A Novel Approach to Multi-Layer-Perceptron Training using Quadratic Interpolation Flower Pollination Neural Network on Non-Binary Datasets

Yulianto Triwahyuadi Polly[1], Sri Hartati[2]*, Suprapto[3], Bambang Sumiarto[4]
Department of Computer Science-Faculty of Science and Engineering, Universitas Nusa Cendana, Kupang, Indonesia[1]
Department of Computer Science and Electronics-Faculty of Mathematics and Natural Science, Universitas Gadjah Mada, Yogyakarta, Indonesia[2, 3]
Department of Veterinary Medicine-Faculty of Veterinary Medicine, Universitas Gadjah Mada, Yogyakarta, Indonesia[4]

***Abstract*—Machine Learning (ML) algorithms are widely used in solving classification problems. The biggest challenge of classification lies in the robustness of the ML algorithm in various dataset characteristics. Quadratic Interpolation Flower Pollination Neural Network (QIFPNN) is categorised into ML algorithm. The new QIFPNN's extraordinary capabilities are measured on binary-type datasets. This research ensures that the remarkable ability of QIFPNN also applies to non-binary datasets with balanced and unbalanced data class characteristics. Flower Pollination Neural Network (FPNN), Particle Swarm Optimisation Neural Network (PSONN), and Bat Neural Network (BANN) were used as comparisons. The QIFPNN, FPNN, PSONN, and BANN were used to train Multi-Layer-Perceptron (MLP). The test results on five datasets show that QIFPNN obtains an average classification accuracy higher than its comparison in three datasets with balanced and unbalanced data class characteristics. The three datasets are Iris, Wine, and Glass. The highest classification accuracy obtained by QIFPNN in the three datasets is 97.1462%, 98.6551%, and 73.1979%, respectively. Based on the F1-score test from QIFPNN, it is higher than all the comparisons in four datasets: Iris, Wine, Vertebral column, and Glass. Sequentially, 96.4599%, 98.7155%, 90.7517%, and 60.2843%. It proves that QIFPNN can also classify datasets with non-binary data types with balanced and unbalanced data class characteristics because they are more consistently tested on various datasets and are not susceptible to the influence of variations in dataset characteristics so that they can be applied to various types of data or cases.**

Keywords*—Quadratic interpolation; flower pollination algorithm; neural network; non-binary dataset; multi-layer-perceptron

I. Introduction

The field of Machine Learning (ML) is a subsection of Artificial Intelligence (AI) [1] that allows computers to recognise patterns in data and make predictions based on that information [2,3]. Practical ML algorithms can provide precise results, even when dealing with datasets that present various real-world problems. The UCI Machine Learning Repository [4] is a database offering a range of datasets the AI community uses to evaluate ML algorithms.

ML algorithms are classifiers, meaning they can predict an unknown sample's class based on previous training data. A challenge in classifying data is the variation in dataset characteristics, such as sample size, number of attributes, class count, sample distribution in each class, missing data, and data type. Traditional classification algorithms can sometimes be unreliable, mainly when dealing with datasets with an uneven distribution of class samples or an unbalanced class distribution [5].

Optimisation algorithms are frequently utilised to tackle optimisation problems and have had positive outcomes in optimising Machine Learning (ML) algorithms, such as Neural Networks (NN) [6]. The NN, also known as Multi-Layer Perceptron (MLP), is considered one of the most effective ways to solve classification problems in real-world scenarios [7,8]. The training process of NN involves feedforward and backward procedures. The backward procedure, in which weight adjustment occurs through the conventional gradient method, can be weak and often gets stuck at local optima [9,10]. The metaheuristic algorithm can take the place of this backward procedure. The success of these algorithms lies in their capability to explore the search space through both exploration and exploitation. Exploration involves finding various solutions in the search space, while exploitation involves finding the best solution to improve existing ones. A proper balance between exploration and exploitation can quickly lead to identifying the search space with the optimal solution [11], avoiding any wasted time in areas with insufficient solutions [12,13].

The Flower Pollination Algorithm (FPA) is a metaheuristic approach that balances exploration and exploitation by regulating global and local pollination in each iteration of the population. FPA has been extensively used in classification tasks using public datasets. For instance, in a study by Senthilnath et al. [14], FPA was used to adjust the class centre weights in Euclidean distance training and was compared to other algorithms such as Harmony Search, Bat Algorithm, Differential Evolution, Spider Monkey Optimization, Grey Wolf Optimization, Cuckoo Search, Particle Swarm Optimization, Genetic Algorithm, and K-means. The results showed that FPA performed better, with the lowest Classification Error Percentage, on all tested datasets. Additionally, FPA was applied to update the Probabilistic Neural Network (PNN) weights in classification problems and produced better results than PNN on all 11 datasets [7]. Yazid

et al. [15] used FPNN, which combines FPA and PNN, to classify heart diseases and found it to have higher accuracy than a standard backpropagation neural network based on results from four UCI datasets.

