# Hybrid Multi-layer Perceptron and Metaheuristic Optimizers for Indoor Localization Error Estimation

Jia Liu[1], Jian Sun[2, *]
[1]CangZhou Kindergarten Teachers College, Cangzhou 061001, China
[2]Hebei Software Institute, Baoding 071000, China
E-mail: sunjian8035@163.com
[*]Corresponding author

**Keywords:** average localization error, pelican optimizer algorithm, global situational system, gold rush optimizer, multi-layer perceptron regression

*Indoor localization is hindered by GPS signal weakening in indoor environments. This research formulates machine learning with Multi-layer Perceptron Regression (MLPR) algorithm supported by two metaheuristic optimizers, namely, Gold Rush Optimizer (GRO) and Pelican Optimizer (POA), to yield hybrid models MLGR and MLPO to forecast Average Localization Error (ALE). The dataset organized in a structured form was of size 107 samples with six significant features as follows: anchor ratio, transmission range, node density, trainings, standard deviation of ALE, and ALE as objective. The dataset was split into training (70%), validation (15%), and testing (15%) subsets. Experimental analysis in three prediction layers reveals that MLGR outperformed MLPO and MLPR models in every prediction layer. MLGR exhibited maximum performance at the third test layer with an RMSE of 0.036 and $R^2$ of 0.993, whereas MLPO and MLPR attained RMSE of 0.059 and 0.080 and values of $R^2$ of 0.981 and 0.966, respectively. The findings establish the validity of the introduced hybrid optimization technique to increase accuracy and convergence rate of prediction of ALE in wireless sensor networks.*

*Povzetek: Raziskava uvaja hibrid MLPR z optimizatorjema GRO in POA za napoved ALE v WSN. Model MLGR doseže najboljšo točnost (RMSE 0.036, $R^2$ 0.993), presega MLPO in osnovni MLPR.*

## 1 Introduction

A Wireless Sensor Network (WSN) is designed by several cheap, tiny sensors located all around an area to gauge various physical characteristics or monitor environmental situations [1,2]. Target tracking and precision agriculture are only two of the many industries where this technology finds applicable uses [3,4]. These sensors are often required to make accurate coordinate predictions while utilizing the minimum resources possible [5,6]. Although sensors equipped with an incorporated GPS can quickly determine their situations, cost and size constraints preclude embedding GPS in each sensor. Employing localization frameworks is one alternative strategy, whereby several anchor nodes equipped with incorporated GPS help the hidden nodes to identify their coordinates accurately [7,8].

Numerous localization frameworks have been developed to address diverse problems [9]. These frameworks should be flexible to operate well in different indoor and outdoor structures and topologies. These localization strategies fall into two groups: range-free and range-based frameworks. When using range-based strategies, the unfamiliar nodes' situation is identified by measuring the length between the anchor and unfamiliar sensor nodes [10,11]. Diverse factors are employed, including the Received Signal Strength Indication (RSSI), the angle, and time of arrival [12]. Conversely, range-free methods such as centroid [13] and ad-hoc situationing systems[14] use simple connectivity-related operations to locate hidden nodes. All these frameworks need is for the anchor node's beacon signal to be present in the medium. Range-based frameworks are preferred and used more frequently than range-free frameworks [15–17]. Several bioinspired frameworks have been recommended for the range-based method to help design a less complex algorithm [18]. A node localization technique utilizing Particle Swarm Optimization (PSO) [19] was first described by Gopakumar and Jacob [20], mimicking the behavior of a swarm of fish searching for food. While the framework first yielded promising results, it often became stuck in local optima, leading to premature convergence. Goyal and Patterh [21] showed significant results in reducing localization errors in 2014 when they used Cuckoo Search (CS) for node localization in WSNs. The CS algorithm's tuning settings, which simplified the computation procedure, are responsible for this result. Researchers recently recommended a modification to the CS technique: Cheng and Xia [22], which improved the traditional CS algorithm's rate of convergence. They changed the mutation probability and random walk stage size to improve the search strategy [23,24].

