Wiley
Applied Computational Intelligence and Soft Computing
Volume 2024, Article ID 8694583, 22 pages
https://doi.org/10.1155/2024/8694583

WILEY

Research Article

Fuzzy Neural Network for Fuzzy Quadratic Programming With Penalty Function and Mean-Variance Markowitz Portfolio Model

Izaz Ullah Khan,[1] **Muhammad Aamir**,[1] **Mehran Ullah**,[2] **and Muhammad Shahbaz Shah**[1]

[1]*Department of Mathematics, COMSATS University Islamabad, Abbottabad Campus, Abbottabad, Pakistan*
[2]*School of Business and Creative Industries, University of the West of Scotland, Paisley PA1 2BE, UK*

Correspondence should be addressed to Mehran Ullah; mehran.ullah@uws.ac.uk

Received 16 March 2024; Revised 25 August 2024; Accepted 16 September 2024

Academic Editor: Nur Ezlin Zamri

This research tries to integrate fuzzy neural networks with penalty function to address the quadratic programming based on the mean-variance Markowitz portfolio model. The fuzzy quadratic programming problem with penalty function consists of the lower, central, and upper models. The models utilize fuzzy neural networks to solve the models. The proposed method has been implemented on the six leading stocks in the Pakistan Stock Exchange. The approach identifies the ideal portfolios for potential investors in the Pakistan Stock Exchange. Data of the six popular stocks trading on the stock exchange from January 2016 to October 2020 are taken into consideration. The optimizers are RMSprop, Momentum, Adadelta, Adagrad, Adam, and gradient descent, respectively. The findings of all the optimizers at all three phases (lower, central, and upper) agree on identifying the optimal investment portfolios for investors. The optimizers recommend investing in either one of the two categories. The first group recommends investing in the FFC, ARPL, and UPFL portfolios. The second group recommends LUCK, AGTL, and IGIHL. The first group tends to enhance return, variability, and risk. It is a high-risk group. The second group aims to reduce return variability while lowering risk. It is a risk-averse group. It is evident that all of the optimizers recommend investing in FFC, ARPL, and UPFL, with the exception of the Adam and Adadelta optimizers, which recommends investment in IGIHL, AGTL, and LUCK. RMSprop, Momentum, Adagrad, and gradient descent increase variability, risk, and returns. Adam proves the best optimizer, then RMSprop, and finally, Adagrad. Adam, Adadelta, and RMSprop are sensitive, whereas momentum and gradient descent are irresponsive to fuzzy uncertain data. The percent improvement in the objective is 0.59% and 0.18% for the proposed Adagrad and Adadelta, respectively.

Keywords: fuzzy neural networks; fuzzy quadratic programming; mean-variance portfolio model; Morkowitz portfolio model; penalty function

1. Introduction

Neural network was presented by Rosenblatt [1] in his seminal paper. The concept of neural network has opened new avenues of research in the field of machine learning, data science, and artificial intelligence. The first one who utilized a neural network to recognize the handwritten zip code was LeCun et al. [2]. Jacob explored the steepest descent method in 1988. The momentum method was discovered by Polyak in 1964, and the accelerated gradient method was introduced by Nesterov in 1983. These developments led to the ideas of batch gradient descent (BGD), stochastic gradient descent (SGD), and mini-batch gradient descent (MGD). Moreover, the development of support vector machine proved to be a promising machine learning tool.

The proposed study integrates neural network, quadratic programming, portfolio optimization, and fuzzy uncertainty. Thus, the literature review starts with the inception of neural network and quadratic programming, followed by optimal portfolio selection using quadratic programming and modeling of fuzzy type uncertainty in the system

modeling. Afterward, the elaboration of the proposed work is presented. Thus, the literature review focuses down from a broad concept to the specific concepts relevant to the research study. Thus, the flow of the research concepts follows a systematic review of the literature of the relevant topics.

Initially, few optimization researchers were familiar with the concept of quadratic optimization [3]. In 1956, Markowitz explored quadratic programming problems [4]. Markowitz was acknowledged as the "father of quadratic programming." The idea was further studied in [5–7]. Artificial neural systems were used to address conventional optimization methods. The studies [8, 9] addressed linear and quadratic programming problems with neural networks. In Bouzerdoum and Pattison [10], the authors addressed bounded quadratic programming problems utilizing the neural networks. Linear programming problems were solved using neural networks, as described by Zak et al. [11]. Subsequently, Wu et al. [12] adopted methods to improve their efficacy. Subsequently, Xia and Wang [13] developed basic neural networks for specially solving convex quadratic programming problems. Together, these advances have addressed key issues in optimization and demonstrated the growing capabilities of neural networks to solve increasingly complex mathematical problems. The current method will utilize penalty function method together with the neural network tools to solve contained quadratic programming problems.

The theory of portfolio optimization was first discovered in the seminal paper of Markowitz [14]. He explained the anticipated returns of the portfolio as desired and variance of the returns as undesired measures. Mean-variance optimization (MVO) was defined by the conception of hedging presented by Michaud [15]. The MVO can compute the optimized portfolio, but it can propose a portfolio that is not ideal fiscally Richard and Roncalli [16]. The tremendous developments in the robotic consultants, artificial intelligence, and machine learning attracted researchers to research the conventional portfolio optimization. The fundamental MVO approach developed by Markowitz has impressive significance in portfolio optimization [17]. Markowitz devised quadratic programming for solving optimal portfolio allocation problems [18]. Roa's quadratic entropy maximized the enhancement technique of indexing and presented the clear idea to investors [19]. Analysts and writers [20, 21] have encountered portfolio optimization problems utilizing varying procedures.

Zadeh [22] proposed the idea of fuzzy sets. The idea was then successfully applied in the fields of science, engineering, technology, economics, and management. It has been applied to mathematical programming problems [23–28]. Adaptive dynamic techniques were used in Khan and Aftab [29]. Machine learning techniques were utilized for delay prediction problem in aviation industry [30]. A fuzzy relational approach for quadratic programming problem was adopted in Molai [31]. In Barik and Biswal [32], probability techniques were used for quadratic programming problems. Necessary optimal conditions for fuzzy quadratic programming problems were proved by Zhou, Cao, and Nasseri [33]. Bai and Bao [34] presented fuzzy quadratic programming problems in an uncertain environment. An ABS algorithm for fuzzy quadratic programming problem was proposed by Ghanbari and Moghadam [35]. A new method was proposed by Umamaheswari and Ganesan [36] for fuzzy quadratic programming problems. In Elshafei [37], the fuzzy quadratic programming problem has unrestricted variables. Multiobjective quadratic programming problems were proposed in [38, 39]. An interactive approach for multiobjective quadratic programming was proposed by Khalifa [40]. Fuzzy quadratic programming problems were decomposed for solution in Taghi-Nezhad and Babakordi [41]. Wang, He, and Shi [42] studied portfolio optimization using fuzzy quadratic programming. Malek and Oskoei [43] used neural network to solve constrained quadratic programming problems. Mansoori, Effati, and Eshaghnezhad [44] solved fuzzy nonlinear programming problem using neural network. In Mansoori, Effati, and Eshaghnezhad [45], the authors used neural network to solve quadratic programming problems in fuzzy uncertain environment. Coelho [46] adopted a dual approach to solve real-world fuzzy quadratic programming problems. Chaweewanchon and Chaysiri [47] utilized machine learning techniques to address Markowitz MVO problems. Faturohman and Christian [48] implemented Markowitz MVO approach in Indonesian stock exchange. Dai et al. [49] solved quadratic programming with noised interference utilizing a noise-immune neural network in a fuzzy environment. Hasan and Kanani [50] utilized decagonal fuzzy numbers with membership function to address fuzzy quadratic fractional programming problems. Tadesse et al. [51] used a goal programming technique for solving fuzzy multiobjective quadratic programming problems.

In this study, the penalty function method is used along with neural network to solve fuzzy quadratic programming problems. The method is applied to mean-variance Markowitz portfolio optimization model in the Pakistan stock exchange. Table 1 presents a summary of the novelty of the proposed technique with the existing techniques. It is obvious that researchers have used different methods to solve fuzzy quadratic programming problems. The proposed penalty function method has not been used previously to solve quadratic programming problems in a fuzzy uncertain environment; thus, our proposed techniques have methodological and population gaps with the existing literature.

The motivations for working on this model are three folds: first, the incorporation of the new penalty function method to solve fuzzy quadratic programming problems; second, solving fuzzy quadratic programming with the novel fuzzy neural networks; and third, implementing it to the mean-variance Markowitz portfolio optimization model in the Pakistan stock exchange.

The statement of the proposed research problem is "How fuzzy neural network with penalty function can be used to formulate quadratic Programming problems with fuzzy uncertainty and then apply it to portfolio optimization problem in stock exchange?"

The objectives of the research are given as follows:

Table 1: Methods in the literature for fuzzy quadratic programming problems.

Study	Approach	Gap
Molai [31]	Fuzzy relational approach	Methodological and population
Barik and Biswal [32]	Probabilistic approach	Methodological and population
Ghanbari and Moghadam [35]	ABS algorithm	Methodological and population
Taghi-Nezhad and Babakordi [41]	Decomposition method	Methodological and population
Malek and Oskoei [43]	Neural network for QP	Methodological and population
Mansoori, Effati, and Eshaghnezhad [44, 45]	Neural network	Methodological and population
Coelho [46]	Dual approach	Methodological and population
Faturohman and Christian [48]	Markowitz mean-variance optimization	Methodological and population
Dai et al. [49]	Noise immune neural network	Methodological and population
Hasan and Kanani [50]	Fractional quadratic programming	Methodological and population
Tadesse et al. [51]	Goal programming for multiobjective quadratic programming	Methodological and population

1. To formulate a fuzzy neural network with penalty function for fuzzy quadratic programming problems.
2. To formulate the Lagrange's function for fuzzy quadratic programming problem.
3. To implement the proposed methodology on portfolio optimization problem in the Pakistan Stock Exchange.
4. To analyze the results and propose suggestions for the portfolio optimization problem in the capital market structure.

The main contributions of the proposed study are listed as follows:

1. Incorporation of penalty function for solving fuzzy quadratic programming problems with fuzzy neural network.
2. Development of Lagrange's function with penalty function for fuzzy quadratic programming problem.
3. Implementing the proposed methodology on portfolio optimization problem in the Pakistan Stock Exchange.
4. Analyze the results and propose insightful perspectives to potential investors.

This study focuses on fuzzy neural network for fuzzy quadratic programming with a penalty function, incorporating the mean-variance Markowitz portfolio model. The use of a fuzzy neural network on this context is especially justified because of its capability to deal with the inherent uncertainties and imprecision in economic information, which can be often encountered in portfolio optimization troubles. By incorporating fuzzy logic, this method permits for a robust and flexible modeling of uncertainties, leading to correct and reliable investment and funding strategies.

This study is organized as follows. After the introduction, the proposed penalty function method for solving fuzzy quadratic programming problem with neural network is proposed in Section 2. In Section 3, the proposed approach adopted in 2 is converted into lower, central, and upper models. In Section 4, the proposed methodology is adopted for the mean-variance fuzzy quadratic programming problem in the Pakistan Stock Exchange. Results and discussions are given in Section 5, and the study is concluded in Section 6.