In recent research, Polly et al. [16] looked into classifying real-world swine disease cases. The dataset comprised 158 samples, 68 attributes, a binary data type, and an unbalanced data distribution in 11 classes. The Quadratic Interpolation Flower Pollination Neural Network (QIFPNN) increased accuracy by 22.40% and training speed by 7.61% compared to the Flower Pollination Neural Network (FPNN). QIFPNN is a modification of FPA where the Levy vector is replaced with a random vector based on the step length of quadratic interpolation (QI). The effectiveness of QIFPNN in increasing accuracy and speeding up training time on datasets with binary data types needs to be extended to datasets with non-binary data types. So it needs to be tested on various dataset characteristics with various data types, then compare the results with FPNN, PSONN, and BANN, which are other metaheuristic algorithms. The goal is to prove that QIFPNN can also classify datasets with non-binary data types. As a result, QIFPNN provides better classification accuracy and F1-score on datasets with non-binary data types than its comparators. It proves that QIFPNN is not susceptible to variations in dataset characteristics so that it can be applied to various data types or cases.

This research is structured as follows: in the theory section, we present the Quadratic Interpolation Flower Pollination Neural Network (QIFPNN). In the experimental setup section, we describe the dataset and parameter settings. In the results and discussion section, we present the accuracy measurement and F1-score of the QIFPNN and its comparators, followed by the conclusion section.

II. THEORY

A. Quadratic Interpolation Flower Pollination (QIFP)

Polly et al. [16] introduced QIFP, an improvement from FPA [17]. The improvement lies in the step vector of global pollination and the search space technique. In the first improvement, the step vector γL is replaced with a quadratic interpolation step vector Q, so (1) can be written as (2).

$$x_i^{t+1}=x_i^t+\gamma L(g^*-x_i^t) \quad (1)$$

$$x_i^{t+1}=x_i^t+Q(g^*-x_i^t) \quad (2)$$

where x_i^t represents the i-th pollen in iteration t, g^* is the best pollen, γ is the scaling factor, and L is the random step vector with Levy distribution. The pollen represents the solution vector.

The derivative of the polynomial quadratic function is used to get r^* which gives the minimum or maximum fitness. The illustration can be seen in Fig. 1. Next; the step vector Q is formed by:

1) Generate a vector Q with the normal distribution $Q \sim N(\mu, \sigma)$, where the mean $\mu=r^*$, deviation standard $\sigma=f_1$, and the f_1 are fitness value of objective functions of g^*.

2) Replace the element value of the vector Q by 20% from d by using (3).

$$Q_c=\begin{cases}0, & c=l\\ Q_c, & \text{otherwise}\end{cases} \quad (3)$$

where $l \in randi[1,d]$; *randi* is *rand integer*, $c=1,2,\ldots,d$ and d is the dimension of the solution vector.

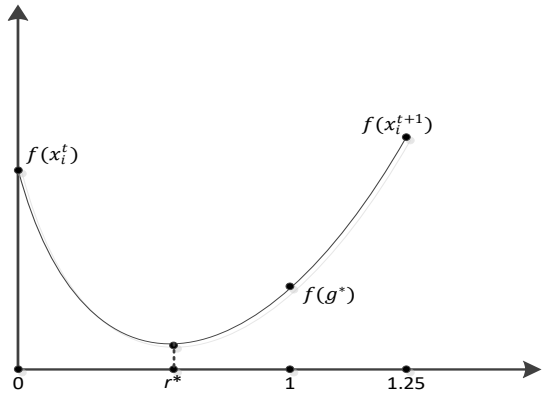

Fig. 1. Illustration of a polynomial quadratic function.

The second improvement, the FPA search space is directly carried out in the natural search space [*LbReal*,*UbReal*]. The QIFP search space starts from a small search space, which is then expanded gradually. Expansion is done by:

1) Identify search space first:

$$[\text{Lb,Ub}]^d=[-1,1]^d \quad (4)$$

2) Expanding the next search space:

$$[Lb,Ub]^d=\begin{cases}[Lb-1,Ub+1]^d & t \text{ MOD } 50=0\\ [Lb,Ub]^d & \text{otherwise}\end{cases} \quad (5)$$

3) Repeat step 2) until $Lb = LbReal$ and $Ub = UbReal$.

where the value of *LbReal* and *UbReal* has an integer type, and zero is the result of the sum of *LbReal* and *UbReal*.

B. Quadratic Interpolation Flower Pollination Neural Network (QIFPNN)

The evaluation of weights for classification problems in MLP is to minimise the mean square error (MSE) value. In QIFPNN, QIFP is used as a weight adjustment [16]. The flowchart of QIFPNN can be seen in Fig. 2, which represents the QIFPNN algorithm. The novelty can be seen in the two rectangles marked with bold lines. The first rectangle refers to (2), while the second rectangle refers to (4) and (5).

