IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 15 July 2024, accepted 10 August 2024, date of publication 20 August 2024, date of current version 4 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3446708

RESEARCH ARTICLE

A Neural Network Architecture for Maximizing Alpha in a Market Timing Investment Strategy

JAVIER H. OSPINA-HOLGUÍN[1] **AND ANA M. PADILLA-OSPINA**[2]
[1]Department of Accounting and Finance, Universidad del Valle, Cali 760042, Colombia
[2]Department of Administration and Organizations, Universidad del Valle, Cali 760042, Colombia

Corresponding author: Javier H. Ospina-Holguín (javier.ospina@correounivalle.edu.co)

This work was supported in part by Universidad del Valle under Grant "Convocatoria Interna 131-2021 para presentación de Proyectos de Investigación y Creación Artística en las Ciencias, las Artes, las Humanidades, las Tecnologías y la Innovación."

ABSTRACT In finance, assuming more risk often corresponds to the expectation of higher, compensating returns. In this setting, alpha stands out as one of the most prevalent and refined measures of risk-adjusted return ever postulated, allowing for the estimation of the excess return that cannot be explained by the risk factors impacting an asset. This article introduces a neural network architecture designed to formulate an investment strategy with the explicit goal of maximizing alpha. The strategy, centered around market timing, determines on a daily basis, based on past returns of the risky asset, whether to fully invest in the risky asset or opt for the risk-free alternative. The neural network architecture comprises two components: a policy network for strategy implementation and an evaluation network for long-term alpha computation during parameter optimization. Employing value-weighted US size decile portfolios as risky assets, the study achieves significant out-of-sample alphas ranging from 3.6% to 8.2% per year under the q^5 asset pricing model (with a transaction cost assumption of one basis point). By construction, these alphas are not generated by risky asset growth. Robustness tests yield similar results with equal-weighted decile portfolios or under the Fama and French six-factor asset pricing model. Variations in transaction cost, number of past returns used as inputs, policy network design, or training sample size produce similar outcomes. This study underscores the effectiveness of reinforcement learning-inspired techniques in uncovering alpha in financial markets.

INDEX TERMS Alpha, asset pricing, reinforcement learning, stock returns, investment decisions, random walk hypothesis, market timing, machine learning, artificial intelligence.

I. INTRODUCTION

Financial alpha is generally considered the main aspiration of active money managers [1, p. 28]. This article introduces a neural network architecture capable of automatically trading to maximize alpha as measured by a given asset pricing model. The alpha-maximizing neural architecture is able to represent a market timing algorithm constructed from daily return data. Each day, it decides whether to invest the entire account balance in a risky portfolio and earn the daily return on that portfolio at the close of the trading day or to instead invest the entire account balance in the risk-free asset and earn the return on that risk-free asset at the close of the trading day.

Alpha, in finance, denotes the additional return that an asset generates compared to a specific benchmark. Typically, the benchmark reflects the expected return based on a given asset pricing model. Alpha represents the return above what the asset pricing model expects the asset to generate, given its exposure to given asset pricing factors. These factors are typically interpreted as sources of risk and are believed to explain the return of any asset. In modern times, alpha has become one of the most important measures of risk-adjusted return (see Section II).

Many previous works in the computer science literature on trading algorithms have shown little connection with the

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Wei Tsai.

finance literature [2]. This is evident when algorithms optimize for total return without considering risk. Any reasonable financial approach requires risk considerations in one way or another. While maximizing profit may seem desirable, algorithms that do so without accounting for risk fail to distinguish between high risk-adjusted profitability and high profitability resulting from taking more risk and simply getting lucky.

When risk is considered in the literature on stock market trading algorithms, common risk-adjusted measures of return are used, most frequently the Sharpe ratio and expected utility (see Section III). These measures are often formulated based on desirable mathematical properties. While they are not inherently flawed, they do not fully capture the significant advancements in asset pricing, especially in empirical asset pricing.

Empirical asset pricing seeks to understand and model financial returns using empirical data. The field aims to identify the factors that actually impact an asset's return based on evidence. For instance, despite being derived from first principles and reasonable mathematical assumptions, the CAPM, considered the first and most basic asset pricing model, was soon found to be empirically incorrect in asset pricing tests [3].

To illustrate the dimensions of today's financial landscape, the financial literature has already identified over 450 financial anomalies. Each of these anomalies represents unexpected (abnormal) returns systematically present across the entire market [4]. In other words, each anomaly is an empirically identified, documented, and in principle validated source of impact on asset returns. This landscape has given rise to what is colloquially known as the ''factor zoo'' [5], an enormous set of potentially new asset pricing factors believed to determine the returns of any asset.

With the passing of time, new anomalies and competing asset pricing models have been continually introduced and tested against the backdrop of previous anomalies and established models. This ongoing research has directly influenced the development of parsimonious asset pricing models, which attempt to summarize the impact of most factors and serve as the basis for calculating alpha today. Therefore, alpha, as derived from these modern parsimonious asset pricing models, represents the culmination of empirical knowledge and contemporary perspectives on explaining the risk-adjusted profitability of assets and portfolios.

Modern parsimonious asset pricing models are thus more refined than ever and are robust to the multitude of systematic abnormal returns empirically encountered. For instance, out of the hundreds of documented anomalies, approximately 65% could not be replicated by [4]. However, of the 150 anomalies replicated by [4], the q^5 model used in this article is capable of explaining all but 23 at standard significance levels, or 6 at more stringent significance levels that account for data mining [6].[1] Generating alpha within the framework of modern asset pricing models is deemed challenging enough that several companies have incorporated it into their names, such as Alphabet[2] (the parent company of Google) and Seeking Alpha (''a crowd-sourced content service for financial markets'').

Additionally, various studies on funds suggest that they are unlikely to consistently achieve alpha, except for a limited subset of such funds [7], [8]. Similar opinions are held regarding individual investors [9]. The significant rise in the popularity of passive investing [10] and smart beta investing [11] underscores the inherent difficulty in consistently generating alpha through active management. Moreover, for institutional investors like hedge funds, alpha emerges as one of the crucial performance measures today, alongside several variants based on it [1, pp. 27–32].

Although it is difficult to determine whether the alpha of a particular investment portfolio is a return generated by active investing or if it is due to the use of an inadequate or incomplete factor model, a question that remains unanswered is whether investment portfolios that maximize alpha through market timing can be automatically constructed when alpha is measured using a prespecified asset pricing model.

The present article provides a positive answer to this question. Previous literature on trading algorithm strategies had largely overlooked alpha, along with other important financial principles. Our literature review revealed only two instances of alpha-optimization trading algorithms: [12], [13]. This article aims to fill this gap in the literature by developing trading strategies that leverage sound financial principles, with the use of alpha as a risk-adjusted measure of return being its primary strategy.

In this context, the contributions of this article are as follows:

First, this article presents the first nontrivial neural network architecture that enables the automatic design of algorithms to maximize alpha based on a given asset pricing model by exploiting patterns in a certain input information set. By using alpha as a performance measure, the resulting algorithm achieves an abnormal return beyond what is expected based on the asset pricing model used, by design. In our framework, it is trivial to modify or reuse the algorithm so that it maximizes alpha in a better asset pricing model if the currently used asset pricing model is deemed to be incorrect or incomplete. After all, as the saying goes, ''one person's alpha is another's beta'' [14], meaning that estimated positive alphas could arise in insufficient asset pricing models that would disappear in more complete models with additional risk factors—a problem similar to the omitted variable problem in regressions [14]. In our proposal, it is also easy to optimize alpha using other types of past information, even

[1] In a robustness check, we also employ the popular six-factor Fama–French model with RMW [58].

[2] See https://abc.xyz/.

though we only use short-term past returns to make trading decisions.

Also following financial practices, the profitability achieved through the algorithm introduced here is measured from a zero-cost arbitrage position. This position involves being long on the asset that the algorithm indicates should be bought (the risky portfolio or the risk-free asset, depending on the case) and short on the underlying portfolio, which is the risky portfolio. In other words, the earnings of the original underlying portfolio are subtracted from the position taken by the algorithm each day. The advantage of this methodology is that it allows measuring the intrinsic profitability of the investment rule and not simply the profitability of the underlying portfolio that is used to invest. For example, if all the algorithm does is buy and hold the underlying risky portfolio for the entire investment period, the return on the zero-cost arbitrage position each day will be zero, and so will the alpha. Often, when a zero-cost arbitrage position is not used when trading and an algorithm reports high returns, it is almost impossible to distinguish whether this is due to the performance of the algorithm itself or because the portfolio on which the algorithm is based has increased the algorithm's performance. For instance, during an investment period in which the underlying portfolio is only increasing, a profitability measure that does not use a zero-cost arbitrage position may suggest that a buy-and-hold algorithm is excellent, given that the algorithm would have simply ridden the "growth wave" of the underlying portfolio value. However, in reality, such an algorithm did not earn anything beyond what the underlying portfolio was earning, and no one would pay much for such an algorithm, since it is useless in comparison to just buying the underlying portfolio.

Similarly, the algorithm we propose to optimize alpha takes into account transaction costs by design. This ensures that the achieved alpha is viable considering a certain level of transaction costs. While this paper uses a low overall transaction cost (though accessible even to non-institutional investors through modern brokerage firms such as Interactive Brokers LLC), it is essential to consider transaction costs in trading algorithms a priori. Some authors do not consider transaction costs [15], or only report breakeven transaction costs (BETC)[3] [12]. However, it is important to factor in transaction costs from the outset, as they can influence and alter the optimal trading strategy choices.

Finally, the alpha reported in our results is calculated out-of-sample and using several distinct portfolios in various conditions. Therefore, the data used to construct the algorithm were not employed to calculate the reported alpha. This strengthens the evidence that the alpha reported in this work is genuine and not simply an artifact of data snooping or overfitting.

[3]The breakeven transaction cost is the highest transaction cost that the algorithm can tolerate without incurring losses.

II. ALPHA: BASIC BACKGROUND

This section reviews the concept of alpha and its interpretation for readers who may not be familiar with it.

In finance, alpha represents the additional return that an asset generates beyond a specific benchmark. Typically, this benchmark is estimated by the expected return predicted by a particular asset pricing model.[4] In a conventional (static and linear) asset pricing model, the expectation of any risky portfolio's excess returns r is expressed as a linear combination of expected premiums:

$$\mathrm{E}[r] = \beta_1 \mathrm{E}[\gamma_1] + \cdots + \beta_k \mathrm{E}[\gamma_k]. \tag{1}$$

Alpha can then be measured as the non-expected return α by estimating the linear equation:

$$r_t = \alpha + \beta_1 \gamma_{1t} + \cdots + \beta_k \gamma_{kt} + \varepsilon_t, \tag{2}$$

using ordinary least squares (OLS), where $t = 1, \ldots, T$.

Typically, asset pricing models are interpreted in terms of risk. In this context, $\boldsymbol{\gamma}_l = (\gamma_{lt})_{t=1}^{T}$ is interpreted as a column vector representing the risk premium varying over time for the l-th risk factor, with $l = 1, \ldots, k$. The scalar β_l, on the other hand, represents the amount of asset exposure to the l-th risk factor. The term ε_t represents idiosyncratic risk, assumed to have an average value of zero. Thus, the scalar constant α denotes the (abnormal) return obtained above (or below) the chosen asset pricing model benchmark. This return cannot be explained by the risk factor exposures in the asset pricing model. (For this reason, it is also called the pricing error.) If the asset pricing model is correct and fully explains the return of every risky asset, α is expected to be zero for any given asset.

For example, in the basic capital asset pricing model (CAPM) [16], [17], where $k = 1$, there is a single risk premium and a single risk exposure. The risk premium in the CAPM is the market risk premium, representing the excess return of the market portfolio. A simple numerical example using the CAPM may better illustrate the meaning and importance of alpha (and beta). Suppose we have a risky asset X with an excess return r^X of 15% per annum and a risk-free rate of 3% per annum. Therefore, the risky asset has a (raw) return of 15%+3% = 18% per annum. This may seem like an excellent investment. However, the perspective can be quite different when accounting for risk exposures.

Relative to the one-factor CAPM model, we can compute alpha (and beta) by running the following time-series regression for asset X:

$$r_t^X = \alpha + \beta_{\mathrm{Mkt}} r_{\mathrm{Mkt}} + \varepsilon_t \tag{3}$$

The only risk factor in the CAPM, given by $r_{\mathrm{Mkt},t}$, is the excess return of the market. This factor $r_{\mathrm{Mkt},t}$ measures the

[4]In our case, the "asset return" is represented by the return—in excess—of the market timing algorithm applied to the underlying risky portfolio *minus* the return—in excess—of that same underlying portfolio. The term "in excess" denotes that the return of the risk-free asset is subtracted from the original raw return. However, according to our previous definition of "asset return," original raw returns may be used instead of excess returns since the risk-free rate gets canceled in the subtraction.

movement of the market's excess return as a whole. The market beta term, β_{Mkt}, represents the exposure to the market risk factor. For example, if the regression (3) for asset X yields an estimated β_{Mkt} of 0.8, this indicates that when the market moves up (or down) by 1%, asset X's excess return will move in the same direction by $0.8 \times 1\% = 0.8\%$.

