal asset weights \mathbf{w} , given known parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

III. CONTINUOUS HOPFIELD NETWORK

A continuous Hopfield network is a type of Hopfield neural network which has continuous states and dynamics [17]. It is an analog computational network for solving optimization problems. For a network of size n , the i -th network element at time t is described by a real input $x_i(t)$, and the network dynamics are governed by

$$\frac{dx_i}{dt} = -p(t)x_i + \sum_{j=1}^n J_{ij}v_j + m_i, \quad (2)$$

where $v_i = g(x_i)$ is a nonlinear activation function, $p(t)$ is an annealing parameter, m_i are the offset biases, and $J_{ij} = [\mathbf{J}]_{ij}$ are elements of the symmetric coupling matrix \mathbf{J} . Should $g(x)$ be a non-decreasing function, then the steady states of the continuous Hopfield network (2) are the minima of the Lyapunov function

$$E = p(t) \sum_{i=1}^n \int_0^{v_i} g^{-1}(x) dx - \frac{1}{2} \sum_{i,j=1}^n J_{ij}v_i v_j - \sum_{i=1}^n m_i v_i. \quad (3)$$

We choose the functional form of $g(x)$, such that when $p(t) \rightarrow 0$, the minima of E occur for $v_i \in [0, 1]$ and correspond to the minima of $-\mathbf{v}^T \mathbf{J} \mathbf{v}$. Therefore, by setting $\mathbf{J} = -\boldsymbol{\Sigma}$, we can minimize the variance $\mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}$ of problem (1). To satisfy the constraints in problem (1) we introduce Lagrange multiplier-like scalars λ_1 , λ_2 and seek to minimize the expression

$$H = \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} + \lambda_1 (\boldsymbol{\mu}^T \mathbf{w} - R)^2 + \lambda_2 (\mathbf{1}^T \mathbf{w} - 1)^2. \quad (4)$$

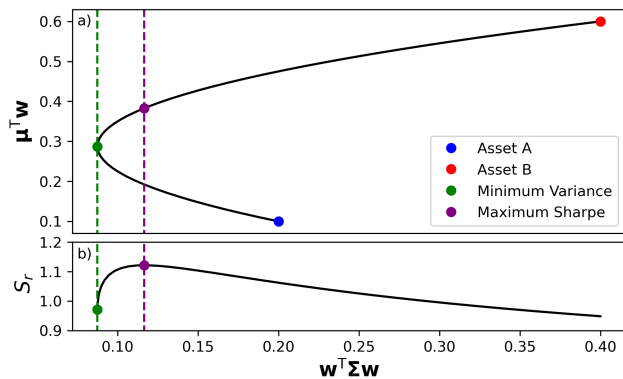


Figure 1. (a) The hyperbola in variance-return space for a portfolio of $n = 2$ assets A and B . The positively sloped portion of this hyperbola is the efficient frontier. The expected returns are $\mu_A = 0.1$ and $\mu_B = 0.6$. The (co)variances are $\sigma_{AA} = 0.2$, $\sigma_{BB} = 0.4$, and $\sigma_{AB} = \sigma_{BA} = -0.1$. The blue circle represents the portfolio consisting only of asset A , and the corresponding investment weights are $\mathbf{w} = [1, 0]^T$. Likewise, the red circle is the portfolio consisting only of asset B . The minimum variance portfolio, shown as a green circle, is the combination of weights \mathbf{w} that minimizes the total variance $\mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}$. The purple circle is the portfolio that maximizes the Sharpe ratio S_r . The Sharpe ratio is defined as $S_r = \boldsymbol{\mu}^T \mathbf{w} / \sqrt{\mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}}$. (b) The Sharpe ratio S_r for each portfolio in the efficient frontier. We now see that the purple circle is indeed the portfolio that maximizes the Sharpe ratio.

Therefore, after discarding constants, we seek to minimize

$$H = -\frac{1}{2} \mathbf{w}^T \mathbf{J} \mathbf{w} - \mathbf{m}^T \mathbf{w}, \quad (5)$$

