2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Cryptocurrency Portfolio Optimization by Neural Networks

Quoc Minh Nguyen*, Dat Thanh Tran*, Juho Kanniainen*, Alexandros Iosifidis†, Moncef Gabbouj*
*Department of Computing Sciences, Tampere University, Finland
†Department of Electrical and Computer Engineering, Aarhus University, Denmark

***Abstract*—Many cryptocurrency brokers nowadays offer a variety of derivative assets that allow traders to perform hedging or speculation. This paper proposes an effective algorithm based on neural networks to take advantage of these investment products. The proposed algorithm constructs a portfolio that contains a pair of negatively correlated assets. A deep neural network, which outputs the allocation weight of each asset at a time interval, is trained to maximize the Sharpe ratio. A novel loss term is proposed to regulate the network's bias towards a specific asset, thus enforcing the network to learn an allocation strategy that is close to a minimum variance strategy. Extensive experiments were conducted using data collected from Binance spanning 19 months to evaluate the effectiveness of our approach. The backtest results show that the proposed algorithm can produce neural networks that are able to make profits in different market situations.**

***Index Terms*—Deep learning, portfolio optimization, financial engineering, cryptocurrency, decision making.**

I. Introduction

Portfolio optimization is the process of distributing wealth over a universe of assets to satisfy specific criteria (e.g., maximize accumulated return, Sharpe ratio, or minimize volatility). One of the pioneering works in this field is the modern portfolio theory [1], which takes advantage of the diversification of the portfolio to reduce volatility, given the same level of portfolio returns. Recent research has incorporated machine learning into the model-based approach or designing a Reinforcement Learning (RL) framework in which an agent learns how to trade from market feedback, or even for training an end-to-end model to output the portfolio weights. Regarding different approaches, portfolio diversification still plays a vital role in the success of portfolio optimization algorithms. However, asset selection can be challenging since the universe of assets can be large and correlation properties between assets change over time.

In modern portfolio theory, the optimal weights are the solution to the optimization problem that uses the expected returns and covariance matrix as parameters. These two quantities are estimated from the historical returns, which have a low signal-to-noise ratio, and this is likely to affect the accuracy of the estimations. Many attempts have been made to bypass the need for the estimation step. One common approach is to design an RL framework [2], [3], [4] that uses an agent to take action based on market information (e.g., prices, returns, volumes) for maximizing the expected accumulated reward. For adapting the RL framework to the portfolio optimization problem, the action could be a vector of allocation weights the agent needs to allocate for constructing the portfolio. At the same time, the state can be created from the previous trading action, asset prices, and volumes. Another less-discussed but also promising approach is the end-to-end framework [5], [6], [7]. In this approach, the asset information forms an input to be introduced to a neural network which outputs the allocation weights. This paper will focus on the latter approach.

Choosing the market from which to construct the portfolio is also an important factor. Cryptocurrency is a type of virtual currency that uses cryptography technology so that it is not possible to make a counterfeit. The best-known example of cryptocurrencies is Bitcoin (BTC), which has a market capitalization of over 800 billion dollars[1] and is the highest market capitalization among other cryptocurrencies. In 2020, Binance, which is the largest cryptocurrency exchange in terms of the daily trading volume of cryptocurrencies, introduced Binance Leverage Tokens (BLVTs), with the first two of them called BTCUP and BTCDOWN. Those tokens are essentially tokenized versions of future positions of BTC allocated by Binance. This investment product lets traders perform hedging or speculate on BTC's future price movements (up or down) and achieve leveraged profits without liquidation risk.

This paper proposes to use an end-to-end machine learning framework to optimize the Sharpe ratio of a portfolio containing BTCUP and BTCDOWN. We estimate the price relation of the two assets and utilize this information to design a loss term that regulates the model's bias towards a specific asset. When combined with the Sharpe ratio, this loss term can enforce the model to make allocations close to a neutral position and profit by taking arbitrage in these assets. In the ideal situation, the portfolio at the neutral position is invariant with respect to changes in BTCUP and BTCDOWN prices, and this position is constant over time. However, due to Binance's proprietary allocation mechanism that affects the leverage effect and the trading activity of BLVTs on the market, this position changes all the time. Hence, our portfolio value can go up or down, even when we initially bought a portfolio of BTCUP and BTCDOWN at a neutral position.

This work makes several contributions. First, we study how

Q. M. Nguyen, D. T. Tran, J. Kanniainen and M. Gabbouj are with the Department of Computing Sciences, Tampere University, Tampere, Finland. (e-mail: {quoc.nguyen,thanh.tran, juho.kanniainen, moncef.gabbouj}@tuni.fi). A. Iosifidis is with the Department of Electrical and Computer Engineering, Aarhus University, Denmark (e-mail: ai@ece.au.dk).

[1]Data reference from https://coinmarketcap.com on April 8, 2022

to construct a portfolio from a pair of BLVTs. Second, we adopt an end-to-end machine learning framework to allocate a profitable BLVT portfolio in no-cost and cost-included settings. Finally, we make further improvements to the portfolio performance by exploiting the negative correlation characteristic of the BLVTs pair through a custom loss term. To the best of our knowledge, this is the first work that uses BLVTs for a portfolio optimization problem.

II. Related Work

Modern portfolio theory [1], or the so-called mean-variance framework, is widely used to benchmark other portfolio optimization strategies. The goal is to maximize returns at a given risk level. Its variant, the Global Minimum Variance portfolio (GMVP) [1], can be constructed by setting the objective function in the optimization step to the portfolio variance. This strategy is suitable for risk-averse investors whose focus is on reducing portfolio risk. Another common benchmark method is the capitalization-weighted portfolio [8] which is used to build market indices such as S&P500. In the capitalization-weighted approach, a large amount of money is distributed over a small set of high-capital assets, while the low-capital assets have little contribution to the portfolio performance. Hence, the diversification of the portfolio is decreased. This drawback is overcome by the Equal-weighted portfolio (EWP) [9], in which the total wealth is distributed equally to all assets. On the other hand, an equally-weighted risk contributions portfolio [10] implements the idea of having the portfolio components contribute equal risk. Authors in [10] state that this method offers a volatility level higher than the GMVP portfolio but lower than the EWP, allowing a trade-off between two approaches in terms of the absolute level of risk, risk budgeting, and diversification.