A Bayesian approach for node localization in WSNs was introduced by Morelande et al. [25]. The suggested algorithm, known as progressive correction, improved their previous work [26]. Both strategies are compared under different situations, with the Cramer-Rao bound (CRB) as the benchmark. Outperforming its predecessor, the newly recommended algorithm achieves superior accuracy. Furthermore, Ghargan et al. [27] recommended a tactic in which three enhancement frameworks, PSO, Backtracking Search Algorithm (BSA), and Gravitational Search Algorithm (GSA), are individually hybridized with ANN. Outperforming alternative methods, the GSA-ANN hybrid achieved low mean absolute distance projection errors: 0.02 meters for outdoor and 0.2 meters for indoor cases. In a current survey by Ahmadi and Bouallegue [28], various innovative ML strategies employed in node localization within WSNs were compiled. The study included a comparison of cumulative localization error distribution curves for different strategies such as ANN, SVM, DT, and Naive Bayes (NB). Analysis of cumulative localization error distributions revealed NB as the superior machine learning technique among those evaluated [29–31].

Bhatti et al. [32] recommended an outlier recognition framework for indoor localization known as iF_Ensemble. This technique utilizes supervised ML methods, RF, and SVM-along with the unsupervised Isolation Forest, and an ensemble technique called stacking. It significantly outperforms the model with a very high localization accuracy of 97.8%.

Wang et al. [33] recommended a new node localization framework called KELM-HQ, which uses Kernel Extreme Learning Machines to boost the accuracy of estimating the situations of hidden nodes. It achieves a 34.6% improvement in localization error compared with fast-SVM, 19.2% with the GADV-Hop algorithm, and 11.9% with the DV-Hop-ELM algorithm. Table 1 represents the comparative summary of state-of-the-Art for indoor localization error prediction.

Table 1: Comparative summary of state-of-the-Art for indoor localization error prediction.

| Method | Reference | Technique | RMSE | MAE | $R^2$ | Limitations and Improvements |
|---|---|---|---|---|---|---|
| PSO-ANN | Gharghan et al. (2016) [34] | Particle Swarm Optimization with ANN | 0.2 (indoor) | 0.12 | N/A | Converges to local minima; lacks robustness under varying topologies. MLGR offers more stable convergence and better generalization. |
| Cuckoo Search | Goyal & Patterh (2014) [35] | Cuckoo Search Metaheuristic | 0.42 | N/A | N/A | Requires tuning-sensitive parameters; slower convergence. MLGR improves convergence speed and accuracy. |
| Bayesian Correction | Morelande et al. (2008) [36] | Progressive Bayesian Estimation | 0.23 | 0.15 | 0.91 | Not ML-based; lower adaptability in dynamic environments. MLGR leverages adaptive learning for real-time localization. |
| KELM-HQ | Wang et al. (2020) [33] | Kernel Extreme Learning Machine | 0.58 | N/A | 0.93 | No optimizer hybridization; poorer error margin. MLGR offers up to 38% better RMSE. |
| MLPR | This Study (Baseline) | Multi-layer Perceptron Regression | 0.080 | 0.071 | 0.966 | Strong nonlinear modeling, but suboptimal without optimization. |
| MLPO | This Study | MLPR + Pelican Optimization | 0.059 | 0.051 | 0.981 | Improves ALE prediction; outperformed by MLGR in convergence and accuracy. |
| MLGR | This Study | MLPR + Gold Rush Optimization | 0.036 | 0.029 | 0.993 | Best overall accuracy and generalization. Efficiently addresses convergence and nonlinearity weaknesses of prior SOTA methods. |