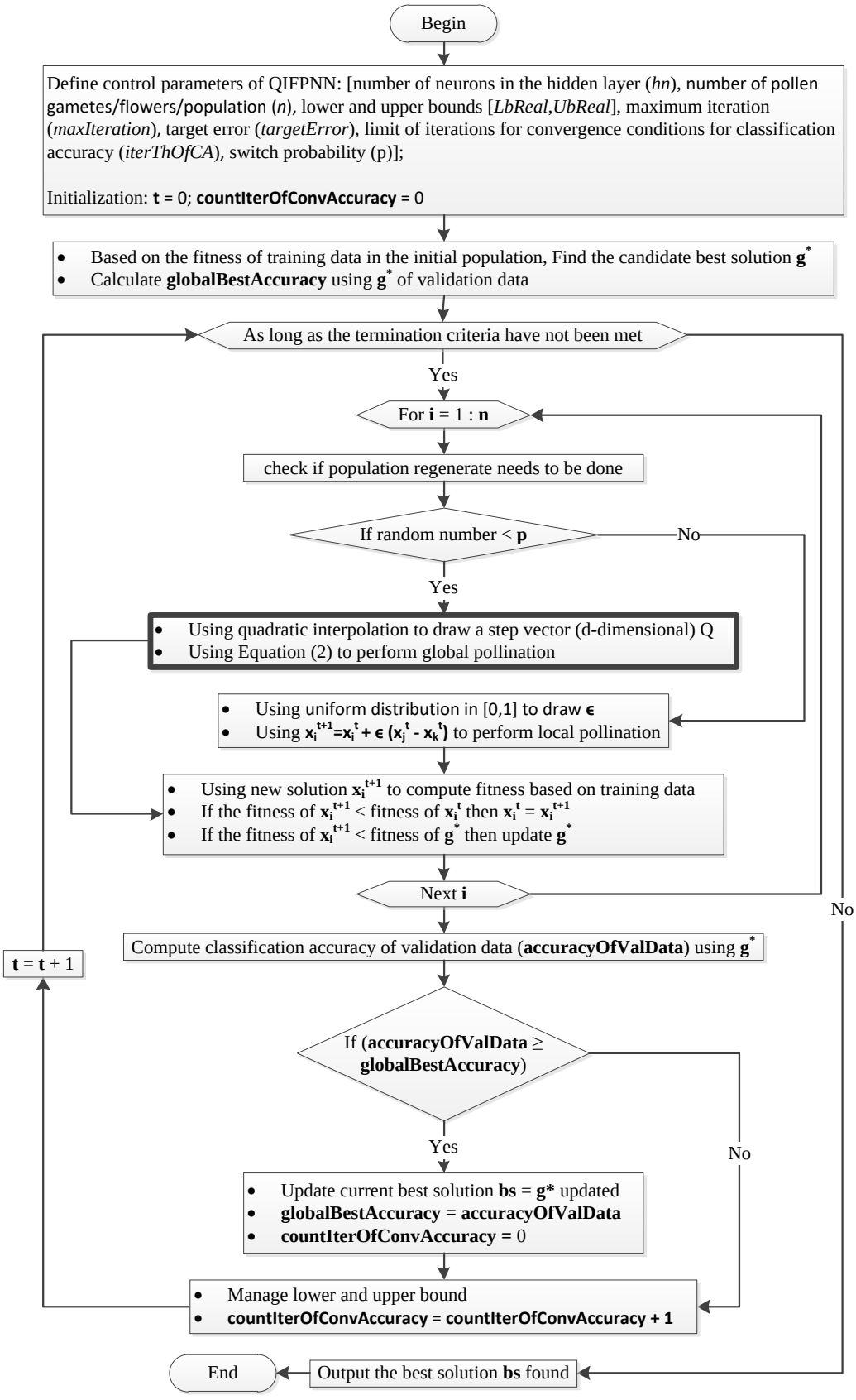

Fig. 2. QIFPNN flowchart.

III. Experimental Setup

A. Dataset Description

This section briefly overviews the five UCI datasets used in the study. The datasets were divided into three parts: training, validation, and testing, with 90% of the data used for training and validation (using 10-fold cross-validation) and 10% for testing

The classification process took place in two stages:

1) The training stage aimed to find the optimal weights and determine the length of the training process. This stage used both training and validation subsets to prevent overfitting.

2) In the testing stage, the weights found in the training stage were applied to the test subsets to measure classification accuracy.

Table I displays the sample size, number of input attributes and classes, missing sample size, and data type of each dataset. Table I can be categorised into datasets with balanced data classes (Iris) and unbalanced data classes (Wine, Lung cancer, Vertebral column, Glass). The Lung cancer dataset has many input attributes but a small sample size (known as a singularity problem) [18]. The Vertebral column dataset has two classes with approximately equal amounts of data in each class. The Glass dataset has six classes with varying amounts of data in each class.