In the machine learning context, a natural approach for portfolio optimization is the RL framework since the problem involves interacting with a market via actions like asset allocation and received rewards such as portfolio returns. In the work of [2], the authors explored how the asset allocation problem can be addressed using the RL method. The experiments conducted with multiple trading agents showed superior results over some model-based portfolio optimization strategies. Deep Portfolio Management (DPM) [3] introduces many improvements to the RL framework in the portfolio optimization problem. The most important proposal in this work is to use a neural network architecture that enables sharing of network weights between different assets. This approach allows the network to assess an asset based on patterns learned from other assets. Since the network weights are not specialized to any asset, this method can update the portfolio's constituents or even the size of the portfolio. Going further toward a real-world trading system, Hierarchical Reinforced trading system for Portfolio Management (HRPM) [4] is an RL framework that decomposes the trading process into hierarchical tasks of portfolio optimization and allocation or trading. In this framework, each task is associated with a separate policy. The high-level policy manages the portfolio optimization task that determines the portfolio weights at a lower frequency, such as days, while the low-level policy operates at a higher frequency and places limit orders to fulfill goals from the high-level policy.

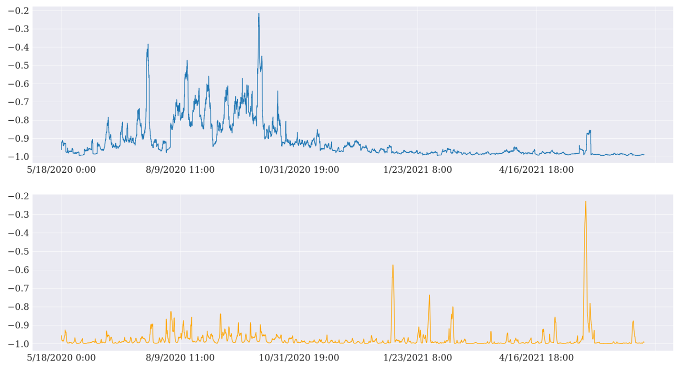


Fig. 1. Rolling correlation of the returns (top) and the prices (bottom) of BTCUP and BTCDOWN from May 2020 to July 2021. The correlation window length is 72 hours, and the timeframe between price samples is 1 hour.

In the RL frameworks, the objective function is an expected cumulative reward, whereas popular risk-adjusted metrics like the Sharpe ratio [11] cannot be decomposed into immediate rewards. To be able to use the Sharpe ratio as the optimization objective, the authors in [5] proposed an end-to-end framework that directly outputs allocation weights from the asset prices and returns via a Long Short-Term Memory (LSTM). Risk-based allocation in [7] takes a similar approach. This method trained a fully connected neural network to output the allocation of risk contribution and solve the weight allocation from that risk contribution using an implicit optimization layer embedded in the neural network. The framework proposed in [5] has the flexibility to be extended to incorporate different types of portfolio constraints such as cardinality, leverage, or maximum position for individual assets by utilizing several custom neural network layers. Overall, the end-to-end framework is simpler than the RL approach in terms of framework complexity, and it can be easier to design specialized components for a specific problem.

A special case of the portfolio optimization problem is the portfolio of Binance Leverage Tokens considered in this paper. Every BLVT is associated with a price called Net Asset Value (NAV). At this price, the owner can redeem their tokens to Binance and receive USDT[1] back. The NAV is updated based on the fluctuation in the value of the future position basket that each BLVT represents. However, the BLVTs can also be traded in the spot market, where their prices follow around NAVs. These prices on the spot market also reflect speculation on the future price movements of the underlying assets in the futures market. The first available BLVTs are BTCUP and BTCDOWN. BTCUP aims to generate leveraged gains when the price of BTC goes up, while BTCDOWN aims to generate leveraged gains when the price of BTC goes down. From that property, every pair of UP and DOWN tokens in BLVTs have

[1]USDT is a stablecoin. This type of cryptocurrency aims to keep cryptocurrency valuations stable. USDT pegs the US dollar, so the value of 1 USDT is very close to 1 US dollar.

a consistent negative correlation over time. Figure 1 illustrates this correlation between BTCUP and BTCDOWN.

III. Method

This section describes the proposed strategy to construct a portfolio that consists of two leveraged tokens that are negatively correlated. In addition, this section presents the formulation of a novel loss function that is used to optimize a neural network generating the portfolio reallocation weights. The proposed loss function consists of two terms. The former maximizes the Sharpe ratio, while the latter is used to regularize the network's bias toward specific assets.

A. Mathematical Formulation

Given an initial amount of capital, an agent allocates all capital into two assets, denoted as asset A and asset B. At each time interval, the agent adjusts the amount of the two assets so that the portfolio contains only asset A and asset B (without any cash). The allocation of each asset in the portfolio is restricted to the long-only and budget constraints. The goal is to design a reallocation strategy that maximizes some portfolio metrics over time, for example, the Sharpe ratio. We start by constructing the mathematical formulation for a portfolio of N assets as in [3]. Later, we restrict $N = 2$ for our specific problem. A portfolio of N assets at time $t-1$ is associated with a weight vector

$$\mathbf{w}_{t-1} = (w_{t-1,0}, w_{t-1,1}, \dots, w_{t-1,N-1}), \tag{1}$$

which is the relative value of each asset compared to the portfolio value. At any time t, the long-only constraint is realized as

$$w_{t,i} \geq 0 \quad \forall i \in \{0, 1, \dots, N-1\}, \tag{2}$$

and the budget constraint can be written as

$$\sum_{i=0}^{N-1} w_{t,i} = 1. \tag{3}$$

Let denote period $[t-1, t]$ is the period between time $t-1$ and time t. At the beginning of period $[t-1, t]$, the asset price vector is denoted as