2. Proposed Fuzzy Artificial Neural Network for Solving Quadratic Programming Problem With Penalty Function

The fuzzy quadratic programming problem in the form of Klir & Yuan [52] is given equation (1). When "s" represents a central value with "l" and "r," its left and right uncertainty range

$$\begin{aligned}&\text{Min } f^{(s,l,r)}(x) = \left(\frac{1}{2}x^t \text{Ax} + b^t x\right)^{(s,l,r)} \quad y \in R^t,\\ &\text{s.t.}\\ &\left(c_i x + d_i - s_i\right)^{(s,l,r)} = (0)^{(s,l,r)},\\ &\left\{\begin{array}{c} s_i^{(s,l,r)} = 0, \quad i \in I = \{0,1,2,\ldots,n-1\} \\ s_i^{(s,l,r)} \geq 0, \quad i \in J = \{n+0, n+1, n+2, \ldots m+n-1\}\end{array}\right\}.\end{aligned} \tag{1}$$

The first "n" constraints are equality constraints, and the next "m" constraints are inequality for a total of "$n+m$" constraints. The Lagrangian with optimal solution $x^{*(s,l,r)}, s^{*(s,l,r)}, \lambda^{*(s,l,r)}$ is given as follows:

$$\begin{aligned}L^{(s,l,r)}\left(x^{(s,l,r)}, s^{(s,l,r)}, \lambda^{(s,l,r)}\right) &= \left(f\left(x^{(s,l,r)}\right) - \sum_{i\in I,J} \lambda_i^{(s,l,r)}\left(c_i x^{(s,l,r)} + d_i^{(s,l,r)} - s_i^{(s,l,r)}\right)\right)^{(s,l,r)},\\ s_i^{(s,l,r)} \geq 0, \quad \lambda_i^{(s,l,r)} \geq 0, \quad &\forall i \in J.\end{aligned} \tag{2}$$

The penalty function P_μ of μ with solutions $x_\mu^{*(s,l,r)}, s_\mu^{*(s,l,r)}, \lambda_\mu^{*(s,l,r)}$ is presented as follows:

$$P_{\mu}^{(s,l,r)}\left(x^{(s,l,r)}, s^{(s,l,r)}, \lambda^{(s,l,r)}\right) = \begin{pmatrix} f\left(x^{(s,l,r)}\right) - \sum_{i\in I,J} \lambda_i^{(s,l,r)}\left(c_i^{(s,l,r)} x^{(s,l,r)} + d_i^{(s,l,r)} - s_i{}^{(s,l,r)}\right) + \\ \\ \frac{1}{2\mu} \quad \sum_{i\in I,J} \lambda_i^{(s,l,r)}\left(c_i^{(s,l,r)} x^{(s,l,r)} + d_i^{(s,l,r)} - s_i{}^{(s,l,r)}\right)^2 \end{pmatrix}^{(s,l,r)}, \tag{3}$$

$$s_i^{(s,l,r)} \geq 0, \quad \lambda_i^{(s,l,r)} \geq 0, \quad \forall\, i \in J.$$

The partial derivatives are equated to zero in equation (4) for optimal points. The optimal value for constraint "i", s_{μ} in equation (5).

$$\frac{\partial P_{\mu}^{(s,l,r)}\left(x_{\mu}^{(s,l,r)}, s_{\mu}^{(s,l,r)}, \lambda_{\mu}^{(s,l,r)}\right)}{\partial s_{\mu,i}^{(s,l,r)}} = 0 = \lambda_{\mu,i}^{(s,l,r)} - \frac{1}{\mu}\left(c_i^{(s,l,r)} x_{\mu}^{(s,l,r)} + d_i^{(s,l,r)} - s_{\mu,i}^{(s,l,r)}\right), \tag{4}$$

$$s_{\mu,i}^{(s,l,r)} = c_i^{(s,l,r)} x_{\mu}^{(s,l,r)} + d_i^{(s,l,r)}. \tag{5}$$

Thus, $s_{\mu,i}^{(s,l,r)} = c_i^{(s,l,r)} x_{\mu}^{(s,l,r)} + d_i^{(s,l,r)}$, and as $\mu \longrightarrow 0$, or when $\lambda_i^{(s,l,r)} = 0$. Recall that $s_{\mu,i}^{(s,l,r)} \geq 0, \forall\, i \in J$. Thus, equation (5) becomes Equation (6).

$$s_{\mu,i}^{*(s,l,r)} = \begin{Bmatrix} 0 & i \in I \\ 0 & c_i^{(s,l,r)} x_{\mu}^{(s,l,r)} + d_i^{(s,l,r)} < \mu\lambda_{\mu,i}^{(s,l,r)}, i \in J \\ c_i^{(s,l,r)} x_{\mu}^{(s,l,r)} + d_i^{(s,l,r)} - \mu\lambda_{\mu,i}^{(s,l,r)} & \text{otherwise} \end{Bmatrix}. \tag{6}$$

Let $s_{\mu,i}^{*(s,l,r)} = (s_{\mu,0}^{*(s,l,r)}, s_{\mu,1}^{*(s,l,r)}, s_{\mu,2}^{*(s,l,r)}, \ldots s_{\mu,m}^{*(s,l,r)})$. The minimization problem equation (3) is updated equation (7).

$$\begin{aligned} \text{Min} P_{\mu}^{(s,l,r)}\left(x_{\mu}^{(s,l,r)}, s_{\mu}^{*(s,l,r)}, \lambda_{\mu}^{(s,l,r)}\right) = f_{\mu,i}^{(s,l,r)}(x) - \sum_{i\in I,J} \lambda_{\mu,i}^{(s,l,r)}\left(c_i^{(s,l,r)} x_{\mu}^{(s,l,r)} + d_{\mu,i}^{(s,l,r)} - s_{\mu,i}^{*(s,l,r)}\right) \\ + \frac{1}{2\mu} \sum_{i\in I,J} \lambda_{\mu,i}^{(s,l,r)}\left(c_i^{(s,l,r)} x_{\mu}^{(s,l,r)} + d_{\mu,i}^{(s,l,r)} - s_{\mu,i}^{*(s,l,r)}\right)^2 \end{aligned} \tag{7}$$

$$\lambda_{\mu,i}^{(s,l,r)} \geq 0.$$

The gradient of equation (7) is equation (8).

$$\begin{aligned} \nabla_x P_{\mu}^{(s,l,r)}\left(x_{\mu}^{(s,l,r)}, s_{\mu}^{*(s,l,r)}, \lambda_{\mu}^{(s,l,r)}\right) = 0, \\ A^{(s,l,r)} x_{\mu}^{(s,l,r)} + b_{\mu,i}^{(s,l,r)} - \lambda_{\mu}^{t(s,l,r)} C^{(s,l,r)} x^{(s,l,r)} + \frac{C^{(s,l,r)}}{\mu}\left(C^{(s,l,r)} x_{\mu}^{(s,l,r)} + d_i^{(s,l,r)} - s_{\mu}^{*(s,l,r)}\right) = 0. \end{aligned} \tag{8}$$

The matrix "$A^{(s,l,r)}$" is assumed to be symmetric and positive definite. Each "$c_i^{(s,l,r)}$" is a row in the constraint matrix "$C^{(s,l,r)}$." Let μ_k be the *kth* element of the sequence of $\mu \longrightarrow 0$, then the following optimal conditions hold. As $x_{\mu k}^{*(s,l,r)} \longrightarrow x^{*(s,l,r)}$, then $P_\mu^{(s,l,r)}(x_{\mu k}^{*(s,l,r)}, s_\mu^{*(s,l,r)}, \lambda^{(s,l,r)}) \longrightarrow L^{(s,l,r)}(x^{*(s,l,r)}, s_\mu^{*(s,l,r)}, \lambda^{(s,l,r)})$.

Karush–Kuhn–Tucker (KKT) points $x_{\mu k}^{*(s,l,r)}$ for $P_\mu^{(s,l,r)}(x^{(s,l,r)}, s_\mu^{*(s,l,r)}, \lambda^{(s,l,r)})$, $\lambda^{(s,l,r)}$ can be refined as shown in equation (9).

$$\lambda_{k+1}^{(s,l,r)} = \lambda_k^{(s,l,r)} - \frac{\left(C^{(s,l,r)} x^{(s,l,r)} + d^{(s,l,r)} - s_\mu^{*(s,l,r)}\right)}{u}. \tag{9}$$

Thus, the Lagrange's multipliers $\lambda^{(s,l,r)} \longrightarrow \lambda^{*(s,l,r)}$. Knowing that $\lambda_i^{(s,l,r)} \geq 0, \forall i \in J$. Thus, equation (9) can be updated as equation (10).

$$\lambda_{\mathrm{ik+1}}^{(s,l,r)} = \begin{Bmatrix} 0 & \lambda_{\mathrm{ik}}^{(s,l,r)} < \dfrac{\left(c_i^{(s,l,r)} x_\mu^{(s,l,r)} + d_i^{(s,l,r)} - s_{i,\mu}^{*(s,l,r)}\right)}{u}, \quad \forall i \in J \\ \lambda_{\mathrm{ik}}^{(s,l,r)} - \dfrac{\left(c_i^{(s,l,r)} x_\mu^{(s,l,r)} + d_i^{(s,l,r)} - s_{i,\mu}^{*(s,l,r)}\right)}{u} & \text{otherwise} \end{Bmatrix}, \tag{10}$$

where $\lambda_\mu^{*(s,l,r)}$ is the limit of the sequence of $\lambda_k^{*(s,l,r)} = (\lambda_{0k}^{(s,l,r)}, \lambda_{1k}^{(s,l,r)}, \lambda_{2k}^{(s,l,r)}, \ldots \lambda_{\mathrm{mk}}^{(s,l,r)})$ for a given μ. Thus, we have equation (11)

$$\begin{aligned} &P_\mu^{(s,l,r)}\left(x_\mu^{*(s,l,r)}, s_\mu^{*(s,l,r)}, \lambda_\mu^{*(s,l,r)}\right) \longrightarrow L^{(s,l,r)}\left(x^{*(s,l,r)}, s^{*(s,l,r)}, \lambda^{*(s,l,r)}\right) = \\ &f^{*(s,l,r)}\left(x^{(s,l,r)}\right) - \sum_{i \in I,J} \lambda^{*(s,l,r)}\left(c_i^{(s,l,r)} x^{*(s,l,r)} + d_i^{(s,l,r)} - s^{*(s,l,r)}\right) \\ &s^{*(s,l,r)} \geq 0, \lambda^{*(s,l,r)} \geq 0, \forall\, i \in J, \quad \text{as } \mu \longrightarrow 0. \end{aligned} \tag{11}$$

Thus, equation (11) presents the optimal solution of the QP problem.

3. Proposed Fuzzy Artificial Neural Network for Solving Lower, Central, and Upper Bounds for QP Problem

The fuzzy quadratic programming problem presented in equation (1) is translated into the triangular fuzzy form equation (12) [52]. Here, "*s*" represents a central value with "*l*" and "*r*" in its left and right uncertainty range.

$$\begin{aligned} &\mathrm{Min} f^{(s-l,s,s+r)}(x) = \left(\frac{1}{2} x^t \mathrm{A} x + b^t x\right)^{(s-l,s,s+r)} \quad x \in R^n, \\ &\text{s.t.} \\ &\left(c_i x + d_i - S_i\right)^{(s-l,s,s+r)} = (0)^{(s-l,s,s+r)}, \\ &\begin{Bmatrix} S_i^{(s-l,s,s+r)} = 0, & i \in I = \{0, 1, 2, \ldots, n-1\} \\ S_i^{(s-l,s,s+r)} \geq 0, & i \in J = \{n+0, n+1, n+2, \ldots m+n-1\} \end{Bmatrix}. \end{aligned} \tag{12}$$

With "n" equality and "m" inequality constraints for a total of "$n+m$" constraints, the Lagrangian with optimal solution $x^{*(s-l,s,s+r)}, s^{*(s-l,s,s+r)}, \lambda^{*(s-l,s,s+r)}$ is presented in equation (13).

$$L^{(s-l,s,s+r)}\left(x^{(s-l,s,s+r)}, s^{(s-l,s,s+r)}, \lambda^{(s-l,s,s+r)}\right) = \left(f\left(x^{(s-l,s,s+r)}\right) - \sum_{i\in I,J} \lambda_i^{(s-l,s,s+r)}\left(c_i^{(s-l,s,s+r)} x^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} - s_i^{(s-l,s,s+r)}\right)\right)^{(s-l,s,s+r)} \quad (13)$$
$$s_i^{(s-l,s,s+r)} \geq 0, \lambda_i^{(s-l,s,s+r)} \geq 0, \quad \forall i \in J.$$

Let us define penalty function $P_\mu^{(s-l,s,s+r)}$ of μ with solutions $x_\mu^{*(s-l,s,s+r)}, s_\mu^{*(s-l,s,s+r)}, \lambda_\mu^{*(s-l,s,s+r)}$ equation (14).