This paper uses machine learning to advance the scheme for ALE prediction. In this paper, the authors used MLPR. MLPR was chosen for its known capability to represent complex, nonlinear relationships between input values typical in wireless sensor network datasets. In contrast to linear or tree-based regressors, MLPR supports complex feature interactions natively and does not require explicit feature transformation. Its design is highly flexible and adaptable, reflecting the need to predict ALE in varied network topologies. Its efficiency in other similar application areas has been previously proved, providing both accuracy and computational expense as needed, making it an appropriate tool to be applied to this case. The use of MLPR to predict ALE has several advantages. This framework performs very well in detecting intricate relationships within data and, hence, is suitable for those applications where the underlying pattern is complex and nonlinear. It is believed that MLPR can learn with diverse traits and complex adjustments of the ALE prediction task. This is because such a type of neural network is pretty robust, providing good results on big data and allowing generalization for unseen examples. GRO and POA were recommended as additional optimizers in a quest for a better level of accuracy for ALE prediction. The results and discussion also presented the hybridized framework's performance against others. Such a comparative study will go a long way in determining which model achieves better accuracy and precision in ALE forecasting. Consequently, the GRO inclusion, POA, and model hybridization in the present study assist in acquiring an in-depth understanding of the strengths and shortcomings of the adopted predictive frameworks.

## 2 Mathematical frameworks

### 2.1 MLPR

A concise overview of the single-layer perceptron, the fundamental building block of Multi-Layer Perceptrons (MLPs) [37], is presented to foster understanding. This most straightforward form of a neural network entails a single output neuron connected to all inputs. Each input connection has a corresponding weight, denoted as $w_i$ (where $i$ ranges from 0 to $n$, with n being the total number of inputs). These weights influence the output $(y)$, wrepresentingthe projected binary class. The input values $(x_i)$ correspond to the considered traits or variables [38]. Every input feature value is multiplied by its corresponding weight, $v_i a_i$, during the weighting stage. The second stage then involves adding together these products $(v_0 a_0 + v_1 a_1 + \cdots + v_n a_n)$. The third stage is the transfer stage, where an activation function is applied to the total. denoted as f (or transfer function). This process yields an output, y, expressed as:

$$y = f(z) \; and \; z = \sum_{i=0}^{n} v_i a_i \tag{1}$$

$a_0 = 1$ represents the output $y$, where $a_0$ is the threshold or bias.

Eq. (1) empowers a perceptron, suited for learning functions that a straight line can separate. When dealing with a two-dimensional input featuring two variables, this function is visually represented by a line [39]. The representation shifts to a plane in the context of three dimensions and three traits. However, when dealing with input in n dimensions, the corresponding function is symbolized by a hyperplane. The equation governing the hyperplane is articulated as:

$$\sum_{i=0}^{n} v_i a_i = 0 \tag{2}$$

Eq. (2) represents the dot product of weight vector V and input vector A.

$$V.A = 0 \tag{3}$$

The subsequent stage, referred to as the learning or training stage, is initiated upon obtaining the responses from the input training data. This stage aims to optimize the weights by minimizing a cost function. This function typically measures the squared error between the framework's projected outputs and the real responses. Analytical methods such as gradient descent can compute the optimal weight vector. Convergence of the framework to a solution is imperative for the network to become operational. It also checks the model's generalization capability by validating it with new data. An SLP architecture is created by connecting several perceptrons in parallel, which is used with multiple outputs.

The perceptron and single-layer perceptron have limited ability to solve nonlinearly separable problems. To remove this limitation, multiple layers can be added in series to develop an MLP network. In a neural network, the first and last layers are referred to as the input and output layers, respectively, while all other in-between layers are called the hidden layers. The output from each layer acts as the input for the subsequent layer, and this process goes on iteratively for every additional layer.

As defined in [40], MLP is a feedforward neural network architecture where information flows only in one direction from the input layer to the output layer via the hidden layers. Each link between neurons is weighted, and all neurons in one layer have the same activation function. The hidden layers use the sigmoid function for nonlinear activation, while the output layer could be either sigmoid or linear, depending on the application.