$$\mathbf{y}_{t-1} = (y_{t-1,0}\,, y_{t-1,1}\,, \dots, y_{t-1,N-1}), \tag{4}$$

and the volume vector for the whole period is

$$\mathbf{v}_{t-1} = (v_{t-1,0}\,, v_{t-1,1}\,, \dots, v_{t-1,N-1}). \tag{5}$$

The asset value vector at time $t-1$, which denotes the absolute value (monetary value) of N assets, is

$$\mathbf{a}_{t-1} = \mathbf{y}_{t-1} \odot \mathbf{v}_{t-1} \tag{6}$$

where $\odot$ is the element-wise multiplication operator. The portfolio value is p_{t-1}, which is computed by summing over the absolute value of N assets

$$p_{t-1} = \mathbf{1}_N^T\, \mathbf{a}_{t-1} = \sum_{i=0}^{N-1} y_{t-1,i}\, v_{t-1,i}, \tag{7}$$

where $\mathbf{1}_N$ is the one-vector of size N.

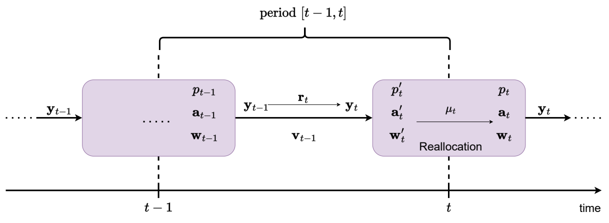


Fig. 2. Illustration of the effect of price change and allocation to the portfolio quantities. In period $[t-1, t]$, the price of assets in the portfolio changes from $\mathbf{y}_{t-1}$ to $\mathbf{y}_t$, which results in the return vector $\mathbf{r}_t$. This price change adjusts p_{t-1}, $\mathbf{a}_{t-1}$, and $\mathbf{w}_{t-1}$ to p'_t, $\mathbf{a}'_t$, and $\mathbf{w}'_t$. The price change does not affect the volume vector $\mathbf{v}_{t-1}$. At the end of period $[t-1, t]$, the reallocation activity adjusts the previously mentioned quantities to p_t, $\mathbf{a}_t$, and $\mathbf{w}_t$. By the effect of transaction fee and other fees, the portfolio value p'_t is shrunk to p_t by a factor of μ_t. By the reallocation, the volume vector has a new value.

At the end of period $[t-1, t]$, before we make any reallocation, the portfolio value is denoted as p'_t, and the asset price vector is $\mathbf{y}_t$. Here we assume that we only reallocate the portfolio at the end of each period, and asset prices at the beginning of the incoming period are equal to the prices at the end of the current period. The asset returns vector in period $[t-1, t]$ is

$$\begin{aligned} \mathbf{r}_t &= ((\mathbf{y}_t - \mathbf{y}_{t-1}) \oslash \mathbf{y}_{t-1}) \\ &= \left(\frac{y_{t,0}}{y_{t-1,0}} - 1, \frac{y_{t,1}}{y_{t-1,1}} - 1, \dots, \frac{y_{t,N-1}}{y_{t-1,N-1}} - 1 \right), \end{aligned} \tag{8}$$

where $\oslash$ denotes the element-wise division operator. Right before making the reallocation in period $[t-1, t]$, the volume vector is determined by

$$\mathbf{v}_{t-1} = p_{t-1} \mathbf{w}_{t-1} \oslash \mathbf{y}_{t-1}, \tag{9}$$

and the asset value vector is

$$\mathbf{a}'_t = \mathbf{y}_t \odot \mathbf{v}_{t-1} = p_{t-1} \mathbf{w}_{t-1} \odot (\mathbf{y}_t \oslash \mathbf{y}_{t-1})\,. \tag{10}$$

Following Eq. (8) and Eq. (10), the portfolio value at time t, before making reallocation, is

$$p'_t = \mathbf{1}_N^T\, \mathbf{a}'_t = p_{t-1} \sum_{i=1}^{N-1} w_{t-1,i}(1 + r_{t,i}). \tag{11}$$

The corresponding portfolio weight vector for this portfolio value is

$$\mathbf{w}'_t = \frac{1}{p'_t}(\mathbf{y}_t \odot \mathbf{v}_{t-1}). \tag{12}$$

At the end of period $[t-1, t]$, the agent needs to decide the weight vector $\mathbf{w}_t$ for the next period $[t, t+1]$ and makes the reallocation accordingly. This reallocation is executed via trading on the market, which requires trading fee. The cost reduces the portfolio value by a shrinkage parameter

$$\mu_t = \frac{p_t}{p'_t}, \tag{13}$$

where $0 < \mu_t \leq 1$. If there is no trading fee, the portfolio value will remain unchanged after the reallocation, which means μ_t is 1. Figure 2 summaries the relationship between portfolio quantities in period $[t-1, t]$.

The portfolio return in period $[t-1,t]$ is computed by comparing the portfolio value at the start of the current and incoming periods

$$R_t = \frac{p_t}{p_{t-1}} - 1 = \frac{\mu_t p'_t}{p_{t-1}} - 1 = \mu_t \left(\sum_{i=0}^{N-1} w_{t-1,i}(1 + r_{t,i}) \right) - 1. \tag{14}$$

The portfolio return of consecutive periods will be used to calculate the Sharpe ratio for training the neural networks model. This return is an essential input to compute other portfolio performance metrics.

B. Transaction and Management Fee

Transaction fee resulting from portfolio reallocation in the general case of N assets has no analytical formula [12], but it can be solved iteratively [3]. In our portfolio optimization problem, there are only two assets, called A and B, and cash is not allowed. When trading occurs, exactly one asset needs to be sold, and all the cash received afterward is used to buy the other asset. We can find the closed form of shrinkage parameter using this portfolio properties.