B. Parameter Settings

The parameters used to configure each algorithm consist of four parts, namely:

- Parameters that apply to all algorithms, including (1) the number of neurons in the hidden layer (*hn*), (2) the population (*population*), (3) the lower and upper bounds of the actual search space [*LbReal*,*UbReal*], (4) the maximum iteration (*maxIteration*), (5) the target error (*targetError*), (6) the limit of iterations for convergence conditions for classification accuracy (*iterThOfCA*);
- Parameters specific to QIFPNN and FPNN, namely the switch probability (*p*);
- Parameters specific to PSONN, namely the learning parameters (α and β);
- Parameters were specific to BANN: the loudness (*loudness*) and the pulse rate (*pulseRate*).

The QIFPNN, FPNN, PSONN, and BANN use MLP architecture with one hidden layer by determining the number of neurons in that layer according to (6) [16]. One hidden layer can approximate any function with arbitrary accuracy [19].

$$\mathrm{hn} = \sqrt{(\mathrm{si}+\mathrm{so})}+\mathrm{ii} \tag{6}$$

Where, respectively, hn, si, and so, are the number of neurons in the hidden, input, and output layers, while ii is an integer set to be 1. The determination of $ii = 1$ aims to simplify the size of the MLP architecture in order to expedite the learning process.

It is essential to control parameters in each iteration of a metaheuristic algorithm to obtain an optimal solution. However, there is no known effective strategy to produce a variety of parameters [20]. All parameters are set based on Polly et al. [16], namely population n=30 [16,21], [*LbReal*,*UbReal*]=[−80,80] , maximum iteration *maxIteration*=4000, target error *targetError*=0.001, and limit of iterations for convergence conditions for classification accuracy *iterThOfCA*=700 . Three variables control the stopping criteria: *maxIteration*, *targetError*, and *iterThOfCA*. Another parameter set for QIFPNN and FPNN is switch probability p=0.8 [16,17,22–24]. Specifically for PSONN, the learning parameters are $\alpha \approx \beta \approx 2$, while for BANN, the parameter *loudness*=0.25 and the parameter *pulseRate*=0.5 [25].

TABLE I. Characteristics of the Datasets

Dataset	Sample Size	Number of Input Attributes	Sample Size in Each Class	Missing Sample Size	Data Type
Iris	150	4	• *Class* 1 Iris Setosa = 50 data (33.3%) • *Class* 2 Iris Versicolour = 50 data (33.3%) • *Class* 3 Iris Virginica = 50 data (33.3%)	-	Real
Wine	178	13	• *Class* 1 = 59 data (33.2%) • *Class* 2 = 71data (39.9%) • *Class* 3 = 48 data (26.97%)	-	Integer, real
Lung Cancer	27	56	• *Class* 1 = 8 data (29.6%) • *Class* 2 = 10 data (37.04%) • *Class* 3 = 9 data (33.3%)	5	Integer
Vertebral Column	310	6	• *Class* 1 Abnormal = 210 data (67.7%) • *Class* 2 Normal = 100 data (32.3%)	-	Real
Glass	214	9	• *Class* 1 Building windows float processed = 70 data (32.7%) • *Class* 2 Building windows non-float processed = 17 data (7.9%) • *Class* 3 Vehicle windows float processed = 76 data (35.5%) • *Class* 4 Vehicle Windows non-float processed = 0 data (0%) • *Class* 5 Containers = 13 data (6.1%) • *Class* 6 Tableware = 9 data (4.2%) • *Class* 7 Headlamps = 29 data (13.6%)	-	Real

This study set the population parameter to 30, referring to Chakraborty et al. [21], as the four datasets used in this research were also employed in their study. Furthermore, the findings of Polly et al. [16] provided additional support by utilising the same population size and yielding satisfactory solutions. The rationale for determining the parameter [*LbReal*,*UbReal*] = [-80, 80] was based on an intuitive approach to solution search techniques, starting from the smallest search space and gradually expanding it. The expansion of the search space was performed every 50 iterations, and the most extensive range of the search space, [−80,80], was determined by setting the maximum iteration to 4000. The target error parameter was set to 0.001 to achieve high accuracy on the training data.

The parameter *iterThOfCA* was set to 700 based on documented test results in Table II. The row labelled "Average Training and Validation Accuracy" shows that using the *iterThOfCA*=700 parameter yields slightly lower accuracy than the *iterThOfCA*=1000 and 1800 parameters. However, the difference with the highest accuracy is tiny, only 0.22. In the row labelled "Training Time," the *iterThOfCA*=700 parameter results in a shorter time than the *iterThOfCA*=1000 and 1800 parameters. Therefore, the *iterThOfCA*=700 parameter is selected as the appropriate option. The switch probability parameter is set to 0.8 based on preliminary parametric studies indicating that a value of p=0.8 can provide better performance for most applications [17,23,24]. Expressly, for PSONN, the learning parameters ("α and β") are set to 2, while for BANN, the normal values for the loudness parameter are 0.25, and the pulse rate parameter is set to 0.5, following standard practices in PSONN and BANN [25].