The performance of the MLP model depends on different factors, including the count and arrangement of hidden layers and nodes, the volume of training data, the selection of variables, etc. On the other hand, training parameters, such as the number of cycles, learning rate, momentum governing weight adjustments, and so on, may hugely impact the model's overall performance.

### 2.2 GRO

For the challenge of damage recognition in optimization, researchers implemented GRO. This framework boasts faster convergence compared to traditional methods. Unlike other strategies, GRO is inspired by human reasoning and decision-making. It begins by creating a set of randomly distributed "operators" within the search domain, mimicking the initial search stage in human problem-solving.

During each cycle, these operators move together. When one detects a significant change in a monitored parameter (analogous to increased sound), the entire group pauses to investigate. In addition to their observations, each operator also considers information from others, particularly those detecting stronger "signals." This collaborative approach guides the group towards the area with the most significant readings, ultimately leading to precisely identifying the damage location. Refer to Fig. 1 for the GRO algorithm's flowchart.

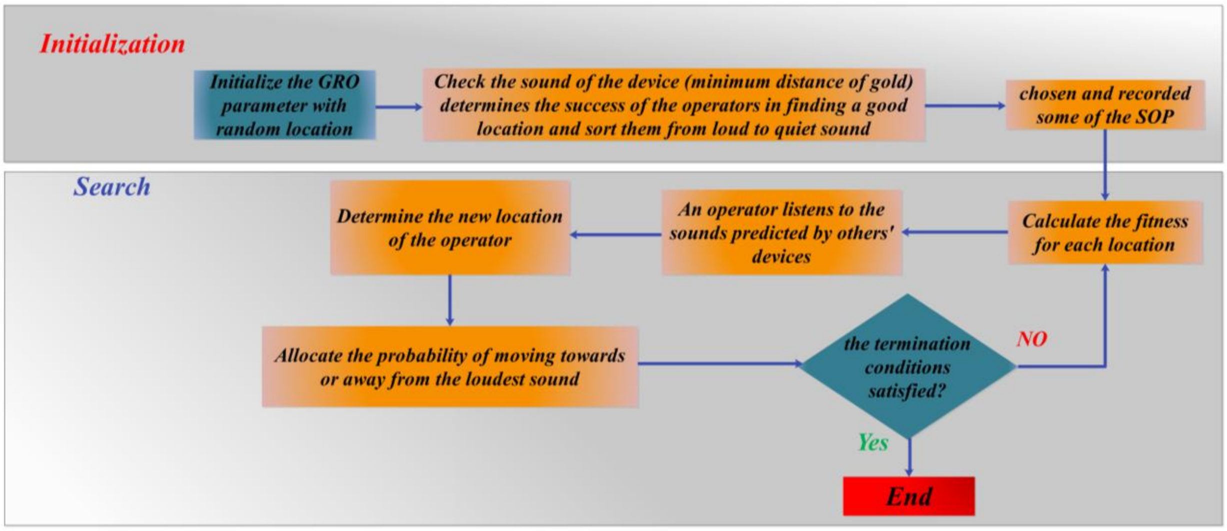

Initialization
Initialize the GRO parameter with random location
Check the sound of the device (minimum distance of gold) determines the success of the operators in finding a good location and sort them from loud to quiet sound
chosen and recorded some of the SOP
Search
Calculate the fitness for each location
An operator listens to the sounds predicted by others' devices
Determine the new location of the operator
Allocate the probability of moving towards or away from the loudest sound
the termination conditions satisfied?
NO
Yes
End

Figure 1: The flowchart of the GRO algorithm

*First stage: initialization*

Eq. (4) illustrates the random allocation of a place for each operator within the search domain. The variable Rand represents a random number within the interval of [0...1], and the search domains for the operators are delineated by $ub_i$ and $lb_i$.

$$location_i^{(0)} = lb_i + (ub_i - lb_i) \times rand, \quad i = 1,2,\dots,n \tag{4}$$

*Second stage: Monitoring involves choosing the best places.*

The SOP is the identifier used to indicate that the operator has successfully identified the ideal location. At the end of every cycle, it is recommended that the top 10% of operators in the SOP be kept.