At the end of period $[t-1,t]$, we need to adjust the weight vector from $\mathbf{w}'_t$ to $\mathbf{w}_t$ by reallocation. If the new weight for asset A satisfied

$$w'_{t,A}\, p'_t > w_{t,A}\, p_t \Longleftrightarrow w'_{t,A} > \mu_t\, w_{t,A}, \tag{15}$$

then the allocation for this asset is decreased. In this case, we need to sell asset A, and the money obtained is used to buy more asset B. It can be shown that when the portfolio contains only two assets, the selling condition above is equivalent to $w'_{t,A} > w_{t,A}$. Similarly, asset B is needed to buy more if and only if $w'_{t,B} < w_{t,B}$.

We suppose that at the end of period $[t-1,t]$ the sold asset is asset A and the bought asset is asset B. When we perform buying and selling, these actions induce transaction fee. The change in the portfolio value before and after reallocation is caused only by these transaction fee. Therefore, the total trading fee from reallocation is $p'_t - p_t$. This cost can be broken down into the cost associated with selling and buying. We set the cost rate for selling and buying is equal and is denoted as $0 \le c < 1$. The cost generated from selling asset A is

$$c\,(w'_{t,A}\, p'_t - w_{t,A}\, p_t) = c\, p'_t\, (w'_{t,A} - w_{t,A}\, \mu_t). \tag{16}$$

The change in absolute value after reallocation of the bought asset B is

$$w_{t,B}\, p_t - w'_{t,B}\, p'_t = p'_t\, (w_{t,B}\, \mu_t - w'_{t,B}). \tag{17}$$

However, to account for the trading fee, the buying order is larger than this value change by a multiple of $1/(1-c)$. Then, the cost generated from buying asset B is

$$c\, \frac{p'_t(w_{t,B}\, \mu_t - w'_{t,B})}{1-c}. \tag{18}$$

The sum of selling cost and buying cost equals the difference between the portfolio values after and before reallocation

$$c\, p'_t(w'_{t,A} - w_{t,A}\, \mu_t) + c\, \frac{p'_t(w_{t,B}\, \mu_t - w'_{t,B})}{1-c} = p'_t - p_t. \tag{19}$$

Solving μ_t from this equation, the shrinkage parameter of the portfolio for reallocation is

$$\mu_t = \frac{(1-c) + c\,(w'_{t,B} - w'_{t,A}(1-c))}{(1-c) + c\,(w_{t,B} - w_{t,A}(1-c))}. \tag{20}$$

In case the sold asset is B, and the bought asset is A, we just need to exchange the index A and B of the weights in Eq. (20) to obtain the right formula.

BLVTs are also associated with the management fee. This fee is charged at 00:00 UTC directly on the net portfolio value. To consider this type of fee, we multiply μ_t by a multiplier $(1-m)$ every 24 hours, where $0 \le m < 1$ is the management fee rate.

C. Baseline Loss Function

The Sharpe ratio integrates two important aspects of portfolio performance, that is, profitability and risk, into one measure. More specifically, it is defined as the excess portfolio expected return over the portfolio volatility. Since the portfolio return distribution is unknown, we estimate the Sharpe ratio with the portfolio return samples. For T trading periods $\{[0,1],[1,2],\ldots,[T-1,T]\}$, the sample mean of portfolio return is

$$E_R = \frac{1}{T} \sum_{t=1}^{T} R_t, \tag{21}$$

where R_t is determined by Eq. (14) and the portfolio volatility is computed by the sample standard deviation of portfolio returns, which is

$$\sigma_R = \sqrt{\frac{1}{T-1} \sum_{t=1}^{T} (R_t - E_R)^2}. \tag{22}$$

The Sharpe ratio, which omits the risk-free rate for simplicity, is

$$SR_T = \frac{E_R}{\sigma_R}. \tag{23}$$

In this paper, we train a neural network model, represented by the function $\mathcal{F}_{\boldsymbol{\theta}}(\cdot)$, to output the portfolio weights in an end-to-end manner [5].

$$\mathbf{w}_t = \mathcal{F}_{\boldsymbol{\theta}}(\mathbf{x}_t), \tag{24}$$

where $\mathbf{x}_t$ is the market information available at time t. The Sharpe ratio in Eq. (23) is a function of portfolio weights. Therefore, it can be used as the loss function for training. We refer to this approach as the baseline method, which involves training a neural network model using the negative Sharpe ratio as the loss function.

$$L_{BL} = -SR_T. \tag{25}$$

Minimizing this loss function is equivalent to maximizing the Sharpe ratio. The long-only and budget constraints are fulfilled by adding the Softmax activation layer as the final layer in the neural network model.

D. Neutral Position Constrain

In this section, we form a constrain for the training of neural networks for the optimal allocation of two tokens: BTCUP and BTCDOWN. This is based on linear regression to model the relationship between the price of the two tokens:

$$y_{t,u} = \alpha + \beta^{market} y_{t,d} + \epsilon_t, \tag{26}$$

The terms $y_{t,u}$ and $y_{t,d}$ denote the prices of BTCUP and BTCDOWN at time t, respectively. The error term ϵ_t is a zero-mean random variable with variance of σ^2. We assume that ϵ_t is independent with $y_{t,d}$. The coefficients β^{market} and α are unknown parameters. Figure 3 illustrates the price relationship of two tokens and the corresponding estimated values of β^{market}.

Suppose that in period $[t, t+1]$, the price of BTCDOWN changes by $\Delta y_{t+1,d}$ to

$$y_{t+1,d} = y_{t,d} + \Delta y_{t+1,d}. \tag{27}$$

Then Eq. (26) implies that

$$y_{t+1,u} = y_{t,u} + \Delta y_{t+1,d}\, \beta^{market} + (\epsilon_{t+1} - \epsilon_t). \tag{28}$$

At the beginning of period $[t, t+1]$, we hold a portfolio of BTCUP and BTCDOWN with volumes $v_{t,u}$ and $v_{t,d}$, respectively. The portfolio value at time t is

$$p_t = y_{t,u}\, v_{t,u} + y_{t,d}\, v_{t,d}. \tag{29}$$

The portfolio value at the end of the period $[t, t+1]$ just before the portfolio reallocation is

$$\begin{aligned} p'_{t+1} &= y_{t+1,u}\, v_{t,u} + y_{t+1,d}\, v_{t,d} \\ &= p_t + (\beta^{market}\, v_{t,u} + v_{t,d}) \Delta y_{t+1,d} + v_{t,u}\, \Delta\epsilon_{t+1}, \end{aligned} \tag{30}$$

where $\Delta\epsilon_{t+1} = \epsilon_{t+1} - \epsilon_t$.

