IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

OPTIMAL RISK RECURRENT NEURAL NETWORK IN FINANCE PORTFOLIO MANAGEMENT

Manya shahi

student

punjab university

Abstract

It becomes a more problematic issue in a world where markets, competition, organizations, products, and services are increasing in complexity. In a global Portfolio framework characterized by a high level of capital intensity, technological complexity, and industrial B2B complexity, decision-makers and product managers are confronted with greater repercussions. The constraint produces a convex optimization problem that ensures the drawdown risk restriction is adhered to. The profitable outcome of the precise analysis in the development of the stock market can be linked to the implications of the stock's financial behavior and the precise implications of the algorithm selection method for the development of the financial. Based on an expanded Kelly criterion, we present a brand-new objective function for optimizing recurrent neural networks in this research. We construct a limit for the probability of drawdown and replace the original risk with this limit. Numerical experiments demonstrate that our bound-on drawdown probability, calculated using Monte Carlo simulation, is reasonably close to the actual drawdown risk.

Keywords- Portfolio Management, Financial, Investment, Drawdown probability, Recurrent Neural Network

1.Introduction

Activities such as adopting new strategies, introducing new management structures, and strengthening public services are carried out across programmes and portfolios [1]. In recent years, project portfolio management has garnered increased interest as companies introduce more concurrent initiatives [2]. Project portfolio management considers the viability of all projects to achieve the organization's goals, whereas project management focuses on the completion of specific or individual projects [3]. Consequently, the most crucial aspect of portfolio management is the selection of the most suitable initiatives at the correct moment to achieve the organization's

goals [4]. Activities such as adopting new strategies, introducing new management structures, and strengthening public services are carried out across programmes and portfolios [5]. In recent years, project portfolio management has garnered increased interest as companies introduce more concurrent initiatives [6]. Project portfolio management considers the viability of all projects to achieve the organization's goals, whereas project management focuses on the completion of specific or individual projects [7].

Because of this, the most crucial aspect of portfolio management is choosing the best initiatives at the correct moment to achieve the organization's goals. The relevance of project portfolio management as a method for selecting and managing an organization's projects has grown, and it is now widely seen as pivotal to the successful and efficient management of [8] One definition of a project portfolio is "a collection of projects being managed by the same entity that must compete for the same pool of resources". There are a few different definitions floating in the literature, but they all very much agree on the same things. According to [9]"managerial activities that relate to the initial screening, selection, and prioritisation of project proposals, the concurrent reprioritization of projects in the portfolio, andthe allocation and reallocation of resources to projects according to priority" is the definition of project portfolio management.. Thus, project selection is a method for evaluating potential projects or groups of projects and deciding which ones to put into action to best achieve the organization's objectives [10]. The study's goal is to establish realistic suggestions for the case firms' IT project portfolio management procedures. This research is being performed because Projektiyhditys (PRY), the case organisation, is interested in learning more about the condition of IT project portfolio management among its member enterprises and advancing those practises. The most common, popular, and dominant approach to managing transition toward future enhancements of services, goods, investments, additions, strategies, and capabilities is through the use of projects, programmes, and portfolios [12]. The likelihood of financial loss and project failure is lower in companies that have developed and implemented effective project portfolio management practises. Business literature consistently emphasises the importance of efficient and successful project portfolio management procedures as a means to reaping significant rewards. The purpose of this thesis is to provide a set of guidelines for IT project portfolio management by examining the existing state of such practises in the case firms and making recommendations based on those analyses. Another artificial network that processes natural languages in order to identify spoken words is the recurrent neural network. On the basis of feature engineering and model training, this recurrent neural network is enforced.

1.1.Background of Scope

The introduction of AI and other forms of technology into the financial sector, however, does not come without a price. It is anticipated that considerable employment losses would occur in the medium term as a result of aggressive technological implementations in portfolio management [13]. The purpose of this study is to provide an overview of the various applications of machine learning in portfolio management, including challenges and opportunities, so that a more accurate assessment of the impact of ML (a subset of artificial intelligence) on the

investment management industry and the economy as a whole can be made. While the trending topic of machine learning and portfolio management can cover a variety of important topics, including client profiling, security analysis, portfolio rebalancing and monitoring of portfolio performance, asset allocation and portfolio construction, and monitoring of portfolio performance, our attention is solely directed toward the former two. The ability to combine various financial assets, each of which has its own individual risk and return characteristics, in order to meet the desired level of risk and expected return for the portfolio is the core objective of portfolio construction. Traditionally, the decisions that are made to make such allocations efficient and finally optimal are based on security analysis, the majority of which is driven by fundamental analysis and technical analysis. An strategy based on deep reinforcement learning is anticipated to be effective in this scenario for the management of the portfolio task, specifically for the purpose of optimising returns as much as possible. Along with a new target policy in regard to the trading agent, a new target policy has been suggested in this regard with regard to the preference of the low-risk activities. The adoption of this new aim strategy has been reflected in terms of the modification of greediness through the use of optimal action. In addition to this, the hyper parameters of the ideal activities have to be chosen. It serves as a tool for verifying the management of the performance level in relation to the bitcoin market. According to the findings of a study that was conducted by [14], the market for cryptocurrencies has been discovered to have served as one of the trading agents due to the excessive amount of data that has been generated and gathered in a minute manner. In addition, another factor that may have contributed to the occurrence of this event is the extremely volatile character of the bitcoin market. It has been regarded as an experimental analysis, and the findings have shown that, in conjunction with the market strategy, it is one of the investment points with the lowest risk associated with them under the current portfolio management approach. In this situation, where the market volatility becomes extreme or the training periods become shorter, it may be simple to adjust a generalised performance along with the provision of a less risky investment policy. This may be the case when the market volatility becomes extreme or when the training periods become shorter.

1.2. Research Objective

A profitable outcome has been linked with the implication of the future behaviour of the stock and the accurate implication of the selection of algorithmic methods for the development of the financial stock analysis as a result of the accurate prediction of the development of the stock market, which has led to an accurate prediction of the development of the stock market. However, the correct method of market analysis can help in the development of an efficient market through the use of artificial neural networking (ANN). This can help in the development of the ability to accurately assess the risk that is associated with the financial assets of the stock market, which is something that can help in the development of an efficient market. Aside from that, the correct method of portfolio management has limited the high level of volatility in the stock association with the low level of volatility of developed stock, which can recommend a development action in the stock association for the correct method of actual portfolio analysis. Lastly, the correct method of actual portfolio analysis has shown that the

correct method of actual portfolio analysis is the correct method. In addition, the process of machine learning has been helpful in the construction of portfolios that are able to make suggestions for alpha components in the design process. This can help to maintain the management of a financial portfolio that has a strong asset allocation strategy. The utilisation of artificial intelligence, on the other hand, has resulted in an increase in the trading implications of the management, which can develop the investment platform for the true discussion of investment in the market's financial side.

Historical stock return estimates have been made using linear models like AR, ARMA, and ARIMA [15]. These models have a drawback in that they are limited to a specific type of time series data (i.e., the established model may work well for one type of equity or index fund but not for another [16]. To address this issue, deep learning models were used in financial evaluation. Due to the possibility of extremely complex and interconnected inputs, deep neural networks (sometimes referred to as artificial neural networks) are good estimators. However, they may also predict the relationship between input data and anticipated outputs. Over the past few decades, many academics have developed numerous algorithms for intensive learning. Convolutional neural networks (CNNs), repeated neural networks (RNNs) [17], and long-term memory (LSTMs) are a few instances of estimations. However, none of these rules specifically account for macroeconomic factors.

learning stock return estimation techniques. To the best of our knowledge, not a lot of study has specifically looked into the connection between hidden information, macroeconomic conditions, and financial asset returns. In this study, we intend to build a feed-forward neural network model with a hidden layer and an output layer using a set of well-known and significant economic variables as the input layer (see Appendix A for details). Proposal. Our goal is to create a predictive model that illustrates the relationship between the return on financial investment instruments and the monthly log-percentage change of macroeconomic data. We divide the historical monthly log-percentage change for each macroeconomic measure and return on investment assets into a training set and a test set in order to look into these correlations. A three-layer neural network's training model is coupled to the training set and tested on the test set. The process involves finding the precise number of hidden neurons that minimises the mean squared error between the estimated and actual values as well as changing the hyperparameters. This artificial neural network mimics economic parameters as inputs and generates outputs for investment asset returns.

2.Literature Review

Prioritization and selection in project portfolio management often involve management methodologies or optimization algorithms that employ specific project selection criteria [18] and . The portfolio of initiatives must aid the firm or organisation in achieving its strategic objectives. It demonstrates that the design of project portfolios involves several factors. There are essentially two approaches to choose a portfolio of projects: one is based on a tried-and-true method, and the other is based on quantitative analysis employing cutting-edge methods discovered through searching [19]. This study focuses on the multi-criteria decision-making approach used to

objectively analyse the project portfolio selection problem. Based on the literature, numerous studies on project selection utilising a multi-criteria decision-making methodology have been published. For instance, Meade and Presley [20] employed the analytic network process (ANP) to select the best project based on a variety of parameters. However, [21]developed an integrated framework that uses a decision support system (DSS) and fuzzy theory to select projects for portfolio management. [21] utilised multi-objective swarm optimization to tackle the project selection problem. It also utilised data envelopment analysis (DEA) to select the most profitable portfolio of projects. The analytical hierarchy process (AHP) was also employed by [22] as a decision-making technique for project portfolios. Data Envelopment Analysis (DEA) and De Novo Optimization to identify the lowest budget required to create the best project portfolio.