$$P_\mu^{(s-l,s,s+r)}\left(x^{(s-l,s,s+r)}, s^{(s-l,s,s+r)}, \lambda^{(s-l,s,s+r)}\right) = \begin{pmatrix} f\left(x^{(s-l,s,s+r)}\right) - \sum_{i\in I,J} \lambda_i^{(s-l,s,s+r)}\left(c_i^{(s-l,s,s+r)} x^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} - s_i^{(s-l,s,s+r)}\right) + \\ \frac{1}{2\mu} \sum_{i\in I,J} \lambda_i^{(s-l,s,s+r)}\left(c_i^{(s-l,s,s+r)} x^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} - s_i^{(s-l,s,s+r)}\right)^2 \end{pmatrix}^{(s-l,s,s+r)} \quad (14)$$
$$s_i^{(s-l,s,s+r)} \geq 0, \quad \lambda_i^{(s-l,s,s+r)} \geq 0, \quad \forall i \in J.$$

For optimality, we calculate the partial derivatives and equate them to zero in equation (15). Now, we consider the optimal value for constraint "i," $s_\mu^{(s-l,s,s+r)}$ in equation (16).

$$\frac{\partial P_\mu^{(s-l,s,s+r)}\left(x_\mu^{(s-l,s,s+r)}, s_\mu^{(s-l,s,s+r)}, \lambda_\mu^{(s-l,s,s+r)}\right)}{\partial s_{\mu,i}^{(s-l,s,s+r)}} = 0, \quad (15)$$
$$\lambda_{\mu,i}^{(s-l,s,s+r)} - \frac{1}{\mu}\left(c_i^{(s-l,s,s+r)} x_\mu^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} - s_{\mu,i}^{(s-l,s,s+r)}\right) = 0,$$
$$s_{\mu,i}^{(s-l,s,s+r)} = c_i^{(s-l,s,s+r)} x_i^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} - \mu\lambda_{\mu,i}^{(s-l,s,s+r)}. \quad (16)$$

The slacks are $s_{\mu,i}^{(s-l,s,s+r)} = c_i^{(s-l,s,s+r)} x_\mu^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)}$, as $\mu \longrightarrow 0$, or when $\lambda_i^{(s-l,s,s+r)} = 0$. Knowing that $s_{\mu,i}^{(s-l,s,s+r)} \geq 0, \forall i \in J$. Thus, equation (16) becomes equation (17).

$$s_{\mu,i}^{*(s-l,s,s+r)} = \begin{Bmatrix} 0 & i \in I \\ 0 & c_i^{(s-l,s,s+r)} x_\mu^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} < \mu\lambda_{\mu,i}^{(s-l,s,s+r)}, \quad i \in J \\ c_i^{(s-l,s,s+r)} x_\mu^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} - \mu\lambda_{\mu,i}^{(s-l,s,s+r)} & \text{otherwise} \end{Bmatrix}. \quad (17)$$

Let $s_{\mu,i}^{*(s-l,s,s+r)} = (s_{\mu,0}^{*(s-l,s,s+r)}, s_{\mu,1}^{*(s-l,s,s+r)}, s_{\mu,2}^{*(s-l,s,s+r)}, \ldots s_{\mu,m}^{*(s-l,s,s+r)})$. The minimization problem equation (14) can be updated as equation (18).

$$\begin{aligned}
&\text{Min} \quad P_{\mu}^{(s-l,s,s+r)}\left(x_{\mu}^{(s-l,s,s+r)}, s_{\mu}^{*(s-l,s,s+r)}, \lambda_{\mu}^{(s-l,s,s+r)}\right) = \\
&= f_{\mu,i}^{(s-l,s,s+r)}(x) - \sum_{i\in I,J} \lambda_{\mu,i}^{(s-l,s,s+r)}\left(c_i^{(s-l,s,s+r)} x_{\mu}^{(s-l,s,s+r)} + d_{\mu,i}^{(s-l,s,s+r)} - s_{\mu,i}^{*(s-l,s,s+r)}\right) \\
&\quad + \frac{1}{2\quad \mu} \sum_{i\in I,J} \lambda_{\mu,i}^{(s-l,s,s+r)}\left(c_i^{(s-l,s,s+r)} x_{\mu}^{(s-l,s,s+r)} + d_{\mu,i}^{(s-l,s,s+r)} - s_{\mu,i}^{*(s-l,s,s+r)}\right)^2, \\
&\lambda_{\mu,i}^{(s-l,s,s+r)} \geq 0.
\end{aligned} \tag{18}$$

The gradient of equation (18) is equation (19).

$$\begin{aligned}
&\nabla_x P_{\mu}^{(s-l,s,s+r)}\left(x_{\mu}^{(s-l,s,s+r)}, s_{\mu}^{*(s-l,s,s+r)}, \lambda_{\mu}^{(s-l,s,s+r)}\right) = 0, \\
&A^{(s-l,s,s+r)} x_{\mu}^{(s-l,s,s+r)} + b_{\mu,i}^{(s-l,s,s+r)} - \lambda_{\mu}^{t(s-l,s,s+r)} C^{(s-l,s,s+r)} x^{(s-l,s,s+r)} \\
&\quad + \frac{C^{(s-l,s,s+r)}}{\mu}\left(C^{(s-l,s,s+r)} x_{\mu}^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} - s_{\mu}^{*(s-l,s,s+r)}\right) = 0.
\end{aligned} \tag{19}$$

The matrix "$A^{(s-l,s,s+r)}$" assumed to be symmetric and positive definite. Each "$c_i^{(s-l,s,s+r)}$" is a row in the constraint matrix "$C^{(s-l,s,s+r)}$." Let μ_k be the *kth* element of the sequence as $\mu \longrightarrow 0$, then the following optimal condition holds.

It is evident that as $x_{\mu k}^{*(s-l,s,s+r)} \longrightarrow x^{*(s-l,s,s+r)}$, then the value $P_{\mu}^{(s-l,s,s+r)}(x_{\mu k}^{*(s-l,s,s+r)}, s_{\mu}^{*(s-l,s,s+r)}, \lambda^{(s-l,s,s+r)}) \longrightarrow L^{(s-l,s,s+r)}(x^{*(s-l,s,s+r)}, s_{\mu}^{*(s-l,s,s+r)}, \lambda^{(s-l,s,s+r)})$.

Since the $x_{\mu k}^{*(s-l,s,s+r)}$ representing KKT points for $P_{\mu}^{(s-l,s,s+r)}(x^{(s-l,s,s+r)}, s_{\mu}^{*(s-l,s,s+r)}, \lambda^{(s-l,s,s+r)}), \lambda^{(s-l,s,s+r)}$ can be refined in equation (20).

$$\lambda_{k+1}^{(s-l,s,s+r)} = \lambda_k^{(s-l,s,s+r)} - \frac{\left(C^{(s-l,s,s+r)} x^{(s-l,s,s+r)} + d^{(s-l,s,s+r)} - s_{\mu}^{*(s-l,s,s+r)}\right)}{u}. \tag{20}$$

Thus, the Langrange's multipliers $\lambda^{(s-l,s,s+r)} \longrightarrow \lambda^{*(s-l,s,s+r)}$. We know that $\lambda_i^{(s-l,s,s+r)} \geq 0, \forall i \in J$. Thus, equation (20) can be updated as equation (21).

$$\lambda_{ik+1}^{(s-l,s,s+r)} = \begin{cases} 0 & \lambda_{ik}^{(s-l,s,s+r)} < \dfrac{\left(c_i^{(s-l,s,s+r)} x_{\mu}^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} - s_{i,\mu}^{*(s-l,s,s+r)}\right)}{u} \quad \forall \quad i \in J \\ \lambda_{ik}^{(s-l,s,s+r)} - \dfrac{\left(c_i^{(s-l,s,s+r)} x_{\mu}^{(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} - s_{i,\mu}^{*(s-l,s,s+r)}\right)}{u} & \text{otherwise} \end{cases}. \tag{21}$$

where $\lambda_{\mu}^{*(s-l,s,s+r)}$, the limit of the sequence of $\lambda_k^{*(s-l,s,s+r)} = (\lambda_{0k}^{(s-l,s,s+r)}, \lambda_{1k}^{(s-l,s,s+r)}, \lambda_{2k}^{(s-l,s,s+r)}, \ldots \lambda_{mk}^{(s-l,s,s+r)})$ for a given μ. Thus, we have equation (22).

$$
\begin{aligned}
&P_{\mu}^{(s-l,s,s+r)}\left(x_{\mu}^{*(s-l,s,s+r)}, s_{\mu}^{*(s-l,s,s+r)}, \lambda_{\mu}^{*(s-l,s,s+r)}\right) \longrightarrow L^{(s-l,s,s+r)}\left(x^{*(s-l,s,s+r)}, s^{*(s-l,s,s+r)}, \lambda^{*(s-l,s,s+r)}\right)\\
&L^{(s-l,s,s+r)}\left(x^{*(s-l,s,s+r)}, s^{*(s-l,s,s+r)}, \lambda^{*(s-l,s,s+r)}\right) =\\
&f^{*(s-l,s,s+r)}\left(x^{(s-l,s,s+r)}\right) - \sum_{i\in I,J} \lambda^{*(s-l,s,s+r)}\left(c_i^{(s-l,s,s+r)} x^{*(s-l,s,s+r)} + d_i^{(s-l,s,s+r)} - s^{*(s-l,s,s+r)}\right)\\
&s^{*(s-l,s,s+r)} \geq 0, \lambda^{*(s-l,s,s+r)} \geq 0, \quad \forall\, i \in J, \text{ as } \mu \longrightarrow 0.
\end{aligned}
\tag{22}
$$

The model equation (22) represents the required optimal model of the $P_{\mu}^{(s-l,s,s+r)}$ of equation (14).

4. Proposed Fuzzy Quadratic Programming Problem of Portfolio Optimization With Fuzzy Neural Network

In this section, the proposed technique is applied to six leading stocks in the Pakistan Stock Exchange. In this study, we shall concentrate on price-based forecasts for clarity's concern. The companies considered include Fauji Fertilizer Company (FFC), Unilever Pakistan Foods Limited (UPFL), Lucky Cement Limited (LUCK), Al-Ghazi Tractors Limited (AGTL), Archroma Pakistan Limited (ARPL), and IGI Holdings Limited (IGIHL) for stock returns. The monthly time series for the "Total Return" for each asset between January 2016 and October 2020 were retrieved from [53].

Let K_{il} stands for the "total returns" for assets $i = 1, 2, 3, \ldots, 6$ and $l = 1, 2, 3, \ldots, L$ where $i = 0$ relate to January 2016 and $l = L$ to October 2020, respectively. The source data K_{il}, $l = 0, 1, 2, \ldots, L$ are converted into rates of return r_{il} for assets using (23).

$$
r_{\text{il}} = \frac{k_{i,l} - k_{i,l-1}}{k_{i,l-1}}, \quad i = 1, 2, 3, \ldots, 6. \tag{23}
$$

Table 2 calculates from equation (23), the rate of returns, from January 2016 to October 2020.

We compute the arithmetic mean equations (24) and (25) on return for each asset. Furthermore, volatility in equation (26) is determined from covariance values in equation (25).

$$
a_i = \frac{1}{L}\sum_{l=1}^{L} r_{\text{il}}. \tag{24}
$$

Table 3 shows the covariance table using equation (25).

$$
\text{cov}\left(R_i, R_j\right) = \frac{1}{L}\sum_{l=1}^{L}\left(r_{\text{il}} - a_i\right)\left(r_{\text{jl}} - a_j\right). \tag{25}
$$

To calculate the volatility of each asset's rate of return, we utilize equation (26).

$$
\eta_i = \sqrt{\text{cov}\left(R_i, R_j\right)}. \tag{26}
$$

Finally, Table 4 displays the correlation matrix.