where $\mathbf{J} = -2\boldsymbol{\Sigma} - 2\lambda_1 \boldsymbol{\mu} \boldsymbol{\mu}^T - 2\lambda_2 \mathbf{1} \mathbf{1}^T$, and $\mathbf{m} = 2R\lambda_1 \boldsymbol{\mu} + 2\lambda_2 \mathbf{1}$. Equation (5) can be directly encoded into the Hopfield network (2), and if required, \mathbf{m} can be absorbed into \mathbf{J} by introducing an additional auxiliary spin. The non-decreasing monotonic function $g(x)$ is chosen to be the logistic function $g(x) = 1/[1 - \exp(-x)]$ to limit possible values of v_i such that $0 \leq v_i \leq 1$. We illustrate the Hopfield network dynamics in Fig. (2) for a randomly generated covariance matrix $\boldsymbol{\Sigma}$ and expected return vector $\boldsymbol{\mu}$. The energy minimization properties of Hopfield networks make them particularly suitable for solving combinatorial optimization problems. Further extensions have been proposed to increase convergence to optimal states in challenging optimization problems. For example, the first-order Eq. (2) can be momentum-enhanced and replaced with a second-order equation leading to Microsoft's analog iterative machine [18] or Toshiba's bifurcation machine [19].

IV. LOW-RANK APPROXIMATION

We now focus on calculating a low-rank approximation of the covariance matrix, which will be used in (1). If

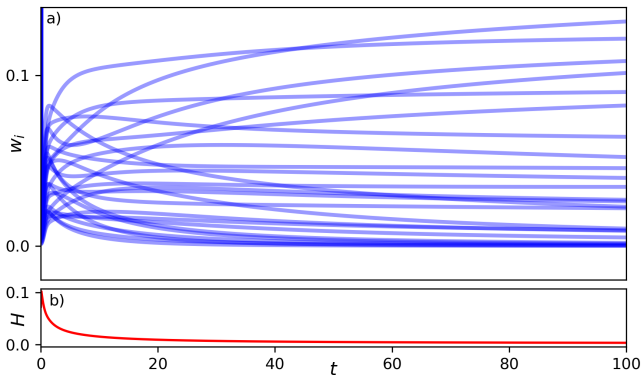


Figure 2. (a) Hopfield network dynamics for a portfolio of $n = 25$ assets with $R = \lambda_1 = \lambda_2 = 1$. The dynamical system evolves according to Eq. (2), which in turn minimizes Eq. (5). Each line represents one asset weight w_i . (b) The value of expression (5) during the network dynamics. Covariance matrix Σ and expected return vector μ are calculated from sampling $N = 10$ observations of returns \mathbf{x} from IID random normal variables $x_j \sim N(1, 1)$, where $j = 1, 2, \dots, N$. The low number of observations N results in a noisy positive semidefinite covariance matrix Σ whose pairwise entries σ_{ij} are nonzero. The externally controlled annealed parameter has form $p(t) = p_0(1 - t/T)$, where T is the total annealing period. Here, $p_0 = 0.01$ and $T = 100$.

$\mathbf{x}_i \in \mathbb{R}^n$ are the i -th sample of asset returns over N total samples, and we assume that $\mathbb{E}[\mathbf{x}] = \mathbf{0}$, then the sample covariance matrix is

$$\mathbf{S} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T. \quad (6)$$

When the number of samples N is of the same magnitude as n , then the sample covariance matrix usually suffers a large estimation error [2, 20]. Many low-rank factor analysis techniques exist to improve the covariance matrix estimate. Here, we consider asset returns \mathbf{x} as random variables that follow the model

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{e}, \quad (7)$$

where $\mathbf{x} \in \mathbb{R}^n$ is the observed data, $\mathbf{A} \in \mathbb{R}^{n \times r}$ is a factor loading matrix, $\mathbf{s} \in \mathbb{R}^r$ is the latent variable, and $\mathbf{e} \in \mathbb{R}^n$ is uncorrelated random noise, where $r \ll n$ [21]. Here, \mathbf{s} represents macroeconomic factors like the growth rate of the GDP, inflation, unemployment, etc. We assume that \mathbf{s} and \mathbf{e} are uncorrelated and that data samples are independent and identically distributed. The covariance matrix is then $\Sigma = \mathbb{E}[\mathbf{x}\mathbf{x}^T]$. This gives

$$\Sigma = \mathbf{A}\mathbb{E}[\mathbf{s}\mathbf{s}^T]\mathbf{A}^T + \mathbb{E}[\mathbf{e}\mathbf{e}^T] \quad (8)$$

$$= \mathbf{A}\mathbf{P}\mathbf{A}^T + \Psi, \quad (9)$$

where $\mathbf{P} \equiv \mathbb{E}[\mathbf{s}\mathbf{s}^T] \in \mathbb{R}^{r \times r}$ has $\text{rank}(\mathbf{P}) \leq r$, $\text{rank}(\mathbf{A}) \leq r$, and Ψ is a diagonal matrix containing the variance of

noise on its diagonal

$$\Psi = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & & 0 \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_n^2 \end{bmatrix}. \quad (10)$$