*Third stage: The link between fitness and distance is called fitness-distance.*

Eq. (5) is used to examine each operator's readings (rate). This calculation helps determine which operator possesses the most promising data for locating the damage.

$$rate(i) = \frac{D_i}{P_i} \times \frac{sound\ (highest\ volume) - sound\ (i)}{(sound\ (highest\ volume) - sound\ (lowestvolume) + \varepsilon)} \tag{5}$$

To reduce mistakes caused by the environment, the coefficients in Eqs. (6-7) are utilized, represented by $D_i$and$P_i$ , respectively.

$$D_i = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + \cdots} \tag{6}$$

$$P_i = 2 - \frac{ite}{max_{ite}} \tag{7}$$

During this stage, each operator makes unique decisions influenced by a combination of sounds, a process encapsulated by Eq. (8).

$$fresh\ place\ (i) = place\ (i) + md \times [(rate(j) - rate(i)) \times (place\ (j) - place\ (i)) \times rand\ ] \tag{8}$$

The $md$ coefficients, delineating the path of movement, are expressed in Eq. (9):

$$md = \begin{cases} +1 \Rightarrow towards\ a\ loudest\ sound & a > rand \\ -1 \Rightarrow away\ from\ a\ loudest\ sound & a < rand \end{cases} \tag{9}$$

*Fourth stage: location correction*

If Eq. (8) is unable to yield a satisfactory location, then the coefficients $\beta$ and $\gamma$ are chosen from the interval $0 < \beta < \gamma < 1$, and Eq. (10) is employed to produce new locations.

$$new\ location\ (i) = \begin{cases} choose\ a\ neighboring\ location & rand < \beta \\ select\ a\ new\ location\ randomly & \beta < rand < \gamma \\ do\ not\ move & \gamma < rand \end{cases} \tag{10}$$

Fifth stage: termination

Iteratively performing stages 4 through 5 in a loop until one of the subsequent circumstances is satisfied:

1. There is no longer a peak count of strives permitted.
2. The ideal place doesn't alter.
3. A predefined anticipated threshold is attained When the disparity between the values of SOP and the global optimum is smaller than a given value.
4. Less than a certain degree of precision separates the objective values of the best and worst sites.

GRO is used in this research to tune the internal parameters of the MLPR model, i.e., the weights and biases, to minimize prediction error in terms of ALE. This optimizes convergence rate and model precision by effectively traversing the search space and escaping local minima, making the process of learning more robust.

## 2.3 POA

Dehghani and Trojovský initiated the POA in 2022, an inventive nature-inspired method that draws influence from pelicans' social behaviors and hunting strategies [41]. Pelicans are huge birds with strong beaks that use a wide neck pouch to catch and eat their prey. These birds comprise the targeted population and typically reside in large groups. Using the following equation, the population members are initialized at random:

$$x_{i,j} = l_j + rand.\left(u_j - l_j\right), \quad i = 1,2,3,\dots,N\ , j = 1,2,3,\dots,m \tag{11}$$

The value of the $jth$ variable for the $ith$ candidate solution is indicated by the formula $x_{i,j}$. The number of people in the group and the overall count of dilemma variables are indicated by the parameters $N$ and $m$, respectively. In addition, $rand$ denotes a randomly generated number within [0.1], and $l_j$ and $u_j$ represent the problem variables' lower and upper restrictions. Eq. (12) is used to create the population matrix, which represents the individuals in the set of possible solutions:

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} X_{1,1} & \cdots & X_{1,j} & \cdots & X_{1,m} \\ \vdots & & \vdots & & \vdots \\ X_{i,1} & \cdots & X_{i,j} & \cdots & X_{i,m} \\ \vdots & & \vdots & & \vdots \\ X_{N,1} & \cdots & X_{N,j} & \cdots & X_{N,m} \end{bmatrix}_{N\times m} \tag{12}$$