It is widely used to evaluate numerous factors, compare and rate multiple alternatives based on those criteria, and ultimately rank solutions in order of importance. This method is referred to as multi-criteria decision making (MCDM [23] Society on MCDM defines it as the study of techniques and procedures for integrating numerous incompatible factors in order to obtain a conclusion [24] The variety of decision makers (DMs) in an MCDM problem aids in providing quantitative and qualitative measurements for evaluating the performance of each alternative in relation to the criteria, as well as determining the relative weight of the evaluation criteria in relation to the overall judgments [25]; Tzeng and Huang 2011). MCDM strategies focus on enhancing decision quality by making the process more clear, logical, and reliable [26]. There are various publications, applications, software packages, specialised scientific journals, and university courses on MCDM approaches [27]. MCDM techniques can be used to manage and resolve project portfolio issues including a variety of criteria or alternatives (projects). Consequently, several methods have been published in the literature to address these problems. AHP and TOPSIS are the two most prevalent MCDM methodologies [28]. The AHP method, one of the most popular approaches, is regularly utilised in candidate evaluation and selection [29] developed one of the most effective MCDM strategies, AHP, which has been extensively studied in the literature [30]. AHP decision makers base their conclusions or judgments on pairwise comparisons between each set of criteria or alternatives using Saaty's scale (Kumar, Rahman, and Chan 2017; Mangla, Kumar, and Barua 2015). The AHP is a technique widely employed to handle decision-making issues. The AHP has been utilised in numerous fields, including industry, management, agriculture, governance, resource allocation, and distribution, to make vital strategic decisions [31]. According to [32] the AHP method is also versatile, as it may be used with other techniques such as linear programming, quality function deployment, fuzzy logic, etc. However, the AHP technique does not account for the ambiguity of linguistic judgement linked with the related operation [33]. To overcome these issues, researchers have merged AHP with fuzzy theory to reduce uncertainty [34]. To illustrate the ambiguity of the DMs, integrated fuzzy number preference scales with language considerations. Fuzzy set theory effectively turns ill-defined or ambiguous information into meaningful data using fuzzy numbers and membership functions [35]. In order to address ambiguous judgments, the AHP is merged with the fundamental concepts of fuzzy sets theory to create fuzzy AHP. Using fuzzy aggregation operators and fuzzy arithmetic, the Fuzzy AHP addresses the

problem of hierarchical organisation [36]. Several authors have used fuzzy AHP successfully in a variety of applications to evaluate the fuzziness of decision-making problems, according to the literature [37].

In addition, the TOPSIS strategy is one of the most well-known methods for addressing MCDM challenges. Hwang and Yoon [38] proposed using the concept that the closest and farthest points in distance from the positive and negative ideal solution should be utilised to determine the ranking from a confined set of alternatives [39] The selected alternative should be the closest to the ideal positive solution and the farthest from the ideal superb solution. The TOPSIS technique has the advantage of requiring less subjective input from DMs and quickly identifying the best alternative [40]. In addition, the TOPSIS technique may be an effective way for DMs to incorporate missing data [41].Numerous TOPSIS-approach studies with diverse applications may be found in the academic literature [42]

Furthermore, Chen's developed Fuzzy TOPSIS is particularly well-suited to challenges in the decision-making application in a fuzzy environment, and it allows for effective handling of ambiguity in evaluations and verdicts [43]. To guarantee that projects' contributions are maximised and matched to corporate success, Levine [44] defines project portfolio management (PPM) as a set of methods linking traditional operations management and project management disciplines. Business strategies [45] can be put into action with the help of PPM by vetting potential projects, picking the ones that are the best fit, giving them the highest priority, and assigning the necessary resources to them. Portfolio management's objectives have been established through extensive research on portfolio practises in a variety of organisations [46], and they include optimising portfolio value, achieving a balanced mix of projects, and guaranteeing alignment to company strategies. Top-down PPM activities are divided into three categories by Müller et al. [47]: (1) aligning projects with business strategy and prioritising them (portfolio selection); (2) continuously monitoring and communicating project priorities and progress at the portfolio-level (portfolio reporting); and (3) making rational and jective choices to accelerate, kill, or reprioritize projects within the portfolio (portfolio decision-making).

As with applying modern portfolio theory to an investment portfolio of varied financial instruments, McFarlan proposed utilising the portfolio model to manage overall risk exposures for IT projects. By establishing and using a set of selection criteria across benefits, costs, and risks associated with the competing project investment options, the US General Accounting Office suggests a portfolio investment method to select, control, and assess IT projects. Information technology portfolio management (IT PPM) is defined by Maizlish and Handler as "a combination of people, processes, and corresponding information and technology that sensed and responded to change by reprioritizing/rebalancing investments and assets, value-based risk assessment of existing assets, eliminating redundancies while maximising reuse, optimal resource allocation, and continuous monitoring & measuring."

Aligning business objectives and IT strategy, as achieved through project portfolio management, is crucial [48] for realising the projected business value from IT-enabled investments. An analysis of the literature on project management identifies two distinct approaches to project management: an external system of governance that

employs centralised monitoring and controls to keep project outcomes in line with strategic objectives, and an internal system that strengthens the organization's capacity to work together toward common goals. In order to facilitate project management in all six dimensions, [49] suggest a framework (goal setting, incentives, monitoring, coordination, decision-making, and capability building). The process of selecting projects involves a number of difficult decision variables. Given that no one method is sufficient to resolve selection difficulties, an integrated strategy is required [50]. The literature study shows that some researchers have adopted fuzzy AHP and fuzzy TOPSIS and put them to use in a wide range of settings. But research shows that fuzzy AHP-TOPSIS portfolio management of projects is underutilised. Consequently, this study's objective is to use AHP and TOPSIS techniques inside a fuzzy setting to select the best possible course of action from among several potential projects. To determine the relative importance of each criterion, the Fuzzy AHP method was utilised. The fuzzy TOPSIS technique uses the criteria weights to increase the separation between potential solutions (projects) and zero in on the best one for getting the job done.

3. Deep learning and Methodology

For the purposes of financial portfolio management, "state space" can be thought of as the sum of various market conditions and investor attitudes for the future of the financial markets, as well as the ratio of a portfolio's assets at the present time. Since it is impossible to derive a full state space from unknown high-dimensional properties of the financial world, approximations are typically obtained by employing observable samples to simplify its state-space representation [51]. A trader's "action space" is the range of options available to them while deciding whether to buy, sell, or hold a certain asset. Separate areas for buying and selling assets are defined as continuous and discrete. What we mean by "transition probability" is the likelihood of changing the current state into the next one immediately following the action. A predetermined, risk-adjusted profit is included in the reward function. The most recent past n-period time of states and actions sequence can be used to learn one or multiple future m-period time of expected rewards, allowing the policy function to have a non-Markovian quality. The accompanying deep learning process has been expanding our understanding of how well humans can judge the actual formulation of mathematical and machine learning algorithms used in risk assessment. We create a reliable method of comparing DL technologies for cost estimation. Additionally, the application of DL technology with appropriate algorithms can be created as a subfield for the contribution of machine learning, which has integrated the process of learning in the commutation problem with the development of many trials and errors. In recent years, deep learning (DL) technology has emerged as a subfield of machine learning that shares certain similarities with AI. In addition, deep learning's incorporation can create a training set that can be used in a fresh data set for the association of the dynamic way of learning to change an action based on a process of continuous feedback to ensure the correct way of maximisation as a reward. In addition, the research lends credence to the use of model-free DL techniques in the implementation of financial investment policy for enhanced market association. The traditional method of optimising the allocation of diverse capital can raise the performance of the stock of capital in the stock market, and this approach can be linked to the process of decision making based on optimization of the allocation in capital requirements.

3.1. Deep Learning

To anticipate an output Y from an input X, Deep Learning is a machine learning technique. To fully grasp deep learning, it is necessary to first grasp its foundational components.

Perceptron

Given a finite number of m inputs, we multiply each input by a weight, add a bias, and then run the result through a non-linear activation function to generate the output y.

By using the bias 0, it is possible to expand the input space by one more level. Thus, even if the input vector is all zeros, the activation function will still return a value. It's the part of the result that can't be explained by the parameters of the experiment.

Neural Network

Algorithms called neural networks are trained to spot patterns in data in a manner similar to the human brain. By using a form of machine perception, they classify and organise raw sensory data. All real-world data, including as images, sounds, texts, and time series, must be converted into its numerical, vector-based pattern recognition language.

Neural networks can be conceptualised as a series of perceptrons interconnected at their inputs and their outputs. Multilayer Perceptrons are a common name for neural networks that contain a hidden layer.

Non-linearities are what the activation function is supposed to bring to the network. When the input distribution is fixed, the output decision made by the activation function is also fixed. The variety of activation functions in Figure 2 demonstrates how non-linearities can be used to improve approximations of arbitrarily complex functions. Here are a few examples of activation functions:

A measure of an algorithm's modelling skill is the Loss function (P.S, also known as the Objective/Error/Cost function). The loss function will produce a large value if the forecasts are drastically off. The loss function's output will be smaller if the forecasts are accurate. As the algorithm makes adjustments to the model, the loss function is the best measure of whether or not the adjustments are on the correct track.

It's possible to use the cross-entropy measure, the mean squared error, or even the absolute difference between the forecast and the actual value as the loss function.

For each data point in a dataset, Absolute Error determines the largest positive or negative discrepancy between the anticipated value and the target value.

$\mathscr{L}=1n\Sigma|y(i)-y\hat{}(i)|ni=1$ [1]

Error Quantity Squared

A common simple loss function is the Mean Squared Error (MSE). The mean squared error (MSE) is determined by squaring the difference between the forecasts and the ground truth and then averaging that number over all of the data in the collection.

$$\mathcal{L} = 1n\Sigma\, ni = 1 \quad (2)$$

$$(y(i) - y\hat{}(i)) \quad (3)$$

Cross Entropy Loss / Log Loss

In the context of classification problems, log loss is a popular loss function. It's a logarithmic transformation of the likelihood function.

By using backpropagation [52], we can re-route the neural network's internal error signal that was generated when making a prediction of the result.When the parameters of the neural network are adjusted, the error that is generated by the network also shifts. We use an optimization method to alter the settings and optimization Algorithms. By using the model's internal learnable parameters to calculate the target values from the set of predictors, optimization methods can help minimise or maximise an Objective/Loss/Error/Cost function. Weights and bias are examples of learnable parameters in neural network models.

The First Order Optimization Algorithms

To do so, they calculate the gradients of the Loss function as a function of the model parameters and then either maximise it or reduce it. If the function is decreasing or rising at a certain location, we can find out from the First Order Derivative. In essence, the first-order derivative of the Loss function provides us with a line that is tangent to a given point on the function's underlying surface.

Second-Order Optimisation Algorithms,

In order to either reduce or maximise the Loss function, these techniques make use of the second order derivative, often known as the Hessian. As a matrix of partial derivatives of the second order, the Hessian is a useful tool for analysing and understanding complex systems. The quadratic surface we obtain from the second derivative is convex to the curved surface of the loss function.