Since $\text{rank}(\mathbf{A}\mathbf{B}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}))$, then $\text{rank}(\mathbf{A}\mathbf{P}\mathbf{A}^T) \leq r$. Therefore, we have decomposed the covariance matrix Σ into a positive semidefinite low-rank matrix plus a positive semidefinite diagonal matrix. Defining $\mathbf{M} \equiv \mathbf{A}\mathbf{P}\mathbf{A}^T$, low-rank factor analysis concerns the estimation of \mathbf{M} and Ψ . To calculate \mathbf{M} and Ψ we solve the minimization problem

$$\begin{aligned} \min_{\mathbf{M}, \Psi} \quad & \|\mathbf{S} - \mathbf{M} - \Psi\|_F^2 \\ \text{s.t.} \quad & \text{rank}(\mathbf{M}) \leq r, \\ & \Sigma \geq 0, \end{aligned} \quad (11)$$

where $\|\cdot\|_F$ denotes the Frobenius norm [22]. A common classical procedure for calculating matrix \mathbf{M} in problem (11) using digital computers is given by the following steps:

1. Construct the singular value decomposition (SVD) of \mathbf{S} . Since \mathbf{S} is symmetric, its eigendecomposition is the same as the SVD, and we write $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where \mathbf{U} is the matrix of eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues.
2. Derive from $\mathbf{\Lambda}$ the matrix $\mathbf{\Lambda}_r$ formed by replacing with zeros the $n - r$ smallest eigenvalues on the diagonal of $\mathbf{\Lambda}$.
3. Compute and output $\mathbf{M} = \mathbf{U}\mathbf{\Lambda}_r\mathbf{U}^T$ as the rank- r approximation to \mathbf{S} .

Under the assumption $\mathbb{E}[\mathbf{x}] = \mathbf{0}$, the SVD method exactly replicates principal component analysis (PCA). The rank of \mathbf{M} is at most r : this follows from the fact that $\mathbf{\Lambda}_r$ has at most r non-zero values. Indeed, the Eckart-Young-Mirsky theorem proves that this procedure yields the matrix of rank less than or equal to r with the lowest possible Frobenius error [23]. The diagonal matrix is estimated as $\Psi = \text{diag}(\mathbf{S} - \mathbf{M})$, where $\text{diag}(\cdot)$ represents a diagonal matrix whose elements are $[\Psi]_{ii} = [\mathbf{S} - \mathbf{M}]_{ii}$ and $[\Psi]_{ij} = 0$ for $i \neq j$ [24]. In addition, we constrain $[\Psi]_{ii} \geq 0$, since the diagonal elements correspond to variances of the error variables. This guarantees that Σ is positive semidefinite. The eigendecomposition presented here becomes computationally expensive as the data size grows. Alternatively, autoencoders – particularly when implemented using stochastic gradient descent – can handle larger datasets and higher-dimensional data more efficiently than PCA [25]. Additionally, when integrating dimensionality reduction as part of a larger neural network framework, an autoencoder can be easily embedded within the pipeline, whereas PCA would need to be applied as a separate pre-processing step [26].