The objective function is derived from Eq. (13) as provided.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times m} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1} \tag{13}$$

The objective function vector, $F$, is made up of the objective function values that are allocated to each candidate solution; for example, $F_i$ for the $ith$ candidate solution is the objective function vector. There are two stages to a pelican's hunting process: exploration and exploitation. Pelicans travel forward searching for their prey in the exploration stage and glide smoothly across the water's surface in the exploitation phase. Pelicans locate the prey by using a randomly generated location to approach it during the first exploration stage. The $POA'$s capacity for exploration is enhanced by this random component in the prey's location [42]. Eq. (14) illustrates the first stage's mathematical expression:

$$x_{i,j}^{p_1} = \begin{cases} x_{i,j} + rand.\left(p_j - I.x_{i,j}\right), & F_p < F_i; \\ x_{i,j} + rand.\left(x_{i,j} - p_j\right), & else, \end{cases} \tag{14}$$

Here, $x_{i,j}^{p_1}$ represent the $ith$ pelican's renewed situation in the $jth$ aspect after the primary stage. The update is dependent on two random variables: $F_p$, which represents the prey's objective function value, and $P_j$, which indicates the prey's situation in the $jth$ aspect. A pelican's revised location is considered acceptable in the $POA$ algorithm if it increases the value of the objective function at that specific spot. This process, known as efficient updating, prevents the framework from coinciding to suboptimal locations. This is depicted mathematically as:

$$x_i = \begin{cases} X_i^{p_1}, F_i^{p_1} < F_i \\ X_i \quad else \end{cases} \tag{15}$$

The hunting action is represented mathematically as: The renewed situation of the $ith$ pelican after the second stage is denoted by $X_i^{P_1}$, and the objective function value of the pelican obtained from this stage is indicated by $F_i^{P_1}$. In the second stage, pelicans tn the water's surface lift their wings upward to increase their chances of catching more fish [43]. As a result, they trap the meal in their throat pouches. This stage significantly improves the $POA$ algorithm and makes it easier for improved solutions to converge within the hunting region.

$$x_{i,j}^{p_1} = x_{i,j} + R.\left(1 - \frac{t}{T}\right).(2.rand - 1).x_{i,j} \tag{16}$$

In the second stage, many parameters are taken into consideration to establish the renewed situation of the $ith$ pelican in the $jth$ aspect, which is represented as $X_{i,j}^{P_2}$. Among these is the constant $R$, which has been given a value of 0.2. The phrase $(1 - t/T)$, where $t$ displays the cycle count and $T$ displats the peak count of cycles, affects the neighborhood radius of $x_{i,j}$. At this point, an efficient updating procedure is also implemented, and Eq. (17) is used to determine whether to admit or deny the new pelican situation.

$$x_i = \begin{cases} X_i^{p_2}, F_i^{p_2} < F_i \\ X_i \quad else \end{cases} \tag{17}$$

The renewed status of the $ith$ pelican is represented by $X_i^{P_2}$, and the related objective function value for that pelican is represented by $F_i^{P_2}$. After every population member has been updated, the next cycle and the procedures listed in Eqs start. (14)-(17) are repeated until the execution process is finished [44]. The $POA$ flowchart displayed in Fig. 2 illustrates the iterative nature of this procedure.

The parameters of the MLPR model, i.e., weights and biases, are adjusted using POA to minimize the prediction error of the ALE. During training, by simulating cooperative hunting behavior, POA speeds up the search process. This improves the overall generalization and stability, and ensures predictions are made with higher accuracy in varied network environments.

The hybrid optimization strategy improves localization accuracy by integrating MLPR's potential to capture nonlinear patterns along with GRO and PO's capabilities to perform global searching. These optimizers refine weight tuning as well as convergence, reducing errors during predictions. The synergy provides more accurate ALE estimations in different network scenarios.