Ascending Gradient Method

To enhance the model's learning as a whole, the Gradient Descent [8] algorithm calculates the gradient of an Error function with respect to the learnable parameters, then feeds that information back into the network to adjust the parameters in the direction of the gradient.

Gradient Descent Variants

This kind of gradient descent uses each gradient descent iteration to analyse every training example. However, batch gradient descent is computationally expensive if the number of training examples is huge. Therefore, batch gradient descent is not recommended when the number of training examples is huge. Mini-batch gradient descent or stochastic gradient descent is what we employ instead.

Stochastic Gradient Descent

One training example is used in each iteration of this gradient descent method. This means that even after only one cycle, where a single example has been analysed, the parameters are being modified. In comparison to batch gradient descent, this is far quicker. However, even when there are many training examples, the system only processes one at a time, which can be inefficient because of the high number of iterations.

Mini Batch Gradient Descent

This gradient descent method outperforms the more common batch gradient descent and stochastic gradient descent in terms of speed. In this case, for each iteration, we'll process b instances in which bm. For this reason,

it doesn't matter whether there are a lot of training examples; they will be handled in batches of b training examples at a time. This means that the training examples can be larger while yet requiring fewer iterations.

Acceleration

Gaining momentum can assist speed up SGD in the right direction and reduce fluctuations. A portion of the update vector from the previous time step is added to the current update vector to achieve this. Simply put, we use momentum like we're pushing a ball down a slope. As it travels downhill, the ball gains speed as its momentum increases (up until the point where it encounters its terminal velocity due to air resistance, if 1). This also occurs with any changes we make to our parameters: For dimensions with parallel gradients, the momentum term increases, while for dimensions with perpendicular gradients, it decreases the magnitude of the updates. The outcome is a decrease in oscillation and a speeding up of the convergence process.

Adam

Another technique that estimates adaptive learning rates for each parameter is Adaptive Moment Estimation (Adam). Like momentum, Adam preserves an exponentially decaying average of past gradients mt as well as a squared average of past gradients vt. Adam likes flat minima in the error surface, analogous to a heavy ball with friction rather than the rolling motion of momentum. Deteriorating gradient averages mt and vt are calculated as follows:

$$mt = \beta 1 mt - 1 + (1 - \beta 1) gtvt = \beta 2 vt - 1 + (1 - \beta 2) gt \quad (3)$$

As the name suggests, this technique is named after the moments estimated: the first moment (the mean) and the second moment (the uncentered variance) of the gradients. The Adam authors note that when the decay rates are minimal (i.e. 1 and 2 are near to 1), the bias towards zero in mt and vt becomes more pronounced, as they are initialised as vectors of 0.

Bias-corrected first and second moment estimates are then computed to account for these deviations.

They then apply these to the parameters in order to arrive at the Adam update rule. They provide actual evidence that Adam is effective in practise and shows that it performs well in comparison to other adaptive learning-method algorithms.

The neural network has not yet learned the features that would allow it to make accurate predictions. As the neural network is trained with more and more examples of data, it will become more accurate in its predictions. We will feed the neural network data during training and see what it predicts for an outcome. If the neural network fails to produce the desired outcome, we will feed it the proper target value and determine how much of a difference there is. Error is the name given to the deviation.

Constructing an Objective Function or Loss Function allows one to determine the amount of divergence. In order to improve its accuracy, the neural network's goal is to minimise the margin of error associated with its forecasts. Locates the local minimum of the Loss Function.

During training, neural networks' weights change such that the resulting mapping more closely matches the outputs of the training data. Neural network learning can be supervised, where the proper output option is provided in training data, or unsupervised, where no output is provided.

The purpose of using training data to teach a neural network is to ensure that the network is able to take what it has learned about a particular set of features and apply it to new data. Good results indicate that the neural network has learned as efficiently as possible. However, there are cases where the network learns exceptionally well on the training dataset by achieving minimum prediction error, but struggles to generalise to the testing dataset. The phrase "Overfitting" describes this type of circumstance where a model is overly tailored to its initial training data.

Many strategies exist to prevent overfitting. You can find a few examples of these below:

3.1.2.Regularization

Regularization [53] involves adding an extra element to the loss function, which punishes our model for being too complex or, in simple words, for using too high values in the weight matrix. This way we try to limit its flexibility, but also encourage it to build solutions based on multiple features.

Cost Function = Loss Function + Regularization Term (4)

Two popular versions of this method are:

L1 - Least Absolute Deviations (LAD)

L2 - Least Square Errors (LS).

$$CostFunction = Loss + \lambda 2m * \Sigma \|w\|2 \quad (5)$$

Regularization Rate

Regularization Rate λ, is the regularization parameter. It is the hyper-parameter whose value is optimized for better results.

3.3. Dropout

Dropout [54] is another widely used regularization technique used in Deep Learning to avoid overfitting. At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connection.

Dropout Regularization

As we can see in the figure, dropout randomly drops nodes and removes their incoming and outgoing connections. This way, the neural network doesn't over-fit to a certain set of features and weights. The dropout rate is a hyper-parameter that needs to be passed to the neural

3.4. Proposed Methodology

Our proposed objective function for optimizing neural networks via backpropagation is derived from a generalized Kelly criterion [55]. This criterion maximizes the average geometric growth rate of a bettor's profit

during betting on multiple horses with known win probabilities. In our approach, each training sample is treated as a bettor whose optimal bets (allocation fractions) for betting on multiple classes (horses) are determined by the generalized Kelly criterion [56]. In gambling theory, every horse has a nonzero win probability

In Kelly gambling, we place a fixed fraction of our total profit (assumed positive) on n bets. We denote the fractions as b ∈ R n , so b ≥ 0 and 1 T b = 1, where 1 is the vector with all components 1. The n bets have a random nonnegative payoff or return, denoted r ∈ R n +, so the profit after the bet changes by the (random) factor r T b. We will assume that all bets do not have infinite return in expectation, i.e., that E ri < ∞ for i = 1, . . . , n. We will also assume that the bet n has a certain return of one, i.e., rn = 1 almost surely. This means that bn represents the fraction of our profit that we do not wager, or hold as cash. The bet vector b = en corresponds to not betting at all. We refer to the bets 1, . . . , n − 1 as the risky bets. We mention some special cases of this general Kelly gambling setup.

Two outcomes. We have n = 2, and r takes on only two values: (P, 1), with probability π, and (0, 1), with probability 1 − π. The first outcome corresponds to winning the bet (π is the probability of winning) and P > 1 is the payoff. • Mutually exclusive outcomes. There are n − 1 mutually exclusive outcomes, with return vectors: r = (Pkek, 1) with probability πk > 0, for k = 1, . . . , n − 1, where Pk > 1 is the payoff for outcome k, and ek is the unit vector with kth entry one and all others 0. Here we bet on which of outcomes 1, . . . , n − 1 will be the winner (e.g., the winner of a horse race). • General finite outcomes. The return vector takes on K values r1, . . . , rK, with probabilities π1, . . . , πK. This case allows for more complex bets, for example in horse racing show, place, exacta, perfecta, and so on. • General returns. The return r comes from an arbitrary infinite distribution (with rn = 1 almost surely). If the returns are log-normal, the gamble is a simple model of investing (long only) in n − 1 assets with log-normal returns;

RISK-CONSTRAINED KELLY GAMBLING

the nth asset is risk free (cash). More generally, we can have n − 1 arbitrary derivatives (e.g., options) with payoffs that depend on an underlying random variable. 3.2.1 Profit growth The gamble is repeated at times (epochs) t = 1, 2, . . ., with IID (independent and identically distributed) returns. Starting with initial profit w1 = 1, the profit at time t is given by wt = (r T 1 b)· · ·(r T t−1 b), where rt denotes the realized return at time t (and not the tth entry of the vector). The profit sequence {wt} is a stochastic process that depends on the choice of the bet vector b, as well as the distribution of return vector r. Our goal is to choose b so that, roughly speaking, the profit becomes large. Note that wt ≥ 0, since r ≥ 0 and b ≥ 0. The event wt = 0 is called ruin, and can only happen if r T b = 0 has positive probability. The methods for choosing b that we discuss below all preclude ruin, so we will assume it does not occur, i.e., wt > 0 for all t, almost surely. Note that if bn > 0, ruin cannot occur since r T b ≥ bn almost surely. With vt = log wt denoting the logarithm of the profit, we have vt = log(r T 1 b) + · · · + log(r T t−1 b). Thus {vt} is a random walk, with increment distribution given by the distribution of log(r T b). The drift of the random walk is E log(r T b); we have E vt = (t−1) E log(r T b), and var vt = (t − 1) var log(r T

b). The quantity E log(r T b) can be interpreted as the average growth rate of the profit; it is the drift in the random walk {vt}. The (expected) growth rate E log(r T b) is a function of the bet vector b.

In neural network optimization, the other bettors are virtual and b c vi is simply the class posterior probability estimated by the softmax layer. This way, for each training sample vi , the belief probabilities form a vector bvi = (b 1 vi , b2 vi , ..., b|C| vi). Similar to the win probabilities, the belief probabilities should adhere to the real life situations. That is, P c∈C b c vi = 1, 0 < b c vi < 1, ∀c ∈ C. To guarantee these, if b c vi = 1 then it gets modified to b c vi = 1 − and if b c vi = 0 then it gets modified to b c vi = /(|C| − 1). The win and the belief probabilities define the revenue rate r c vi and the expected revenue erc vi of vi in betting on the c th class. These parameters are given by r c vi = 1/bc vi and erc vi = p c vi /bc vi , (2) and are prevented from getting infinity by 0 < bc vi < 1.

An input layer, a hidden layer made up of memory block cell assemblies, and an output layer make up the original LSTM design. Three different types of specialized gate units that can learn to preserve, use, or destroy this state as necessary make up each memory block, which is made up of memory cell units that maintain state across time-steps. The LSTM training algorithm adjusts all units' incoming connections as it back propagates errors from the output units across the memory blocks, but it then truncates the back-propagated errors. Therefore, second-order networks with units positioned between memory blocks and the input layer cannot be successfully trained using the LSTM training technique. Recurrent Neural network is shown in Fig.1.