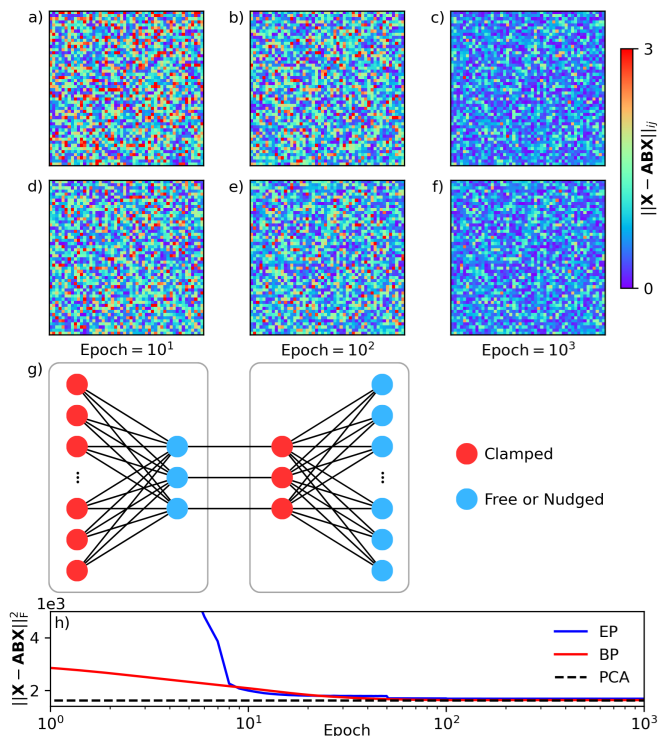


Figure 3. Training a linear autoencoder via (a)-(c) backpropagation (BP), and (d)-(f) with equilibrium propagation (EP). Input and output layers have size 50, while the single hidden layer has size 5. The networks are trained on 50 vectors $\mathbf{x}_1, \dots, \mathbf{x}_{50}$ of size 50 whose elements are randomly sampled from the normal distribution $N(0, 1)$. (a)-(f) illustrate the element-wise absolute difference between the 50×50 matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_{50}]$ and its reconstructed output \mathbf{ABX} at different epochs. (g) An illustrative example of the encoder/decoder Hopfield network structure trained with EP. (h) The overall network loss for BP and EP over epoch time. The black horizontal dashed line corresponds to the loss of the equivalent SVD/PCA method described in Section IV.

V. LINEAR AUTOENCODERS

A linear autoencoder is a classic neural network model for unsupervised learning that is trained to learn the identity function. The output and input layers have the same number of nodes, while the middle layer has a fewer nodes. It aims to approximate the input through learning linear encodings and decodings between input and latent space. The encoder $\mathbf{B} \in \mathbb{R}^{r \times n}$ maps input $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{n \times N}$ into a low-dimensional latent space $[\mathbf{s}_1, \dots, \mathbf{s}_N]$, and the decoder $\mathbf{A} \in \mathbb{R}^{n \times r}$ maps $[\mathbf{s}_1, \dots, \mathbf{s}_N]$ back to the original representation \mathbf{X} . We therefore recover the same model as in Eq. (7), and training the linear autoencoder becomes the minimization problem [27]

$$\min_{\mathbf{A}, \mathbf{B}} \|\mathbf{X} - \mathbf{ABX}\|_F^2. \quad (12)$$

We do not explicitly express the learnable biases in the network as these may be absorbed into the encoder \mathbf{B} and

decoder \mathbf{A} by introducing an auxiliary row into \mathbf{X} that is permanently clamped to values of 1. We illustrate the training of a linear autoencoder in Fig. (3)(a)-(c) with the backpropagation method and compare it to the PCA in Fig. (3)(h). A linear autoencoder is related to the PCA. Indeed, under mild nondegeneracy conditions, any \mathbf{A} at a local minimizer recovers the top rank- r eigenspace of \mathbf{XX}^T [28]. However, unlike the actual PCA, the coordinates of the output of the middle layer in the network are correlated and are not sorted in descending order of variance [29]. Autoencoder neural networks typically use backpropagation to train the weights. However, backpropagation is energy-intensive and not biologically plausible.

VI. EQUILIBRIUM PROPAGATION

On dedicated analog hardware, equilibrium propagation is an energy-efficient alternative to backpropagation [30]. Therefore, in the supervised learning setting studied here, it may be used to train the weights of a linear autoencoder. Equilibrium propagation is an energy-based model because it relies on the concept of energy minimization to learn and make predictions. We consider the continuous Lyapunov function (3) with $p(t) = 1$ and $m_i = 0$ for all i , where here the symmetric coupling weights J_{ij} are to be learned, and nonlinear activation function $g(x)$ need not be the same as in Section III. Neurons x_i are split in three sets: the input neurons, which are always clamped, the hidden neurons, and the output neurons. The discrepancy between the desired output \mathbf{y} and the realized output $\hat{\mathbf{x}}$ is measured by the cost function

$$C = \frac{1}{2} \|\mathbf{y} - \hat{\mathbf{x}}\|_2^2, \quad (13)$$

which forms part of the total energy function $F = E + \beta C$. The clamping factor $\beta \geq 0$ is a real-valued scalar that allows the output neurons to be weakly clamped [31]. The continuous-time dynamical system evolves according to the differential equation of motion

$$\frac{dx_i}{dt} = -\frac{\partial F}{\partial x_i} = -\frac{\partial E}{\partial x_i} - \beta \frac{\partial C}{\partial x_i}, \quad (14)$$

which is formed of two parts. The first is the internal force induced by the internal Hopfield energy, given by Eq. (2) for all i , and the second, the external force, is induced by the cost function C as

$$-\beta \frac{\partial C}{\partial x_i} = \beta (y_i - x_i), \quad i \in \mathcal{Y}, \quad (15)$$