If the CAPM is the correct asset pricing model, the estimated α should be zero for every asset. Suppose that asset X actually has an estimated alpha of zero, an estimated beta of 0.8, and the aforementioned excess return, r_t^X, of 15% per annum. While the asset's performance seems quite impressive in terms of excess return, the value of alpha reveals that it is not. After all, asset X simply moves up (or down) in the same direction as (and probably in response to) the movements of the whole market. An investor could achieve exactly the same excess return by simply investing 80% of her portfolio money in a low-cost market index and holding 20% in cash. No advanced investment techniques are required to replicate the excess returns of asset X.

On the other hand, consider a very different set of assumptions. Suppose that X actually has an estimated alpha of 15%, an estimated beta close to zero, and assume the same excess returns for X of $r^X = 15\%$ annually. In this case, it makes no difference whether the entire market goes down or up; the excess return will always be (on average) 15% per annum. Furthermore, it will be impossible to replicate this return by simply investing a constant amount of money (over time) in a market index and the rest in cash. Knowing how to replicate X's return is now not only highly non-trivial but also highly desirable. If the only major risk in investing, as recognized by the CAPM, is the movement of the overall market, then asset X is indeed immune to that risk and will always deliver a 15% annual excess return regardless of what happens in the market. This is perhaps why Pedersen [1, p. 28] asserts that alpha is clearly the most desirable term in the regression, referring to an equation similar to (3).

Alpha also plays a crucial role in understanding investment styles. Within the context of alpha, investors can be classified in two classic ways: passive and active investors. A passive investor typically subscribes to the belief that markets are efficient and that a given asset pricing model is accurate. If the asset pricing model being utilized is (sufficiently) accurate, it will reliably predict the expected return of any asset (or portfolio), and it will not be possible to earn anything beyond the benchmark of what is expected given the risk exposures [18, p. 703]. Consequently, there would be no incentive to seek positive alpha, at least not consistently.

This implies that for a passive investor, there would be no incentive to seek out and analyze special information about the assets in the portfolio or to employ sophisticated trading strategies to achieve alpha or "beat the benchmark" [18, p. 233]. The acquisition of such information and the development of trading strategies would only result in unrecovered costs [18, p. 233]. Therefore, the passive investor would typically aim to minimize trading costs by opting to buy and hold a sufficiently diversified portfolio, such as a market index [18, p. 234].

An active investor, on the other hand, would specifically aim to earn more than her chosen benchmark—that is, beyond what is expected based on her exposure to the risk factors. In essence, she would strive to maximize alpha. To achieve this objective, she would likely seek to gather any special information about the assets in the portfolio and capitalize on this information through specialized trading strategies. Alternatively, she might seek to exploit other investors' behavioral biases. (Even newer investment approaches, such as smart beta or factor index investing, can also be understood within the context of alpha.)

III. LITERATURE REVIEW

For the past two decades, return forecasting models have been a popular subject in the literature. Initially, these models were often based on classical econometric algorithms [19]. With the rise of machine learning, new stock market forecasting techniques have emerged. Modern stock market forecasting models often originate from machine learning approaches utilizing supervised learning algorithms. In contrast to classical econometric analysis, which focuses on parameter estimation, supervised learning aims to predict outcomes directly by discovering a forecast function that exploits complex and often nonlinear patterns in the relationship between input and output variables in a generalizable way [20]. Supervised learning often involves *regularization*, a technique intended to prevent overfitting. A *regularizer* attempts to constrain the complexity of the forecast function being sought or constructed. Therefore, the construction of the forecast function usually involves two steps: "The first step is, conditional on a level of complexity, to pick the best in-sample loss-minimizing function. The second step is to estimate the optimal level of complexity using empirical tuning," such as out-of-sample cross-validation [20]. This approach fine-tunes the forecast function for optimal out-of-sample performance using a hold-out sample not previously seen during the algorithm's construction [20]. The algorithm's final performance is then typically reported on a third testing sample that has not been previously seen.

In the context of stock market forecasting, two main classes of supervised learning models exist: regression and classification [21]. Regression models predict future asset return values or price levels, while classification models predict the future direction of the return—whether it will go upwards or downwards.

Accuracy has been the most used measure in evaluating the performance of models predicting stock return direction through classification [22]. However, other related measures, such as hit ratio, precision, recall, F_1 score, and balanced accuracy, have also been reported [23]. These measures assess the success of classification in various ways, often by comparing the forecasted return direction to the actual return direction, as observed in a testing sample.

As in the classical econometric approach, supervised learning regression models are traditionally evaluated using error-based metrics, such as root mean square error (RMSE), or more advanced ones like mean absolute prediction error (MAPE) [23], [24]. Other commonly used metrics in this field have a similar nature [23], [24]. Additionally, paired *t*-tests have been used for statistical comparisons between regression-based returns and a benchmark [23].

The common emphasis on minimizing prediction error or loss function, rather than maximizing return or risk-adjusted return through a trading rule, is a challenge shared by classical and supervised learning models. In their review, Kumbure et al. [23] noted that only a minority of machine learning studies reported return-based measures, such as the rate of return or average return. When utilizing forecasts to inform a market timing algorithm, determining the appropriate action based solely on predicted information can be challenging. Specifically, deciding when to buy or sell the risky portfolio may be unclear. As Neely et al. [25] stated, ''the forecasting problem is not equivalent to finding an optimal trading rule, although the two are clearly linked. A profitable trading rule may forecast rather poorly much of the time, but perform well overall because it is able to position the trader on the right side of the market during large moves.''

The limitations of trading to minimize forecast error become more pronounced when transaction costs are factored into the trading algorithm. Even a market timing algorithm with perfect forecasting, which predicts whether the return of the risky portfolio will be greater or less than the return of the risk-free asset and buys or sells accordingly, may not be profitable due to the substantial number of trades and associated costs. In other words, the algorithm may propose an excessive number of transactions that fail to offset the full costs incurred. Therefore, even when similar risk-adjusted measures of return, such as those used in this study, are reported—as seen in [26], which assesses the sources of risk in forecasts and their performance under transaction costs—those forecasts are not necessarily optimized to consider the sources of risk or transaction costs, as we do in this work.

The literature most closely related to this proposal involves the use of reinforcement learning (RL) or evolutionary computation (EC) to guide investment decisions. In reinforcement learning, the emphasis is on discovering an optimal trading algorithm rather than an optimal forecast. A reinforcement learning algorithm or *policy* determines the next *action* to take based on the current *state*, which describes the *environment*. For instance, it decides whether to buy or sell a risky asset based on the current information about its past returns. The algorithm learns the optimal policy of actions based on the actions taken and the *rewards* earned. In this way, the trading algorithm is explicitly designed to optimize a reward measure, rather than the more common focus on forecast error.

In the context provided, a standard market timing algorithm can be viewed as a dynamic (intertemporal or multiperiod) portfolio optimization problem featuring only two assets: the risky portfolio and the risk-free asset. Analytical solutions for such problems in continuous time, optimizing expected utility, have existed since at least 1969 [27]. These solutions even include considerations for transaction costs since at least 1988 [28]. However, these problems often become intractable when formulated in more realistic terms [29, p. 225]. Nevertheless, they prove to be well-suited for the reinforcement learning framework, as acknowledged in discrete time since at least 2001 [30]. Indeed, the reinforcement learning framework facilitates the direct search for policies that optimize returns or risk-based measures of returns, even when using more realistic descriptions of states, environments, or rewards.

Another area of related research that has influenced the use of reinforcement learning in finance involves the endeavor to extend the single-period mean-variance portfolio optimization problem proposed by Markowitz [31] to a discrete multiperiod setting through reinforcement learning [32]. It is important to note that in the mean-variance portfolio choice problem, the return measure is represented by the mean of the return series, and risk is quantified by the variance of the return series. A clear and insightful description of the relationship between discrete-time portfolio optimization and reinforcement learning is given in [33]:

> One can postulate that the portfolio optimization problem can be reformulated as a discrete-time (partially observable) Markov Decision Process (MDP) and hence as a stochastic optimal control, where the system being controlled in discrete time is a portfolio consisting of multiple investments, and the control is the portfolio weights (fractions of capital allocation). The problem is then solved by a sequential maximization of portfolio returns as rewards in a Bellman optimality equation. If the MDP is fully deterministic (or state transition probabilities are known) and if a reward function is also known, the Bellman optimality equation can be solved using a recursive backward value iteration method of Dynamic Programming (DP). If, on the other hand, the system dynamics is unknown and the optimal policy should be computed from samples, one can use model-free Reinforcement Learning (RL) to solve the problem. In portfolio optimization, neither the future returns of investments nor the state transition probabilities are known. Consequently, the MDP is nondeterministic and one can use RL for the problem.

Model-free RL approaches have become increasingly popular because they do not rely on investment return modeling [33]. These approaches do not require an understanding of the underlying return dynamics because they can approximate a Bellman optimality equation using only sample data [33].

The commonly used reward measure in this literature is profitability or total profit [34], [35]. However, alternative approaches have sought to maximize risk-based measures of return, such as the Sharpe ratio or the Sortino ratio, along with their variants [34], [35], [36]. (The Sharpe ratio assesses risk-adjusted return by dividing the average excess return—over a target—by the overall volatility of the portfolio, as measured by the standard deviation of excess return. The Sortino ratio replaces the entire standard deviation with the ''downside deviation,'' which excludes positive portfolio fluctuations and solely considers the variability associated

with losses relative to the target return.) Some forms of expected utility have also been optimized as a reward [37]. Additionally, alphas from several modern asset pricing models were reported in the results of an RL algorithm in a prior study [38]. However, it is worth noting that the algorithm examined in [38] optimizes for the out-of-sample Sharpe ratio, while our work focuses specifically on optimizing for alpha. To the best of our knowledge, no (nontrivial) neural network architectures have been proposed with the specific goal of maximizing alpha, as described in this study.

There is yet another series of studies that have aimed to discover optimal rules for trading—typically through evolutionary computation (EC)—but not necessarily utilizing modern neural network architectures. EC is a field of Natural Computation inspired by the evolutionary mechanisms of nature, as understood from a neo-Darwinian perspective [39]. Neo-Darwinism integrates the principles of Darwinian evolution with modern knowledge of its basic mechanisms such as DNA [39]. EC attempts to solve an optimization problem through the evolution of a population of candidate solutions that coexist in parallel [39]. To ''breed'' each new generation, genetic search operators (typically *mutation* and *crossover*) as well as *selection* operators are used. Mutation and crossover enable the creation of new candidate solutions from modifications of existing ones through operators analogous to mutation and sexual reproduction in organic species. The selection process allows the identification of the best among the ''bred'' candidate solutions so that they constitute the next generation of the population. Specifically, selection refers to the process by which solutions with the worst *fitness* are discarded from the population. Here, fitness is a quantitative measure of the effectiveness of the candidate solution in solving the specific problem at hand [39]. The search and selection processes are iteratively repeated until optimal or satisfactory solutions are found. Within the framework of EC, each strategy or transaction rule is typically represented by a candidate solution, with fitness indicating a desirable characteristic of the strategy, such as its overall profitability.

Evolutionary algorithms offer a number of advantages over more traditional optimization methods, including: First, greater flexibility. These algorithms can be applied to problems with non-differentiable or discontinuous objective functions, such as the space of certain potential rules [25], [40]. Second, due to the stochastic nature of the search and selection operators, evolutionary algorithms are less likely to converge to a local optimum than other methods [40]. Finally, evolutionary algorithms can be employed for problems with very large search spaces and are easily parallelizable [40]. For example, the manual searches conducted in past decades for the most profitable combination of technical indicators (i.e., buy or sell indicators or signals based on past prices and transaction volumes) can be efficiently automated using EC [39].

The majority of studies in EC in finance evaluate their rules by simulating their operation in the market and reporting total return or profits [41]. While some works also report the Sharpe ratio as a measure of risk-adjusted return [41], the fitness function usually directly incorporates these measures of return or risk-adjusted return. Among these studies, most consider transaction costs and compare their results with an index or a buy-and-hold strategy [41]. Other researchers focus on metrics that are more closely related to prediction, such as RMSE, MAPE, hit rate, mean absolute error (MAE), and accuracy [41]. In portfolio theory applications, there has also been progress in incorporating more realistic constraints into single-period mean-variance portfolio optimizations through EC, or by utilizing advanced risk measures beyond variance, such as mean absolute downside semi-deviation, value-at-risk, and expected shortfall [39].

Within the area of EC, we identified two instances closely related to our work. Both instances focus on optimizing for alpha. In [12], genetic programming (GP) is used to maximize Carhart's [42] (also known as Fama and French four-factor) alpha of a zero-cost arbitrage trading strategy based on an algorithm applied to each of the US volatility decile portfolios. The trading rules are encoded by basic functional trees with four levels of operations: Boolean operators (''if-then-else'', ''and'', ''or'') determining buy or sell signals at the first level, relational operators (''$<$'' or ''$>$'') returning 0 or 1 values at the second level, real functions encompassing various technical analysis indicators and applicable to both numerical values and time series of numbers at the third level, and inputs (primarily prices or returns) at the last level. Each strategy decides whether to take a long position in the risky asset, a short position, or hold the risk-free asset. The fitness function computes alpha but explicitly excludes candidate solutions (trading strategies) with alphas having a small p-value in the Carhart [42] regression and solutions unable to withstand trading costs of at least 25 basis points [12]. Time-averaged rolling out-of-sample alphas, averaged across multiple optimization runs, are reported and found to be substantial [12].