TABLE II. THE DETERMINATION OF THE *ITERTHOFCA* PARAMETER WAS BASED ON THE AVERAGE TRAINING ACCURACY, VALIDATION ACCURACY, AND TRAINING TIME USING 10-FOLD CROSS-VALIDATION, CONDUCTED OVER 5 REPETITIONS

	***iterThOfCA* Parameter**		
	700	**1000**	**1800**
The average training and validation accuracies (%)	**87.9191**	88.0872	88.1432
The average training time (seconds)	**5934.45**	7497.80	10555.65

C. Testing

The experiment measures the performance of four algorithms by evaluating their average classification accuracy and training time. The tests were repeated 20 times in each fold of the 10-fold cross-validation method, and the results were recorded as the average of these trials. Additionally, the F1-score was also calculated as part of the testing procedure.

IV. RESULT AND DISCUSSION

The tests were conducted on five datasets, as outlined in Table I. The results of the tests, including the average of the classification accuracy and the average of the training time, are shown in Table III and Fig. 3 to 4. Table IV shows the results of the F1-score test.

As seen in Table III, the QIFPNN algorithm produced a higher average classification accuracy in the training subset than the other algorithms for all five datasets. It includes the Iris, Wine, Lung cancer, Vertebral column, and Glass datasets with an accuracy of 98.9103%, 99.3056%, 90.7478%, 87.6424%, and 73.855%, respectively. The QIFPNN model has the lowest mean square error and does not experience premature convergence across various datasets. The F1-score further supports this in the training subset, which was higher for all five datasets, including the Iris, Wine, Lung cancer, Vertebral column, and Glass datasets, with scores of 97.7294%, 99.3488%, 90.0801%, 90.9789%, and 64.8062% respectively, as shown in Table IV.

The average classification accuracy obtained from the training, validation, and test subsets from QIFPNN is higher than all the comparisons in the three datasets, namely the Iris, Wine, and Glass datasets. The average acquisition accuracy of the classification can be seen in Table III and Fig. 3, namely 97.1462%, 98.6551%, and 73.1979%. FPNN is only higher in two datasets than all the comparisons in the Lung cancer and Vertebral column datasets, respectively 78.7861% and 87.4692%. Still, QIFPNN ranks second highest after FPNN in both datasets, with gains of 76.5826% and 87.3895%. Based on the F1-score test from QIFPNN in Table IV, it is higher than all the comparisons in the four datasets: Iris, Wine, Vertebral column, and Glass. Sequentially, 96.4599%, 98.7155%, 90.7517%, and 60.2843%. Meanwhile, FPNN is higher than all its comparisons only in the Lung cancer dataset, namely 75.3369%, but QIFPNN ranks second highest with an acquisition of 73.1106%. It proves that the QIFPNN training model is the best among all comparisons because it is more consistently tested on various datasets and is not susceptible to the influence of variations in dataset characteristics so that it can be applied to multiple other data/cases.

The average PSONN training time is faster than QIFPNN, FPNN, and BANN on three datasets, namely Iris, Vertebral column, and Glass, with values of 384.4917 seconds, 55.6039 seconds, and 678.6673 seconds. QIFPNN is faster than FPNN, BANN, and PSONN on two datasets, namely Wine and Lung Cancer, with values of 179.7063 seconds and 135.8932 seconds, which can be seen in Table III and Fig. 4. It proves that, in general, PSONN is faster than its comparators. However, QIFPNN has a speedy training time compared to the comparison specifically for datasets that experience singularity problems, such as the Lung cancer dataset. The training time is also quick for datasets like Wine, which have unbalanced data class characteristics. This fact shows that the quadratic interpolation concept accommodated by QIFPNN works very quickly in the training process for datasets with characteristics similar to those of the Lung cancer and Wine datasets.

Based on the results of classification accuracy and F1-score measurements, it can be said that the QIFPNN is suitable for classifying non-binary datasets with balanced and unbalanced data class characteristic models. The slow training time of QIFPNN on the Iris, Vertebral column, and Glass datasets is due to the QIFPNN fitness evaluation being twice the fitness evaluation of FPNN, BANN, and PSONN in each individual (flower/pollen gamete) per iteration.

TABLE III. RECAPITULATION OF TESTS ON THE IRIS, WINE, LUNG CANCER, VERTEBRAL COLUMN, AND GLASS DATASETS BASED ON THE AVERAGE FOR CLASSIFICATION ACCURACY AND TRAINING TIME OF 20 TRIALS ON ALL FOLDS