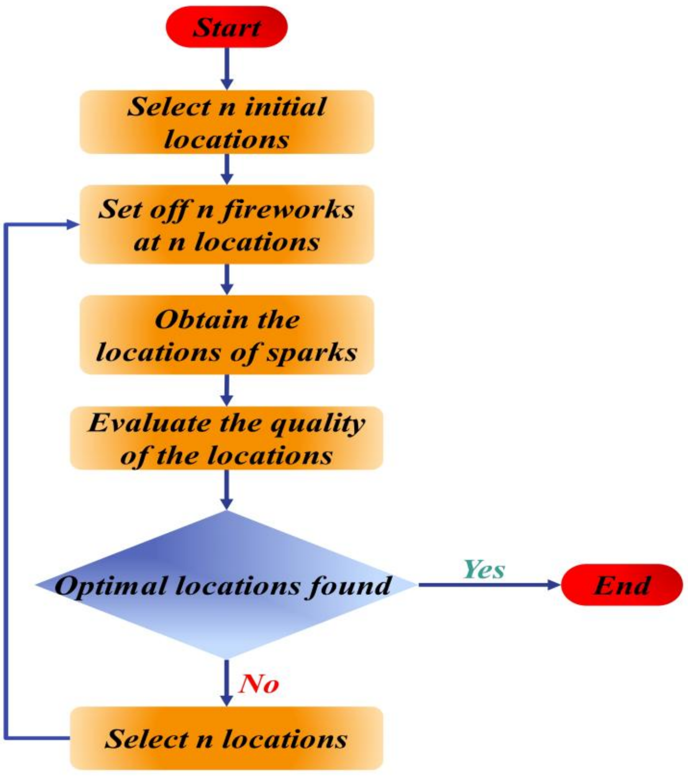

Start
Select n initial locations
Set off n fireworks at n locations
Obtain the locations of sparks
Evaluate the quality of the locations
Optimal locations found
Yes
End
No
Select n locations

Figure 2: The flowchart of $POA$

## 2.4 Performance evaluation metrics

This part organizes diverse gauges to examine the compound frameworks' productivity by measuring their correlation and error levels. The metrics covered encompass RMSE, Index of agreement (IOA), MAE, Coefficient Correlation ($R^2$), and Mean Relative Absolute Error (MRAE). These chosen metrics, RMSE, MAE, IOA, $R^2$, and MRAE, provide a complete assessment of accuracy, consistency, and relative error of predictions. The absolute errors are measured by RMSE and MAE, explained variance by $R^2$, agreement by IOA, and proportional deviation by MRAE, balancing evaluation across model behaviors and sensitivity in errors. The formulas for each of these metrics are provided below.

$$MRAE = \frac{1}{n}\sum_{i=1}^{n}\frac{|m_j - b_j|}{|m_j - \bar{m}|} \tag{18}$$

$$IOA = 1 - \frac{\sum_{i=1}^{n}(x_{i,a} - x_{i,p})^2}{\sum_{i=1}^{n}(|x_{i,p} - \overline{x_a}| + |x_{i,a} - \overline{x_a}|)^2} \tag{19}$$

$$MARE = \frac{1}{N}\sum_{i=1}^{N}\frac{|e_i - m_i|}{m_i} \tag{20}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(m_i - b_i)^2} \tag{21}$$

$$R^2 = \left(\frac{\sum_{i=1}^{n}(b_i - \bar{b})(m_i - \bar{m})}{\sqrt{\left[\sum_{i=1}^{n}(b_i - \bar{b})^2\right]\left[\sum_{i=1}^{n}(m_i - \bar{m})^2\right]}}\right)^2 \tag{22}$$

The factors are displayed below:

- $n$ $display$ the sample magnitude.
- $b_i$ $display$ the projected value.
- $\bar{m}$ and $\bar{b}$ represent the gauged and mean projected values, respectively.
- $m_i$ $display$ the gauged value is.
- The critical value from the t-spread is based on the intended grade of assurance and the degrees of freedom displayed by $t$.
- The mean of the predictor variable in the database is represented by $\bar{x}$.