In [13], a different approach is taken, using the simplest neural network, a perceptron, or McCulloch–Pitts neuron—a linear combination of inputs introduced by [43] in 1943. This model aims to maximize alpha through differential evolution (DE), another evolutionary algorithm. The inputs consist of past contiguous returns of US size decile portfolios, and each strategy is represented by a Heaviside function of a linear combination of these returns plus a constant (bias), where the output 1 represents being 100% in the risky portfolio and 0 represents being 100% in the risk-free asset. An a priori transaction cost of 1 basis point is imposed for trading the risky asset. The study identifies Fama and French five-factor [44] sizable alphas, as well as Carhart's [42] sizable alphas, for a zero-cost arbitrage trading strategy in the test sample. The main takeaway from [13] is that an evolutionary algorithm can achieve a form of RL, as previously suggested by [45] using evolution strategies. Section V-D) compares the approach of the present work with the state-of-the-art approach in [13], positioning the current work as an intermediate point between two lines of research: those employ-

ing RL and those maximizing alpha through evolutionary techniques.

IV. METHODOLOGY

A. PROBLEM

This article proposes a neural network architecture capable of trading automatically, choosing to invest 100% of the capital in a risky asset or 100% of the capital in a risk-free asset, in such a way that alpha is maximized in a given asset pricing model. In the reinforcement learning context, the neural network architecture can be interpreted as an agent that takes actions in an environment, in this case, the market [46, p. 48].

To describe the way the neural network architecture was built, it is worth first stating the problem in a simple way:

Suppose there is a policy a_{t-1} that decides at each moment $t-1$ what action to carry out based on the state of the market or environment s_{t-1}; that is, the state stores the information used to make the decision to carry out that action. In this algorithm, the action taken yesterday a_{t-1} can be only one of two possible values that we will designate as 1 or 0.

When the action taken yesterday $a_{t-1}= 0$, the algorithm orders for the next day to be (or continue) "out of the market". In other words, if the investor was 100% in the risky portfolio yesterday, the next day, the investor will sell the risky portfolio and buy the risk-free asset. If the investor was 100% in the risk-free asset yesterday, the investor will continue with this risk-free asset the next day. Thus, when $a_{t-1}= 0$, the return for the investor from yesterday to today by being long in the algorithm is:

$$\tilde{r}_t = r_t^f \tag{4}$$

where r_t^f is the return on the risk-free asset.

However, in this work, we want to determine how much additional profitability the algorithm achieves over the risky portfolio return. The purpose of measuring additional returns in this way is not to attribute to the algorithm the returns that the underlying risky portfolio may already have. To do this, the return r_t on a zero-arbitrage position, long in the asset that the algorithm indicates should be bought (the risky portfolio or the risk-free asset, as the case may be) and short in the underlying portfolio, that is, short in the risky portfolio, is measured as:

$$r_t = \tilde{r}_t - R_t \tag{5}$$

where R_t is the return of the risky portfolio. Thus, when $a_{t-1}= 0$,

$$r_t = r_t^f - R_t. \tag{6}$$

Similarly, when the action taken yesterday $a_{t-1} = 1$, the algorithm orders for the next day to be (or continue) "in the market". If the investor was 100% in the risk-free asset yesterday, the next day, the investor will sell the risk-free asset and buy the risky portfolio. If the investor was 100% in the risky portfolio, the investor will continue with this risky portfolio the next day. According to the above, when $a_{t-1} = 1$, the return from yesterday to today for the investor being long in the algorithm is:

$$\tilde{r}_t = R_t \tag{7}$$

where R_t is the return of the risky portfolio.[5] Therefore, when $a_{t-1} = 1$, the return r_t on the zero-arbitrage position, long on the asset that the algorithm indicates should be bought and short on the underlying portfolio, which is the risky portfolio, is:

$$r_t = \tilde{r}_t - R_t = R_t - R_t = 0. \tag{8}$$

In general, note how the return from yesterday to today on the zero-arbitrage position can be written in terms of the action taken yesterday as follows:

$$\begin{aligned} r_t &= R_t a_{t-1} + (1 - a_{t-1})\, r_t^f - R_t \\ &= \left(R_t - r_t^f\right) a_{t-1} - \left(R_t - r_t^f\right) \\ &= \left(R_t - r_t^f\right) (a_{t-1} - 1)\,. \end{aligned} \tag{9}$$

The above expression can be further generalized to include a one-way transaction cost C when the risky portfolio is bought or sold. Following, for example, [47], [48], we may assume that there is no cost for trading the risk-free asset. Then, if the function representing the transaction cost is c, such a function can be expressed through the actions taken yesterday and the day before yesterday as follows:

$$\begin{aligned} c\,(a_{t-1}, a_{t-2}) &= \begin{cases} 0, & \text{if } a_{t-2} = 1 \wedge a_{t-1} = 1 \\ C, & \text{if } a_{t-2} = 1 \wedge a_{t-1} = 0 \\ 0, & \text{if } a_{t-2} = 0 \wedge a_{t-1} = 0 \\ C, & \text{if } a_{t-2} = 0 \wedge a_{t-1} = 1 \end{cases} \\ &= C\left(1 - \delta_{a_{t-1}a_{t-2}}\right) \\ &= C\,|a_{t-1} - a_{t-2}| \end{aligned} \tag{10}$$

where C denotes the one-way cost of buying or selling the risky portfolio, δ_{ij} represents the Kronecker delta and $|\cdot|$ denotes the absolute value. Introducing this cost function c, the yesterday to today return of the zero-arbitrage position can be written in terms of the actions of yesterday and the day before yesterday in the following form:

$$\begin{aligned} r_t &= \left(R_t - r_t^f\right)(a_{t-1} - 1) - c(a_{t-1}, a_{t-2}) \\ &= \left(R_t - r_t^f\right)(a_{t-1} - 1) - C\,|a_{t-1} - a_{t-2}|. \end{aligned} \tag{11}$$

Now, our purpose is to optimize alpha (hopefully in an out-of-sample generalizable way) given the vector of zero-arbitrage position returns in time $\boldsymbol{r} = (r_t)_{t=3}^T$ and given the vectors of each of the k asset pricing factors in time $\boldsymbol{\gamma}_l = (\gamma_{lt})_{t=3}^T$, with $l = 1, \ldots, k$. That is, the algorithm must

[5] Asset-pricing models customarily use the excess risky portfolio return here, but since in this work we examine the return r_t on a zero-arbitrage position, using the raw return for the risky portfolio is also possible (see footnote 1).

take a sequence of actions represented by the vector $\boldsymbol{a} = (a_t)_{t=1}^{T-2}$ that maximizes the alpha (α) of a given asset pricing model defined in terms of the values of its factors $\boldsymbol{\gamma}_l = (\gamma_{lt})_{t=3}^{T}$, with $l = 1, \ldots, k$.

From (2), we have, in matrix notation:

$$\begin{aligned} r_t &= \alpha + \sum_{l=1}^{k} \beta_l \gamma_{lt} + \varepsilon_t \\ \boldsymbol{r} &= \begin{pmatrix} 1 & \boldsymbol{\gamma}_1 & \boldsymbol{\gamma}_2 & \cdots & \boldsymbol{\gamma}_k \end{pmatrix} \cdot \\ &\qquad \begin{pmatrix} \alpha & \beta_1 & \beta_2 & \cdots & \beta_k \end{pmatrix}^\top + \boldsymbol{\varepsilon} \\ \boldsymbol{r} &= \boldsymbol{\Gamma B} + \boldsymbol{\varepsilon}, \end{aligned} \tag{12}$$

where $t = 3, \ldots, T$, and $\boldsymbol{\varepsilon} = (\varepsilon_t)_{t=3}^{T}$. Here, $\boldsymbol{B} = (\alpha\ \boldsymbol{\beta})^\top = (\alpha\ \beta_1\ \beta_2\ \cdots\ \beta_k)^\top$ is a $(1+k) \times 1$ column vector, $\boldsymbol{\beta} = (\beta_1\ \beta_2\ \cdots\ \beta_k)$ is a $1 \times k$ vector of factor exposures, and $\boldsymbol{\Gamma} = \begin{pmatrix} 1 & \boldsymbol{\Gamma}^\star \end{pmatrix}$, where $\boldsymbol{\Gamma}^\star = \begin{pmatrix} \boldsymbol{\gamma}_1 & \boldsymbol{\gamma}_2 & \cdots & \boldsymbol{\gamma}_k \end{pmatrix}$. Each $\boldsymbol{\gamma}_l$ is a $(T-2) \times 1$ column vector containing the l-th risk factor values over time $t = 3, \ldots, T$, with $l = 1, \ldots, k$.

Therefore, we can compute an estimated alpha using ordinary least squares (OLS) from (12) as:

$$\begin{aligned} \hat{\alpha} &= \left(\hat{\boldsymbol{B}}\right)_1 \\ &= \left(\left(\boldsymbol{\Gamma}^\top \boldsymbol{\Gamma}\right)^{-1} \boldsymbol{\Gamma}^\top \boldsymbol{r}\right)_1 \\ &= \left(\boldsymbol{\Gamma}^{-1} \boldsymbol{P}\, \boldsymbol{r}\right)_1, \end{aligned} \tag{13}$$

where $\hat{\boldsymbol{B}}$ is the OLS estimate of $\boldsymbol{B}$, $\boldsymbol{r}$ is the vector of returns (generated by $\boldsymbol{a}$), $(\boldsymbol{v})_1$ denotes the first element of the vector $\boldsymbol{v}$, and $\boldsymbol{P}$ is the projection matrix $\boldsymbol{\Gamma}\left(\boldsymbol{\Gamma}^\top \boldsymbol{\Gamma}\right)^{-1} \boldsymbol{\Gamma}^\top$.

Using this notation, we aim to minimize the total reward:

$$\begin{aligned} &\min_{(a_t)_{t=1}^{T-2}} \left[-\hat{\alpha}\right] \\ &= \min_{(a_t)_{t=1}^{T-2}} \left[-(\hat{\boldsymbol{B}})_1\right] \\ &= \min_{(a_t)_{t=1}^{T-2}} \left[-\left(\left(\boldsymbol{\Gamma}^{-1}\boldsymbol{P}\right)\boldsymbol{r}\right)_1\right], \end{aligned} \tag{14}$$

since this is equivalent to maximizing the OLS estimate of alpha. Note that $\boldsymbol{\Gamma}^{-1}\boldsymbol{P}$ does not depend on the actions, but $\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{a})$ does.

The optimization problem is recursive and complex, especially since an optimal action a_{t-1} depends on the choice previously made a_{t-2} due to (11) and specifically due to the form of the expression $c(a_{t-1}, a_{t-2})$. Since this is valid at any time $t \geq 3$, a_{t-1} is potentially a function of the sequence $a_1, a_2, a_3, \ldots, a_{t-2}$.

The neural network architecture we propose to solve this problem in a simple way is composed of two neural networks: a policy network and an evaluation network. The policy network simultaneously determines the vector of actions $\boldsymbol{a}$ as a parametric function where each $a_{t-1} = f(\boldsymbol{\theta}, \boldsymbol{s}_{t-1})$, $\boldsymbol{\theta}$ is a vector of trainable parameters, and $\boldsymbol{s}_{t-1}$ is the state of the market (the environment) at $t-1$. In our case, $\boldsymbol{s}_{t-1}$ is fully observable and does not contain actions. Moreover, the function f is a proper parametric stochastic function (for example, a neural network) and is stable in time, as is $\boldsymbol{\theta}$. So,

$$\begin{aligned} \boldsymbol{a} &= (a_t)_{t=1}^{T-2} \\ &= \left(f(\boldsymbol{\theta}, \boldsymbol{s}_t)\right)_{t=1}^{T-2} \\ &= \left(f(\boldsymbol{\theta}, \boldsymbol{s}_1), \cdots, f(\boldsymbol{\theta}, \boldsymbol{s}_{T-2})\right). \end{aligned} \tag{15}$$

According to the policy network, to obtain a sequence of optimal actions, it is enough to obtain the optimal vector of parameters $\boldsymbol{\theta}$.

The above simplification is consistent with the way a significant fraction of rule-based investors trade, for example, technical analysts. In some of the trading rules they use, technical analysts assume that a market timing trade decision can be made based on past price history (we use past return history instead). In this context, their transaction rules are stable over time (even deterministic), and the state of the environment does not contain the history of the decisions made.

For example, consider a standard double moving average crossover rule in technical analysis, which suggests holding the risky asset at time t if the shorter moving average at time $t-1$ is greater than the longer moving average at time $t-1$, or holding the risk-free asset otherwise. Here, a moving average at time $t-1$ represents the average of daily prices from $t-m-1$ up to $t-1$, where $m \in N$. The parameter m takes the value m_s for the shorter moving average and m_l for the longer moving average, with $1 \leq m_s < m_l$ [49].

While the profitability of technical analysis remains a matter of research and debate [50], the existence of possible patterns of predictability in the market, especially in portfolios, has been empirically demonstrated in several studies [51], [52]. However, how do you incorporate the fact that we want to maximize alpha (and not directly, say, total return)? Or how do you incorporate the fact that the investment rule must consider transaction costs that may depend on the entire sequence of previous actions? To answer both questions, the evaluation network is introduced.

The evaluation network calculates the agent's final reward based on the actions it has taken and depends on the expression (14). Remember that if $\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{a})$ is the set of all zero-arbitrage returns that the investor earns over time, the final reward is the alpha $(\hat{\boldsymbol{B}})_1$ calculated from this $\boldsymbol{r}$ and a certain asset pricing model described and specified by $\boldsymbol{\Gamma}$.