Our regularization strategy is based on the idea that the value of the portfolio at $t+1$ just before the portfolio reallocation, p'_{t+1}, is *immune* for changes in $y_{t+1,d}$. In this way, our strategy is not dynamically dependent on instantaneous changes in the token prices, but is static. To exclude the second term in Eq. (30), which captures the dependency on $y_{t+1,d}$, we set

$$\frac{v_{t,d}}{v_{t,u}} = -\beta^{market}. \tag{31}$$

This does not guarantee that $p'_{t+1} = p_t$ as $\epsilon_{t+1} - \epsilon_t$ has certain variance, and for that reason, the portfolio value can change due to the residual component. However, assuming that ϵ_{t+1} does not correlate with $y_{t+1,d}$, this strategy minimizes the influence of $\Delta y_{t+1,d}$ on the $\Delta p'_{t+1} = p'_{t+1} - p_t$.

We call the weight values that satisfy Eq. (31) the neutral weights and denote them as $w^*_{t,u}$, and $w^*_{t,d}$. Then by rewriting the volumes in terms of portfolio weights, we get

$$\frac{w^*_{t,d}}{w^*_{t,u}} = -\beta^{market} \left(\frac{y_{t,d}}{y_{t,u}} \right). \tag{32}$$

Suppose we explicitly set the portfolio weights to neutral weights in every period. In that case, the portfolio has no bias toward specific assets and is neutral regarding asset price change. One can expect the portfolio value to be stable in the successive trading period, and the maximum drawdown over the long term will be low. This strategy resembles the GMVP strategy because both aim to minimize portfolio volatility. However, when using the above analysis instead of GMVP, we directly exploit the linearity relation of a pair of BLVTs.

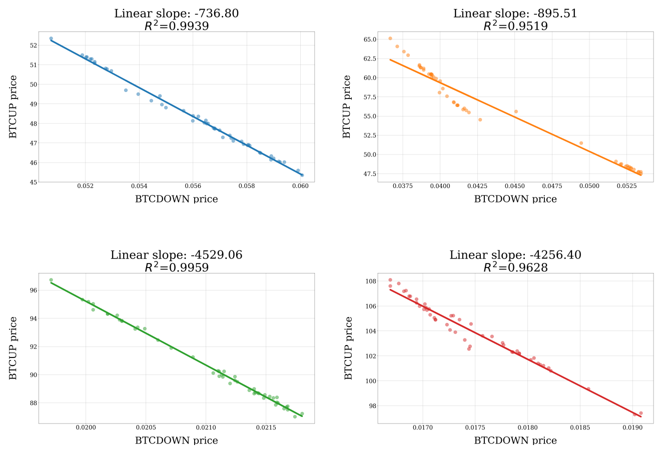


Fig. 3. Scatter plot and ordinary least squares linear regression line of the prices of BTCUP and BTCDOWN. In each plot, the prices are sampled every 1 hour and there are 48 samples. Even though the price relation is a straight line, simple linear regression does not fit well in some cases.

To compute the value of these neutral weights, we estimate the value of β^{market} using the ordinary least squares method using data from the K most recent period. Then

$$\frac{\widehat{w}_{t,d}}{\widehat{w}_{t,u}} = -\widehat{\beta}^{market}_t \left(\frac{y_{t,d}}{y_{t,u}} \right), \tag{33}$$

where $\widehat{\beta}^{market}_t$ is the estimate at time t of β^{market}, and $\widehat{w}_{t,d}$, and $\widehat{w}_{t,u}$ are the estimates of $w^*_{t,d}$, and $w^*_{t,u}$, respectively. Eq. (33), together with the budget constraint, determines the value of $\widehat{w}_{t,u}$, and $\widehat{w}_{t,d}$.

E. Variance-controlled Loss Terms

Controlling the neutrality of a portfolio has some benefits. First, gearing the portfolio towards neutral position will lower directional risk, hence possibly improving the Sharpe ratio. Second, reducing the network's bias toward a specific assets can reduce portfolio lost in case the asset has worse performance compare to its past. Finally, the region around the neutral weights by a margin will always contain some weight values that the model can make profits or at least preserve the portfolio value. If we encourage model outputs to be in this region, it might be easier for the network to learn how to take profits rather than searching from random portfolio weight values.

To control the neutrality of the portfolio, we first rewrite the approximate portfolio variance

$$\begin{aligned} \mathrm{Var}[p'_{t+1}] &\approx (\beta^{market}\, v_{t,u} + v_{t,d})^2\, \mathrm{Var}[\Delta y_{t+1,d}] \\ &= v^2_{t,u} (\beta^{market} - \beta^{model}_t)^2\, \mathrm{Var}[\Delta y_{t+1,d}], \end{aligned} \tag{34}$$

where

$$\beta^{model}_t = -\frac{v_{t,d}}{v_{t,u}} = -\frac{y_{t,u}\, w_{t,d}}{y_{t,d}\, w_{t,u}} \tag{35}$$

Here β_t^{model} is a parameter computed from model allocation weights. At training time, we use the estimate $\widehat{\beta}_t^{market}$ for β^{market}. Our purpose is to constrain the variance via the term $(\beta_t^{market} - \beta_t^{model})^2$ within a margin

$$\begin{aligned} &(\beta_t^{market} - \beta_t^{model})^2 \leq (\gamma \beta^{market})^2 \\ \iff (1+\gamma)\beta^{market} &\leq \beta_t^{model} \leq (1-\gamma)\beta_t^{market}. \end{aligned} \tag{36}$$