for nodes in the output layer \mathcal{Y} . Equilibrium propagation has two modes: the free phase and the weakly clamped phase. In the free phase $\beta = 0$ and only the inputs are clamped. The network then converges to a fixed point \mathbf{x}^* and the output units are read out. In the weakly clamped

phase $\beta > 0$, which induces an external force that acts on the output units as in Eq. (15). This force nudges the outputs from their fixed point values in the direction of the target values y_i . This perturbation propagates among the hidden neurons before a new fixed point \mathbf{x}_β^* is found. Then, another weakly clamped phase is executed, this time with $\beta \rightarrow -\beta$ leading to the weakly clamped equilibrium $\mathbf{x}_{-\beta}^*$. It was shown that the weakly clamped phase implements the propagation of error derivatives with respect to the synaptic weights [31]. In the limit $\beta \rightarrow 0$, the update rule is

$$\Delta J_{ij} \propto \frac{1}{\beta} \left(\left. \frac{\partial F}{\partial J_{ij}} \right|_{\mathbf{x}_\beta^*} - \left. \frac{\partial F}{\partial J_{ij}} \right|_{\mathbf{x}_{-\beta}^*} \right), \quad (16)$$

which is a second-order approximation to the standard backpropagation derivative [32]. The process is iterated, at each step updating the weights J_{ij} to minimize the loss function C . We choose activation function

$$g(x) = \begin{cases} x & \text{if } |x| \leq c \\ c \cdot \text{sgn}(x) & \text{otherwise,} \end{cases} \quad (17)$$

with constant c , so that under the condition that $|x_i| \leq c$ for all i , Eq. (2) is linear and can thus represent a linear autoencoder. In this case, the output $\hat{\mathbf{x}}$ of Eq. (2) is then the solution to the linear differential equation $d\mathbf{x}/dt = (\mathbf{J} - \mathbf{I})\mathbf{x}$, and therefore

$$\hat{\mathbf{x}} = \lim_{t \rightarrow \infty} \mathbf{x}(t) = \lim_{t \rightarrow \infty} \exp\{(\mathbf{J} - \mathbf{I})t\} \mathbf{x}(0). \quad (18)$$

The constant c in Eq. (17) is chosen to be large enough such that after training, all neurons obey $|x_i| \leq c$, and we can associate the Hopfield network as a linear autoencoder. To achieve a steady state in Eq. (18), at least one eigenvalue of $\mathbf{J} - \mathbf{I}$ should be zero, with all others having a negative real part.

Proposition. *We state, with proof given in Ref. [28], that for any fixed $n \times r$ matrix \mathbf{A} , Eq. (12) attains its minimum for $\mathbf{B} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$.*

Lemma. *The $n \times n$ matrix $\mathbf{J} - \mathbf{I}$, where $\mathbf{J} = \mathbf{A}\mathbf{B}$, has at least one zero eigenvalue, with all others having negative real part.*

Proof. $\mathbf{J} = \mathbf{A}\mathbf{B} = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$, and therefore

$$\mathbf{J}^2 = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \quad (19)$$

$$= \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T, \quad (20)$$

which shows that \mathbf{J} is idempotent, that is $\mathbf{J}^2 = \mathbf{J}$. It follows that \mathbf{J} is a projection operator on the column space $C(\mathbf{J})$ along its null space $N(\mathbf{J})$. The n eigenvalues λ_i of \mathbf{J} are either 0 or 1: $\lambda_i \mathbf{x}_i = \mathbf{J}\mathbf{x}_i = \mathbf{J}^2 \mathbf{x}_i = \lambda_i \mathbf{J}\mathbf{x}_i = \lambda_i^2 \mathbf{x}_i$, which implies $\lambda_i \in \{0, 1\}$. By construction, \mathbf{J} has rank at most r , and therefore there are at least $n - r$ zero eigenvalues. It follows that there are between 1 and r nonzero eigenvalues of \mathbf{J} , which must have value $\lambda_i = 1$.

Since $\mathbf{J} - \mathbf{I}$ has eigenvalues $\mu_i = \lambda_i - 1$, then $\mu_i \in \{-1, 0\}$. Therefore, $\mathbf{J} - \mathbf{I}$ has between 1 and r zero eigenvalues, with all others being equal to -1 .