Dataset	Method	The Average for Classification Accuracy of 10-fold on			The Average for Classification Accuracy of The Training, Validation, and Test Subsets (%)	The Average Training Time of 10 fold (seconds)
		Training subset (%)	*Validation subset* (%)	*Test subset* (%)		
Iris	QIFPNN	**98.9103**	98.4615	94.0667	**97.1462**	534.3663
	FPNN	98.3390	98.6978	92.2000	96.4123	488.9779
	BANN	89.7546	89.1566	84.3000	87.7371	508.0139
	PSONN	82.0805	82.9890	78.8000	81.2898	**384.4917**
Wine	QIFPNN	**99.3056**	98.1875	98.4722	**98.6551**	**179.7063**
	FPNN	99.0104	99.0313	95.5278	97.8565	709.1112
	BANN	90.6319	89.5938	89.3333	89.8530	702.8217
	PSONN	85.1424	87.0000	83.6389	85.2604	943.0891
Lung Cancer	QIFPNN	**90.7478**	70.0000	69.0000	76.5826	**135.8932**
	FPNN	87.0249	88.5000	60.8333	**78.7861**	784.5162
	BANN	71.9989	65.4167	49.0000	62.1385	640.8733
	PSONN	63.6818	67.4167	47.6667	59.5884	826.6481
Vertebral Column	QIFPNN	**87.6424**	89.5099	85.0161	87.3895	727.1914
	FPNN	85.9383	90.3241	86.1452	**87.4692**	555.9895
	BANN	86.2841	88.0595	84.5000	86.2812	675.5812
	PSONN	76.2325	80.8082	77.2742	78.1050	**553.6039**
Glass	QIFPNN	**73.8550**	73.0395	72.6991	**73.1979**	1199.2273
	FPNN	64.3742	71.3566	65.6212	67.1173	856.9318
	BANN	60.5766	61.9513	59.0000	60.5093	806.5957
	PSONN	46.6498	51.8789	48.4524	48.9937	**678.6673**

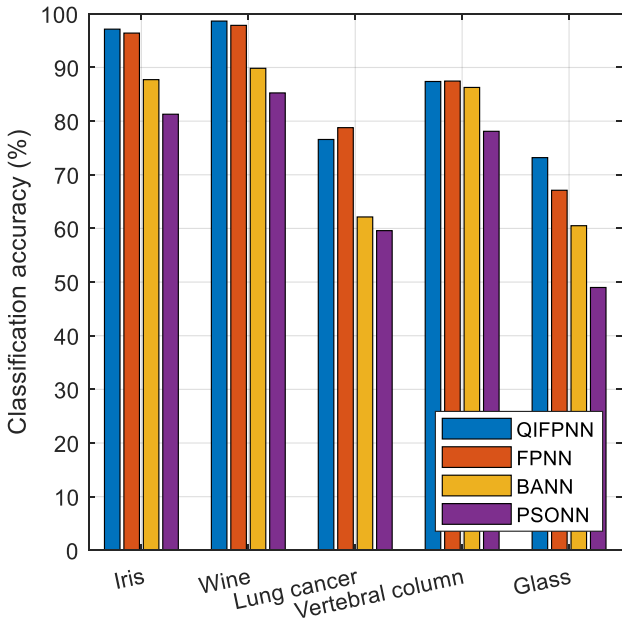

Fig. 3. The average for classification accuracy of the training, validation, and test subsets.

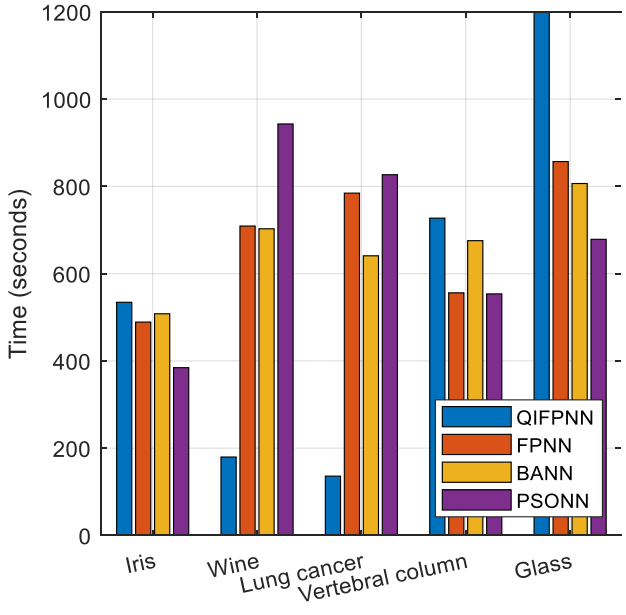

Fig. 4. The average training time of 10 fold.

TABLE IV. RECAPITULATION OF TESTS ON THE IRIS, WINE, LUNG CANCER, VERTEBRAL COLUMN, AND GLASS DATASETS BASED ON THE AVERAGE FOR F1-SCORE OF 20 TRIALS ON ALL FOLDS