This condition is equivalent to

$$\begin{cases} C_1 = \beta_t^{model} - (1+\gamma)\beta_t^{market} \geq 0 \\ C_2 = (1-\gamma)\beta_t^{market} - \beta_t^{model} \geq 0. \end{cases} \tag{37}$$

where $\gamma \geq 0$ is the parameter that controls the degree of neutrality of the portfolio. We design the following loss terms that take the form of hinge loss

$$A_1(\mathbf{w}_t; \gamma) = \max(0, -C_1\, C_2), \tag{38}$$

and

$$A_2(\mathbf{w}_t; \gamma) = \max(0, -C_1)^2 + \max(0, -C_2)^2. \tag{39}$$

These two loss terms will only penalize the model when the β_t^{model} lies outside the constrained region. When β_t^{model} lies in the range defined in Eq. (36), the two loss terms will vanish. Otherwise, the penalized terms will be equal to the following positive terms

$$A_1(\mathbf{w}_t; \gamma) = -C_1 C_2, \tag{40}$$

and

$$A_2(\mathbf{w}_t; \gamma) = \begin{cases} C_1^2 & \text{if } C_1 < 0, \\ C_2^2 & \text{if } C_2 < 0. \end{cases} \tag{41}$$

We have conducted experiments where we train the neural network with loss functions that is the negative Sharpe ratio combined with $A_1(\mathbf{w}_t; \gamma)$ or $A_2(\mathbf{w}_t; \gamma)$. The obtained Sharpe ratios are significantly lower than the baseline. We suspect that the model output in a fraction form $w_{t,d}/w_{t,u}$ in the proposed loss terms may cause difficulty for the model to learn the optimal allocation. We solve this problem by multiplying the proposed loss terms with square BTCUP volume $v_{t,u}^2$ to eliminate the fraction form and observe better results. The final forms of the proposed loss terms are

$$\begin{aligned} HL_1(\mathbf{w}_t; \gamma) &= v_{t,u}^2\, A_1(\mathbf{w}_t; \gamma) \\ &= \max\left(0, -v_{t,u}^2\, C_1\, C_2\right), \end{aligned} \tag{42}$$

and

$$\begin{aligned} HL_2(\mathbf{w}_t; \gamma) &= v_{t,u}^2\, A_2(\mathbf{w}_t; \gamma) \\ &= \max(0, -v_{t,u}\, C_1)^2 + \max(0, -v_{t,u}\, C_2)^2. \end{aligned} \tag{43}$$

The proposed loss function is the negative Sharpe ratio, combined with the variance-controlled loss terms

$$L_1 = -SR_T + \xi\, \frac{1}{T} \left(\sum_{t=1}^{T} HL_1(\mathbf{w}_t; \gamma) \right), \tag{44}$$

and

$$L_2 = -SR_T + \xi\, \frac{1}{T} \left(\sum_{t=1}^{T} HL_2(\mathbf{w}_t; \gamma) \right), \tag{45}$$

where ξ is the parameter that controls the effects of the additional loss terms. We define the proposed methods as training a machine learning model that uses the proposed loss functions L_1 or L_2.

TABLE I
DATA RANGE FOR TRAINING AND TESTING.

	Train data range	Test data range
Period 1	2020-05-15 00:00 to 2021-07-03 23:00	2021-07-04 00:00 to 2021-09-01 23:00
Period 2	2020-05-15 00:00 to 2021-09-01 23:00	2021-09-02 00:00 to 2021-10-31 23:00
Period 3	2020-05-15 00:00 to 2021-10-31 23:00	2021-11-01 00:00 to 2021-30-12 23:00

IV. Datasets and Evaluations

In this section, we present empirical experiments on the baseline and proposed methods. Before that, details about the dataset, model architecture, feature selection, and hyperparameters setting are described.

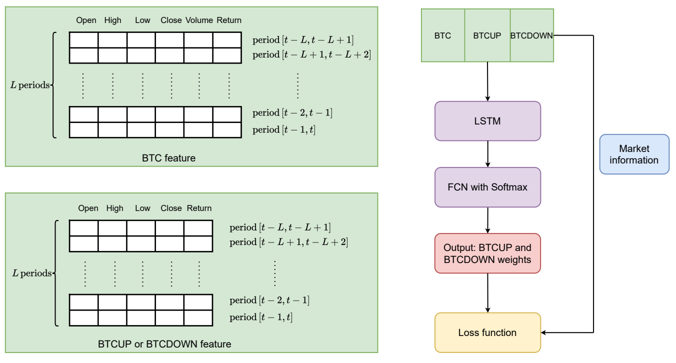
A. Dataset Description

We pulled the 1-hour timeframe of OHLCV data of three assets with tickers name BTCUSD (BTC), BTCUPUSDT (BTCUP), and BTCDOWNUSDT (BTCDOWN) using Binance API. OHLCV is an aggregate form of market data standing for Open, High, Low, Close, and Volume. The entire dataset spans 19 months and starts from the introduction of BTCUP and BTCDOWN. The data is split following the forward validation split scheme, where the previous training and the test dataset are merged to form a new training dataset. The new test dataset is the data available right after the new training dataset. In our experiment, we use three test datasets. A summary of the data range in training and testing is shown in Table I. The test periods are selected so that all have a duration of two months and try to cover different price trends of BTC. In the first and second periods, BTC has an uptrend price movement, whereas BTC lost about 25% of its value in the third period.

B. Model Architecture and Feature Selection

The input features for the neural network are derived from OHLCV data and return data for all the assets in the portfolio. Returns for individual assets are computed from the closing prices of two sequential periods, as defined in Eq. (8). Given the heterogeneity in price and volume ranges among assets, we apply normalization for all data, including returns. For each type of data, the mean and standard deviation in L_{norm} consecutive periods are computed, and these statistics are then used to perform the z-score normalization for the successful L_{norm} periods. This forwarding normalization scheme guarantees that information from the future does not flow backward in time.

After normalization, the feature of each ticker is organized into a matrix where the columns are concatenated from the OHLCV and return features. The features of the three assets are combined along the column dimension. Figure 4a visualizes the input feature. To estimate the portfolio weights $\mathbf{w}_t$,



(a) Feature selection (b) Model architecture

Fig. 4. In (a), the elements of feature matrices are the z-score normalization data. (b) shows the forward computation of the model and its structure. The market information are the BTCUP and BTCDOWN close prices, their future returns, and β_t^{market}.

the model uses a lookback window containing data from the most recent L periods, including period $[t-1, t]$.