The Lemma guarantees that should equilibrium propagation learn the weights that minimize Eq. (12), the corresponding Hopfield network will converge to a steady state. Yet, during training, this will, in general, not be the case, and positive eigenvalues of $\mathbf{J} - \mathbf{I}$ will produce exponential growth in Eq. (18). However, Eq. (18) only holds in the linear regime of the activation function (17). Exponential growth is prevented by the symmetric clipping incorporated into the nonlinear activation function $g(x)$ for neurons with $|x_i| > c$.

In the linear regime, the overall network dynamics is represented by the square matrix $\lim_{t \rightarrow \infty} \exp\{(\mathbf{J} - \mathbf{I})t\}$, which for linear autoencoders we seek to decompose into its non-square constituent parts: encoder \mathbf{B} and decoder \mathbf{A} . We achieve this by treating the encoder and decoder as separate Hopfield networks, as shown in Fig. (3)(g), each with their own energy function. The encoder settles into an equilibrium representing the latent vector \mathbf{s} without taking into account the decoder. \mathbf{s} is then used as a fixed input to the decoder which then settles into its own equilibrium. The decoder then undergoes the weakly clamped phases, and its weights are updated according to Eq. (16). The encoder weights also need to be optimized to lower the reconstruction loss at the decoder output, which is achieved by setting

$$\frac{\partial C}{\partial x_i} = \lim_{\beta \rightarrow 0} \frac{1}{2\beta} \left(\left. \frac{\partial F}{\partial x_i} \right|_{\mathbf{x}_\beta^{(\text{dec})}} - \left. \frac{\partial F}{\partial x_i} \right|_{\mathbf{x}_{-\beta}^{(\text{dec})}} \right), \quad i \in \mathcal{Y}, \quad (21)$$

in Eq. (14), where $\mathbf{x}_\beta^{(\text{dec})}$ is the weakly clamped decoder equilibrium state, and Eq. (21) only pertains to neurons in the encoder output layer \mathcal{Y} . Equation (21) follows from the fact that it can be shown that equilibrium propagation also allows for finding the gradient of the loss with respect to the input [33]. We note that \mathbf{J} , which contains the couplings of the continuous Hopfield network, is now a $(n + r) \times (n + r)$ matrix on account of the number of nodes in the encoder and decoder networks. Nonetheless, the factor loading matrix \mathbf{A} can be recovered as the $n \times r$ block corresponding to the nodes of the decoder output layer. The equilibrium propagation training procedure is illustrated in Fig. (3)(d)-(f) and compared to backpropagation and the PCA in Fig. (3)(h).

VII. RESULTS

We present our results in an order consistent with the portfolio optimization process a company or institution may take. We start with raw data samples and construct a low-rank covariance matrix approximating the true covariance matrix. This is a common procedure because the number of observations is often smaller than the number of assets, leading to significant noise that can distort

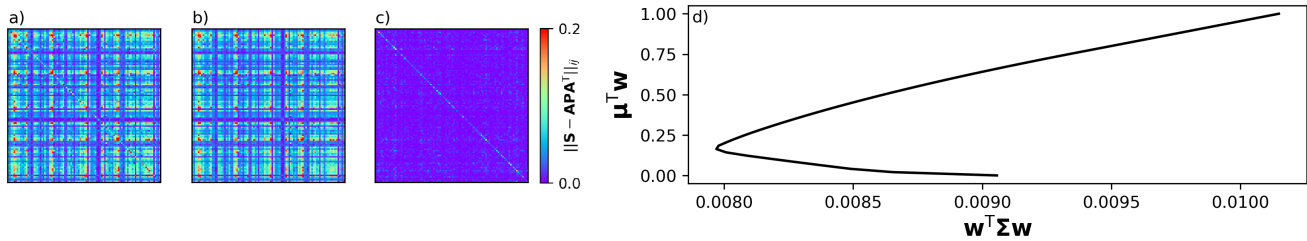


Figure 4. (a) The sample covariance matrix \mathbf{S} for $n = 100$ financial stocks selected from the S&P 500 index. \mathbf{S} is calculated from Eq. (6), with $N = 50$ time series samples. (b) The $r = 10$ low-rank approximation $\mathbf{A}\mathbf{P}\mathbf{A}^T$ of the covariance matrix, as calculated by training a continuous Hopfield network via equilibrium propagation. (c) The element-wise absolute difference between the sample covariance matrix and its low-rank approximation. (d) The hyperbola in variance-return space for possible portfolios. Each point along the hyperbola is calculated by solving (1) for a specific return value R using Eq. (2).

the underlying relationships between the assets. Next, we calculate the efficient frontier, which produces optimal portfolios for each level of desired expected return R .