According to this, the evaluation network maximizes alpha as follows:

$$\begin{aligned} &\min_{(a_t)_{t=1}^{T-2}} \left[-\hat{\alpha}\right] \\ &= \min_{(a_t)_{t=1}^{T-2}} \left[-(\hat{\boldsymbol{B}})_1\right] \\ &= \min_{(a_t)_{t=1}^{T-2}} \left[\left(\left(\boldsymbol{\Gamma}^{-1}\boldsymbol{P}\right)\boldsymbol{r}\right)_1\right], \end{aligned} \tag{16}$$

where $\boldsymbol{r}(\boldsymbol{a})$ is the vector of returns in terms of the actions, $(\boldsymbol{v})_1$ denotes the first element of the vector $\boldsymbol{v}$, and $\boldsymbol{P}$ is the

projection matrix $\boldsymbol{\Gamma}\left(\boldsymbol{\Gamma}^{\top}\boldsymbol{\Gamma}\right)^{-1}\boldsymbol{\Gamma}^{\top}$. As noted above, only $\boldsymbol{r}=\boldsymbol{r}(\boldsymbol{a})$ depends on the actions (as $\left(\boldsymbol{\Gamma}^{-1}\boldsymbol{P}\right)$ does not), and thanks to the simplification made through the policy network, according to which $a_t = f(\boldsymbol{\theta}, \boldsymbol{s}_t)$, for a given functional form f and given state $\boldsymbol{s}_t$, $\boldsymbol{r}$ only depends on the vector of parameters $\boldsymbol{\theta}$.

B. THE NEURAL NETWORK ARCHITECTURE MODEL: BASE MODEL

In the neural network architecture model, the policy network is represented by a function $f(\boldsymbol{\theta}, \boldsymbol{s}_{t-1})$ that depends on a vector of trainable parameters $\boldsymbol{\theta}$ and the state $\boldsymbol{s}_{t-1}$ to decide which action $a_{t-1} = f(\boldsymbol{\theta}, \boldsymbol{s}_{t-1})$ to take every previous day. In the base model, the neural network f used to describe the policy is a simple two-layer feed-forward neural network. This neural network is composed of an initial linear layer (with biases) with a size-two output, a logistic sigmoid activation function, and a second final linear layer (without biases) with a size-two output. Finally, a softmax layer is applied to assign actions to 0 or 1. That is, for each state, carrying out the policy leads to either 1, representing being ''in the market'', or 0, representing being ''out of the market''. Only the first output of the softmax layer is used.

In the context of reinforcement learning, the information used by the policy network to decide which daily action to take is given by the characteristics encoded into the state. In this way, the state contains the variables that describe the environment in which the agent acts. In the base model, the state is the vector that represents the returns of the risky portfolio over the last $L = 10$ days (cf. [53]), that is,

$$\boldsymbol{s}_{t-1} = (R_\tau)_{\tau=t-(L-1)-1}^{t-1}. \tag{17}$$

Note how the state does not include information about past trading actions (in contrast with [53], for example). Such information is only encoded in the evaluation network. Also note how consecutive states are not independent. Although this form of state is quite simple, the complete neural network architecture is capable of obtaining enough information from it to generate alpha.

In the base model, the neural network architecture then uses the q^5-model [6], a five-factor model, as the asset pricing model in the evaluation network. This means that the definitions of $\boldsymbol{\Gamma}$ and $\boldsymbol{P}$ in (16) refer to the q^5-model factors.[6] The q^5-model improved on the original Hou-Xue-Zhang q-factor model [54] by adding an additional factor, for a total of five factors.

The first factor in the q^5-model refers to the market factor. The market factor is built as the market return over the risk-free rate. This fundamental factor was already present as early as in the original CAPM, the first asset-pricing model.

The second factor in the q^5-model is a size factor. This factor is constructed as the return on a portfolio of stocks of small capitalization companies minus the return on a portfolio of stocks of large capitalization companies. This factor attempts to capture the size effect, according to which small companies tend to outperform large ones [54].

The third factor in the q^5-model is an investment factor. The investment factor is constructed as the return on a portfolio of stocks of companies with low investment minus the return on a portfolio of stocks of companies with high investment. According to [54], given expected cash flows, the net present values of new capital projects, and therefore the investment itself, diminish when the cost of capital is high, and vice versa. For this reason, investment and returns are related.

The fourth factor in the q^5-model is a profitability factor, constructed as the return on a portfolio of stocks of companies with high ROE (return on equity) and the return on a portfolio of stocks of companies with low ROE. According to [54], ROE is related to returns because, given low investment, high expected ROE requires high discount rates, and vice versa. If the discount rates were not higher when expecting a higher ROE, firms would deduce high net present values of new capital and invest more instead, and vice versa [54].

The fifth factor added to the q^5-model is an expected growth factor. This factor is constructed as described in [6]. According to [6, p. 2]:

> In the investment theory, firms with high expected investment growth should earn higher expected returns than firms with low expected investment growth, holding current investment and expected profitability constant. Intuitively, if expected investment is high next period, the present value of cash flows from next period onward must be high. Consisting primarily of the present value of cash flows from next period onward, the benefit of investment this period must also be high. As such, if expected investment is high next period relative to current investment, the discount rate must be high to offset the high benefit of investment this period to keep current investment low.

As stated in the introduction, despite its parsimony, the q^5-model has been very successful at explaining most systematic abnormal returns (or anomalies) that have been previously identified and replicated. For example, of 150 anomalies that were found to be replicable by [4], out of a total of over 450 anomalies examined, the q^5-model [6] could explain all but 23 at standard significance levels. Or all but 6 at more stringent significance levels that take potential data mining into account. This may seem to imply that finding abnormal returns using the q^5-model should be an inherently difficult test to pass for any proposed algorithm.

To sum up the workings of the base model, the evaluation network specifically uses the information from the q^5-model factors to determine $\left(\boldsymbol{\Gamma}^{-1}\boldsymbol{P}\right)$ and calculate alpha, as in (16). The expression $\left(\boldsymbol{\Gamma}^{-1}\boldsymbol{P}\right)$ depends only on the factors, and since alpha is calculated as the first component of the vector $\left(\boldsymbol{\Gamma}^{-1}\boldsymbol{P}\right)\boldsymbol{r}(\boldsymbol{a})$, the other pieces of information required to compute alpha are the returns $\boldsymbol{r}(\boldsymbol{a}) = \boldsymbol{r}\Big(\big(f(\boldsymbol{\theta}, \boldsymbol{s}_1), \cdots, f(\boldsymbol{\theta}, \boldsymbol{s}_{T-2})\big)\Big)$ which do not depend on the factors at all. Indeed, the returns only depend on the states and the vector of trainable parameters $\boldsymbol{\theta}$ as defined in the policy network. Then, alpha is optimized by changing this vector $\boldsymbol{\theta}$.

[6] As published in https://global-q.org/factors.html by the authors of [6].

Note that the policy's functional form and parameters $f(\boldsymbol{\theta}, \cdot)$ in the policy network, although stochastic, remain unchanged over states and therefore remain unchanged over time. Since the function f and the $\boldsymbol{\theta}$ vector remain unchanged for the whole sample during every round of alpha evaluation, what the evaluation network is doing is an optimization where the policy network is successively applied with $\boldsymbol{\theta}$ parameters that are effectively shared in time throughout the optimization.

In line with modern optimization practices, the model is fit in a time-ordered first training set, but then a contiguous time-ordered validation set is used to select the $\boldsymbol{\theta}$ parameters that render the smallest (validation) loss. Both the training and the validation sets are chosen of equal lengths. The use of a logistic function in the policy network, instead of a deterministic Heavisde function as described in [13] or instead of a sign function as described in a similar problem in [30], as well as the stochastic nature of the policy, instead of a deterministic one as in [12] or [13], help allow the loss in (16) to be optimizable through stochastic descent in what would otherwise be a nondifferentiable optimization.

Indeed, the maximization of alpha in the neural network architecture is accomplished in the base model by minimizing the loss, computed as the negative of alpha, through ADAM [55], a variant of stochastic gradient descent that uses an adaptive learning rate. Here, alpha can be seen as a stochastic scalar function differentiable with respect to the parameters in $\boldsymbol{\theta}$. In the ADAM optimization scheme, adaptive estimates of lower-order moments are used for the optimization, whereas the invariance of the learning rate to diagonal rescaling of the gradients is guaranteed [55].

Initially, to prevent overfitting, a global L_2 regularization with a regularization parameter of 0.01 was used. As an additional form of early stopping to prevent overfitting, the optimization algorithm was also initially stopped when the absolute change of the loss did not improve by at least 0.001 for more than five rounds. However, these safeguards were later found to be unnecessary and were omitted in the end.

The results of the optimization are the optimal parameters, $\boldsymbol{\theta}^{\text{opt}}$. They make up the optimal policy. Then, to evaluate the out-of-sample performance of the neural network architecture, it is enough to apply this optimal policy to data outside the original (training and validation) sample and calculate the alpha in a similar way to the one already described in (13), but with $\left(\boldsymbol{\Gamma}^{-1}\boldsymbol{P}\right)\boldsymbol{r}\left(\boldsymbol{a}\right)$ using out-of-sample factors and out-of-sample returns, with the returns determined by $f(\boldsymbol{\theta}^{\text{opt}}, \cdot)$ and the new out-of-sample states.

Out-of-sample alpha in the testing sample is measured (and reported) as annual percentage points by multiplying the daily alpha by 252, where 252 is the average number of trading days per annum.[7] The returns are correspondingly measured as percentage points. Additionally, when computing alpha, a one-way transaction cost of one basis point is used, i.e., 0.01. This cost was on the lower end of the spectrum in the financial literature toward 2000 [47], but it has also been used as a more recent minimum benchmark in modern reinforcement learning studies on finance [13], [53].

V. RESULTS AND DISCUSSION

A. BASE MODEL

To illustrate the ability of the neural network architecture to generate alpha, we use each of the 10 size decile portfolios (i.e., market capitalization decile portfolios) of the three major US stock exchanges (NYSE, NASDAQ, AMEX) as a risky portfolio separately.[8] These portfolios result from ordering all the shares of the NYSE stock market according to the market capitalization of the companies they represent. To do this, the data at the end of June for each company are used, and breakpoints are calculated resulting from grouping the shares ordered by size from the NYSE into ten equally large subsections. These breakpoints are then used to assign the stocks of all three exchanges (NYSE, NASDAQ, AMEX) into ten groups or deciles. Portfolios are rebalanced at the end of each June with the same procedure. Accordingly, the daily return of the p-th risky portfolio is the value-weighted daily return of all stocks on the three exchanges (NYSE, NASDAQ, AMEX) assigned to the p-th decile group. Specifically, the first decile $p = 1$ denotes the decile group of the smallest companies, and the last decile $p = 10$ denotes the decile group of the largest companies. The risk-free rate corresponds to the thirty-day T-bill. The use of value-weighted portfolios in the base model is very important, given that previous literature has suggested that microcaps explain many anomalies and that such anomalies tend to disappear when using value-weighted portfolios instead of equal-weighted portfolios [4].

The evaluation and the policy networks used in the base model are the ones described in Section IV-B. Once the policy network optimal vector, $\boldsymbol{\theta}^{\text{opt}}$, is found, the optimal policy is applied out-of-sample, and the alpha it produces is measured in the testing set. Thus, the alphas are measured using data not previously used during the construction of the optimal policy. Instead, the out-of-sample testing set is simply used to calculate the alpha with the policy already established during training and validation.

The in-sample evaluation uses the $(5400-L-1)$-returns sample from January 18, 1967, to June 28, 1988, as the training set, with the length of the state being $L = 10$ in the base model. The in-sample evaluation also uses a $(5400-L-1)$-returns sample, but from June 29, 1988, to November 10, 2009, as the validation set. The testing set begins on November 11, 2009, and goes until December 31, 2021, for a total of a little over 12 years.

It is important to note that the dataset of states begins in every case $L + 1$ days before the starting date of the returns sample, given that each state has L past returns and given that two consecutive states (and correspondingly two consecutive

[7] While this annualization scheme is technically valid only under independent and identically distributed daily returns, it is the standard in the finance literature.

[8] Available at Kenneth French's website: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/.

actions) are needed to determine the return of the algorithm according to (11). For example, for $L = 10$, the first state in the training sample goes from January 03, 1967, to January 16, 1967, the second from January 04, 1967, to January 17, 1967, and so on.

Table 1 reports the summary statistics of the out-of-sample risky portfolios, and Table 2 reports the out-of-sample alpha achieved by the algorithm. The returns of the risky portfolio are subtracted from the returns obtained from the algorithm's position in the portfolio to obtain a zero-arbitrage position (see Section I); thus, the alpha is not the naïve result of just following the underlying risky portfolio growth (for example, a buy and hold strategy on the risky portfolio would provide a zero alpha in this setting).

The q^5-model alphas achieved are of economic significance, never going below 3.64% per annum and going up to 8.21% per annum in the best scenario, which occurred when using the portfolio with the smallest firms as the underlying portfolio. Furthermore, in eight of the ten portfolios examined, the algorithm achieves a statistically significant alpha at the 95% confidence level.

B. POTENTIAL EXPLANATION OF RESULTS

If the algorithm can achieve such high economic alphas by solely utilizing the last ten days of risky portfolio returns to inform its actions, a pertinent question arises: why does the algorithm perform so well?

To explore this, consider an experiment where actions (either 1 or 0) are randomly chosen with a 50% probability and applied to each portfolio. Repeat this experiment 1000 times for each portfolio and measure alpha each time. An interesting observation arises: the median alpha for each portfolio is less than zero in statistical terms.