Finally, the minimum mean-variance quadratic programming problem is formulated in equation (27).

$$
\begin{aligned}
&\text{Min } 0.004462x_{\text{FCC}}^2 + 0.00403108x_{\text{FCC}}x_{\text{UPFL}} + 0.003149x_{\text{FCC}}x_{\text{LUCK}} + 0.005642x_{\text{FCC}}x_{\text{AGTL}} +\\
&0.00274x_{\text{FCC}}x_{\text{ARPL}} + 0.010724x_{\text{FCC}}x_{\text{IGIHL}} + 0.00916262x_{\text{UPFL}}^2 + 0.005692x_{\text{UPFL}}x_{\text{LUCK}} +\\
&0.007034x_{\text{UPFL}}x_{\text{AGTL}} + 0.00399x_{\text{UPFL}}x_{\text{ARPL}} + 0.01139x_{\text{UPFL}}x_{\text{IGIHL}} + 0.010414x_{\text{LUCK}}^2 +\\
&0.01015x_{\text{LUCK}}x_{\text{AGTL}} + 0.003922x_{\text{LUCK}}x_{\text{ARPL}} + 0.017868x_{\text{LUCK}}x_{\text{IGIHL}} + 0.017993x_{\text{AGTL}}^2 +\\
&0.01064x_{\text{AGTL}}x_{\text{ARPL}} + 0.018486x_{\text{AGTL}}x_{\text{IGIHL}} + 0.006055x_{\text{ARPL}}^2 + 0.006396x_{\text{ARPL}}x_{\text{IGIHL}} +\\
&0.023295x_{\text{IGIHL}}^2,\\
&\text{s.t.}\\
&0.999108x_{\text{FCC}} + 1.013255x_{\text{UPFL}} + 1.005754x_{\text{LUCK}} + 0.99858204x_{\text{AGTL}} + 1.003295x_{\text{ARPL}} + 0.997214x_{\text{IGIHL}} \geq 0.25,\\
&x_{\text{FFC}} + x_{\text{UPFL}} + x_{\text{LUCK}} + x_{\text{AGTL}} + x_{\text{ARPL}} + x_{\text{IGIHL}} = 1.
\end{aligned}
\tag{27}
$$

Table 2: Rate of returns of the stocks from January 2016 to October 2020.

Date	FFC	UPFL	LUCK	AGTL	ARPL	IGIHL
Feb 2016	−0.017	−0.086	0.009	−0.005	−0.065	−0.011
Mar 2016	−0.023	−0.007	0.092	0.214	0.013	−0.037
—	—	—	—	—	—	—
Sep 2020	−0.0001	0.428	0.053	−0.103	−0.038	−0.022
Oct 2020	−0.021	−0.14	0.040	−0.020	−0.029	−0.094
Arithmetic mean a_i	0.001	0.017	0.010	0.006	0.006	0.007
Volatility	0.066	0.095	0.102	0.134	0.077	0.152

Table 3: Covariance of the stocks.

Covariance	FFC	UPFL	LUCK	AGTL	ARPL	IGIHL
FFC	0.004	0.002	0.003	0.002	0.001	0.005
UPFL	0.002	0.009	0.002	0.003	0.001	0.005
LUCK	0.003	0.002	0.010	0.005	0.001	0.008
AGTL	0.002	0.003	0.005	0.017	0.005	0.009
ARPL	0.001	0.001	0.001	0.005	0.006	0.003
IGIHL	0.005	0.005	0.008	0.009	0.003	0.023

Table 4: Correlation matrix of the six stocks.

Correlation	FFC	UPFL	LUCK	AGTL	ARPL	IGIHL
FFC	1	0.315	0.461	0.314	0.263	0.525
UPFL	0.315	1	0.291	0.273	0.267	0.389
LUCK	0.461	0.291	1	0.370	0.246	0.573
AGTL	0.314	0.273	0.370	1	0.509	0.451
ARPL	0.263	0.267	0.246	0.509	1	0.269
IGIHL	0.525	0.389	0.573	0.451	0.269	1

The model equation (27) is solved using *Python* machine learning software. To train the optimization problem, the RMSProp, Momentum, Adadelta, Adagrad, Adam, and gradient descent optimizers are utilized. Figure 1 depicts the path taken by various optimizers to reach the minimum.

The lower quadratic programming MVO for the portfolio optimization is equation (28).

$$
\begin{aligned}
&\text{Min } 0.004362x_{\text{FCC}}^2 + 0.00393108x_{\text{FCC}}x_{\text{UPFL}} + 0.003049x_{\text{FCC}}x_{\text{LUCK}} + 0.005542x_{\text{FCC}}x_{\text{AGTL}} + \\
&0.00264x_{FCC}x_{\text{ARPL}} + 0.010624x_{\text{FCC}}x_{\text{IGIHL}} + 0.00906262x_{\text{UPFL}}^2 + 0.005592x_{\text{UPFL}}x_{\text{LUCK}} + \\
&0.006934x_{\text{UPFL}}x_{\text{AGTL}} + 0.00389x_{\text{UPFL}}x_{\text{ARPL}} + 0.01129x_{\text{UPFL}}x_{\text{IGIHL}} + 0.010314x_{\text{LUCK}}^2 + \\
&0.01005x_{\text{LUCK}}x_{\text{AGTL}} + 0.003822x_{\text{LUCK}}x_{\text{ARPL}} + 0.017768x_{\text{LUCK}}x_{\text{IGIHL}} + 0.017893x_{\text{AGTL}}^2 + \\
&0.01054x_{\text{AGTL}}x_{\text{ARPL}} + 0.018386x_{\text{AGTL}}x_{\text{IGIHL}} + 0.005955x_{\text{ARPL}}^2 + 0.006296x_{\text{ARPL}}x_{\text{IGIHL}} + \\
&0.023195, \\
&\text{s.t.} \\
&0.999108x_{\text{FCC}} + 1.013255x_{\text{UPFL}} + 1.005754x_{\text{LUCK}} + 0.99858204x_{\text{AGTL}} + 1.003295x_{\text{ARPL}} + 0.997214x_{\text{IGIHL}} \geq 0.10, \\
&x_{\text{FFC}} + x_{\text{UPFL}} + x_{\text{LUCK}} + x_{\text{AGTL}} + x_{\text{ARPL}} + x_{\text{IGIHL}} = 1.
\end{aligned}
\tag{28}
$$

Figure 2 depicts the path taken by various optimizers to reach the minimum.

The upper quadratic programming mean-variance model is equation (29).

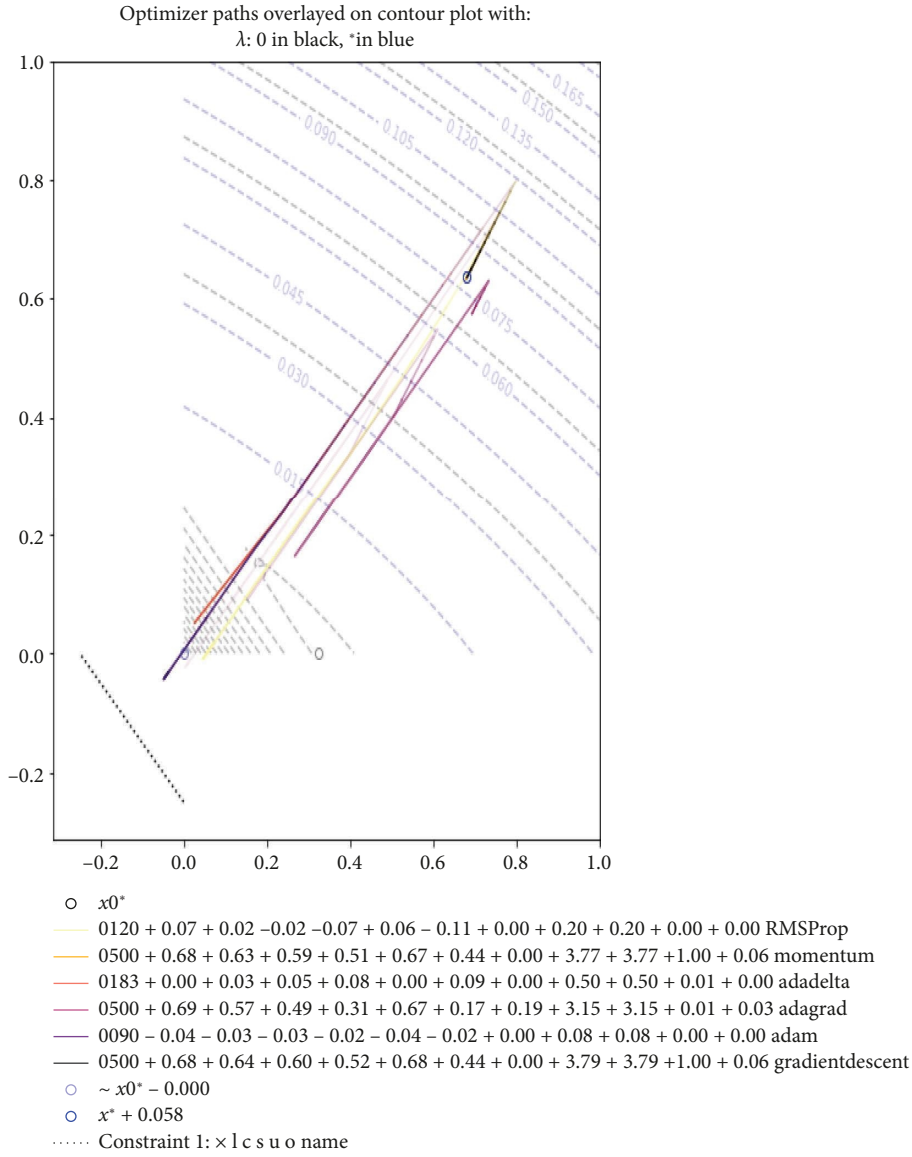

Figure 1: Optimizers at initial RHS = 0.25, lohi = [0, 1] for central solutions.

$$
\begin{aligned}
\text{Min}\quad & 0.004562x_{\text{FCC}}^2 + 0.00413108x_{\text{FCC}}x_{\text{UPFL}} + 0.003249x_{\text{FCC}}x_{\text{LUCK}} + 0.005742x_{\text{FCC}}x_{\text{AGTL}} + \\
& 0.00284x_{\text{FCC}}x_{\text{ARPL}} + 0.010824x_{\text{FCC}}x_{\text{IGIHL}} + 0.00926262x_{\text{UPFL}}^2 + 0.005792x_{\text{UPFL}}x_{\text{LUCK}} + \\
& 0.007134x_{\text{UPFL}}x_{\text{AGTL}} + 0.00409x_{\text{UPFL}}x_{\text{ARPL}} + 0.01149x_{\text{UPFL}}x_{\text{IGIHL}} + 0.010424x_{\text{LUCK}}^2 + \\
& 0.01025x_{\text{LUCK}}x_{\text{AGTL}} + 0.004022x_{\text{LUCK}}x_{\text{ARPL}} + 0.017878x_{\text{LUCK}}x_{\text{IGIHL}} + 0.018093x_{\text{AGTL}}^2 + \\
& 0.01074x_{\text{AGTL}}x_{\text{ARPL}} + 0.018586x_{\text{AGTL}}x_{\text{IGIHL}} + 0.006155x_{\text{ARPL}}^2 + 0.006496x_{\text{ARPL}}x_{\text{IGIHL}} + \\
& 0.023395x_{\text{IGIHL}}^2, \\
\text{s.t.}\quad & \\
& 0.999108x_{\text{FCC}} + 1.013255x_{\text{UPFL}} + 1.005754x_{\text{LUCK}} + 0.99858204x_{\text{AGTL}} + 1.003295x_{\text{ARPL}} + 0.997214x_{\text{IGIHL}} \geq 0.30, \\
& x_{\text{FFC}} + x_{\text{UPFL}} + x_{\text{LUCK}} + x_{\text{AGTL}} + x_{\text{ARPL}} + x_{\text{IGIHL}} = 1.
\end{aligned}
\tag{29}
$$

When equation (29) is solved using Python, the visualization of the results is given in Figure 3.