We begin by collecting real data samples $\mathbf{x}_i \in \mathbb{R}^n$ for $i = 1, 2, \dots, N$ from stock returns of a selection of $n = 100$ stocks in the S&P 500 index. We restrict ourselves to only $N = 50$ observations such that the sample covariance matrix has a tendency to contain significant noise. Two continuous Hopfield networks, structured as the encoder and decoder parts of a linear autoencoder, are trained using equilibrium propagation. The latent variables $[\mathbf{s}_1, \dots, \mathbf{s}_N]$ are calculated as the subset \mathcal{Y} of steady-state solutions of the encoder network, while the factor loading matrix \mathbf{A} is the $n \times r$ block of the decoder matrix representation $\lim_{t \rightarrow \infty} \exp\{(\mathbf{J} - \mathbf{I})t\}$ corresponding to its output layer \mathcal{Y} . In practice, we cannot take the limit to infinity, and instead, we use a suitably large value of t such that $\exp\{(\mathbf{J} - \mathbf{I})t\}$ changes minimally from t to $t + 1$. The low-rank approximation to the covariance matrix is calculated as $\mathbf{A}\mathbf{P}\mathbf{A}^T$, where $\mathbf{P} = \mathbb{E}[\mathbf{s}\mathbf{s}^T]$. We depict the full-rank sample covariance matrix and the equilibrium propagation-based low-rank approximation in Figs. (4)(a) and (b) respectively. Figure (4)(c) then illustrates the element-wise absolute difference between these two covariance matrices. The low-rank approximation is plugged into (1) and solved for the portfolio weights \mathbf{w} using the continuous Hopfield network of Eq. (2). We minimize the portfolio variance subject to the constraint $\mu^T \mathbf{w} = R$ for incremental values of R . In Fig. (4)(d), we plot the corresponding variances and returns for range $R = [0, 1]$. The efficient frontier is identified, and an optimal portfolio can be selected based on risk appetite. In particular, the minimum variance and maximum Sharpe ratio portfolios can be established.

Analog Hopfield networks can be implemented as electronic circuits [34] and photonic neural networks [35]. Photonic systems operate on picosecond to femtosecond timescales as high bandwidth signals flow through a single optical waveguide. Consequently, such implementations can have dense connectivity while maintaining fast convergence times. However, physical analog platforms are subject to noise sensitivity, thermal effects, and non-

idealities in circuit components which can degrade performance. In addition, real-world portfolio optimization problems often involve complex constraints such as transaction costs, market liquidity, regulatory requirements, cardinality constraints, and tax considerations. While some of these can be readily incorporated into the objective function (5), for example, an ℓ^1 -norm can enforce sparsity to satisfy a cardinality constraint, others take more complex forms. For instance, a hybrid approach that combines analog Hopfield networks with digital computing could be explored.

VIII. CONCLUSIONS

This paper introduces a fully analog pipeline for portfolio optimization problems. Starting with raw data samples, the proposed pipeline leverages the energy-efficient analog operation of continuous Hopfield networks to calculate optimal portfolio weights. The analog pipeline distinguishes itself from traditional digital methods by its speed and scalability, with applications in time-sensitive domains such as high-frequency trading. At the heart of the pipeline are continuous Hopfield networks, used in two separate applications: autoencoder neural networks and minimum variance portfolios.

The autoencoder network yields the latent variables and factor loading matrix, which are then used to calculate a low-rank approximation of the covariance matrix. After that, a Hopfield network applies the quadratic mean-variance model to determine optimal portfolio weights. By shifting to analog architectures, we reduce the reliance on binary logic operations typical of digital systems, paving the way for a more energy-efficient approach to computation. This efficiency can reduce power consumption in data centers and other computing environments, addressing the growing energy demands of digital computing. Specifically, companies can reduce their energy consumption while optimizing large portfolios as part of their risk management processes.

ACKNOWLEDGMENTS

J.S.C. acknowledges the PhD support from the EP-SRC. N.G.B. acknowledges support from HORIZON EIC-2022-PATHFINDERCHALLENGES-01 HEISING-BERG Project 101114978 and Weizmann-UK Make Connection Grant 142568.