Now, compute an optimal sequence of actions for each portfolio following the proposed base model algorithm. Then, calculate the ratio ϕ_p of 1-actions to the total number of actions, for every p-th decile portfolio. This ϕ_p ratio indicates the likelihood of investing in the risky asset if optimal actions are followed. Now use this ratio ϕ_p to define a Bernoulli distribution to randomly choose actions (either 1 or 0). Repeat this experiment 1000 times. Despite the median alpha remaining statistically less than zero for every portfolio, each new median alpha is economically higher in every portfolio compared to the previous median alpha computed when actions were chosen with a 50% probability.[9]

All of this suggests that the ϕ_p ratio provides useful information about when to invest in the risky or risk-free asset. However, this information alone is insufficient to achieve a statistically positive alpha when the sequence of actions is randomized. Thus, the order, and therefore the history, of the optimal actions are crucial.

Interestingly, the trained policy networks for each portfolio are essentially the same, sharing identical weights and biases (denoted as $\boldsymbol{\theta}^{\text{opt}}$ above). Even when the optimizer's random seeds are changed, they converge to the same $\boldsymbol{\theta}^{\text{opt}}$ for every portfolio, indicating that the network exploits similar patterns across different portfolios. Furthermore, optimization for each portfolio is achieved in only one round, encompassing all returns in the training and validation sets simultaneously, as per (16). While it may seem unlikely that $\boldsymbol{\theta}^{\text{opt}}$ is exactly the same for different portfolios, it appears that the ADAM optimization converges to the same local optimum for each portfolio.

The former suggests that differences between any two optimal sequences arise from variations in input states rather than differences in the policy functions themselves. Nevertheless, optimal actions across portfolios are also highly similar: the percentage of common elements in out-of-sample optimal actions between any two distinct portfolios ranges from 81.3% to 95.1%.

Consequently, the policy network appears to learn similar patterns from the input data across different portfolios, although these specific patterns are challenging to pinpoint. For instance, it was observed that returns from risky portfolios exhibit temporal patterns that are likely exploitable. This conclusion emerged from applying a standard variance ratio test for a random walk to each portfolio.[10] However, even when the in-sample time series of risky portfolios are shuffled randomly before training, the out-of-sample alpha for the deciles does not disappear. This suggests that the algorithm does not only learn temporal patterns in the data.

In summary, the algorithm's efficacy appears to stem from its ability to recognize complex patterns in the data. These patterns include not only temporal variations in the states, which are inherited from the risky portfolio series but also potentially cross-sectional and temporal patterns in the factors and the risk-free rate series. The algorithm effectively captures the interrelations between these elements, which collectively contribute to generating alpha.

C. ROBUSTNESS TESTS OF THE NEURAL NETWORK ARCHITECTURE MODEL

In robustness tests, various modifications of the base model were evaluated. For this, the following cases were considered:

1) EQUAL-WEIGHTED RISKY PORTFOLIOS INSTEAD OF VALUE-WEIGHTED RISKY PORTFOLIOS

In this scenario, the base model is applied to the size (i.e., market capitalization) decile portfolios of the three main US market exchanges (NYSE, NASDAQ, AMEX) but constructed with equal weights. This means that the daily return of the p-th risky portfolio is calculated as the daily return of an equal-weighted portfolio of all the shares of the

[9] Similar results are obtained even when estimating a discrete-time, two-state Markov process for each portfolio and simulating it 1000 times.

[10] A random walk test using wild bootstrapping [67], conducted in overlapping segments of exactly ten days with 2000 bootstrap iterations, was applied to the entire return series of each value-weighted US size decile portfolio. Only the returns of the decile 9 portfolio were found to be "random," meaning they did not reject the random walk hypothesis at the 90% confidence level.

TABLE 1. Summary statistics of value-weighted US size decile portfolios.

Decile	Mean	Std. Dev.	Skewness	Kurtosis	Sharpe R.
Small	13.62*** [0.00]	20.75	−0.05	0.05	0.63
2	16.45*** [0.00]	24.53	−0.03	0.04	0.65
3	16.63*** [0.00]	23.80	−0.04	0.04	0.68
4	15.38*** [0.00]	22.81	−0.03	0.04	0.65
5	15.94*** [0.00]	22.11	−0.03	0.05	0.70
6	15.69*** [0.00]	21.33	−0.04	0.06	0.71
7	15.23*** [0.00]	19.96	−0.04	0.06	0.74
8	17.27*** [0.00]	18.92	−0.05	0.06	0.89
9	15.79*** [0.00]	17.89	−0.04	0.07	0.86
Large	15.82*** [0.00]	16.96	−0.03	0.07	0.91

Descriptive statistics for the value-weighted US size decile portfolio returns (each represented by R_t in the base model), as used in the testing sample from November 11, 2009, to December 31, 2021. "Std. Dev." stands for standard deviation and "Sharpe R." for Sharpe ratio. A robust one-sample sign test for the null hypothesis that the median return is zero was applied to each portfolio. P-values are in square brackets. *** denotes significance at the 1% level for that test.

TABLE 2. Algorithm's alphas using value-weighted US size decile portfolios: Base model case.

Decile	α	β_{Mkt}	β_{Me}	$\beta_{I/A}$	β_{Roe}	β_{Eg}
Small	**8.21**** (2.57)	−0.38*** (−12.93)	−0.32*** (−9.43)	0.10** (2.06)	0.05 (0.80)	0.11 (1.15)
2	4.20 (1.18)	−0.46*** (−13.30)	−0.36*** (−8.90)	0.04 (0.78)	−0.01 (−0.19)	0.15* (1.77)
3	**7.01**** (2.10)	−0.47*** (−13.25)	−0.32*** (−8.68)	0.07 (1.44)	−0.06 (−0.94)	0.16* (1.76)
4	**7.66**** (2.41)	−0.48*** (−14.69)	−0.28*** (−8.20)	0.08* (1.87)	−0.06 (−1.13)	0.16** (2.03)
5	**6.64**** (2.17)	−0.48*** (−13.98)	−0.20*** (−5.64)	0.05 (1.15)	−0.05 (−0.83)	0.18** (2.27)
6	**6.61**** (2.19)	−0.47*** (−12.39)	−0.14*** (−4.11)	0.08** (2.00)	−0.08 (−1.39)	0.21** (2.46)
7	**7.98***** (2.87)	−0.45*** (−12.98)	−0.06** (−2.10)	0.10** (2.40)	−0.07 (−1.21)	0.22** (2.42)
8	**5.57**** (2.15)	−0.45*** (−14.34)	0.00 (−0.01)	0.06 (1.49)	−0.06 (−1.28)	0.17** (2.11)
9	**5.09**** (2.02)	−0.44*** (−12.89)	0.05* (1.95)	0.02 (0.51)	−0.06 (−1.22)	0.16** (1.98)
Large	3.64 (1.51)	−0.41*** (−9.81)	0.10*** (2.82)	0.02 (0.50)	−0.06 (−1.35)	0.04 (0.70)

Out-of-sample results for the q^5-model time-series regression (12) are given, using the daily algorithmic zero-arbitrage position returns based on each value-weighted US size decile portfolio as the risky asset. In the algorithmic zero-arbitrage position, the risky asset portfolio return is subtracted from the return determined (the day before) by the algorithm, resulting from being either 100% in the risky asset or 100% in the risk-free asset. Thus, if the algorithm solely buys and holds the risky asset, the algorithmic zero-arbitrage position returns would be null by definition. A transaction cost of one basis point and a ten-past-daily returns state vector are employed. The policy network is represented by a vanilla two-layer feed-forward neural network. The q^5-model alphas are annualized and presented in percentage. The in-sample data from January 18, 1967, to November 10, 2009, were used to determine the optimal parameters $\boldsymbol{\theta}^{\text{opt}}$ that describe the policy network, using the last half of this sample for validation. The policy is tested out-of-sample in a testing set from November 11, 2009, to December 31, 2021. Robust Newey–West [56] t values are provided in parentheses. ***, **, and * denote significance at the 1%, 5%, or 10% level, respectively. Underlined alphas are positive (of economic significance), and bold alphas are statistically significant at least at the 10% level.

three exchanges (NYSE, NASDAQ, AMEX) assigned to the p-th decile group, instead of using value weighting. The rest of the details are left as equal to those in the base case.

Table 3, Scenario #1, gives the q^5-model alphas obtained by the algorithm. Nine out of the ten calculated alphas are significant at least at the 90% confidence level, ranging from 4.15% per annum to 9.72% per annum. It is interesting to highlight that the best six alphas achieved were all significant at least at a 95% confidence level. Equal-weighted portfolios tend to protect against outliers (in this case, outliers in size), but they often need additional rebalancing. The algorithm handles these portfolios at least as well as the value-weighted ones.

In the following two scenarios, we return to the base model of Section V-A, but with details from the policy network definition changed to test for the robustness of the results under policy variations.

2) SHORTER STATE VECTOR

In this shorter state vector scenario, the base model is computed using the standard sample of value-weighted risky portfolios but with a state that contains less information, specifically, fewer historical returns. Indeed, in this test, the state is the vector that represents the returns of the risky portfolio containing the last eight days instead of the usual ten days of the base model. Table 3, Scenario #2, reports the q^5-model alphas obtained by the algorithm. All of them are of economic significance, ranging from 2.74% per annum to 7.75% per annum. In half of the portfolios, the algorithm was able to achieve a statistically significant alpha at the 90% confidence level.

As would be expected, the decreasing of available information when passing from ten past risky portfolio returns input state vectors to eight past risky portfolio returns input state vectors weakens alpha, both statistically and economically.

3) DIFFERENT POLICY NEURAL NETWORK

In this scenario, the base model is computed as always, using US value-weighted risky portfolios, but with a different functional form for the policy network. Everything else is left unchanged in comparison with the base case.

Here, instead of a vanilla policy network, a self-normalizing network [57] is used. Specifically, the self-normalizing network is composed of an initial linear layer (with biases) with a size-two output, a SELU (Scaled Exponential Linear Unit) activation function, a dropout layer, and a second final linear layer (without biases) with a size-two output. With a probability of 0.2, the dropout layer randomly sets the input variables to zero while maintaining the input's

mean and variance (i.e., alpha dropout is used [57]). A softmax layer is then added at the end to assign actions to 0 or 1. In other words, implementing the policy results in either a 1, signifying being "in the market," or a 0, signifying being "out of the market", for each state. The softmax layer's first output is the only one that is used.

Self-normalizing networks tend to be quite flexible and general (at least in classification problems), so it is interesting to examine whether they can adequately represent the alpha-optimizing 0 or 1 actions. Unfortunately, only one of the alphas was significant, and only at the 10% level (Table 3, Scenario #3). However, all alphas were still economically positive, ranging from 0.74% per annum to 5.98% per annum. This illustrates the superiority of the vanilla policy network, perhaps due to the more complicated policy tending to overfit. (It is worth mentioning that self-normalizing neural networks often assume mean 0 and variance 1 data, a transformation we did not apply since the data were already stationary with a close-to-null mean).

4) LARGER TRAINING AND VALIDATION SETS

In this scenario, we use a larger training set and a larger validation set (Table 3, Scenario #4). Since the available data are the same, this means that the testing set was shorter. The in-sample evaluation used the $(6400-L-1)$-returns sample from January 18, 1967, to June 11, 1992, as the training set and the $(6400-L-1)$-returns sample from June 12, 1992, to October 20, 2017, as the validation set. The testing set ranges from October 23, 2017, to December 31, 2021, for a total of slightly over 4 years. Other parameters are left unchanged with respect to the base model, including the length of the state $L = 10$.

The algorithm's alphas were even more impressive, perhaps due to the additional training and validation information (Table 3, Scenario #4). All alphas were economically over 6.51% per annum. Seven of them surpassed 10% per annum, with six of those showing statistical significance at the 5% level.

5) HIGHER TRANSACTION COST

In this scenario, the base model using value-weighted risky portfolios is applied, but with a transaction cost of five basis points instead of one basis point. As would be expected, alphas deteriorate due to the higher cumulative transaction costs. All alphas stopped being significant at the 10% level. However, they remain economically important, ranging from 0.16% per annum to 4.50% per annum (Table 3, Scenario #5).

6) FF6 MODEL INSTEAD OF THE q^5 MODEL

In this scenario (Table 3, Scenario #6), the base model using the standard value-weighted risky portfolios is applied, but using the Fama-French six-factor asset pricing model [58], instead of the q^5 model. In [58], Fama and French developed an extension of their original asset pricing models: their three-factor model [59] and their five-factor model [44], adding a momentum factor, as in [42].

TABLE 3. Algorithm's alphas in robustness scenarios.