In this study, unique parameter settings were crucial to achieving suitable performance. The fuzzy neural network

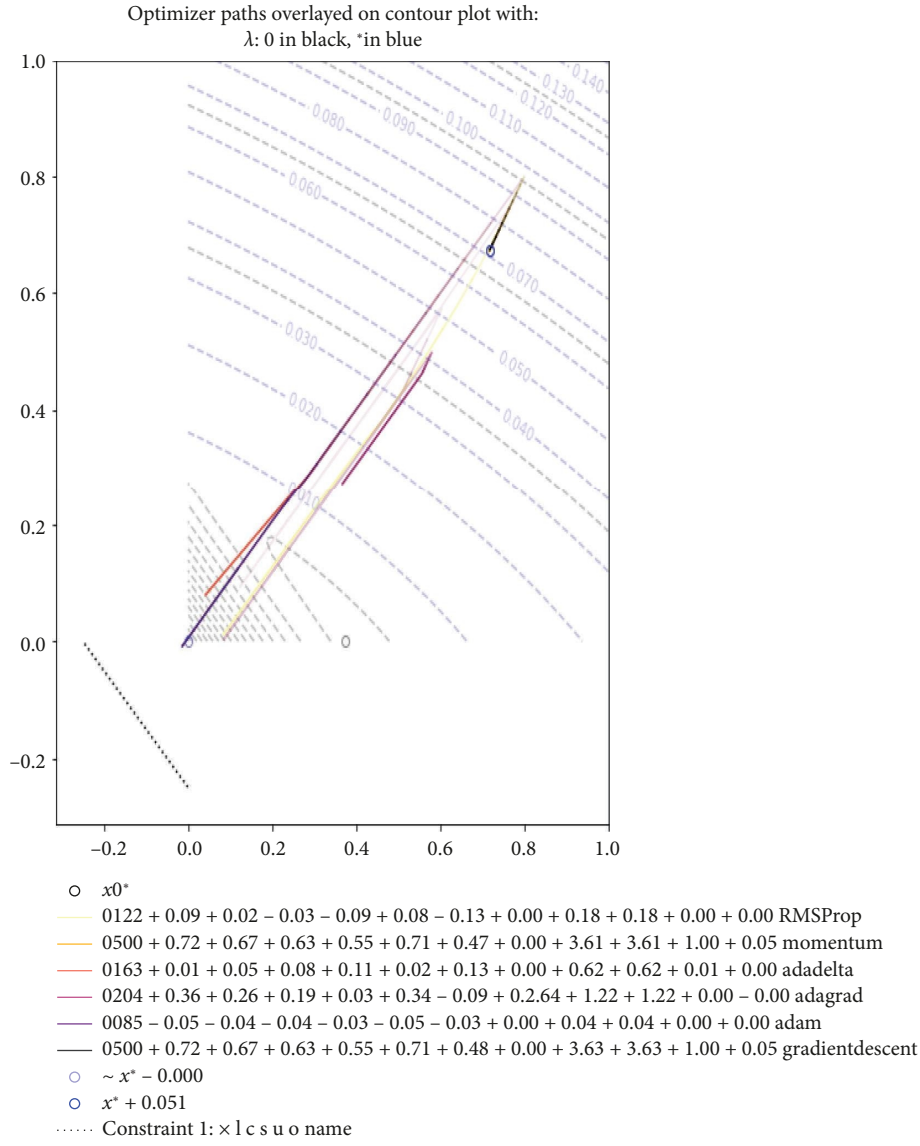

Figure 2: Optimizers at initial RHS = 0.25, lohi = [0, 1] for lower solutions.

was configured with a momentum of 0.9, a learning rate of 0.01, and a maximum of a thousand epochs for training. The research was conducted using an Intel Core i7 processor having 16 GB of RAM, NVIDIA GTX 1080 GPU. The high processing and computational power ensure efficient evaluation of the computational load. The software package utilized Python 3.8 with Tensor Flow 2.4 for neural network implementation. These particular parameters were utilized to leverage the excessive computational strength and superior machine learning capabilities. Thus, these specific hardware and software platforms were used enabling the accurate and efficient execution of complex fuzzy neural network models.

5. Results and Discussions

The profitable portfolios identified by the proposed methods are presented in Table 5 and Figures 4, 5, 6. Table 5 and Figure 4 show the investment allocations based on the lower programming model.

According to the optimizers, the investors should invest in two categories. The first one suggests investing in the portfolios of FFC, ARPL, and UPFL. The second one suggests LUCKY AGTL and IGIHL. The first category tends to increase return variability and risk. As a result, it is considered a high-risk category. The second category seeks to reduce both return variability and risk. It is a risk-averse

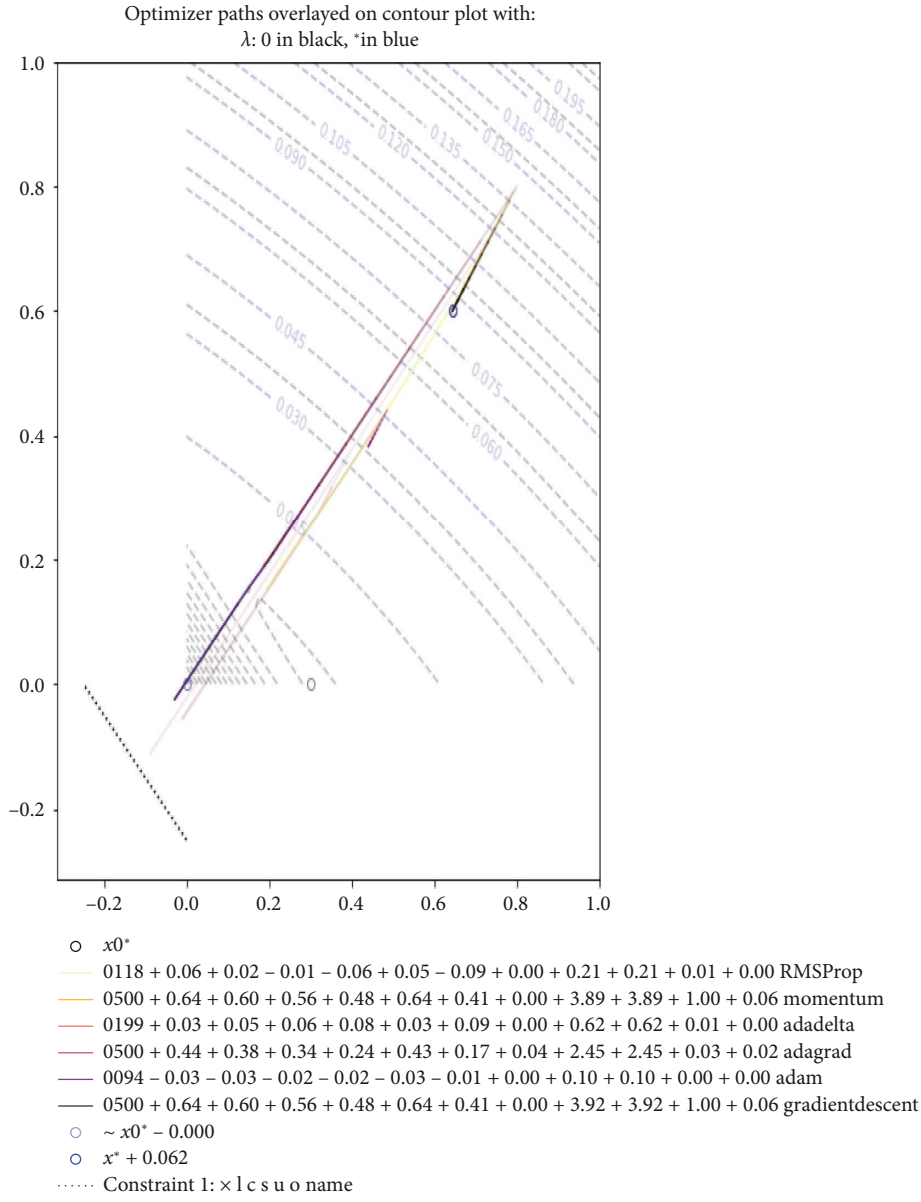

Figure 3: Optimizers at initial RHS = 0.25, lohi = [0, 1] for upper solutions.

Table 5: Investment allocations based on the (lower, central, and upper) programming model.

Name	At lohi [0, 1.1]	At lohi [0, 1], RHS = 0.25	At lohi [0, 2], RHS = 0.25
Investment allocations with RMSprop optimizer			
FFC	(0.09, 0.07, 0.06)	(0.09, 0.07, 0.06)	(0.09, 0.07, 0.06)
UPFL	(0.02, 0.02, 0.02)	(0.02, 0.02, 0.02)	(0.02, 0.02, 0.02)
LUCKY	(−0.03, −0.02, −0.01)	(−0.03, −0.02, −0.01)	(−0.03, −0.02, −0.01)
AGTL	(−0.09, −0.07, −0.06)	(−0.09, −0.07, −0.06)	(−0.09, −0.07, −0.06)
ARPL	(0.08, 0.06, 0.05)	(0.08, 0.06, 0.05)	(0.08, 0.06, 0.05)
IGIHL	(−0.13, −0.11, −0.09)	(−0.13, −0.11, −0.09)	(−0.13, −0.11, −0.09)
For investment		FFC, ARPL, UPFL	
Investment allocations with momentum optimizer			
FFC	(0.72, 0.68, 0.64)	(0.72, 0.68, 0.64)	(0.72, 0.68, 0.64)
UPFL	(0.67, 0.63, 0.6)	(0.67, 0.63, 0.6)	(0.67, 0.63, 0.6)
LUCKY	(0.63, 0.59, 0.56)	(0.63, 0.59, 0.56)	(0.63, 0.59, 0.56)

Table 5: Continued.

Name	At lohi [0, 1.1]	At lohi [0, 1], RHS = 0.25	At lohi [0, 2], RHS = 0.25
AGTL	(0.55.0.51, 0.48)	(0.55, 0.51, 0.48)	(0.55, 0.51, 0.48)
ARPL	(0.71, 0.67, 0.64)	(0.71, 0.67, 0.64)	(0.71, 0.67, 0.64)
IGIHL	(0.47, 0.44, 0.41)	(0.47, 0.44, 0.41)	(0.47, 0.44, 0.41)
Best for investment		FFC, ARPL, UPFL, LUCKY	
Investment allocations with Adadelta optimizer			
FFC	(0.01, 0, 0.03)	(0.01, 0, 0.03)	(0.01, 0, 0.03)
UPFL	(0.05, 0.03, 0.05)	(0.05, 0.03, 0.05)	(0.05, 0.03, 0.05)
LUCKY	(0.08, 0.05, 0.06)	(0.08, 0.05, 0.06)	(0.08, 0.05, 0.06)
AGTL	(0.11, 0.08, 0.08)	(0.11, 0.08, 0.08)	(0.11, 0.08, 0.08)
ARPL	(0.02, 0, 0.03)	(0.02, 0, 0.03)	(0.02, 0, 0.03)
IGIHL	(0.13, 0.09, 0.09)	(0.13, 0.09, 0.09)	(0.13, 0.09, 0.09)
Best for investment		IGIHL, AGTL, LUCKY	
Investment allocations with Adagrad optimizer			
FFC	(0.36, 0.69, 0.44)	(0.36, 0.69, 0.44)	(0.36, 0.69, 0.44)
UPFL	(0.26, 0.57, 0.38)	(0.26, 0.57, 0.38)	(0.26, 0.57, 0.38)
LUCKY	(0.19, 0.49, 0.34)	(0.19, 0.49, 0.34)	(0.19, 0.49, 0.34)
AGTL	(0.03, 0.31, 0.24)	(0.03, 0.31, 0.24)	(0.03, 0.31, 0.24)
ARPL	(0.34, 0.67, 0.43)	(0.34, 0.67, 0.43)	(0.34, 0.67, 0.43)
IGIHL	(−0.09, 0.17, 0.17)	(−0.09, 0.17, 0.17)	(−0.09, 0.17, 0.17)
Best for investment		FFC, ARPL, UPFL, LUCKY	
Investment allocations with adam optimizer			
FFC	(−0.05, −0.04, −0.03)	(−0.05, −0.04, −0.03)	(−0.05, −0.04, −0.03)
UPFL	(−0.04, −0.03, −0.03)	(−0.04, −0.03, −0.03)	(−0.04, −0.03, −0.03)
LUCKY	(−0.04, −0.03, −0.02)	(−0.04, −0.03, −0.02)	(−0.04, −0.03, −0.02)
AGTL	(−0.03, −0.02, −0.02)	(−0.03, −0.02, −0.02)	(−0.03, −0.02, −0.02)
ARPL	(−0.05, −0.04, −0.03)	(−0.05, −0.04, −0.03)	(−0.05, −0.04, −0.03)
IGIHL	(−0.03, −0.02, −0.01)	(−0.03, −0.02, −0.01)	(−0.03, −0.02, −0.01)
Best for investment		No identification.	
Investment allocations with gradient decent optimizer			
FFC	(0.72, 0.68, 0.64)	(0.72, 0.68, 0.64)	(0.72, 0.68, 0.64)
UPFL	(0.67, 0.64, 0.6)	(0.67, 0.64, 0.6)	(0.67, 0.64, 0.6)
LUCKY	(0.63, 0.6, 0.56)	(0.63, 0.6, 0.56)	(0.63, 0.6, 0.56)
AGTL	(0.55, 0.52, 0.48)	(0.55, 0.52, 0.48)	(0.55, 0.52, 0.48)
ARPL	(0.71, 0.68, 0.64)	(0.71, 0.68, 0.64)	(0.71, 0.68, 0.64)
IGIHL	(0.48, 0.44, 0.41)	(0.48, 0.44, 0.41)	(0.48, 0.44, 0.41)
Best for investment		FFC, ARPL, UPFL, LUCKY	