Dataset	Method	The Average for F1-score of 10 Fold on			The Average for F1-score of The Training, Validation, and Test Subsets (%)
		Training Subset (%)	*Validation Subset (%)*	*Test Subset (%)*	
Iris	QIFPNN	**97.7294**	98.5092	93.1412	**96.4599**
	FPNN	96.7000	98.7036	90.7623	95.3886
	BANN	85.3531	85.9876	80.4231	83.9213
	PSONN	75.4134	77.5504	72.9403	75.3014
Wine	QIFPNN	**99.3488**	98.2541	98.5435	**98.7155**
	FPNN	99.0587	99.0560	95.6972	97.9373
	BANN	88.4922	87.3719	87.1803	87.6815
	PSONN	83.9748	85.7330	82.5509	84.0862
Lung Cancer	QIFPNN	**90.0801**	66.4461	62.8056	73.1106
	FPNN	86.6670	85.5936	53.7500	**75.3369**
	BANN	68.2836	62.7826	39.8333	56.9665
	PSONN	60.7273	64.1797	38.7500	54.5523
Vertebral Column	QIFPNN	**90.9789**	92.3511	88.9251	**90.7517**
	FPNN	89.7003	92.9143	89.5973	90.7373
	BANN	89.9682	91.2794	88.4099	89.8858
	PSONN	83.1972	86.5253	83.7587	84.4937
Glass	QIFPNN	**64.8062**	55.9833	60.0634	**60.2843**
	FPNN	45.7893	48.3887	44.9519	46.3766
	BANN	41.4919	38.9253	36.9897	39.1356
	PSONN	23.9903	27.5101	25.1203	25.5403

However, QIFPNN is faster in obtaining solutions compared to FPNN, BANN, and PSONN on the Lung cancer and Wine dataset, so it can be explained that the quadratic interpolation concept on QIFPNN can increase the QIFPNN training speed than the levy distribution concept on FPNN, the idea of acoustic echolocation on BANN, and the concept of adjusting the trajectories of individual agents, called particles, as the piecewise paths formed by positional vectors in a quasi-stochastic manner in PSONN in both datasets.

V. CONCLUSION

In previous research, the new algorithm QIFPNN was tested on real-world cases involving binary data types. Findings from that study indicated that QIFPNN significantly outperformed FPNN, PSONN, and BANN. In this study, QIFPNN was tested using five UCI datasets with variations in data characteristics, such as non-binary data types, balanced and imbalanced classes, and singularity issues. The main objective of this research was to investigate the reliability of QIFPNN in dealing with various characteristics present in these datasets. Reliability measurements used classification accuracy, F1-score, and training time as evaluation metrics. The classification accuracy measurements on the training subset consistently showed that QIFPNN outperformed all other models across all datasets, indicating that QIFPNN did not suffer from premature convergence.

Moreover, the average classification accuracy on the training, validation, and test subsets demonstrated that QIFPNN performed superiorly on three datasets: Iris, Wine, and Glass. Although on the other two datasets, QIFPNN ranked second after FPNN; the difference was marginal. Furthermore, the F1-score test results revealed that QIFPNN significantly outperformed other models on four datasets, including Iris, Wine, Vertebral column, and Glass. However, on the Lung cancer dataset, QIFPNN ranked second after FPNN with a slight difference. These findings demonstrate that QIFPNN exhibits reliable performance in handling various dataset characteristics. Based on the measurements of classification accuracy and F1-score, it can be concluded that the QIFPNN training model is a reliable choice for various dataset characteristics in different cases.

Additionally, the training time measurements indicated that PSONN was faster on three datasets, namely Iris, Vertebral column, and Glass, while QIFPNN was faster on two datasets. In the QIFPNN algorithm, the fitness evaluation is performed twice, resulting in excessive training time consumption. We recommend improving the Q step vector by introducing an additional parameter as a multiplier factor. The aim is to enhance the performance of QIFPNN.

ACKNOWLEDGMENT

The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation. The Department of Computer Science and Electronic Universitas Gadjah Mada supports this work.

REFERENCES

[1] Y. Shambharkar, S. Salagrama, K. Sharma, O. Mishra, and D. Parashar, "An Automatic Framework for Number Plate Detection using OCR and Deep Learning Approach," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 4, pp. 8–14, 2023.

[2] S. Rauschert, K. Raubenheimer, P. E. Melton, and R. C. Huang, "Machine learning and clinical epigenetics: A review of challenges for diagnosis and classification," *Clin. Epigenetics*, vol. 12, no. 1, pp. 1–11, 2020, doi: 10.1186/s13148-020-00842-4.

[3] C. Krittanawong, H. J. Zhang, Z. Wang, M. Aydar, and T. Kitai, "Artificial Intelligence in Precision Cardiovascular Medicine," *J. Am. Coll. Cardiol.*, vol. 69, no. 21, pp. 2657–2664, 2017, doi: 10.1016/j.jacc.2017.03.571.

[4] D. Dua and C. Graff, "UCI Machine Learning Repository," *Irvine, CA: University of California, School of Information and Computer Science*, 2019. http://archive.ics.uci.edu/ml.

[5] H. Patel, D. Singh Rajput, G. Thippa Reddy, C. Iwendi, A. Kashif Bashir, and O. Jo, "A review on classification of imbalanced data for wireless sensor networks," *Int. J. Distrib. Sens. Networks*, vol. 16, no. 4, pp. 1–15, 2020, doi: 10.1177/1550147720916404.