	Base	#1	#2	#3	#4	#5	#6
D	α	α	α	α	α	α	α
S	**8.21**** (2.57)	**9.72***** (3.61)	**7.37**** (2.15)	**5.98*** (1.87)	**15.16**** (2.37)	4.50 (1.40)	**8.74***** (2.67)
2	4.20 (1.18)	4.84 (1.35)	**7.75**** (2.00)	1.32 (0.37)	9.00 (1.32)	0.16 (0.04)	4.70 (1.30)
3	**7.01**** (2.10)	**6.35*** (1.81)	4.53 (1.22)	2.39 (0.70)	**13.12**** (2.06)	3.07 (0.92)	**7.38**** (2.17)
4	**7.66**** (2.41)	**8.65***** (2.65)	**6.41*** (1.96)	2.89 (0.90)	**13.67**** (2.21)	3.74 (1.17)	**8.08**** (2.53)
5	**6.64**** (2.17)	**6.46**** (1.98)	3.19 (0.98)	2.94 (0.97)	**10.37*** (1.74)	2.82 (0.92)	**7.29**** (2.39)
6	**6.61**** (2.19)	**7.52**** (2.47)	**5.67*** (1.78)	2.26 (0.77)	**11.97**** (1.98)	2.88 (0.94)	**7.11**** (2.34)
7	**7.98***** (2.87)	**8.20***** (2.87)	**5.42*** (1.95)	3.79 (1.38)	**14.03**** (2.54)	4.48 (1.60)	**8.47***** (3.03)
8	**5.57**** (2.15)	**6.42**** (2.34)	3.34 (1.30)	2.20 (0.85)	**10.80**** (2.12)	2.14 (0.82)	**6.06**** (2.34)
9	**5.09**** (2.02)	**5.53**** (2.12)	2.74 (1.19)	1.89 (0.74)	**8.33*** (1.65)	1.83 (0.72)	**5.47**** (2.17)
L	3.64 (1.51)	**4.15*** (1.66)	3.26 (1.59)	0.74 (0.31)	6.51 (1.33)	0.41 (0.17)	3.57 (1.49)

In the base model (the same as in Table II), out-of-sample results for the q^5-model time-series regression (12) are given, using the daily algorithmic zero-arbitrage position returns based on each value-weighted US size decile portfolio as the risky asset. "D" stands for Decile, "S" stands for Small, and "L" stands for Large. In the algorithmic zero-arbitrage position, the risky asset portfolio return is subtracted from the return determined (the day before) by the algorithm, resulting from being either 100% in the risky asset or 100% in the risk-free asset. Thus, if the algorithm solely buys and holds the risky asset, the algorithmic zero-arbitrage position returns would be null by definition. A transaction cost of one basis point and a ten-past-daily returns state vector are employed. The policy network is represented by a vanilla two-layer feed-forward neural network. The q^5-model alphas are annualized and presented in percentage. The in-sample data from January 18, 1967, to November 10, 2009, were used to determine the optimal parameters $\boldsymbol{\theta}^{\text{opt}}$ that describe the policy network, using the last half of this sample for validation. The policy is tested out-of-sample in a testing set from November 11, 2009, to December 31, 2021. In Scenario #1, equal-weighted US size decile portfolios are used instead of value-weighted US size decile portfolios. In Scenario #2, an eight-past-daily returns state vector is employed instead of a ten-past-daily returns state vector. Consequently, the in-sample data ranges from January 16, 1967, to November 12, 2009, and the testing set spans from November 13, 2009, to December 31, 2021. In Scenario #3, the policy network is represented by a self-normalizing network instead of a vanilla two-layer feed-forward neural network. Scenario #4 involves larger in-sample training and validation sets, with the in-sample data ranging from January 18, 1967, to October 20, 2017, and the testing set from October 23, 2017, to December 31, 2021. Scenario #5 uses a transaction cost of five basis points instead of one basis point. Scenario #6 employs the Fama–French six-factor model for the time-series regression instead of the q^5 model. Robust Newey–West [56] t values are provided in parentheses. ***, **, and * denote significance at the 1%, 5%, or 10% level, respectively. Underlined alphas are positive (of economic significance), and bold alphas are statistically significant at least at the 10% level.

Although the Fama-French six-factor model is believed to be outperformed by the q^5 model [6], it is another of the recent models with wide diffusion. The previous Fama-French five-factor model [44] explains asset returns using five risk factors: a market factor (*Mkt*), a size factor (*SMB*), a value factor (*HML*), a profitability factor (*CMA*), and an investment factor (*RMW*). The six-factor model [58] adds an additional factor to the five-factor model: a momentum factor

(*UMD*). The factor data are available at Kenneth French's website.[11]

Using the Fama-French six-factor model on the usual value-weighted risky portfolios, the algorithm delivers economically positive alphas for every portfolio. Furthermore, eight alphas, ranging from 5.47% per annum to 8.74% per annum, are significant at least at the 5% level. Such results distinctively show that the algorithm's efficacy is not due to the asset pricing model initially chosen but can be replicated successfully with other standard asset pricing factors.

D. COMPARISON TO OTHER METHODS

1) COMPARISON TO THE METHOD IN [13]

As highlighted in Section III, one of the two most similar predecessors to the present work is [13], making it valuable to draw a comparison with the method proposed therein. In [13], the action taken by the trading strategy at a given time is represented by a Heaviside function with a range of $\{0, 1\}$. Similar to this work, a value of 0 signifies being 100% invested in the risk-free asset, while a value of 1 denotes being 100% invested in the risky asset within a market timing strategy. Specifically, to determine the appropriate action, [13] computes a linear combination of past days' returns plus a constant (a bias) and applies a Heaviside function. States based on $L = 10$ past days returns, akin to the base model in this paper, are used. Alpha is directly calculated as per (13), after its expression in terms of actions contingent on Heaviside functions. The optimization function is nonlinear and non-differentiable due to the utilization of Heaviside functions, comprising fragments of parallel hyperplanes at different heights on the alpha axis. Nevertheless, this is not problematic as the optimization is conducted by differential evolution (DE), which is robust to nonlinearity and non-differentiability.

In the original study by [13], Carhart's (i.e., Fama and French four-factor) alphas [42] and Fama and French five-factor alphas [44] were computed by applying their algorithm to US size decile portfolios and assessing the return of a zero-cost arbitrage position. For comparison, the alpha of the q^5-model applied to the method in [13] is calculated here. It is worth noting that the use of the q^5-model in this context establishes a more stringent benchmark for evaluating the method in [13] than the one originally employed, given that the Fama and French models are considered to be surpassed by the q^5-model, which also boasts greater explanatory power for the anomalies documented in the literature [6].

The portfolios in this comparison are identical to those in the base model: Each of the 10 value-weighted size decile portfolios (i.e., market capitalization decile portfolios) of the three major US stock exchanges (NYSE, NASDAQ, AMEX) is used separately as a risky portfolio. The comparison involves the same a priori one basis point transaction cost for trading the risky portfolio. Given that the method in [13] does not involve training and validation, only the validation data is utilized to derive parameters that maximize the in-sample alpha. In other words, the in-sample evaluation also utilizes the $(5400-L-1)$-returns sample from June 29, 1988, to November 10, 2009. (Using the sample from January 18, 1967, to November 10, 2009, yielded qualitatively similar results but with a lower alpha for the decile of the smallest firms.) Out-of-sample results are computed and reported in a test set. The test set is the same as in the base case, spanning from November 11, 2009, to December 31, 2021, for a total of over 12 years. Table 4, Model #1, presents the outcomes of applying the method in [13] to the test set, which are fully comparable to those obtained in the base case.

The first thing to note is that the method in [13] performs well for the decile portfolio with the smallest firms. This aligns with the base model, which also reports the highest alpha for the same portfolio. The alpha achieved by the [13] method is of a comparable magnitude and significance ($\alpha = 7.88\%$ per annum with a $t = 2.42$ using [13] versus $\alpha = 8.21\%$ per annum with a $t = 2.57$ in the base case using the exact same test sample). This positive alpha suggests that the [13] algorithm can effectively exploit patterns in the underlying decile portfolio to generate risk-adjusted returns. However, it is essential to note that the method in [13] does not yield positive and significant alphas in any of the other underlying decile portfolios.

This observation might imply that the [13] algorithm is effective only in the portfolio where the signal is stronger or easier to exploit for extracting alpha. Looking at the in-sample alphas, they are notably higher than the in-sample alphas of the base model presented in this paper. Specifically, the in-sample alphas of the base model in the present study range from 4.38% per annum (for the decile of the largest firms) to 12.9% per annum (for the decile of the smallest firms). In contrast, the in-sample alphas using the method described in [13] range from 10.10% per annum (for the second decile of the largest firms) to 21.46% per annum (for the decile of the smallest firms). This discrepancy suggests that the method in [13] may be overfitting. This is not entirely surprising, since the method described in [13] does not incorporate the use of hold-out information to improve its out-of-sample performance.

In short, although the method described in [13] is significantly easier to program and execute, it is much less flexible and generalizable. It is less flexible by default, as it does not attempt to avoid overfitting through the use of a hold-out sample (in fact it tended to overfit in the conducted comparison). Moreover, its lack of flexibility extends to its failure to leverage the extensive knowledge available about neural networks that have already been developed and optimized. For example, the model presented in this work is not only agnostic to the specific architecture of the neural network used as a policy, making it easily adaptable to other types of networks, but it also takes advantage of well-known neural network training methods like automatic differentiation [60]. This adaptability could prove highly valuable when extending the method to more realistic and

[11] https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/.

complex scenarios, such as employing states with hundreds or thousands of variables, as opposed to the ten used in the base case.

2) COMPARISON TO SUPERVISED LEARNING METHODS[11]

In this section, we make an additional comparison between the proposed algorithm and standard supervised learning methods. In supervised learning, the goal is to find a general function that connects input variables to an output variable based on multiple examples of input-output pairs. Therefore, the objective of this approach is to directly construct the policy function $f(\boldsymbol{\theta}, \cdot)$ that determines the actions $a_{t-1} = f(\boldsymbol{\theta}, \boldsymbol{s}_{t-1})$, using examples of pairs of state vectors $\boldsymbol{s}_{t-1}$ and optimal actions a_{t-1}.

The main difficulty with this approach is that the actions optimizing alpha are not predetermined; therefore, there are no examples of outputs to be paired with inputs. First, alpha depends on a comprehensive history of actions, determining stock returns $\boldsymbol{r}$, as observed in (13). Second, within this framework, each action recursively depends on all preceding actions in a complex manner, as explained after (14). Third, alpha is influenced by variables not directly dictated by the actions, such as the risk factor vectors of the asset pricing model used.

These are precisely the reasons why a reinforcement learning approach was originally preferred in this work instead of supervised learning. However, to compare the proposed algorithm with supervised learning models, it is essential to have examples of optimal actions in some form. To facilitate this comparison, the optimality criterion will need to shift away from alpha maximization, as an algorithm that establishes optimal actions in this sense is currently unknown (other than those introduced in [12], [13], and this work, all of which directly provide $f(\boldsymbol{\theta}, \cdot)$ without requiring supervised learning).

Instead, we will seek to maximize the one-step ahead return (from a zero-arbitrage position of the algorithm). For simplicity, we will refer to this concept of optimality as r_t-optimality to distinguish it from the usual optimality in this work, which focuses on alpha maximization. It is important to note that r_t-optimality does not necessarily imply alpha optimality, as an r_t-optimal algorithm aims to maximize daily return alone, whereas an alpha optimization algorithm aims to maximize a complex risk-adjusted return based on an asset pricing model.

In general, finding a multi-period solution that maximizes total return is highly complex (cf. [61]). However, it is straightforward to establish an analytical criterion for one-step-ahead r_t-optimality, provided we have future return data associated with each past action in advance, as is the case with the training and validation samples.

It is important to note that given an action a_{t-2} at time $t-2$, the return r_t on the day after making the decision a_{t-1} at time $t-1$ is, according to (11):

$$r_t = \left(R_t - r_t^f\right)(a_{t-1} - 1) - C\,|a_{t-1} - a_{t-2}|. \tag{18}$$

Then, if $r(a_{t-1} = 1, a_{t-2})$ is greater than $r(a_{t-1} = 0, a_{t-2})$, the best empirical decision that could have been made at $t-1$ is $a_{t-1} = 1$. Otherwise, it is $a_{t-1} = 0$. Therefore, the condition for $a_{t-1} = 1$ can be expressed, in terms of $r_t(a_{t-1}, a_{t-2})$, as:

$$\begin{aligned} r_t(1, a_{t-2}) &> r_t(0, a_{t-2}) \\ -C\,|1 - a_{t-2}| &> -\left(R_t - r_t^f\right) - C\,|-a_{t-2}| \\ \left(R_t - r_t^f\right) - C\,(1 - 2a_{t-2}) &> 0, \end{aligned} \tag{19}$$

where the facts that $a_{t-2} \in \{0, 1\}$, $|-a_{t-2}| = a_{t-2}$ and $|1 - a_{t-2}| = 1 - a_{t-2}$ were used.

From this condition, an empirical sequence of r_t-optimal actions can be computed period by period, starting from an initial value, for example $a_1 = 0$, and calculating consecutively

$$a_{t-1} = H\left(\left(R_t - r_t^f\right) - C\;(1 - 2a_{t-2})\right) \tag{20}$$

from $t = 3$ until the end of the training and validation samples, where $H(x)$ is the Heaviside function defined as:

$$H(x) \equiv \begin{cases} 1, & x \geq 0 \\ 0, & x < 0. \end{cases} \tag{21}$$

Once the sequence of empirical r_t-optimal actions is determined, it becomes feasible to train any supervised learning algorithm using input–output pairs to establish the policy function $f(\boldsymbol{\theta}, \cdot)$ that maps the state vector $\boldsymbol{s}_{t-1}$ to the action $a_{t-1} = f(\boldsymbol{\theta}, \boldsymbol{s}_{t-1})$.

Table 4, Models #2–#5, shows the out-of-sample alphas achieved when applying the policy $f(\boldsymbol{\theta}, \cdot)$ in the test sample. This policy was obtained using four standard supervised learning models: a (self-normalizing) neural network, a gradient boosting machine, a logistic regression, and nearest neighbors. The same training and validation sample as the original algorithm were used.