category. With the exception of the Adam and Adadelta optimizers, all optimizers recommend investing in FFC, ARPL, and UPFL. These two optimizers significantly minimize the unpredictability and risk of returns. The others increase the variability, risk, and returns.

Table 5 and Figure 5 show the investment allocations based on the central programming model.

Table 5 and Figure 6 depict the upper programming model's investment allocations.

Figures 4, 5, and 6 show the investment allocations based on the lower, central, and upper programming models, respectively. Analytically, this situation is presented in Table 5. On analysis and visualizations of the trends of the optimizers, it is deduced that the investors should invest in two categories: firstly, investing in the portfolios of FFC, ARPL, and UPFL and secondly, investing in LUCKY AGTL and IGIHL. The first category tends to increase return variability and risk. As a result, it is considered a high-risk category. The second category seeks to reduce both return variability and risk. It is a risk-averse category. With the exception of the Adam and Adadelta optimizers, all optimizers recommend investing in FFC, ARPL, and UPFL.

According to Table 6, Adam is the best optimizer followed by RMSprop. This optimizer avoids risk and unpredictability in returns. Adam optimizer outperforms all others. Furthermore, Adam, Adadelta, and RMSprop are sensitive to fuzzy data. Adagrad is somewhat responsive to fuzzy uncertainty, whereas Momentum and gradient descent are not.

Finally, Table 7 shows a comprehensive comparison of the proposed methods over [12, 43–46]. The table depicts that the proposed Adagrad and proposed Adadelta improve the objective by 0.59% and 0.18%, respectively. Thus, with respect to the optimizing efficiency, Adagrad is the best optimizer followed by Adadelta. Dual of Wu et al. [12] and New model three of Malek and Oskoei [43] bring improvement of 0.00004%. Mansoori, Effati, and Eshaghnezhad [44] bring an improvement of 0.10%, and Mansoori, Effati, and Eshaghnezhad [45] bring an improvement of 0.15% at specific uncertainty level. Coelho [46] recorded an

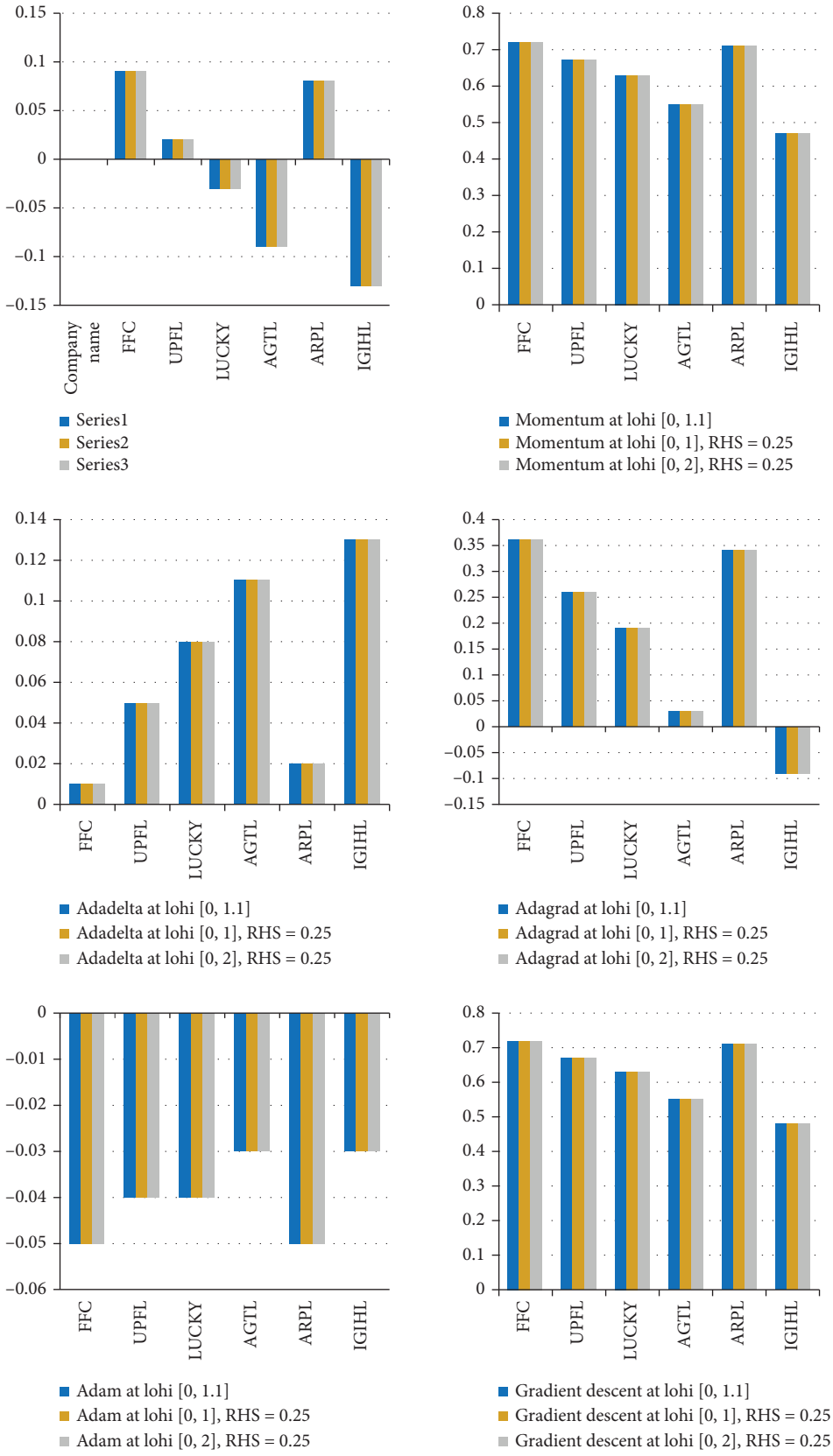

Figure 4: Investment allocations with different optimizers for the lower model.

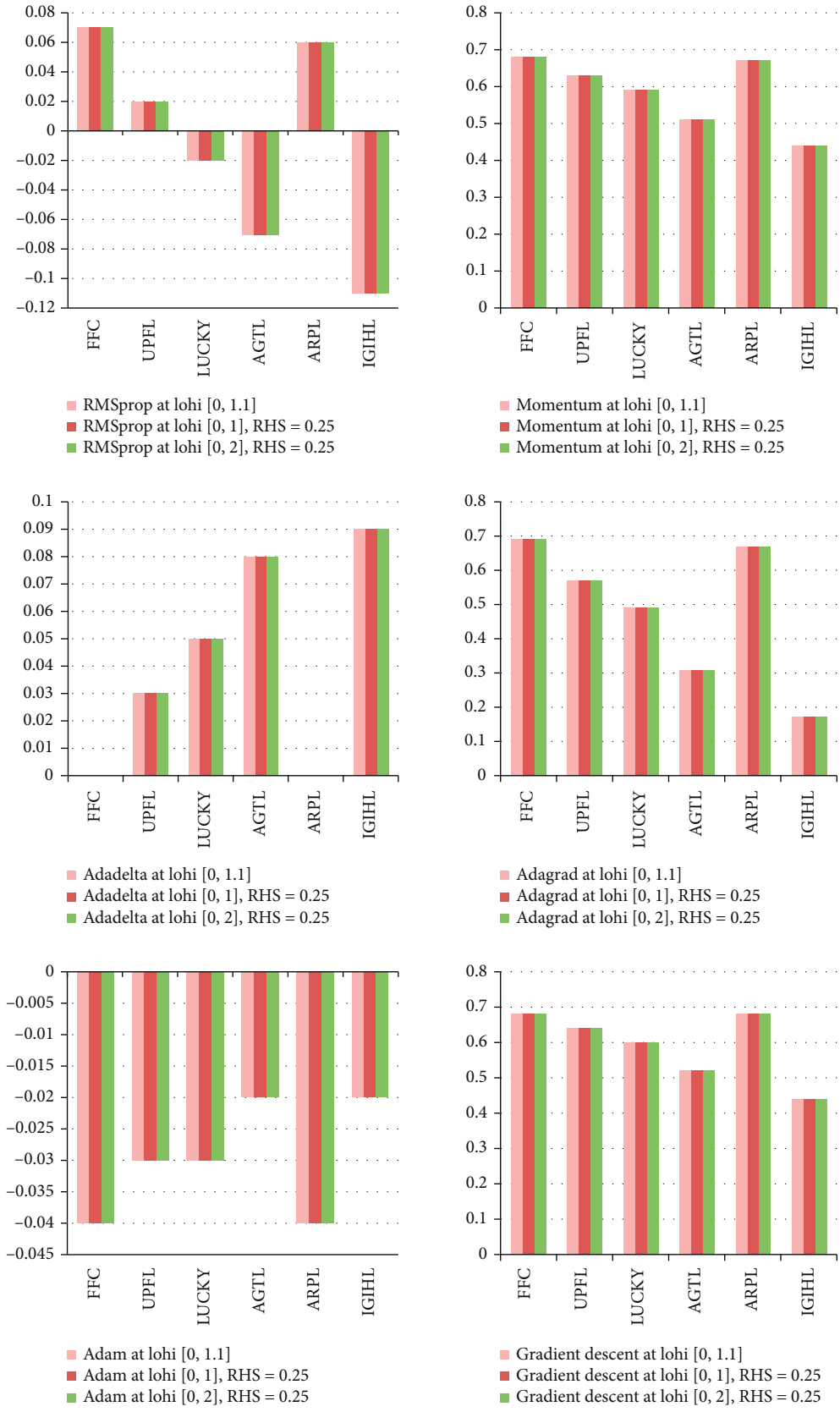

Figure 5: Investment allocations with different optimizers for the central model.