[6] D. Devikanniga, K. Vetrivel, and N. Badrinath, "Review of meta-heuristic optimisation based artificial neural networks and its applications," *J. Phys. Conf. Ser.*, vol. 1362, no. 1, 2019, doi: 10.1088/1742-6596/1362/1/012074.

[7] M. Alweshah, M. A. Qadoura, A. I. Hammouri, M. S. Azmi, and S. AlKhalaileh, "Flower Pollination Algorithm for Solving Classification Problems," *Int. J. Adv. Soft Comput. its Appl.*, vol. 12, no. 1, pp. 15–34, 2020.

[8] M. Alweshah, A. I. Hammouri, and S. Tedmori, "Biogeography-based optimisation for data classification problems," *Int. J. Data Mining, Model. Manag.*, vol. 9, no. 2, pp. 142–162, 2017, doi: 10.1504/IJDMMM.2017.085645.

[9] Q. T. Bui, "Metaheuristic algorithms in optimising neural network: a comparative study for forest fire susceptibility mapping in Dak Nong, Vietnam," *Geomatics, Nat. Hazards Risk*, vol. 10, no. 1, pp. 136–150, 2019, doi: 10.1080/19475705.2018.1509902.

[10] M. Mavrovouniotis and S. Yang, "Training neural networks with ant colony optimisation algorithms for pattern classification," *Soft Comput.*, vol. 19, no. 6, pp. 1511–1522, 2015, doi: 10.1007/s00500-014-1334-5.

[11] J. A. Villaruz, B. D. Gerardo, A. O. Gamao, and R. P. Medina, "Scouting Firefly Algorithm and its Performance on Global Optimization Problems," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 3, pp. 445–451, 2023.

[12] B. Morales-Castañeda, D. Zaldívar, E. Cuevas, F. Fausto, and A. Rodríguez, "A better balance in metaheuristic algorithms: Does it exist?," *Swarm Evol. Comput.*, vol. 54, p. 100671, 2020, doi: 10.1016/j.swevo.2020.100671.

[13] M. Crepinsek, S. H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 1–33, 2013, doi: 10.1145/2480741.2480752.

[14] J. Senthilnath, S. Kulkarni, S. Suresh, X. S. Yang, and J. A. Benediktsson, "FPA clust: evaluation of the flower pollination algorithm for data clustering," *Evol. Intell.*, no. 0123456789, Jun. 2019, doi: 10.1007/s12065-019-00254-1.

[15] M. Haider Bin Abu Yazid, M. Shukor Talib, and M. Haikal Satria, "Flower Pollination Neural Network for Heart Disease Classification," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 551, no. 1, 2019, doi: 10.1088/1757-899X/551/1/012072.

[16] Y. T. Polly, S. Hartati, Suprapto, and B. Sumiarto, "Modified Flower Pollination Algorithm For Disease Identification In Swine," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 6, pp. 616–628, Dec. 2021, doi: 10.22266/ijies2021.1231.55.

[17] X.-S. Yang, "Chapter 11 - Flower Pollination Algorithms," in *Nature-Inspired Optimization Algorithms*, 2014, pp. 155–173.

[18] L. F. Chen, H. Y. M. Liao, M. T. Ko, J. C. Lin, and G. J. Yu, "A New LDA-based Face Recognition System Which Can Solve the Small Sample Size Problem," *Pattern Recognit.*, vol. 33, pp. 1713–1726, 2000.

[19] H. Chiroma *et al.*, "A new approach for forecasting OPEC petroleum consumption based on neural network train by using flower pollination algorithm," *Appl. Soft Comput.*, vol. 48, pp. 50–58, Nov. 2016, doi: 10.1016/j.asoc.2016.06.038.

[20] X. S. Yang, *Cuckoo Search and Firefly Algorithm*, vol. 516. Cham: Springer International Publishing, 2014.

[21] D. Chakraborty, S. Saha, and S. Maity, "Training feedforward neural networks using hybrid flower pollination-gravitational search algorithm," *Futur. Trends Comput. Anal. Knowl. Manag. (ABLAZE), 2015 Int. Conf.*, pp. 261–266, 2015, doi: 10.1109/ABLAZE.2015.7155008.

[22] K. Lu, Z. Ma, X. Yang, and H. Zhou, "An Improved Flower Pollination Algorithm for Global Numerical Optimization," in *2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, Oct. 2020, pp. 271–274, doi: 10.1109/DCABES50732.2020.00077.

[23] X.-S. Yang, "Flower Pollination Algorithm for Global Optimization," in *Unconventional Computation and Natural Computation*, vol. 7445, 2012, pp. 240–249.

[24] A. Al Bataineh, D. Kaur, and S. M. J. Jalali, "Multi-Layer Perceptron Training Optimization Using Nature Inspired Computing," *IEEE Access*, vol. 10, pp. 36963–36977, 2022, doi: 10.1109/ACCESS.2022.3164669.

[25] X.-S. Yang, *Nature-Inspired Optimization Algorithms*. Elsevier Inc, 2014.