Our choice of network is a Long Short-Term Memory (LSTM) [13] with one layer and the hidden feature size of 64, followed by a fully connected layer with the Softmax activation function. While there exist numerous specialized network architectures, such as [14], [15], [16], [17], [18], [19], proposed for the purpose of financial forecasting, our choice leans towards LSTM (Long Short-Term Memory). This is primarily due to its robust validation through extensive testing in portfolio optimization [5]. Figure 4b summarizes the model architecture.

C. Experiment Protocols

We perform a hyperparameter search using the following parameters. The batch size is selected from the set $\{64, 128, 256, 512\}$, while the number of epochs ranged between 80 and 140 in increments of 20. We use Adam optimizer [20] and a cosine annealing learning rate scheduler [21] with the start learning rate chosen from the set $\{1\text{e-}5, 3\text{e-}5, 5\text{e-}5, 1\text{e-}4, 3\text{e-}4, 5\text{e-}4, 1\text{e-}3\}$. The weight decay is chosen from $\{0.0, 1\text{e-}4, 3\text{e-}4, 5\text{e-}4, 1\text{e-}3\}$. The value of γ is chosen from the set $\{0.0, 0.1, 0.2, \dots, 1.0\}$. The parameter ξ is selected from the set $\{1\text{e-}6, 3\text{e-}6, 5\text{e-}6, 1\text{e-}5, 3\text{e-}5, 5\text{e-}5, 1\text{e-}4, 3\text{e-}4, 5\text{e-}4, 1\text{e-}3\}$. We conduct experiments under both fee-included and no-fee configurations. In the no-cost scheme, μ_t is set to 1, and no management fee is imposed. Conversely, the cost-included scheme incorporated the Binance trading fee for BLVTs, setting the trading fee rate c at 0.075%, and the daily management fee m at 0.01%. Lastly, the length of the normalization window is fixed at $L_{norm} = 12$ and the number of periods used to estimate β^{market} and the length of the lookback window is fixed at $L = K = 48$.

Due to the random initialization of the model parameters, the performance of the trained models differs for each training session. Each parameter configuration is run five times, and the reported metric is the mean and standard deviation of the average Sharpe ratio over three testing periods across five runs.

D. Experimental Results

We compare our proposed methods with the baseline method (NS). The proposed methods are presented under the name SVC$_1$ for model training with loss function L_1, and SVC$_2$ for loss function L_2. We also include other benchmark methods for a comprehensive comparison. The Neutral-Weight Portfolio (NWP) involves the allocation of estimated neutral weights as outlined in Eq. (33). The Equal-Weight Portfolio (EWP) [9] uniformly distributes weights between BTCUP and BTCDOWN for each allocation period. The Global Minimum Variance Portfolio (GMVP) [1] determines optimal weights that minimize the estimated return covariance matrix, using a window length of 48 hours for estimation. Finally, the performance of the underlying cryptocurrency of BTCUP and BTCDOWN, which is BTC, will be presented for reference. For holding BTC, the performance when the trading fee and management fee are included does not affect the portfolio performance since we do not conduct any trading, and holding BTC is not required the management fee.

The findings presented in Table II indicate that our proposed methods enhance the performance of the baseline method in most cases. Other strategies such as NWP, EWP, and GMVP yield Sharpe ratios near zero, underscoring the advantages of deploying neural network-based methods over conventional ones. The performance of BTC is much lower than the performance of neural network-based methods because the profits of BTC gains in the first and second periods are eroded by the downtrend price movement in the third period, while the neural network-based methods can put more weight on BTCDOWN to speculate on the decreasing BTC price. Therefore, these results show the benefit of holding a pair of BLVTs over the underlying asset.

In Tables III, and IV, we choose the median result in the 5 runs to show a breakdown of the performance of all methods over three testing periods to observe how each method performs in different market situations. In addition, the final accumulated portfolio value (fAPV) is presented as a profitability metric. We set the initial portfolio value to 1. Then the fAPV will reflect the accumulated return over each testing period. The Maximum Drawdown (MDD) is also considered to highlight the risk associated with each method.

These tables show that the NWP, EWP, and GMVP methods usually have the lowest MDD in all test periods, whether the trading and management fee are considered. While these methods are effective in reducing portfolio risk, they do so at the expense of portfolio profitability. The fAPVs of these methods show that they can only preserve the original portfolio value and cannot make profits. The Sharpe ratios from the proposed methods outperform the baseline method in mostly every period in different settings. When BTC loses one-fourth of its value in the third period, the proposed methods can profit and achieve a higher Sharpe ratio than both BTC and the baseline method.

TABLE II
AVERAGE SHARPE RATIO OVER THREE TESTING PERIODS.

	NS	SVC_1 (proposed)	SVC_2 (proposed)	NWP	EWP	GMVP	BTC
No fee	0.033 ± 0.008	$\mathbf{0.035 \pm 0.005}$	0.033 ± 0.004	-0.005	0.004	-0.001	0.012
Fee included	0.030 ± 0.007	$\mathbf{0.033 \pm 0.003}$	0.032 ± 0.002	-0.008	-0.003	-0.010	0.012

TABLE III
EXPERIMENT RESULTS WITH NO TRADING FEE AND MANAGEMENT FEE.