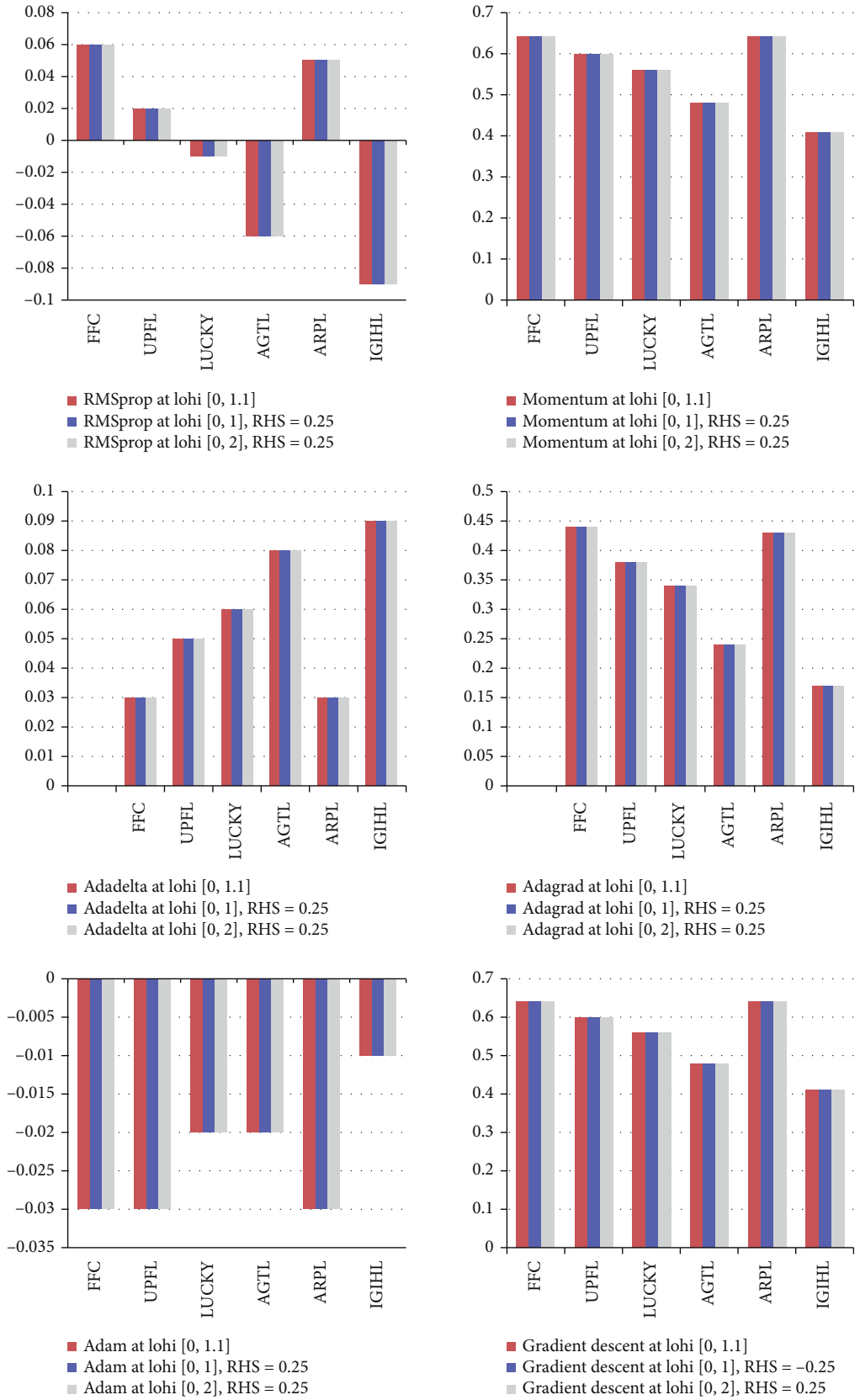

Figure 6: Investment allocations with different optimizers for the upper model.

Table 6: Combine the lower, central, and upper models' solutions.

Optimizer	Objective value at lohi = [0, 1.1]	Objective value at lohi = [0, 1], rhs constraint = 0.25	Objective value at lohi = [0, 2], rhs constraint = 0.25	Responsiveness to fuzzy data
RMSProp				
Triangular fuzzy	(122, 120, 118)	(122, 120, 118)	(122, 120, 118)	
$(s-l,\ s,\ s+r)$	(120, −2, −2)	(120, −2, −2)	(120, −2, −2)	**Responsive to fuzzy data.**
$(s,\ l,\ r)$				
Percent improvement	0.03	0.03	0.03	
Momentum				
Triangular fuzzy	(500, 500, 500)	(500, 500, 500)	(500, 500, 500)	
$(s-l,\ s,\ s+r)$	(500, 0, 0)	(500, 0, 0)	(500, 0, 0)	Irresponsive to fuzzy data
$(s,\ l,\ r)$				
Percent improvement	0	0	0	
Adadelta				
Triangular fuzzy				
$(s-l,\ s,\ s+r)$	(163, 183, 199)	(163, 183, 199)	(163, 183, 199)	**Responsive to fuzzy data.**
$(s,\ l,\ r)$	(183, 20, 16)	(183, 20, 16)	(183, 20, 16)	
Percent improvement	0.18	0.18	0.18	
Adagrad				
Triangular fuzzy				
$(s-l,\ s,\ s+r)$	(204, 500, 500)	(204, 500, 500)	(204, 500, 500)	**Responsive to fuzzy data**
$(s,\ l,\ r)$	(500, 296, 0)	(500, 296, 0)	(500, 296, 0)	
	0.59	0.59	0.59	
Adam				
Triangular fuzzy $(s-l,\ s,\ s+r)$				
$(s,\ l,\ r)$	(85, 90, 94)	(85, 90, 94)	(85, 90, 94)	**Responsive to fuzzy data**
Percent improvement	(90, 5, 4)	(90, 5, 4)	(90, 5, 4)	
	0.09	0.09	0.09	
Gradient decent				
Triangular fuzzy $(s-l,\ s,\ s+r)$	(500, 500, 500)	(500, 500, 500)	(500, 500, 500)	Irresponsive to fuzzy data
$(s,\ l,\ r)$	(500, 0, 0)	(500, 0, 0)	(500, 0, 0)	
Percent improvement	0	0	0	

improvement of 0.14%. Thus, from the table, it is clear that Adagrad is the best followed by Adadelta bringing improvements of 0.59% and 0.18%, respectively, when concerning the optimizing efficiency.

In this research, the penalty function approach is used to solve quadratic programming problems. The penalty function approach is implemented on the powerful computational technique of neural network in a fuzzy uncertain environment. The penalty function method ensures the constraints satisfaction problem in such a way that if the value of the function goes beyond the provided close range of the constraints, the function is penalized with a number so that to keep it in the provided range. In this way, the constraints satisfaction problem is deemed possible. Comparing the proposed results with the renowned studies depicts that the proposed study brings an improvement of 59% and 18% for the Adagrad and Adadelta optimizers, respectively [12, 43–46]. A 4% improvement was recorded by Wu et al. [12] and new model three by Malek and Oskoei [43]. When fuzziness was incorporated into the model of Mansoori, Effati, and Eshaghnezhad [44], the objective function was improved by 10%. Moreover, in Mansoori, Effati, and Eshaghnezhad [45], 15% improvement was reported in the objective function by the incorporation of fuzzy modeling techniques. Coelho [46] recorded an improvement of 14%. From this analysis, it is clear that the Adagrad and Adadelta optimizers proposed in this study recorded an improvement of 59% and 18%, respectively. In this regard, it is clear that the proposed methods are better than the methods previously in the literature.

Table 7: Comparison of the proposed methods with literature.

	Starting value/final value	Difference	Percent difference
Proposed RMSprop	122 118	3	0.03
Proposed Adadelta	199 163	36	0.18
Proposed Adagrad	500 204	296	0.59
Proposed Adam	94 85	9	0.09
Mansoori, Effati, and Eshaghnezhad [45] at uncertainty level $\alpha = 0.2, w_1 = 1/4, w_2 = 3/4$ Mansoori, Effati, and Eshaghnezhad [45] at uncertainty level $\alpha = 0.5, w_1 = 1/4, w_2 = 3/4$	−1.5559 −1.8000	−0.2441	0.156
Primal of the Wu et al. [12] model 5, and Malek, and Oskoei [43] new 3 model, Table 5 Dual of the Wu et al. [12] model 5, and Malek, and Oskoei [43] new 3 model, Table 5	−224.99 −225.02	−0.01	0.00004
Mansoori, Effati, and Eshaghnezhad [44] at uncertainty level $\alpha = 0.0, w_1 = 1/2, w_2 = 1/2$, Table 1 Mansoori, Effati, and Eshaghnezhad [44] at uncertainty level $\alpha = 1, w_1 = 1/2, w_2 = 1/2$, Table 1	−18.97 −21.00	−2.03	0.10
Coelho [46] at uncertainty level $\alpha = 0.0$ Coelho [46] at uncertainty level $\alpha = 1$	439.57 504.11	64.54	0.1468

6. Conclusions

In this study, fuzzy neural networks with penalty functions are utilized to formulate the quadratic programming based on mean-variance Markowitz portfolio model. Fuzzy artificial neural networks are used in this article to address fuzzy quadratic programming problems. The problem's lower, central, and upper models are first developed. The fuzzy neural networks are then used to solve the models. The proposed method is then applied in the capital market to identify optimal portfolios to potential investors in the Pakistan Stock Exchange. From January 2016 to October 2020, six stocks traded on the stock exchange were analyzed. Lower, central, and upper quadratic programming models were solved using six distinct optimizers, namely, RMSprop, Momentum, Adadelta, Adagrad, Adam, and gradient descent. The optimizers agree on identifying the ideal investment portfolios for the investors at all three levels (lower, central, and upper). The optimizers divide the investors into two groups and assign them to one of the two groups. The first group recommends investing in the portfolios FFC, ARPL, and UPFL, while the second group recommends investing in LUCKY, AGTL, and IGIHL. Based on the analysis, it is evident that all of the optimizers recommend investing in FFC, ARPL, and UPFL, with the exception of the Adam and Adadelta optimizers, recommending investment in IGIHL, AGTL, and LUCKY. These two optimizers lower return variability and risk. The others raise the level of variability, risk, and return. The proposed Adagrad and proposed Adadelta improve the objective by 0.59% and 0.18%, respectively. The implementation has promising results to address uncertainties in financial investment problems. Future work can be extended to explore its application to other areas of economics and finance. Likewise, such implementations will be in risk management and optimal derivative pricing. Moreover, the development of sophisticated fuzzy logic methodologies can be deemed possible. Additionally, the incorporation of deep learning techniques could further improve the applicability and performance of these models [54]. Furthermore, cloud computing and parallel processing resources may offer potential improvements in the scalability and processing speed [55].

Data Availability Statement

Data can be provided on reasonable request. For further information, we explore the data portal, Pakistan Stock Exchange. Data portal-Pakistan Stock Exchange (PSX).

Conflicts of Interest

The authors declare no conflicts of interest.

Author Contributions

Izaz Ullah Khan conceptualized the study, developed methodology, contributed software, programming, and project management, and write-up, and validated and supervised the study. Muhammad Aamir contributed methodology, write-up, validation, visualization, formal Analysis, and investigation. Mehran Ullah contributed investigation, resources, data curation, formal analysis, review, and project management. Muhammad Shahbaz Shah contributed graphics and visualization, programming, data curation, and validation.

Funding

This study was funded by no agency/grant.

Acknowledgments

Appreciation is extended to the institutions, computational intelligence laboratories, and facilities that made this work possible.

References

[1] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review* 65, no. 6 (1958): 386–408.

[2] Y. LeCun, B. Boser, J. S. Denker, et al., "Back Propagation Applied to Handwritten Zip Code Recognition," *Neural Computation* 1, no. 4 (1989): 541–551.

[3] H. B. Mann, "Quadratic Forms With Linear Constraints," *The American Mathematical Monthly* 50, no. 7 (1943): 430–433.

[4] H. M. Markowitz, "The Optimization of a Quadratic Function Subject to Linear Constraints," *Naval Research Logistics Quarterly* 3, no. 1-2 (1956): 111–133.

[5] B. C. Eaves, "On Quadratic Programming," *Management Science* 17, no. 11 (1971): 698–711.

[6] M. Frank and P. Wolfe, "An Algorithm for Quadratic Programming," *Naval Research Logistics Quarterly* 3, no. 1-2 (1956): 95–110.

[7] P. Wolfe, "The Simplex Method for Quadratic Programming," *Econometrica* 27, no. 3 (1959): 382–398.