For the self-normalizing neural network [57], only one training round was applied, and a very high L_2 regularization with a regularization parameter of 10 was used to prevent overfitting. The LightGBM implementation of the gradient boosting machine also used the same regularization parameter of 10. In the case of logistic regression, only one training round was applied, with an L_2 regularization parameter of 10 again used. For nearest neighbors, a high distribution smoothing parameter was employed.

All supervised methods produce alphas of economic significance for the decile of the smallest firms. However, except for the supervised learning neural network method, the performance of the other supervised learning methods is subpar.

The results from the supervised learning neural network are particularly promising. This method achieved alphas of economic significance for seven out of the ten portfolios,

[11] We appreciate the advice of an anonymous reviewer on the importance of including this comparison.

TABLE 4. Algorithm's alphas using value-weighted US size decile portfolios with alternative methods.

	Base	#1	#2	#3	#4	#5
D	α	α	α	α	α	α
S	**8.21**** (2.57)	**7.88**** (2.42)	**8.12***** (2.61)	0.76 (0.25)	5.77* 1.71	1.29 0.43
2	4.20 (1.18)	−8.18** (−2.26)	4.39 (1.28)	−8.14** (−2.47)	−2.66 (−0.73)	−7.34** (−2.12)
3	**7.01**** (2.10)	−6.53* (−1.93)	3.08 (0.89)	−5.92* (−1.81)	−1.71 (−0.47)	−4.43 (−1.35)
4	**7.66**** (2.41)	−5.31 (−1.52)	3.85 (1.19)	−5.14 (−1.49)	−3.13 (−0.99)	−3.41 (−1.04)
5	**6.64**** (2.17)	−4.93 (−1.53)	0.26 (0.09)	−5.07 (−1.63)	−1.85 (−0.61)	−1.47 (−0.51)
6	**6.61**** (2.19)	−2.35 (−0.69)	2.24 (0.79)	−2.51 (−0.79)	0.21 0.07	0.16 0.05
7	**7.98***** (2.87)	0.98 (0.37)	0.36 (0.12)	0.18 (0.07)	0.47 0.18	−2.18 (−0.80)
8	**5.57**** (2.15)	−3.50 (−1.21)	−0.25 (−0.09)	−3.57 (−1.30)	−0.85 (−0.34)	−3.93 (−1.42)
9	**5.09**** (2.02)	−8.88** (−2.57)	−0.64 (−0.28)	−1.55 (−0.70)	−1.93 (−0.85)	−1.83 (−0.74)
L	3.64 (1.51)	−0.98 (−0.42)	−3.17 (−1.45)	−0.54 (−0.20)	−4.11 (−1.55)	−5.07** (−2.11)

Out-of-sample results for the q^5-model time-series regression (12) are given, using the daily algorithmic zero-arbitrage position returns based on each value-weighted US size decile portfolio as the risky asset. "D" stands for Decile, "S" stands for Small, and "L" stands for Large. In the algorithmic zero-arbitrage position, the risky asset portfolio return is subtracted from the return determined (the day before) by the algorithm, resulting from being either 100% in the risky asset or 100% in the risk-free asset. Thus, if the algorithm solely buys and holds the risky asset, the algorithmic zero-arbitrage position returns would be null by definition. A transaction cost of one basis point and a ten-past-daily returns state vector are employed. The q^5-model alphas are annualized and presented as percentages. The base model corresponds to the proposed algorithm as shown in Table II. In Model #1, the policy network is represented by a Heaviside function applied to a linear combination of the returns of the last ten days plus a bias (a constant). The in-sample data from June 29, 1988, to November 10, 2009, were used to determine, via differential evolution, the alpha-maximizing optimal parameters that describe the policy network. (Using the sample from January 18, 1967, to November 10, 2009, provided qualitatively similar results, but with a lower alpha for the decile with the smallest firms.) In Models #2–#5, every policy is constructed using a supervised learning method, where the "r_t-optimal" actions for training were determined by empirically maximizing the day-ahead return (not alpha) in sample data from January 18, 1967, to November 10, 2009, with the last half of this sample used for validation. The supervised models then learned from pairs of state vectors (returns of the last ten days) and r_t-optimal actions, employing a self-normalizing neural network (#2), a gradient boosting machine (#3), a logistic regression (#4), and nearest neighbors (#5), respectively. Each of the six policies (the base model and the five models for comparison) had its out-of-sample alpha determined in a test set from November 11, 2009, to December 31, 2021, not previously used during training or validation. Robust Newey–West [56] t-values are provided in parentheses. ***, **, and * denote significance at the 1%, 5%, or 10% level, respectively. Underlined alphas are positive (of economic significance), and bold alphas are positive and statistically significant at least at the 10% level.

although statistical significance was only observed for the decile of the smallest firms. Economically, it outperformed all other comparison methods in terms of alpha, including the specialized method in [13]. This underscores the versatility achieved by the self-normalizing neural network in supervised learning, likely attributable to its universal approximation properties [62], along with effective strategies implemented to prevent overfitting: validation sample usage, early stopping, and rigorous regularization. While similar techniques were employed in the other supervised learning methods, they were not as effective.

However, when compared with the method introduced in this work, the results were economically inferior across all portfolios, except for the two decile portfolios containing the smallest firms, where the alphas achieved by the neural network supervised learning method were comparable to those of the proposed algorithm. Additionally, the proposed algorithm consistently yielded economically positive alphas, whereas the algorithm based on the neural network supervised learning method produced negative alphas in all three decile portfolios containing the largest firms. As our experiments have shown, these portfolios pose the greatest challenge when trying to identify and capitalize on patterns.

It is also interesting to compare the self-normalizing reinforced learning neural network used in a robustness test (Table 3, Scenario #3) with the self-normalizing supervised learning neural network (Table 4, Model #2). The supervised learning neural network exhibited highly competitive performance compared to the reinforcement learning neural network of the same structure, achieving economically superior alphas in the four decile portfolios containing the smallest firms, where signals are presumably easier to extract. However, only the reinforcement learning neural network consistently delivered economically positive alphas across all portfolios. This demonstrates that the exact same neural network structure can exhibit very different behaviors depending on the type of learning—supervised or reinforced—and the optimization criteria applied.

VI. CONCLUSION

It is well known that investors are willing to take greater risks when they expect higher returns. Consequently, an investor is interested not only in the net return on their investment but also primarily in the risk-adjusted return. Among the various measures of risk-adjusted returns in finance, alpha from an asset pricing model is perhaps the most widely used and developed today.

Through alpha, asset pricing models enable the calculation of an asset's return beyond what is expected from its exposure to impacting risk factors. Achieving significant positive alpha in investments is challenging. For instance, research on mutual funds has demonstrated that most funds do not generate alpha using relatively modern asset pricing models [8], [63].

Despite the recent increase in machine learning applications in stock market trading algorithms, no alternative had been presented thus far that allows for the direct discovery of alpha through nonlinear neural networks. The present work accomplishes this by introducing a neural network architecture capable of training a market timing algorithm that maximizes the alpha of a given linear asset pricing model. Although market timing models generally face profitability

challenges [64], the introduced algorithm has achieved economically significant and often statistically significant alphas out of sample, suggesting that these results are not due to in-sample optimization.

In studies that postulate the existence of alpha, such as the present one, inquiring into data snooping is a valid concern. Nevertheless, in robustness tests, the algorithm continued to perform well out-of-sample under a wide range of different specifications, changing the algorithm details, the risky portfolios, or the in-sample dates.

The algorithm's novelty is rooted in its use of a neural network architecture to directly maximize alpha. This architecture consists of two key components: a policy network and an evaluation network. The policy network represents a parametric investment rule that makes daily decisions about investing in a risky asset or risk-free asset based on historical returns. The evaluation network focuses on the long-term computation of alpha during the optimization of the policy network's shared-in-time parameters. By employing this dual-network structure, the algorithm learns to exploit patterns in past return data and the cross-section of returns and risk factors to make investment decisions that yield alpha, i.e., returns exceeding those predicted by the chosen asset pricing model. Furthermore, the algorithm is designed to incorporate real-world constraints, including transaction costs, in the decision-making process, and uses a zero-cost arbitrage position for performance evaluation. This approach ensures that the resulting alpha is realistic and representative of actual profitability, not an artifact of unrealistic assumptions or inflated by the underlying portfolio's performance.

Evidently, the validity of any obtained alpha depends on the validity of the asset pricing model used. In this paper, two of the most complete parsimonious modern asset pricing models were used: the q^5 model [6] and, in robustness tests, the Fama and French 6-factor model [58]. It is always possible to argue that if an alpha was obtained, it is because the model is not capturing other types of risks that should be considered. However, the neural network architecture is trivially generalizable to any other asset pricing model that is linear in its risk factors, even if it includes new factors, so that if exploitable patterns persist in the architecture's inputs, positive alpha should again be achievable.

Additionally, the introduced neural network architecture achieved its results relatively easily: the computational method is fast and requires very little information compared to alternative models for alpha optimization, such as those of [12], [13]. One of the most interesting features of this new proposal is that it can potentially be enhanced with more complex neural network models and more efficient optimization algorithms.

Another strength of the algorithm is that it not only considers transaction costs but is also easily applicable to other types of costs or restrictions, such as taxes. Additionally, although the historical returns of risky assets were used as input state vectors, the algorithm allows the incorporation of any other type of historical information to determine optimal actions, such as macroeconomic or accounting information specific to the risky portfolios. In the future, it is conceivable to develop more advanced versions of the algorithm that use further information, perhaps including supervised learning forecasts as inputs.

In summary, the present work adds a robust, generalizable, and easy-to-use tool to the existing arsenal of asset management tools. Nevertheless, the implications of automatic alpha-maximizing trading strategies for asset pricing theories remain to be explored. In principle, new anomalies and trading factors could be easily and programmatically detected in the future, but the financial sources and potential interpretation of these anomalies may remain opaque. Additionally, as these same anomalies can be easily detected by other investors, it is uncertain how persistent they will be [65]. Furthermore, a deeper exploration of the market-related consequences of these strategies, particularly in terms of market stability and systemic risk, would be valuable for future research. This is particularly important since alpha-earning active managers require a counterparty loser for every move they make in order to be successful [66].

ACKNOWLEDGMENT

The following tools and services were used to improve grammar and style: DeepL Write, Google Translate, Google Bard, ChatGPT, American Journal Experts, and Cambridge Proofreading. The authors would like to thank four anonymous reviewers and Dr. Chun-Wei Tsai, the associate journal editor, for their valuable feedback, which greatly improved the quality of this article. The content of the manuscript is the sole responsibility of the authors.

REFERENCES

[1] L. H. Pedersen, *Efficiently Inefficient: How Smart Money Invests and Market Prices Are Determined*. Princeton, NJ, USA: Princeton Univ. Press, 2015, doi: 10.2307/j.ctt1287knh.

[2] A. Brabazon, M. Kampouridis, and M. O'Neill, "Applications of genetic programming to finance and economics: Past, present, future," *Genetic Program. Evolvable Mach.*, vol. 21, nos. 1–2, pp. 33–53, Jun. 2020, doi: 10.1007/s10710-019-09359-z.

[3] D. B. Keim, "Financial market anomalies," in *The New Palgrave Dictionary of Economics*, 2nd ed., S. N. Durlauf and L. E. Blume, Eds., Basingstoke, NY, USA: Palgrave Macmillan, 2008.

[4] K. Hou, C. Xue, and L. Zhang, "Replicating anomalies," *Rev. Financial Stud.*, vol. 33, no. 5, pp. 2019–2133, May 2020, doi: 10.1093/rfs/hhy131.

[5] G. Feng, S. Giglio, and D. Xiu, "Taming the factor zoo: A test of new factors," *J. Finance*, vol. 75, no. 3, pp. 1327–1370, Jun. 2020, doi: 10.1111/jofi.12883.

[6] K. Hou, H. Mo, C. Xue, and L. Zhang, "An augmented q-factor model with expected growth," *Rev. Finance*, vol. 25, no. 1, pp. 1–41, Feb. 2021, doi: 10.1093/rof/rfaa004.

[7] K. Cuthbertson, D. Nitzsche, and N. O'Sullivan, "Mutual fund performance: Measurement and evidence," *Financial Markets, Inst. Instrum.*, vol. 19, no. 2, pp. 95–187, May 2010, doi: 10.1111/j.1468-0416.2010.00156.x.

[8] E. F. Fama and K. R. French, "Luck versus skill in the cross-section of mutual fund returns," *J. Finance*, vol. 65, no. 5, pp. 1915–1947, Oct. 2010, doi: 10.1111/j.1540-6261.2010.01598.x.

[9] Clark Center Forum. *Diversified Investing*. Accessed: Mar. 10, 2023. [Online]. Available: https://www.igmchicago.org/surveys/diversified-investing-2/

[10] K. Anadu, M. Kruttli, P. McCabe, and E. Osambela, "The shift from active to passive investing: Risks to financial stability?" *Financial Analysts J.*, vol. 76, no. 4, pp. 23–39, Oct. 2020, doi: 10.1080/0015198x.2020.1779498.

[11] T. Smith and S. Anderson. *Smart Beta: Explanation, Strategy and Examples*. Accessed: Mar. 10, 2023. [Online]. Available: https://www.investopedia.com/terms/s/smart-beta.asp

[12] J. Brogaard and A. Zareei, "Machine learning and the stock market," *J. Financial Quant. Anal.*, vol. 58, no. 4, pp. 1431–1472, Jun. 2023, doi: 10.1017/s0022109022001120.