-
- [1] H. Markowitz, *The Journal of Finance* **7**, 77 (1952).
- [2] O. Ledoit and M. Wolf, *Journal of Financial Econometrics* **20**, 187 (2022).
- [3] J. M. Tabcart, S. L. Dance, A. S. Lawless, N. K. Nichols, and J. A. Waller, *Tellus A: Dynamic Meteorology and Oceanography* **72**, 1 (2020).
- [4] J. Bai and S. Ng, *Econometrica* **70**, 191 (2002).
- [5] Q. Liu, *Journal of Applied Econometrics* **24**, 560 (2009).
- [6] N. Goumatianos, I. Christou, and P. Lindgren, *Procedia Economics and Finance* **5**, 298 (2013).
- [7] F. A. Ziegelmann, B. Borges, and J. F. Caldeira, *Brazilian Review of Econometrics* **35**, 23 (2015).
- [8] P. Filipiak and P. Lipinski, in *Applications of Evolutionary Computation: 20th European Conference, Proceedings, Part I 20* (Springer, 2017) pp. 34–50.
- [9] S. K. Vadlamani, T. P. Xiao, and E. Yablonovitch, *Proceedings of the National Academy of Sciences* **117**, 26639 (2020).
- [10] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, arXiv preprint quant-ph/0001106 (2000).
- [11] A. Lucas, *Frontiers in Physics* **2**, 5 (2014).
- [12] N. G. Berloff, M. Silva, K. Kalinin, A. Askitopoulos, J. D. Töpfer, P. Cilibrizzi, W. Langbein, and P. G. Lagoudakis, *Nature Materials* (2017).
- [13] K. P. Kalinin, A. Amo, J. Bloch, and N. G. Berloff, *Nanophotonics* **9**, 4127 (2020).
- [14] J. E. Beasley, in *Theory Driven by Influential Applications* (Informs, 2013) pp. 201–221.
- [15] R. Z. Wang, J. S. Cummins, M. Syed, N. Stroeve, G. Pastras, J. Sakellariou, S. Tsintzos, A. Askitopoulos, D. Veraldi, M. C. Strinati, *et al.*, arXiv preprint arXiv:2406.01400 (2024).
- [16] G. De las Cuevas and T. S. Cubitt, *Science* **351**, 1180 (2016).
- [17] J. J. Hopfield, *Proceedings of the National Academy of Sciences* **81**, 3088 (1984).
- [18] K. Kalinin, G. Mourgiyas-Alexandris, H. Ballani, N. G. Berloff, J. H. Clegg, D. Cletheroe, C. Gkantsidis, I. Haller, V. Lyutsarev, F. Parmigiani, L. Pickup, *et al.*, arXiv preprint arXiv:2304.12594 (2023).
- [19] H. Goto, *Scientific Reports* **6**, 1 (2016).
- [20] R. Zhou, J. Ying, and D. P. Palomar, *IEEE Transactions on Signal Processing* **70**, 4020 (2022).
- [21] P. Stoica and P. Babu, *IEEE Transactions on Signal Processing* **71**, 1699 (2023).
- [22] C. D. Manning, *Introduction to Information Retrieval* (Syngress Publishing, 2008).
- [23] C. Eckart and G. Young, *Psychometrika* **1**, 211 (1936).
- [24] D. Bertsimas, M. S. Copenhaver, and R. Mazumder, *Journal of Machine Learning Research* **18**, 1 (2017).
- [25] D. P. Kingma and M. Welling, arXiv preprint arXiv:1312.6114 (2013).
- [26] Y. LeCun, Y. Bengio, and G. Hinton, *Nature* **521**, 436 (2015).
- [27] T. Li, R. Mehta, Z. Qian, and J. Sun, in *ICML Workshop on Uncertainty and Robustness in Deep Learning* (2020).
- [28] P. Baldi and K. Hornik, *Neural Networks* **2**, 53 (1989).
- [29] E. Plaut, arXiv preprint arXiv:1804.10253 (2018).
- [30] T. Van Der Meersch, J. Deleu, and T. Demeester, in *Associative Memory & Hopfield Networks in 2023* (2023).
- [31] B. Scellier and Y. Bengio, *Frontiers in Computational Neuroscience* **11**, 24 (2017).
- [32] A. Laborieux, M. Ernoult, B. Scellier, Y. Bengio, J. Grollier, and D. Querlioz, *Frontiers in Neuroscience* **15**, 633674 (2021).
- [33] B. Scellier, arXiv preprint arXiv:2103.09985 (2021).
- [34] F. Cai, S. Kumar, T. Van Vaerenbergh, X. Sheng, R. Liu, C. Li, Z. Liu, M. Foltin, S. Yu, Q. Xia, *et al.*, *Nature Electronics* **3**, 409 (2020).
- [35] A. N. Tait, T. F. De Lima, E. Zhou, A. X. Wu, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, *Scientific Reports* **7**, 7430 (2017).