[8] C. Y. Maa and M. A. Shanblatt, "Linear and Quadratic Programming Neural Network Analysis," *IEEE Transactions on Neural Networks* 3, no. 4 (1992): 580–594.

[9] Y. Xia, "A New Neural Network for Solving Linear and Quadratic Programming Problems," *IEEE Transactions on Neural Networks* 7, no. 6 (1996): 1544–1548.

[10] A. Bouzerdoum and T. R. Pattison, "Neural Network for Quadratic Optimization With Bound Constraints," *IEEE Transactions on Neural Networks* 4, no. 2 (1993): 293–304.

[11] S. H. Zak, V. Upatising, and S. Hui, "Solving Linear Programming Problems With Neural Networks: A Comparative Study," *IEEE Transactions on Neural Networks* 6, no. 1 (1995): 94–104.

[12] X. Y. Wu, Y. S. Xia, J. Li, and W. K. Chen, "A High-Performance Neural Network for Solving Linear and Quadratic Programming Problems," *IEEE Transactions on Neural Networks* 7, no. 3 (1997): 643–651.

[13] Y. Xia and J. Wang, "Primal Neural Networks for Solving Convex Quadratic Programs," in *Proceedings of the International Joint Conference on Neural Networks, (Cat. No. 99CH36339)* (Washington, DC, USA: IEEE, July 1999), 582–587.

[14] H. M. Markowitz, "Foundations of Portfolio Theory," *The Journal of Finance* 46, no. 2 (1991): 469–477.

[15] R. O. Michaud, "The Markowitz Optimization Enigma: Is 'optimized' Optimal? Financial," *Analyst Journal* 5, no. 1 (1989): 31–42.

[16] J. C. Richard and T. Roncalli, "Constrained Risk Budgeting Portfolios: Theory, Algorithms, Applications and Puzzles," *SSRN Constrained Risk Budgeting Portfolios: Theory, Algorithms, Applications & Puzzles by Jean-Charles Richard, Thierry Roncalli* (Rochester, NY, USA: SSRN, 2019).

[17] J. Brodie, I. Daubechies, C. De. Mol, D. Giannone, and I. Loris, "Sparse and Stable Markowitz Portfolios," *Proceedings of the National Academy of Sciences of the United States of America* 106, no. 30 (2009): 12267–12272.

[18] R. W. Cottle and G. Infanger, "Harry Markowitz and the Early History of Quadratic Programming," in *Handbook of Portfolio Construction*, ed. J. B. Guerard (Boston, MA: Springer, 2010), 179–211.

[19] B. Carmichael, G. B. Koumou, and K. Moran, "Rao's Quadratic Entropy and Maximum Diversification Indexation," *Quantitative Finance* 18, no. 6 (2018): 1017–1031.

[20] E. Lezmi, H. Malongo, T. Roncalli, and R. Sobotka, "Portfolio Allocation with Skewness Risk: A Practical Guide," *SSRN Portfolio Allocation with Skewness Risk: A Practical Guide by Edmond Lezmi, Hassan Malongo, Thierry Roncalli, R Sobotka* (Rochester, NY: SSRN, 2018).

[21] J. Gonzalvez, E. Lezmi, T. Roncalli, and J. Xu, "Financial Applications of Gaussian Processes and Bayesian Optimization," *SSRN Financial Applications of Gaussian Processes and Bayesian Optimization by Joan Gonzalvez, Edmond Lezmi, Thierry Roncalli, Jiali Xu* (Rochester, NY: SSRN, 2019).

[22] L. A. Zadeh, "Fuzzy Sets," *Information and Control* 8, no. 3 (1965): 338–353.

[23] I. U. Khan, T. Ahmad, and N. Maan, "An Inverse Feedback Fuzzy State Space Modeling (FFSSM) for Insulin-Glucose Regulatory System in Humans," *Scientific Research and Essays* 8, no. 25 (2013): 1570–1583, https://doi.org/10.5897/SRE12.066.

[24] I. U. Khan, T. Ahmad, and N. Maan, "A Simplified Novel Technique for Solving Fully Fuzzy Linear Programming Problems," *Journal of Optimization Theory and Applications* 159 (2013): 536–546, https://doi.org/10.1007/s10957-012-0215-2.

[25] I. U. Khan, T. Ahmad, and N. Maan, "Revised Convexity, Normality and Stability Properties of the Dynamical Feedback Fuzzy State Space Model (FFSSM) of Insulin-Glucose Regulatory System in Humans," *Soft Computing* 23, no. 21 (2019): 11247–11262.

[26] I. U. Khan and F. W. Karam, "Intelligent Business Analytics Using Proposed Input/Output Oriented Data Envelopment Analysis DEA and Slack Based DEA Models for US-Airlines," *Journal of Intelligent and Fuzzy Systems* 37, no. 6 (2019): 8207–8217, https://doi.org/10.3233/JIFS-190641.

[27] I. U. Khan and F. Rafique, "Minimum-Cost Capacitated Fuzzy Network, Fuzzy Linear Programming Formulation, and Perspective Data Analytics to Minimize the Operations Cost of American Airlines," *Soft Computing* 25, no. 2 (2021): 1411–1429.

[28] I. U. Khan and M. Aftab, "Dynamic Programming Approach for Fuzzy Linear Programming Problems FLPs and Its Application to Optimal Resource Allocation Problems in Education System," *Journal of Intelligent and Fuzzy Systems* 42, no. 4 (2022): 3517–3535.

[29] I. U. Khan and M. Aftab, "Adaptive Fuzzy Dynamic Programming (AFDP) Technique for Linear Programming Problems Lps With Fuzzy Constraints," *Soft Computing* 27, no. 19 (2023): 13931–13949, https://doi.org/10.1007/s00500-023-08462-9.

[30] Faiza and K. Khalil, "Airline Flight Delays Using Artificial Intelligence in COVID-19 With Perspective Analytics," *Journal of Intelligent and Fuzzy Systems* 44, no. 4 (2023): 6631–6653, https://doi.org/10.3233/JIFS-222827.

[31] A. A. Molai, "The Quardratic Programming Problem With Fuzzy Relation Inequality Constraints," *Computers and Industrial Engineering* 62, no. 1 (2012): 256–263.

[32] S. K. Barik and M. P. Biswal, "Probabilistic Quadratic Programming Problems With Some Fuzzy Parameters," *Advances in Operations Research* 14 (2012).

[33] X.-G. Zhou, B.-Y. Cao, and S. H. Nasseri, "Optimality Conditions for Fuzzy Number Quardratic Programming With Fuzzy Coefficients," *Journal of Applied Mathematics* 13 (2014).

[34] E. D. Bai and Yu.E. Bao, "Study on Fuzzy Quadratic Programming Problems," in *Proceedings of the International Conference on Applied Mechanics, Mathematics, Modelling and Simulations 2018* (Hong Kong, China, November 2018), 424–429.

[35] R. Ghanbari and K. G. Moghadam, "Solving Fuzzy Quadratic Programming Problems Based on ABS Algorithm," *Soft Computing* 23 (2019): 11343–11349.

[36] P. Umamaheswari and K. Ganesan, "A New Approach for the Solution of Fuzzy Quadratic Programming Problems," *Journal of Advanced Research in Dynamical and Control Systems* 11, no. 1 (2019): 342–349.

[37] M. M. Elshafei, "Fully Fuzzy Quadratic Programming With Unrestricted Fully Fuzzy Variables and Parameters," *Journal of progressive research in mathematics* 15, no. 3 (2019): 2654–2667.

[38] P. K. Rout, S. Nanda, and S. Acharya, "Muti-Objective Fuzzy Probabilistic Quadratic Programming Problem," *International Journal of Operational Research* 34, no. 3 (2019): 387–408.

[39] A. Biswas and A. K. De, "Methodology for Solving Multi-Objective Quadratic Programming Problems in a Fuzzy Stochastic Environment," in *Multi-objective Stochastic Programming in Fuzzy Environment* (Hershey, PA: IGI Global Publisher, 2019), 177–217.

[40] H. A. Khalifa, "Interactive Multiple Objective Programming in Optimization of the Fully Fuzzy Quadratic Programming Problems," *International Journal of Applied Operational Research* 10, no. 1 (2020): 21–30.

[41] N. A. Taghi-Nezhad and F. Babakordi, "Fuzzy Quadratic Programming With Non-Negative Parameters: A Solving Method Based on Decomposition," *Journal of Decisions and Operations Research* 3, no. 4 (2020): 325–332.

[42] J. Wang, F. He, and X. Shi, "Numerical Solution of a General Interval Quadratic Programming Model for Portfolio Selection," *PLoS One* 14, no. 3 (2019): https://doi.org/10.1371/journal.pone.0212913.

[43] A. Malek and H. G. Oskoei, "Numerical Solutions for Constrained Quadratic Problems Using High Performance Neural Networks," *Applied Mathematics and Computation* 169 (2005): 451–471.

[44] A. Mansoori, S. Effati, and M. Eshaghnezhad, "An Efficient Recurrent Neural Network for Solving Fuzzy Non-Linear Programming Problems," *Applied Intelligence* 46 (2017): 308–327.

[45] A. Mansoori, S. Effati, and M. Eshaghnezhad, "A Neural Network to Solve Quadratic Programming Problems With

Fuzzy Parameters," *Fuzzy Optimization and Decision Making* 17 (2018): 75–101.

[46] R. Coelho, "Solving Real-World Fuzzy Quadratic Programming Problems by Dual Parametric Approach," *Fuzzy Logic in Intelligent System Design, Advances in Intelligent Systems and Computing 648*, eds. P. Melin, O. Castillo, J. Kacprzyk, M. Reformat, and W. Melek (Berlin, Germany: Springer, 2018), https://doi.org/10.1007/978-3-319-67137-6_4.

[47] A. Chaweewanchon and R. Chaysiri, "Markowitz Mean-Variance Portfolio Optimization With Predictive Stock Selection Using Machine Learning," *International Journal of Financial Studies* 10 (2022): 64, https://doi.org/10.3390/ijfs10030064.

[48] T. Faturohman and D. Christian, "Predictive Blend: Fundamental Indexing With Markowitz Mean Variance Portfolio in Indonesia Stock Exchange," *Comparative Analysis of Trade and Finance in Emerging Economies (International Symposia in Economic Theory and Econometrics*, eds. W. A. Barnett and B. S. Sergi (Leeds, UK: Emerald Publishing Limited, 2023), 101–111, https://doi.org/10.1108/S1571-038620230000031013.

[49] J. Dai, L. Luo, L. Xiao, et al., "Modified Noise-Immune Fuzzy Neural Network for Solving the Quadratic Programming With Equality Constraint Problem," *IEEE Transactions on Neural Networks and Learning Systems* 1 (2023): https://doi.org/10.1109/TNNLS.2023.3290030.

[50] I. H. Hasan and I. H. A. Kanani, "Optimal Fuzzy Solution for Fully Fuzzy Quadratic Fractional Programming Problems With Decagonal Membership Function and Ranking Function Technique," *AIP Conference Proceedings* 3036 (2024): https://doi.org/10.1063/5.0196008.

[51] A. Tadesse, M. M. Acharya, S. Acharya, and M. Sahoo, "Fuzzy Goal Programming Approach to Solve Fully Fuzzy Multi-Objective Quadratic Programming Problem," *International Journal of System Assurance Engineering and Management* 15, no. 2 (2024): 705–712.

[52] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications* (Upper Saddle River, NJ: Prentice Hall PTR, 1995).

[53] Data Portal Pakistan Stock Exchange, "Data Portal-Pakistan Stock Exchange (PSX)," (2021), https://dps.psx.com.pk/.

[54] S. F. Ahmed, M. S. B. Alam, M. Hassan, et al., "Deep Learning Modelling Techniques: Current Progress, Applications, Advantages and Applications," *Artificial Intelligence Review* 56 (2023): 13521–13617.

[55] H. Tian, A. W. Liew, and H. Shen, "Advances in Parallel and Distributed Computing and its Applications," in *Concurrency and Computation: Practice and Experience* (Hoboken, NJ: John Wiley & Sons, Ltd, 2021), https://doi.org/10.1002/cpe.6667.