[13] J. H. Ospina-Holguín and A. M. Padilla-Ospina, "The search for time-series predictability-based anomalies," *J. Bus. Econ. Manage.*, vol. 23, no. 1, pp. 1–19, Nov. 2021, doi: 10.3846/jbem.2021.15650.

[14] B. T. Kelly and D. Xiu, "Financial machine learning," 2023, doi: 10.2139/ssrn.4501707.

[15] T. L. Meng and M. Khushi, "Reinforcement learning in financial markets," *Data*, vol. 4, no. 3, p. 110, Jul. 2019, doi: 10.3390/data4030110.

[16] W. F. Sharpe, "Capital asset prices: A theory of market equilibrium under conditions of risk," *J. Finance*, vol. 19, no. 3, pp. 425–442, Sep. 1964, doi: 10.1111/j.1540-6261.1964.tb02865.x.

[17] J. Lintner, "The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets," *Rev. Econ. Statist.*, vol. 47, no. 1, pp. 13–37, Feb. 1965, doi: 10.2307/1924119.

[18] Z. Bodie, A. Kane, and A. Marcus, *ISE Essentials of Investments*, 12th ed., Columbus, OH, USA: McGraw-Hill Education, 2021.

[19] G. S. Atsalakis and K. P. Valavanis, "Surveying stock market forecasting techniques—Part I: Conventional methods," in *Computation Optimization in Economics and Finance Research Compendium*, C. Zopounidis, Ed., New York, NY, USA: Nova Science, 2013, pp. 49–104.

[20] S. Mullainathan and J. Spiess, "Machine learning: An applied econometric approach," *J. Econ. Perspect.*, vol. 31, no. 2, pp. 87–106, May 2017, doi: 10.1257/jep.31.2.87.

[21] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. New York, NY, USA: Springer, 2013.

[22] O. Bustos and A. Pomares-Quimbaya, "Stock market movement forecast: A systematic review," *Exp. Syst. Appl.*, vol. 156, Oct. 2020, Art. no. 113464, doi: 10.1016/j.eswa.2020.113464.

[23] M. M. Kumbure, C. Lohrmann, P. Luukka, and J. Porras, "Machine learning techniques and data for stock market forecasting: A literature review," *Exp. Syst. Appl.*, vol. 197, Jul. 2022, Art. no. 116659, doi: 10.1016/j.eswa.2022.116659.

[24] G. Kumar, S. Jain, and U. P. Singh, "Stock market forecasting using computational intelligence: A survey," *Arch. Comput. Methods Eng.*, vol. 28, no. 3, pp. 1069–1101, May 2021, doi: 10.1007/s11831-020-09413-5.

[25] C. Neely, P. Weller, and R. Dittmar, "Is technical analysis in the foreign exchange market profitable? A genetic programming approach," *J. Financial Quant. Anal.*, vol. 32, no. 4, pp. 405–426, Dec. 1997, doi: 10.2307/2331231.

[26] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *Eur. J. Oper. Res.*, vol. 270, no. 2, pp. 654–669, Oct. 2018, doi: 10.1016/j.ejor.2017.11.054.

[27] R. C. Merton, "Lifetime portfolio selection under uncertainty: The continuous-time case," *Rev. Econ. Statist.*, vol. 51, no. 3, pp. 247–257, Aug. 1969, doi: 10.2307/1926560.

[28] J. F. Eastham and K. J. Hastings, "Optimal impulse control of portfolios," *Math. Oper. Res.*, vol. 13, no. 4, pp. 588–605, Nov. 1988, doi: 10.1287/moor.13.4.588.

[29] A. Rao and T. Jelvis, *Foundations of Reinforcement Learning With Applications in Finance*. Boca Raton, FL, USA: CRC Press, 2022, doi: 10.1201/9781003229193.

[30] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 4, pp. 875–889, Jul. 2001, doi: 10.1109/72.935097.

[31] H. Markowitz, "Portfolio selection," *J. Finance*, vol. 7, no. 1, pp. 77–91, Mar. 1952, doi: 10.2307/2975974.

[32] B. Hambly, R. Xu, and H. Yang, "Recent advances in reinforcement learning in finance," *Math. Finance*, vol. 33, no. 3, pp. 437–503, Jul. 2023, doi: 10.1111/mafi.12382.

[33] Y. Sato, "Model-free reinforcement learning for financial portfolios: A brief survey," 2019, *arXiv:1904.04973*.

[34] J. Wang, Y. Zhang, K. Tang, J. Wu, and Z. Xiong, "AlphaStock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Jul. 2019, pp. 1900–1908, doi: 10.1145/3292500.3330647.

[35] X.-Y. Liu, H. Yang, J. Gao, and C. D. Wang, "FinRL: Deep reinforcement learning framework to automate trading in quantitative finance," in *Proc. 2nd ACM Int. Conf. AI Finance*, New York, NY, USA, Nov. 2021, pp. 1–9, doi: 10.1145/3490354.3494366.

[36] J. Moody, L. Wu, Y. Liao, and M. Saffell, "Performance functions and reinforcement learning for trading systems and portfolios," *J. Forecast*, vol. 17, nos. 5–6, pp. 441–470, Sep. 1998, doi: 10.1002/(SICI)1099-131X(1998090)17:5/6%3C441::AID-FOR707%3E3.0.CO;2-%23.

[37] P. N. Kolm and G. Ritter, "Modern perspectives on reinforcement learning in finance," 2019, doi: 10.2139/ssrn.3449401.

[38] L. Cong, K. Tang, J. Wang, and Y. Zhang, "AlphaPortfolio for investment and economically interpretable AI," 2020, doi: 10.2139/ssrn.3554486.

[39] A. Brabazon, M. O'Neill, and I. Dempsey, "An introduction to evolutionary computation in finance," *IEEE Comput. Intell. Mag.*, vol. 3, no. 4, pp. 42–55, Nov. 2008, doi: 10.1109/MCI.2008.929841.

[40] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules," *J. Financial Econ.*, vol. 51, no. 2, pp. 245–271, Feb. 1999, doi: 10.1016/s0304-405x(98)00052-x.

[41] Y. Hu, K. Liu, X. Zhang, L. Su, E. W. T. Ngai, and M. Liu, "Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review," *Appl. Soft Comput.*, vol. 36, pp. 534–551, Nov. 2015, doi: 10.1016/j.asoc.2015.07.008.

[42] M. M. Carhart, "On persistence in mutual fund performance," *J. Finance*, vol. 52, no. 1, pp. 57–82, Mar. 1997, doi: 10.1111/j.1540-6261.1997.tb03808.x.

[43] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943, doi: 10.1007/bf02478259.

[44] E. F. Fama and K. R. French, "A five-factor asset pricing model," *J. Financial Econ.*, vol. 116, no. 1, pp. 1–22, Apr. 2015, doi: 10.1016/j.jfineco.2014.10.010.

[45] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv:1703.03864*.

[46] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., Cambridge, MA, USA: MIT Press, 2018.

[47] A. W. Lynch and P. Balduzzi, "Predictability and transaction costs: The impact on rebalancing rules and behavior," *J. Finance*, vol. 55, no. 5, pp. 2285–2309, Oct. 2000, doi: 10.1111/0022-1082.00287.

[48] P. Balduzzi and A. W. Lynch, "Transaction costs and predictability: Some utility cost calculations," *J. Financial Econ.*, vol. 52, no. 1, pp. 47–78, Apr. 1999, doi: 10.1016/S0304-405X(99)00004-5.

[49] C.-H. Park and S. H. Irwin, "The profitability of technical analysis: A review," 2004, doi: 10.2139/ssrn.603481.

[50] C.-H. Park and S. H. Irwin, "What do we know about the profitability of technical analysis?" *J. Econ. Surv.*, vol. 21, no. 4, pp. 786–826, Sep. 2007, doi: 10.1111/j.1467-6419.2007.00519.x.

[51] K.-P. Lim, W. Luo, and J. H. Kim, "Are US stock index returns predictable? Evidence from automatic autocorrelation-based tests," *Appl. Econ.*, vol. 45, no. 8, pp. 953–962, Mar. 2013, doi: 10.1080/00036846.2011.613782.

[52] A. W. Lo and A. C. MacKinlay, "Stock market prices do not follow random walks: Evidence from a simple specification test," *Rev. Financial Stud.*, vol. 1, no. 1, pp. 41–66, Jan. 1988, doi: 10.1093/rfs/1.1.41.

[53] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, Mar. 2017, doi: 10.1109/TNNLS.2016.2522401.

[54] K. Hou, C. Xue, and L. Zhang, "Digesting anomalies: An investment approach," *Rev. Financial Stud.*, vol. 28, no. 3, pp. 650–705, Mar. 2015, doi: 10.1093/rfs/hhu068.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[56] W. K. Newey and K. D. West, "A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix," *Econometrica*, vol. 55, no. 3, pp. 703–708, May 1987, doi: 10.2307/1913610.

[57] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," 2017, *arXiv:1706.02515*.

[58] E. F. Fama and K. R. French, "Choosing factors," *J. Financial Econ.*, vol. 128, no. 2, pp. 234–252, May 2018, doi: 10.1016/j.jfineco.2018.02.012.
[59] E. F. Fama and K. R. French, "Common risk factors in the returns on stocks and bonds," *J. Financial Econ.*, vol. 33, no. 1, pp. 3–56, Feb. 1993, doi: 10.1016/0304-405x(93)90023-5.
[60] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 5595–5637, Jan. 2017. [Online]. Available: https://dl.acm.org/doi/10.5555/3122009.3242010
[61] S. Boyd, E. Busseti, S. Diamond, R. N. Kahn, K. Koh, P. Nystrup, and J. Speth, "Multi-period trading via convex optimization," *Found. Trends Optim.*, vol. 3, no. 1, pp. 1–76, 2017, doi: 10.1561/2400000023.
[62] P. Kidger and T. Lyons, "Universal approximation with deep narrow networks," 2019, *arXiv:1905.08539*.
[63] L. Barras, O. Scaillet, and R. Wermers, "False discoveries in mutual fund performance: Measuring luck in estimated alphas," *J. Finance*, vol. 65, no. 1, pp. 179–216, Feb. 2010, doi: 10.1111/j.1540-6261.2009.01527.x.
[64] L. Bodson, L. Cavenaile, and D. Sougné, "A global approach to mutual funds market timing ability," *J. Empirical Finance*, vol. 20, pp. 96–101, Jan. 2013, doi: 10.1016/j.jempfin.2012.11.001.
[65] R. D. Mclean and J. Pontiff, "Does academic research destroy stock return predictability?" *J. Finance*, vol. 71, no. 1, pp. 5–32, Feb. 2016, doi: 10.1111/jofi.12365.
[66] R. D. Arnott, C. Brightman, V. Kalesnik, and L. Wu, "Earning alpha by avoiding the index rebalancing crowd," *Financial Analysts J.*, vol. 79, no. 2, pp. 76–97, Apr. 2023, doi: 10.1080/0015198x.2023.2173506.
[67] J. H. Kim, "Wild bootstrapping variance ratio tests," *Econ. Lett.*, vol. 92, no. 1, pp. 38–43, Jul. 2006, doi: 10.1016/j.econlet.2006.01.007.

JAVIER H. OSPINA-HOLGUÍN was born in Cali, Colombia. He received the B.S. degree in physics and the M.Sc. degree in organizational sciences from Universidad del Valle, Cali, in 2001 and 2007, respectively, the M.Sc. degree in economics from the University of Amsterdam, Amsterdam, The Netherlands, in 2011, and the Ph.D. degree in business administration, with a concentration in finance, from Universidad del Valle, in 2018.

He is currently a Full Professor with the Department of Accounting and Finance, Universidad del Valle. His research interests include financial forecasting and econometrics, through machine learning and reinforcement applications to finance, to empirical asset pricing and portfolio theory.

Prof. Ospina-Holguín has received several scholarships and awards throughout his career, such as a Pacific Alliance Scholarship, a Tinbergen Institute Full Scholarship, and a first place standing in Colombia's national high school exit examination (ICFES).

ANA M. PADILLA-OSPINA was born in Cali, Colombia. She received the bachelor's degree in business administration, the M.Sc. degree in organizational sciences, and the Ph.D. degree in administration from Universidad del Valle, in 2007, 2013, and 2019, respectively.

Since 2020, she has been an Assistant Professor in foreign trade and international business with the Faculty of Administrative Sciences, Universidad del Valle. She has also participated in international projects, such as Project 617RT0531 (2018–2022) through the Open Network for Foresight and Innovation for Latin America and the Caribbean funded by the Ibero-American Program of Science and Technology for Development (CYTED) and the Technical-Economic Proposal for the Design and Execution of an Agrifood Prospective study, through the Inter-American Commission on Science and Technology (COMCYT) of the Organization of American States (OAS) (2021–2022) funded by the Organization of American States (OAS) and Minciencias. She is the author of more than 17 articles and a book chapter. Her research interests include innovation financing, innovation activity development, social innovation, competitive development strategies, currency analysis, and economic value-added assessment.

Dr. Padilla-Ospina was a recipient of Grant 617 of 2013 from Colombian Ministry of Science, Technology and Innovation (Minciencias), the Grant for Doctoral Students of 2016 from Universidad del Valle and the Pacific Alliance Scholarship, in 2016.