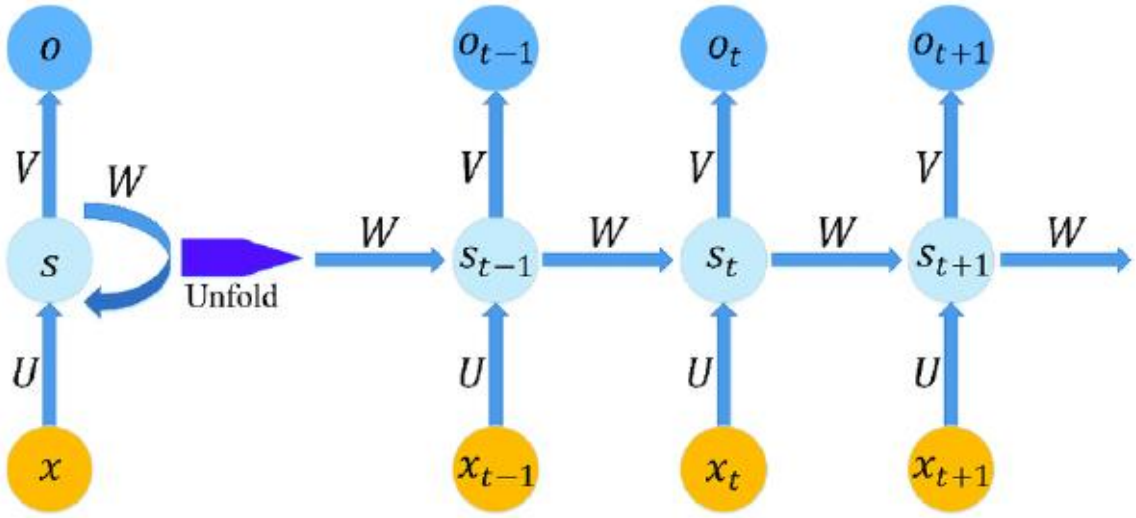

Fig.1 Recurrent Neural Network

1. Activation dynamics

When an input is given, we go through each layer's μ activation in turn, starting with the input layer $\varphi, \rho, \beta, \tau, \sigma$ and ending with the output layerσ. In order to activate a memory cell, its state must pass via its squashing function and be altered by the activation of its output gate in eq.6.

$$A_{\mu y} = \sum W_{\mu y x} B_x \qquad for\ \{\varphi, \rho, \beta, \tau, \sigma\} \qquad (6)$$

$$B_{\mu y} = k_{\mu y}(a_{\mu y}) \qquad for\ \{\varphi, \rho, \tau, \sigma\} \tag{7}$$

$$Q_{\beta y} = J_{\rho y}\, Q^{\sim}{}_{\beta y} + J_{\varphi y} A_{\beta y} \tag{8}$$

$$J_{\beta y} = J_{\tau y} K_{\beta y}\,(Q_{\beta y}) \tag{9}$$

Since peephole connections originate from ths e internal state of the memory cells rather than their activations, one should substitute the memory cell state sci for the transmitting unit activation yi in the equations in this section when taking into account peephole connections.

2. Learning rules

Each unit must keep track of the activity flow across each of its connections over time in order to learn successfully. To achieve this, each unit keeps track of each of its input connections' eligibility, updating this trace as soon as it determines its activation. The history of activations that have passed through a connection but may still be affecting the network's status are included in the validity trace for that connection. In this situation, qualifying traces do not deteriorate over time like those utilized in temporal-difference learning. When a target vector is provided, the validity traces are used to help assign error liabilities to particular connections.

The much more recent activation result to cross the link eq.10 makes up the instantaneous eligibility traces for output gates and output units. For the memory cells in equation (11), forget gates in equation (12), and input gates in equation (13), the suitability traces are simplified partial derivatives of the state of the memory cell with respect to the connection in question. Each prior eligibility trace is enhanced in proportion to how it affects the memory cell's state, and it is preserved in proportion to how much state the memory cell keeps.

$$E_{\mu yx} = J_x \qquad for\ \mu E\{\tau, \sigma\} \tag{10}$$

$$E_{\beta yx} = J_{\rho y}\, E^{\sim}{}_{\beta yx} + J_{xy} J_x \tag{11}$$

$$E_{\rho yx} = J_{\rho y} E^{\sim}{}_{\rho yx} + A_{\beta x} K_{\rho y}(A_{\rho y}) J_x \tag{12}$$

$$E_{\beta yx} = J_{\rho y} E^{\sim}{}_{\varphi yx} + A_{\beta x} K_{\varphi y}(A_{\varphi y}) J_x \tag{13}$$

To compare the network's performance between time-steps of the activation dynamics, a target vector P with all values in the range [0, 1] may be given to the network. The difference between the network output and the target is calculated using the cross-entropy function. Gradient ascent is used to train the network by moving this function in the direction of zero since $E_1 \leq 0$ when the P and B values are in the range [0, 1], as is necessary. The error responsibility $\partial_{\sigma y}$ for the output units is determined by deriving Eq. (32) in relation to the output unit activations. By propagating the mistake backwards through the network, the deltas for the output gates and the remaining components can be obtained. This process is provided hereunder as Mathematical Representation.

$$E_1 = \sum_{y=\theta} \left(P_y \, log\left(J_{\sigma y}\right) + (1 - \left(P_y\right) log\left(1 - J_{\sigma y}\right) \right) \quad (14)$$

$$\partial_{\sigma y} = P_y - J_{\sigma y} \quad (15)$$

$$\partial_{\tau y} = K_{\tau y}\left(A_{\tau y}\right) K_{\beta y}\left(Q_{\beta y}\right) \sum_{f=\theta} \partial_{\sigma_f} \tau_{\sigma_{f\beta y}} \quad (16)$$

$$\partial_{\mu y} = K_{\mu y}\left(Q_{xy}\right) J_{\sigma y} \sum_{f=\theta} \partial_{\sigma_f} \tau_{\sigma_{f\beta y}} \qquad for\ \mu E\ \{\varphi, \rho, \beta\} \quad (17)$$

The learning rate, the error for unit responsibility $\partial_{\mu x}$, and the trace$E_{\mu yx}$, the link weights at very layer in unit μ are updated in eq.18.

$$\Delta W_{\sigma xy} = \alpha \partial_{\mu y} E_{\mu yx} \qquad for\ \mu E\ \{\varphi, \rho, \beta, \tau, \sigma\} \quad (18)$$

Hyperparameter Optimization The feed-forward neural network described in subsection II-D1 is trained (optimized) by the RR separately. Each training trial, with an objective function, is done by a mini-batch-based stochastic gradient descent optimization. In each iteration, a mini-batch, consisting of B volumetric image data sets and their labels, is fed to the network. An epoch is a complete pass of the 30 training data sets through the network. Thus, the maximum number of iterations Nitr is determined by the maximum number of epochs Nep through Nitr = (30 B) × Nep. In the above training, the learning rate adaptations follow an exponential decay strategy, in which the actual learning rate r(t) is given by r(t) = r0 · e −d·nt , where nt is the current iteration number (or units of epochs); r0 ∈ {0.01, 0.02, ..., 0.1} is the initial learning rate and d ∈ {0.01, 0.011, ..., 0.025} is the decay factor. For each objective function, prior to the network training, the hyperparameters of momentum μ, (r0, d), weight decay λ, B, Nep, and number of convolutional layers Li in the i th network stage are optimized. This optimization aims at reducing the unnecessary complexity of the network and improving its generalization. λ is the parameter of a regularizer that encourages small values for the main network parameters to mitigate overfitting. It is set to zero for the learnable parameters of the PReLU activations and is tuned for the rest of the network parameters. The above hyperparameters, except for Nep, are tuned by the Hyperopt software that performs a Bayesian optimization on a tree of Parzen estimators. To mitigate overfitting, Nep is tuned according to the principle of early stopping. That is, the training stops once performance on a held-out validation data set stops improving. Table II shows the optimized hyperparameters for each objective function.

For each objective function, the network is trained by a mini-batch-based stochastic gradient descent optimization using the hyperparameters listed in Table II and multiple random seeds for initializing the convolutional weights. The network parameters are then averaged over multiple seeds trials to yield an average network model. In each training iteration, a mini-batch TB is passed through the network in the forward direction yielding the updated belief probabilities {bvi }vi∈TB . The {bvi }vi∈TB and the win probabilities {pvi }vi∈TB define the current value of the objective function. Then the network parameters are optimized by a backpropagation from the output of the softmax layer (in case of RR and WCE) or from the binary masks of the

segmentations (in case of DC) to the previous network layers. The trainings are done by using the open source library of TensorFlowTM and parallelization on 4 NVIDIA TITAN X R GPUs. Each GPU is occupied at most by 20% of its processing capacity to allow other usages.

In Kelly gambling, we choose b to maximize E log(r T b), the growth rate of profit. This leads to the optimization problem maximize E log(r T b) subject to 1 T b = 1, b ≥ 0, (3.1) with variable b. We call a solution b ? of this problem a set of Kelly optimal bets. The Kelly gambling problem is always feasible, since b = en (which corresponds to not placing any bets) is feasible. This choice achieves objective value zero, so the optimal value of the Kely gambling problem is always nonnegative. The Kelly gambling problem (3.1) is a convex optimization problem, since the objective is concave and the constraints are convex. Kelly optimal bets maximize growth rate of the profit. If ˜b is a bet vector that is not Kelly optimal, with associated profit sequence ˜wt , and b ? is Kelly optimal, with associated profit sequence w ? t , then w ? t /w˜t → ∞ with probability one as t → ∞. (This follows immediately since the random walk log w ? t − log ˜wt has positive drift [49, §XII.2]. Also see [36, §16] for a general discussion of Kelly gambling.) We note that the bet vector b = en is Kelly optimal if and only if E ri ≤ 1 for i = 1, . . . , n−1. Thus we should not bet at all if all the bets are losers in expectation; conversely, if just one bet is a winner in expectation, the optimal bet is not the trivial one en, and the optimal growth rate is positive. We show this in the appendix. This approximation can be useful when finding a solution to the RCK problem (3.7). We first estimate the first and second moments of ρ via Monte Carlo, and then solve the QRCK problem (3.14) (with the estimated moments) to get a solution b qp and a Langrange multipler κ qp. We take these as good approximations for the solution of (3.7), and use them as the starting points for the primal-dual stochastic gradient method, i.e., we set b (1) = b qp and κ (1) = κ qp. This gives no theoretical advantage, since the method converges no matter what the initial points are; but it can speed up the convergence in practice. We now connect the QRCK problem (3.14) to classical Markowitz portfolio selection. We start by defining μ = E ρ, the mean excess return, and S = E ρρT = Σ + (E ρ)(E ρ) T , the (raw) second moment of ρ (with Σ the covariance of the return). We say that an allocation vector b is a Markowitz portfolio if it solves maximize μ T b − γ 2 b TΣb subject to 1 T b = 1, b ≥ 0, (3.15) for some value of the (risk-aversion) parameter γ ≥ 0. A solution to problem (3.14) is a Markowitz portfolio, provided there are no arbitrages. By no arbitrage we mean that μ T b > 0, 1 T b = 1, and b ≥ 0, implies b TΣb > 0. Let us show this. Let b qp be the solution to the QRCK problem (3.14). By (strong) Lagrange duality [12], b qp is a solution of maximize μ T b − 1 2 b T Sb + ν(μ T b − λ+1 2 b T Sb) subject to 1 T b = 1, b ≥ 0 for some ν ≥ 0, which we get by dualizing only the constraint −λ E ρ T b+ λ(λ+1) 2 E(ρ T b) 2 ≤ 0. We divide the objective by 1 + ν and substitute S = μμT + Σ to get that b qp is a 90 ***RISK-CONSTRAINED KELLY GAMBLING***