	Period 1			Period 2			Period 3		
	Sharpe	fAPV	MDD	Sharpe	fAPV	MDD	Sharpe	fAPV	MDD
NS	0.040	1.338	0.147	0.034	1.385	0.221	**0.029**	**1.258**	0.218
SVC_1	**0.079**	**1.878**	0.142	0.033	1.401	0.220	-0.010	0.855	0.398
SVC_2	0.056	1.299	0.060	**0.043**	**1.438**	0.167	0.002	0.973	0.329
NWP	0.003	1.007	0.050	-0.016	0.956	**0.053**	-0.001	0.995	0.053
EWP	-0.006	0.983	0.077	-0.009	0.972	0.088	0.026	1.057	0.038
GMVP	-0.004	0.991	**0.038**	-0.017	0.958	0.054	0.019	1.035	**0.033**
BTC	0.041	1.419	0.156	0.024	1.220	0.231	-0.028	0.754	0.335

TABLE IV
EXPERIMENT RESULTS WITH TRADING FEE $c = 0.075\%$ AND DAILY MANAGEMENT FEE $m = 0.01\%$

	Period 1			Period 2			Period 3		
	Sharpe	fAPV	MDD	Sharpe	fAPV	MDD	Sharpe	fAPV	MDD
NS	0.075	**1.939**	0.097	0.017	1.189	0.311	-0.005	0.848	0.472
SVC_1	0.074	1.891	0.123	0.025	1.319	0.319	**0.003**	**0.995**	0.214
SVC_2	**0.075**	1.862	0.113	**0.026**	**1.324**	0.323	-0.005	0.868	0.459
NWP	0.000	0.998	0.050	-0.019	0.948	**0.056**	-0.005	0.987	0.056
EWP	-0.012	0.965	0.087	-0.015	0.954	0.094	0.018	1.038	0.039
GMVP	-0.013	0.973	**0.046**	-0.025	0.940	0.069	0.009	1.016	**0.034**
BTC	0.041	1.419	0.156	0.024	1.220	0.231	-0.028	0.754	0.335

V. CONCLUSION

This work adopts the approach of using deep learning for the portfolio optimization problem with the Sharpe ratio as the loss function. A pair of BLVTs is chosen to construct a portfolio to benefit from their consistent negative correlation. The portfolio contains only two assets that allow us to investigate the neutral position of the portfolio using simple mathematical analysis. Additional loss terms are designed to control the neutrality of the portfolio. We compare the proposed methods with the baseline and other non-learning approaches. Experimental results show that holding a portfolio containing a pair of BLVTs is superior to holding only the underlying asset, especially for a high-volatility market like cryptocurrency. In addition, the proposed methods show their effectiveness in improving the baseline method and gaining the best results compared to other methods in all different settings.

REFERENCES

[1] H. Markowitz, "Portfolio Selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952, publisher: [American Finance Association, Wiley]. [Online]. Available: https://www.jstor.org/stable/2975974

[2] A. Filos, "Reinforcement Learning for Portfolio Management," Sep. 2019, arXiv:1909.09571 [cs, q-fin, stat] version: 1. [Online]. Available: http://arxiv.org/abs/1909.09571

[3] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," *arXiv preprint arXiv:1706.10059*, 2017.

[4] R. Wang, H. Wei, B. An, Z. Feng, and J. Yao, "Commission Fee is not Enough: A Hierarchical Reinforced Framework for Portfolio Management," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, pp. 626–633, May 2021, number: 1. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16142

[5] Z. Zhang, S. Zohren, and S. Roberts, "Deep learning for portfolio optimization," *The Journal of Financial Data Science*, vol. 2, no. 4, pp. 8–20, 2020, publisher: Institutional Investor Journals Umbrella.

[6] C. Zhang, Z. Zhang, M. Cucuringu, and S. Zohren, "A Universal End-to-End Approach to Portfolio Optimization via Deep Learning," Nov. 2021, arXiv:2111.09170 [q-fin]. [Online]. Available: http://arxiv.org/abs/2111.09170

[7] A. S. Uysal, X. Li, and J. M. Mulvey, "End-to-End Risk Budgeting Portfolio Optimization with Neural Networks," Jul. 2021, arXiv:2107.04636 [q-fin]. [Online]. Available: http://arxiv.org/abs/2107.04636

[8] J. C. Hsu, "Cap-Weighted Portfolios are Sub-Optimal Portfolios," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper 647001, Dec. 2004. [Online]. Available: https://papers.ssrn.com/abstract=647001

[9] V. DeMiguel, L. Garlappi, and R. Uppal, "Optimal Versus Naive Diversification: How Inefficient is the 1/N Portfolio Strategy?" *The Review of Financial Studies*, vol. 22, no. 5, pp. 1915–1953, May 2009. [Online]. Available: https://doi.org/10.1093/rfs/hhm075

[10] S. Maillard, T. Roncalli, and J. Teïletche, "The Properties of Equally Weighted Risk Contribution Portfolios," *The Journal of Portfolio Management*, vol. 36, no. 4, pp. 60–70, Jul. 2010, publisher: Institutional Investor Journals Umbrella. [Online]. Available: https://jpm.pm-research.com/content/36/4/60

[11] W. F. Sharpe, "The Sharpe Ratio," *The Journal of Portfolio Management*, vol. 21, no. 1, pp. 49–58, Oct. 1994, publisher: Institutional Investor Journals Umbrella Section: Primary Article. [Online]. Available: https://jpm.pm-research.com/content/21/1/49

[12] M. Ormos and A. Urbán, "Performance analysis of log-optimal portfolio strategies with transaction costs," *Quantitative Finance*, vol. 13, no. 10, pp. 1587–1597, 2013, publisher: Taylor & Francis.

[13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Oct. 1997, conference Name: Neural Computation.

[14] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *2017 IEEE 19th conference on business informatics (CBI)*, vol. 1. IEEE, 2017, pp. 7–12.

[15] D. T. Tran, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Tensor representation in high-frequency financial data for price change prediction," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2017, pp. 1–7.

[16] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 5, pp. 1407–1418, 2018.

[17] D. T. Tran, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Data-driven neural architecture learning for financial time-series forecasting," *arXiv preprint arXiv:1903.06751*, 2019.

[18] D. T. Tran, N. Passalis, A. Tefas, M. Gabbouj, and A. Iosifidis, "Attention-based neural bag-of-features learning for sequence data," *IEEE Access*, vol. 10, pp. 45 542–45 552, 2022.

[19] D. T. Tran, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Data normalization for bilinear structures in high-frequency financial time-series," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 7287–7292.

[20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[21] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.