It solution of maximize μ T b − η 2 (μ T b) 2 − η 2 b TΣb subject to 1 T b = 1, b ≥ 0, (3.16) for some η > 0. In turn, b qp is a solution of maximize (1 − ημT b qp)μ T b − η 2 b TΣb subject to 1 T b = 1, b ≥ 0, (3.17) since the objectives of problem (3.16) and (3.17) have the same gradient at b qp. If μ T b qp < 1/η then problem (3.17) is equivalent to problem (3.15) with γ = η/(1 − ημT b qp). Assume for contradiction that μ T b qp ≥ 1/η, which

implies (b qp) TΣb qp > 0 by the no artibtrage assumption. Then for problem (3.17) the bet en achieves objective value 0, which is better than that of b qp. As this contradicts the optimality of b qp, we have μ T b qp < 1/η. So we conclude a solution to problem (3.14) is a solution to problem (3.15), i.e., b qp is a Markowitz portfolio.

4. Methodology Analysis

Dataset Description

We have used American Stocks data from Yahoo Finance. We have the stocks data for the following 10 companies: Boeing, Coca Cola, Ford, IBM, GE, JP Morgan, Microsoft, Nike, Walmart, and Exxon Mobil. Out of these ten, we decide to work with M stocks which are chosen randomly.

We have downloaded the data from 2013 to 2022. This means we have 12054 days' worth of data for each stock. The downloaded data contains the Date, Open Price, High Price, Low Price, Close Price, Adjusted Close Price, Volume and Stock Ticker. All the prices are in Dollars, while the volume is defined in the number of shares traded for that ticker on that day. For days, where we don't have the stock data available, we maintain the time series by filling the empty price data with the close price of the previous day and we also set the volume to 0 to indicate that the market is closed on that day.

In order to come up with a general agent which is robust with different stocks, we will normalize the price data. We will divide the opening price, closing price, high price, and low price by the highest closing price of the total period.

The network works on an input tensor of shape [(M+1) x L x N], where M is the number of stocks we select, L is the length of the window and N is the number of features. We add a 1 to M to represent liquid cash that we initially start with.

The assets under consideration are liquid, hence they can be converted into cash quickly, with little or no loss in value. Moreover, the selected assets have available historical data in order to enable analysis.

	Sr. No.	Symbol	Company Name	Market Capitalisation \n(In lakhs)
0	1	RELIANCE	Reliance Industries Limited	7.060358e+07
1	2	TCS	Tata Consultancy Services Limited	6.852230e+07
2	3	HINDUNILVR	Hindustan Unilever Limited	4.975841e+07
3	4	HDFCBANK	HDFC Bank Limited	4.724823e+07
4	5	HDFC	Housing Development Finance Corporation Limited	2.823571e+07
...	...	...	...	...
995	996	MBAPL	Madhya Bharat Agro Products Limited	1.440368e+04
996	997	SANGAMIND	Sangam (India) Limited	1.438887e+04
997	998	CHEMFAB	Chemfab Alkalis Limited	1.405719e+04
998	999	GOKEX	Gokaldas Exports Limited	1.393813e+04
999	1000	ADVANIHOTR	Advani Hotels & Resorts (India) Limited	1.391199e+04

Fig.2. Input Data for Portfolio Management

Pre-processing

```
# Data Cleaning and Correction w.r.t dates
sp500.loc[sp500[sp500['Date first added']=='1983-11-30 (1957-03-04)'].index,'Date first added'] = '1983-11-30'

# Data filtering of the firm after December 2015
sp500['Date first added'] = pd.to_datetime(sp500['Date first added'],format='%Y-%m-%d')
sp500 = sp500[sp500['Date first added']<'2007-01-01']
print("The number of stocks in the universe is:", sp500.shape[0])
```

Fig.3. Pre-processing Analysis

Since RL is the type of ML that offers the ML agent an increasing reward based on his actions, it benefits portfolio managers. RL models work in a predetermined manner and must be replaced by anything. In fact, according to Snow, RL is almost entirely designed for stock trading. "Will commodity prices rise tomorrow?" Compared to the supervised reading that answers the question; Reinforcement reading, 'Should I buy things today?' "Reinforcement learning algorithms are already integrated into the trading strategy," Snow said. However, RL is still not very popular with portfolio managers because the process requires large amounts of data (including SL) and is time consuming to train, costly to test and has great computing power.

Instead of looking at and learning locally, as in RL, SL uses labelled or quoted data sets to train algorithms to evaluate results - usually in the case of portfolio management. To break data based on future value values - or Fed inputs. ML model. , Types of monitored reads include queues and subdivisions.

Although SL has evolved and is more widely used than RL for the reasons mentioned above, some experts predict a gradual integration of the two portfolio management practices. "At the same time, I expect to see significant improvements in the RL trading space so that the RL can use the SL trading methodology without sacrificing their strength," Snow said. "Mentally RL offers a kind of model flexibility where we cannot focus too much on predictive power, which is a practical task, but always goal-oriented improvement."“We defined a model training period of 60 months with an interval of 60 months to determine the optimal weights of the six index funds and to test the data outside the sample. A strategy for capturing short-term momentum and relatively long-term average reversal effects. Therefore, data points are selected in training and sample external testing to balance meaningful data volume, capture such information and gather appropriate horizons to capture useful market signals. We tested different training periods from 36 months to 180 mon and found that the 60-month horizon was a reasonable balance in capturing recent signals while mining significant historical data. Although there are only 60 data points for setting up the neural network and learning the model during the training period, data samples are displayed on a monthly cycle rather than on a daily basis.

This process of 60 months of training and one month of sample testing is repeated daily and continues until the final data point in the sample. For each training period, we begin by generalizing our training data set so that all data is within the normal range. For this purpose, we use the minimum-maximum normalization method for both the monthly log returns of six index funds and the monthly log-percentage change of 11 macroeconomic variables. It accelerates the gradient decent algorithm in setting up neural networks based on generalizations.

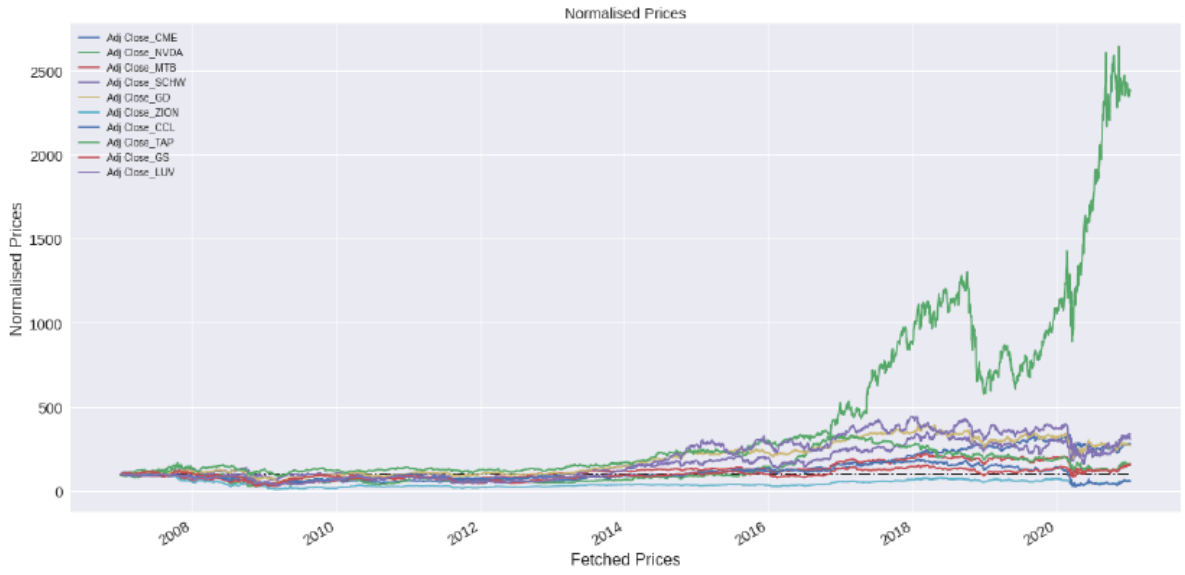

Fig.4. Normalization data Analysis

We define the minimum profit as the infimum of the profit trajectory over time, Wmin = inf t=1,2,... wt . This is a random variable, with distribution that depends on b. With b = en, we have wt = 1 for all t, so Wmin = 1. With b for which E log(r T b) > 0 (which we assume), Wmin takes values in (0, 1]. Small Wmin corresponds to a case where the initial profit drops to a small value before eventually increasing. The drawdown is defined as 1 − Wmin. A drawdown of 0.3 means that our profit dropped 30% from its initial value (one), before increasing (which it eventually must do, since vt → ∞ with probability one). Several other definitions of drawdown are used in the literature. 3.3 Drawdown We define the minimum wealth as the infimum of the wealth trajectory over time, Wmin = inf t=1,2,... wt . This is a random variable, with distribution that depends on b. With b = en, we have wt = 1 for all t, so Wmin = 1. With b for which E log(r T b) > 0 (which we assume), Wmin takes values in (0, 1]. Small Wmin corresponds to a case where the initial wealth drops to a small value before eventually increasing. The drawdown is defined as 1 − Wmin. A drawdown of 0.3 means that our wealth dropped 30% from its initial value (one), before increasing (which it eventually must do, since vt → ∞ with probability one). Several other definitions of drawdown are used in the literature. A large drawdown means that Wmin is small, i.e., our wealth drops to a small value before growing. The drawdown risk is defined as Prob(Wmin < α), where α ∈ (0, 1) is a given target (undesired) minimum wealth. This risk depends on the bet vector b in a very complicated way. There is in general no formula for the risk in terms of b, but we can always (approximately) compute the drawdown risk for a given b using Monte Carlo 1simulation. As an example, a drawdown risk of 0.1 for α = 0.7 means the probability of experiencing more than 30% drawdown is only 10%. The smaller the drawdown risk (with any target), the better. Fractional Kelly gambling It is well known that Kelly optimal bets can lead to substantial drawdown risk. One ad hoc method for handling this is to compute a Kelly optimal bet b ? , and then use the so-called fractional Kelly bet given by b = f b? + (1 − f)en,

DRAWDOWN RISK BOUND 81 where f ∈ (0, 1) is the fraction. The fractional Kelly bet scales down the (risky) bets by f. Fractional Kelly bets have smaller drawdowns than Kelly bets, at the cost of reduced growth rate. We will see that trading off growth rate and drawdown risk can be more directly (and better) handled. 3.3.2 Kelly gambling with drawdown risk We can add a drawdown risk constraint to the Kelly gambling problem (3.1), to obtain the problem maximize E log(r T b) subject to 1 T b = 1, b ≥ 0, Prob(Wmin < α) < β, (3.5) with variable b, where α, β ∈ (0, 1) are given parameters. The last constraint limits the probability of a drop in wealth to value α to be no more than β. For example, we might take α = 0.7 and β = 0.1, meaning that we require the probability of a drawdown of more than 30% to be less than 10%. (This imposes a constraint on the bet vector b.) Unfortunately the problem is, as far as we know, a difficult optimization problem in general. In the next section we will develop a bound on the drawdown risk that results in a tractable convex constraint on b. We will see in numerical simulations that the bound is generally quite good.

Max Drawdown is another important metric in finance, which describes the difference between the peak value and the lowest low of the portfolio. Mathematically, expressed as

$$\text{Maximum Drawdown} = \frac{\text{peak equity value} - \text{owest equity value}}{\text{peak equity value}.} \quad (19)$$

Finally, the last two metrics will be quite important to investors, namely the annualized returns, and the annualized standard deviations. For annualized returns, we can relate it to the total return, xf (T), over a holding period denoted by T, mathematically given as

$$Annualized\ Returns = [1 + xf(T)]^{T-1} - 1 \quad ()$$

To get the annualized standard deviation, we multiply the standard deviation by the square root of the number of periods per year.

$$\sigma \cdot \sqrt{\text{periods}.}$$

5. Result and Evaluation Analysis

In this section, five company's profit, return, and drawdown are analyzed. Following that, Reliance, TCS, Hindunilvr, HDFC bank, HDFC are considered in this result analysis within 2013 to 2022 years. Initially, the Annual Return output is provided hereunder

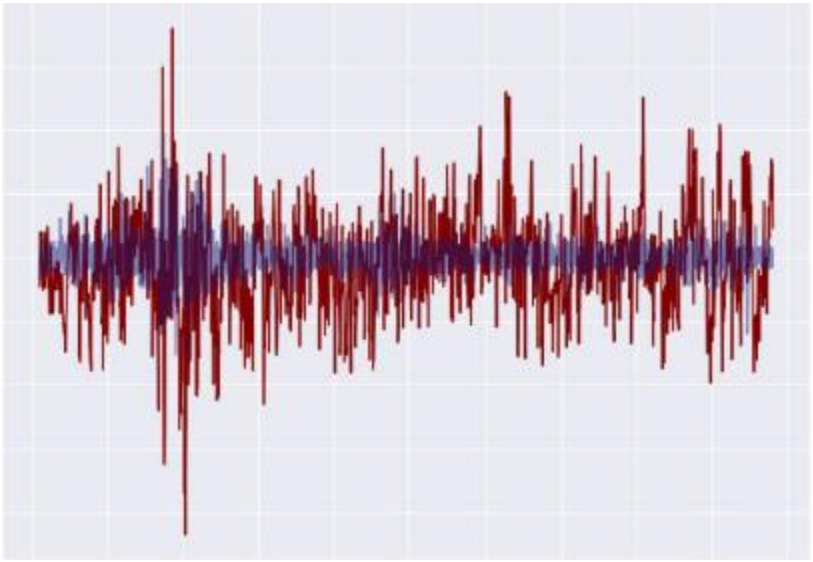

Fig.5. Reliance's Annual Return according to 2013- 2022 years

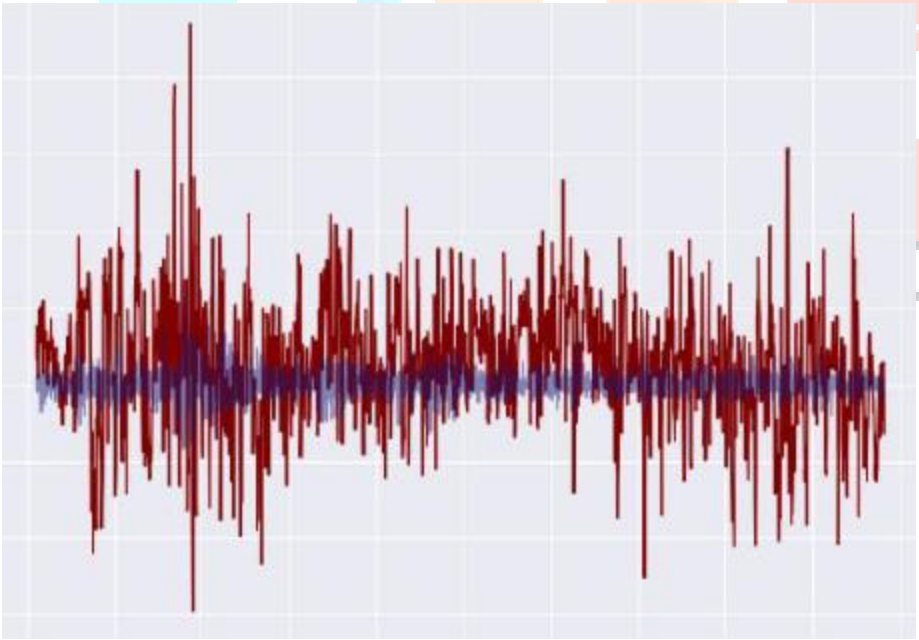

Fig.6. TCS's Annual Return according to 2013- 2022 years

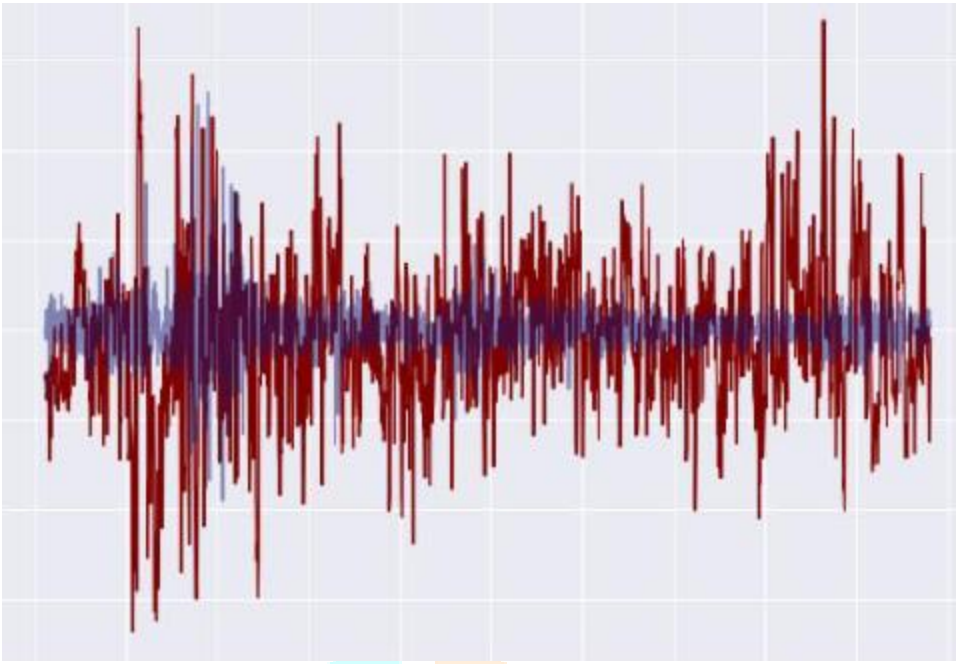

Fig.7. Hindunilvr Annual Return according to 2013- 2022 years

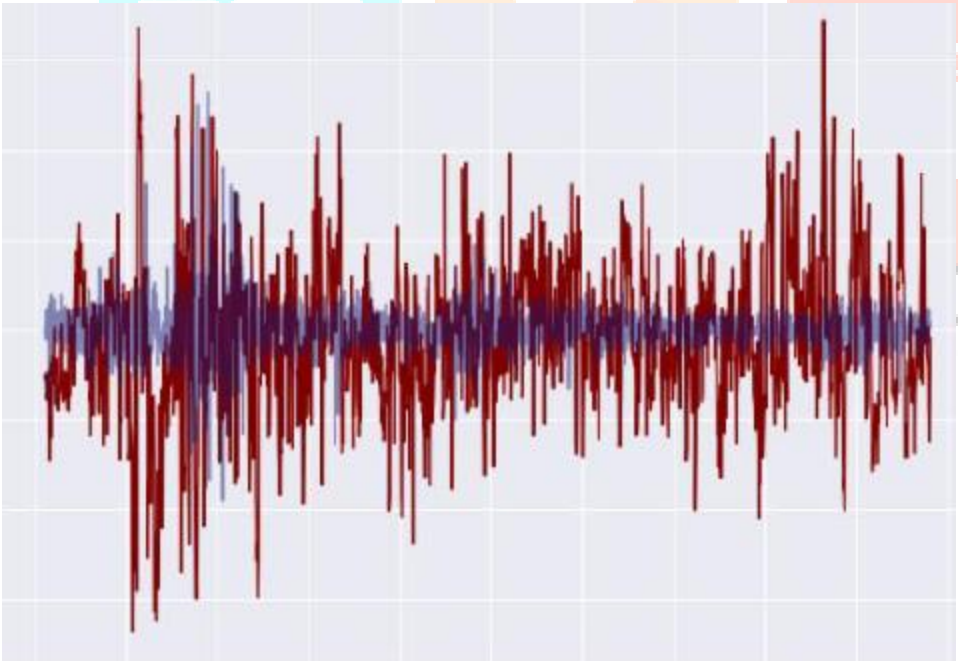

Fig.7. HDFC Bank's Annual Return according to 2013- 2022 years

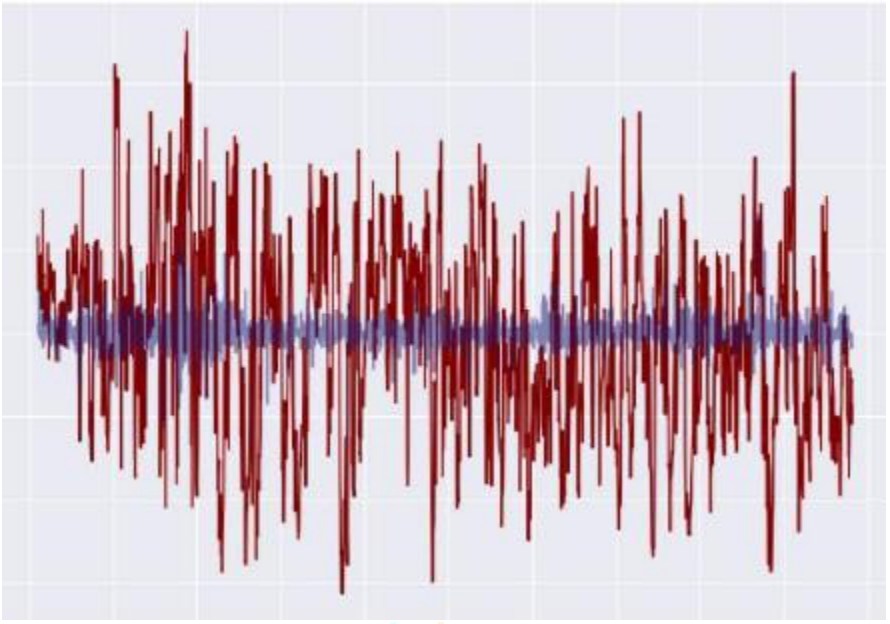

Fig.8. HDFC's Annual Return according to 2013- 2022 years

```
        print("the overall drawdown was:", round(-loss,2)*100,"%")
        print("the overall one time highest profit was seen:",
round(profit,2)*100,"%")
        list1 = [symbol, -loss, profit, date_l, date_p ]
        col = ["Stock name","max loss","max profit", "day of loss","day of
profit"]
        dict1 = dict(zip(col, list1))
        results = pd.DataFrame(data = dict1, index = [0])
        df5 = df5.append(results, ignore_index = True)
        #print(datetime.now())
    except Exception as e:
        print(e)

0.36369453924914663 2022-02-01 00:00:00
0.2940486348122866 2022-02-02 00:00:00
the overall drawdown was: -28.999999999999996 %
the overall one time highest profit was seen: 36.0 %
1.369330453563715 2022-04-01 00:00:00
the overall drawdown was: -137.0 %
the overall one time highest profit was seen: 152.0 %
0.29118325472510076 2021-11-04 00:00:00
0.36950459599958696 2021-11-08 00:00:00
```

Fig.9.Finanace Portfolio Output Analysis

We compare the Kelly optimal bet with the RCK bets for $\alpha = 0.7$ and $\beta = 0.1$ ($\lambda = 6.456$). We then obtain the RCK bets for $\lambda = 5.500$, a value chosen so that we achieve risk close to the specified value $\beta = 0.1$ (as discussed in § 3.5.1). For each bet vector we carry out 105 Monte Carlo simulations of wt for t = 1, . . . , 100. This allows us to estimate (well) the associated risk probabilities. Table 3.1 shows the results. The second column gives the

growth rate, the third column gives the bound on drawdown risk, and the last column gives the drawdown risk computed by Monte Carlo simulation. The Kelly optimal bet experiences a drawdown exceeding our threshold $\alpha = 0.7$ around 40% of the time. For all the RCK bets the drawdown risk (computed by Monte Carlo) is less than our bound, but not dramatically so. (We have observed this over a wide range of problems.) The RCK bet with $\lambda = 6.456$ is guaranteed to have a drawdown probability not exceeding 10%; Monte Carlo simulation shows that it is (approximately) 7.3%. For the third bet vector in our comparison, we decreased the risk aversion parameter until we obtained a bet with (Monte Carlo computed) risk near the limit 10%. The optimal value of the (hard) Kelly gambling problem with drawdown risk (3.9) must be less than 0.062 (the unconstrained optimal growth rate) and greater than 0.043 (since our second bet vector is guaranteed to satisfy the risk constraint). Since our third bet vector has drawdown risk less than 10%, we can further refine this result to state that the optimal value of the (hard) Kelly gambling problem with drawdown risk (3.9) is between 0.062 and 0.047. Figure 3.1 shows ten trajectories of wt in our Monte Carlo simulations for the Kelly optimal bet (left) and the RCK bet obtained with $\lambda = 5.5$ (right). The dashed line shows the wealth threshold $\alpha = 0.7$. Out of these ten simulations, four of the Kelly trajectories dip below the threshold $\alpha = 0.7$, and one of the other trajectories does, which is consistent with the probabilities reported above. Figure 3.2 shows the sample CDF of Wmin over the 105 simulated trajectories of wt , for the Kelly optimal bets and the RCK bets with $\lambda = 6.46$. The upper bound $\alpha\lambda$ is also shown. We see that the risk bound is not bad, typically around 30% or so higher than the actual risk. We have observed this to be the case across many problem instances.

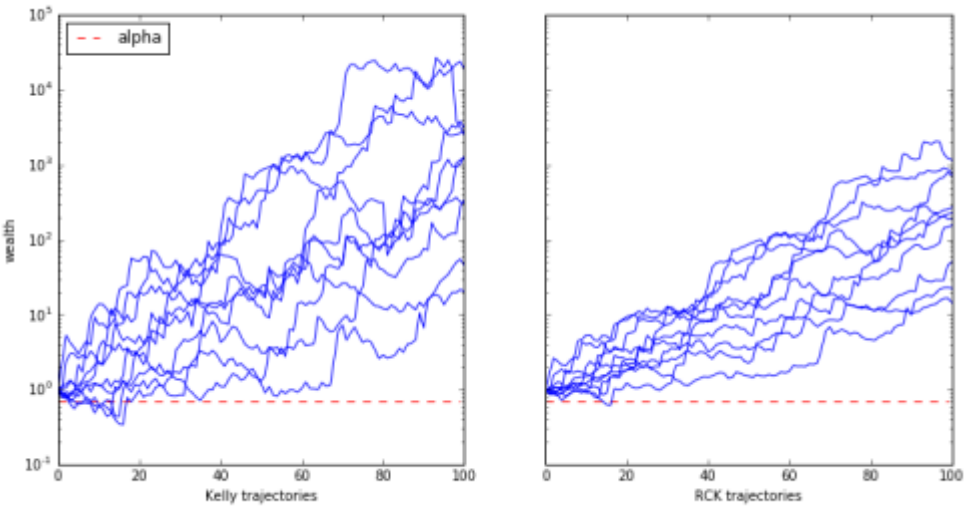

Fig.10. Analysis of Profit Trajectile

Conclusion

Here we present a new objective function for training any recurrent neural network using stochastic gradient optimization with backpropagation and a mini-batch approach. Using a generalised Kelly criterion, we design and test a function that describes the optimal wagering strategy when betting on a pool of horses whose individual odds of winning are already known. Volumetric medical picture segmentation with objective function. The conducted assessments show that the suggested optimization strategy outperforms the state-of-the-art approaches in terms of classification accuracy and training convergence speed. We discover that LSTM can learn from experience and improve its performance over time, while still being able to capture market movement patterns with a small amount of data and characteristics. This effectively resolves the profit, yearly return, and drawdown aspects of the financial portfolio selection issue (a gamble with infinite return distributions).

Future Enhancement

Potential future work could make use of risk indicators related to how the agent allots assets. The investing agent's outcomes could vary widely depending on the level of risk he or she is willing to take. We have considered the portfolio management problem from the perspective of the model-free reinforcement learning paradigm in this thesis. Using model-based reinforcement learning algorithms to teach the agent how to allocate its capital among a portfolio of stocks so as to maximise its total return could be the next step in this line of inquiry. To produce predictions, the environment's dynamics are modelled in model based reinforcement learning. Altering the criterion for success could be still another option. It would be possible to use the risk-adjusted return. We may also try to make the objective function more game-like in nature, with fewer moving parts. In comparison to a more complicated goal, a simpler one would help the agent improve its performance more quickly.

Reference

[1]Abdel-Basset, M., M. Saleh, A. Gamal, and F. Smarandache. 2019. “An Approach of TOPSIS Technique for Developing Supplier Selection with Group Decision Making Under Type-2 Neutrosophic Number.” Applied Soft Computing Journal 77: 438–452. doi:10.1016/j.asoc. 2019.01.035.

[2]Abdullah, L., and N. Zulkifli. 2015. “Integration of Fuzzy AHP and Interval Type-2 Fuzzy DEMATEL: An Application to Human Resource Management.” Expert Systems with Applications 42 (9): 4397–4409. doi:10.1016/j.eswa.2015.01.021.

[3]Akkaya, G., B. Turanoğlu, and S. Öztaş. 2015. “An Integrated Fuzzy AHP and Fuzzy MOORA Approach to the Problem of Industrial Engineering Sector Choosing.” Expert Systems with Applications 42 (24): 9565–9573. doi:10.1016/j.eswa.2015.07.061.

[4]Alhayani, B., and A. A. Abdallah. 2020. “Manufacturing Intelligent Corvus Corone Module for a Secured Two Way Image Transmission Under WSN.” Engineering Computations 37 (9): 1–17. doi:10.1108/EC-02-2020-0107.

[5]Alhayani, B., and H. Ilhan. 2020. “Efficient Cooperative Imge Transmission in One-way Mult-hop Sensor Network.” International Journal of Electrical Engineering Education 57 (2): 321–339.

[6]Al Hayani, B., and H. Ilhan. 2020. “Image Transmission Over Decode and Forward Based Cooperative Wireless Multimedia Sensor Networks for Rayleigh Fading Channels in Medical Internet of Things (MIoT) for Remote Health-Care and Health Communication Monitoring.” Journal of Medical Imaging And Health Informatics 10: 160–168.

[7]Al Hayani, B., and H. Ilhan. 2020. "Visual Sensor Intelligent Module Based Image Transmission in Industrial Manufacturing for Monitoring and Manipulation Problems." Journal of Intelligent Manufacturing 4: 1–14.
[8] Amiri, M. P. 2010. "Project Selection for Oil-fields Development by Using the AHP and Fuzzy TOPSIS Methods." Expert Systems with Applications 37 (9): 6218–6224. doi:10.1016/j.eswa. 2010.02.103.
[9]Arasteh, A., A. Aliahmadi, and M. M. Omran. 2014. "A Multi-stage Multi Criteria Model for Portfolio Management." Arabian Journal for Science and Engineering 39 (5): 4269–4283. doi:10.1007/s13369-014-0987-9.
[10]Behzadian, M., S. Khanmohammadi Otaghsara, M. Yazdani, and J. Ignatius. 2012. "A State-of the- Art Survey of TOPSIS Applications." Expert Systems with Applications 39 (17): 13051–13069. doi:10.1016/j.eswa.2012.05.056.
[11]Benaija, K., and L. Kjiri. 2015. "Project Portfolio Selection: Multi-criteria Analysis and Interactions Between Projects." International Journal of Computer Science Issues 11 (6): 134–143. http:// arxiv.org/abs/1503.05366.
[12]Beringer, C., D. Jonas, and A. Kock. 2013. "Behavior of Internal Stakeholders in Project Portfolio Management and Its Impact on Success." International Journal of Project Management 31 (6): 830–846. doi:10.1016/j.ijproman.2012.11.006.
[13]Beşikçi, E. B., T. Kececi, O. Arslan, and O. Turan. 2016. "An Application of Fuzzy-AHP to Ship Operational Energy Efficiency Measures." Ocean Engineering 121: 392–402. doi:10.1016/j. oceaneng.2016.05.031.
[14]Beskese, A., H. H. Demir, H. K. Ozcan, and H. E. Okten. 2015. "Landfill Site Selection Using Fuzzy AHP and Fuzzy TOPSIS: A Case Study for Istanbul." Environmental Earth Sciences 73 (7): 3513–3521. doi:10.1007/s12665-014-3635-5.
[15]Blichfeldt, B. S., and P. Eskerod. 2008. "Project Portfolio Management – There's More to it Than What Management Enacts." International Journal of Project Management 26 (4): 357–365. doi:10.1016/j.ijproman.2007.06.004.
[16]Boutkhoum, O., M. Hanine, T. Agouti, and A. Tikniouine. 2017. "A Decision-Making Approach Based on Fuzzy AHP-TOPSIS Methodology for Selecting the Appropriate Cloud Solution to Manage Big Data Projects." International Journal of Systems Assurance Engineering and Management 8: 1237–1253. doi:10.1007/s13198-017-0592-x.
[17]da Silva, Edson Filisbino Freire, et al. "A machine learning approach for monitoring Brazilian optical water types using Sentinel-2 MSI." Remote Sensing Applications: Society and Environment 23 (2021): 100577.
[18]Malhotra, Ruchika. "A systematic review of machine learning techniques for software fault prediction." Applied Soft Computing 27 (2015): 504-518.
[19]Pang, Yulei, Xiaozhen Xue, and Akbar Siami Namin. "Identifying effective test cases through k-means clustering for enhancing regression testing." 2013 12th International Conference on Machine Learning and Applications. Vol. 2. IEEE, 2013.
[20]Bhushan, Megha, et al. "A classification and systematic review of product line feature model defects." Software Quality Journal 28.4 (2020): 1507-1550.
[21]Malhotra, Ruchika, Arvinder Kaur, and Yogesh Singh. "Empirical validation of object-oriented metrics for predicting fault proneness at different severity levels using support vector machines." International Journal of System Assurance Engineering and Management 1.3 (2010): 269-281.
[22]Lincke, Rüdiger, Tobias Gutzmann, and Welf Löwe. "Software quality prediction models compared." 2010 10th International Conference on Quality Software. IEEE, 2010.

[23]Xu, Xiaobin, et al. "A belief rule-based evidence updating method for industrial alarm system design." Control Engineering Practice 81 (2018): 73-84.

[24]Zhou, Shuren, and Bo Tan. "Electrocardiogram soft computing using hybrid deep learning CNN-ELM." applied soft computing 86 (2020): 105778.

25. Scaled Agile Framework: SAFe 5.0 Framework - SAFe Big Picture. Scaled Agile Framework (2020). https://www.scaledagileframework.com/

26. Hansen, L.K., Kræmmergard, P.: Discourses and theoretical assumptions in IT project portfolio management: a review of the literature. Int. J. Inf. Technol. Proj. Manag. (IJITPM) 5(3), 39–66 (2014)

27. Hoda, R., Kruchten, P., Noble, J.,Marshall, S.: Agility in context. In: 2010 Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications, pp. 74–88. ACM (2010)

28. Karlström, D., Runeson, P.: Integrating agile software development into stage-gate managed product development. Empir. Softw. Eng. 11(2), 203–225 (2006). https://doi.org/10.1007/s10 664-006-6402-8

29. Abrantes, R., Figueiredo, J.: Resource management process framework for dynamic NPD portfolios. Int. J. Project Manag. 33(6), 1274–1288 (2015)

30. Lee, G., Xia,W.: Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility. MIS Q. 34(1), 87–114 (2010)

31. Theobald, S., Schmitt, A., Diebold, P.: Comparing scaling agile frameworks based on underlying practices. In: Hoda, R. (ed.) XP 2019. LNBIP, vol. 364, pp. 88–96. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_11

32. Alqudah, M., Razali, R.: A review of scaling agile methods in large software development. Int. J. Adv. Sci. Eng. Inf. Technol. 6(6), 828–837 (2016)

33. Lappi, T., Karvonen, T., Lwakatare, L.E., Aaltonen, K., Kuvaja, P.: Toward an improved understanding of agile project governance: a systematic literature review. Proj. Manag. J. 49(6), 39–63 (2018)

34. Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: a systematic literature review. J. Syst. Softw. 119, 87–108 (2016)

35. Ahmad, M.O., Dennehy, D., Conboy, K., Oivo, M.: Kanban in software engineering: a systematic mapping study. J. Syst. Softw. 137, 96–113 (2018)

36. Tranfield, D., Denyer, D., Smart, P.: Towards a methodology for developing evidenceinformed management knowledge by means of systematic review. Br. J. Manag. 14(3), 207–222 (2003)

37. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering (2007)

38. Okoli, C.: A guide to conducting a standalone systematic literature review. Commun. Assoc. Inf. Syst. 37(1), 43 (2015)

39. Wohlin,C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: 2014 Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–10 (2014)

40. Cruzes, D.S., Dyba, T.: Recommended steps for thematic synthesis in software engineering. In: 2011 International Symposium on Empirical Software Engineering and Measurement, pp. 275–284. IEEE (2011)

41. Zhou, X., Jin, Y., Zhang, H., Li, S., Huang, X.: A map of threats to validity of systematic literature reviews in software engineering. In: 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), pp. 153–160. IEEE (2016)

42. Archer, N.P., Ghasemzadeh, F.: An integrated framework for project portfolio selection. Int. J. Project Manag. 17(4), 207–216 (1999)

43. Killen, C.P., Hunt, R.A., Kleinschmidt, E.J.: Managing the new product development project portfolio: a review of the literature and empirical evidence. In: PICMET 2007 Portland International Conference on Management of Engineering & Technology 2007, pp. 1864–1874. IEEE (2007)

44. Martinsuo, M., Lehtonen, P.: Role of single-project management in achieving portfolio management efficiency. Int. J. Project Manag. 25(1), 56–65 (2007)

45. Cao, L., Ramesh, B., Abdel-Hamid, T.: Modeling dynamics in agile software development. ACM Trans. Manag. Inf. Syst. (TMIS) 1(1), 1–26 (2010)

46. Cao, L.,Mohan, K., Ramesh, B., Sarkar, S.: Adapting funding processes for agile IT projects: an empirical investigation. Eur. J. Inf. Syst. 22(2), 191–205 (2013)

47. Moran, A.: Managing Agile: Strategy, Implementation, Organisation and People. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16262-1

48. Sahota, M., Bogsnes, B., Nyfjord, J., Hesselberg, J., Drugovic, A.: Beyond Budgeting: a Proven Governance System Compatible with Agile Culture. BBRT (2014). http://bbrt.co.uk/bbfiles/BeyondBudgetingAgileWhitePaper_2014.pdf

49. Knaster, R., Leffingwell, D.: SAFe 4.5 Distilled: Applying the Scaled Agile Framework for Lean Enterprises. Addison-Wesley Professional, Boston (2018)

50. Racheva, Z., Daneva, M.: Using measurements to support real-option thinking in agile software development. In: Proceedings of the 2008 InternationalWorkshop on Scrutinizing Agile Practice (Shoot-out at the agile corral), pp. 15–18. ACM (2008)

51. Reinertsten, D.G.: The Principles of Product Development Flow: Second Generation Lean Product Development. Celeritas, Redondo Beach (2009)

[52] J. Bergstra, "Hyperopt: Distributed asynchronous hyper-parameter optimization," [web page] https://github.com/hyperopt/hyperopt, 2016.

[53] K. O. Babalola, B. Patenaude, P. Aljabar, J. Schnabel, D. Kennedy, W. Crum, S. Smith, T. Cootes, M. Jenkinson, and D. Rueckert, "An evaluation of four automatic methods of segmenting the subcortical structures in the brain," NeuroImage, vol. 47, no. 4, pp. 1435–1447, 2009.

[54] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge, "Comparing images using the Hausdorff distance," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 9, pp. 850– 863, 1993.