Metrics of model performance depend heavily on critical hyper parameters like learning rate, hidden layers, and cycles of training. In this research, incorrect tuning resulted in both overfitting or under fitting as evidenced by increased RMSE or decreased $R^2$. Accurate adjustment enhanced generalization, particularly in MLGR. Optimal hyper parameters were determined according to validation performance.

## 3 Data collection

Any machine learning project is based on robust data. This work collected data relevant to indoor localization in WSNs with great care. The data comprised several network parameters that were presumed to affect ALE. To ensure generalizability and prevent overfitting of the model, the data was separated into three distinct parts: training, validation, and testing, in a ratio of 70%, 15%, and 15%, respectively. The training data and the model learned from it took the maximum share of this. The validation data set provided a check, as most hyper parameter tuning to avoid overfitting during training is based on the validation database. Lastly, the unseen testing database verifies the model's generalization to real-life predictive power. This careful data acquisition and segmentation strategy provides a firm ground for an efficient and generalizable ML model in indoor localization. The runtime of the models differed with different configurations. For MLPR with the Gold Rush Optimizer, runtime grew with network depth: 44.3 seconds for 1 layer, 54.7 seconds for 2 layers, and 66.1 seconds for 3 layers. Analogously, the MLPR with the Pelican Optimizer recorded 33.8 seconds for 1 layer, 41.2 seconds for 2 layers, and 52.6 seconds for 3 layers. In comparison with the baseline MLPR without any optimizer, the shortest runtimes were seen at 18.9 seconds for 1 layer, 22.8 seconds for 2 layers, and 30.2 seconds for 3 layers. These results have shown both the optimizers to lengthen computational time, with the Gold Rush Optimizer being the most time-consuming.

## 3.1 Data description

The database used in this exploration is taken from published literature [45], including six important input traits to predict ALE using the recommended machine learning frameworks. This database is a comprehensive collection of several parameters relevant to indoor localization systems, which would be very useful to explore the relationship between these parameters and localization accuracy.

### 3.1.1 Input traits

- Anchor ratio: This feature specifies the number of anchor nodes per overall node in the network. Indeed, anchor nodes are vital parts of an indoor localization environment, in which they always act as reference nodes against which the situation of goal nodes is calculated. Enhanced anchor ratios augment location precision by expanding spatial reference points, facilitating superior triangulation. A reduced number of anchors diminishes geometric constraints, resulting in increased estimating uncertainty and more inaccuracy in expected locations.
- Transrange: It defines the range of transmitters in meters in the wireless communication network. The parameter specifies the maximum possible distance over which nodes may communicate effectively and defines the area of coverage of the whole localization system and its accuracy. The transmission range affects the extent and distribution of the signal. A moderate range guarantees dependable connectivity and constant RSSI readings, hence improving localization precision. Excessive range may induce interference, whilst limited range might lead to signal attenuation and inadequate data transmission.
- Node density: Node density characterizes the spatial distribution of the nodes in the network area and quantifies the number of nodes per unit area. It influences the granularities of the measurement in the localization. Increased node density augments the spatial resolution of the network, hence enhancing ALE by providing more comprehensive environmental data. Sparse networks jeopardize adequate coverage, resulting in increased localization mistakes due to a deficiency of proximate references.
- Cycles: This refers to the number of cycles to execute the ML framework training process. Cycles are critical for improving model parameters to ensure predictive performance. Increased iterations provide improved model convergence during optimization and learning, enhancing localization predictions. A reduced number of cycles may result in underfitting, causing the model to inadequately generalize and elevate ALE.
- SDale: The standard deviation of ALE, which reflects the spread or variability of the localization error in different instances or cases. This metric provides insight into the consistency and reliability of the localization system. The standard deviation of ALE indicates the stability and consistency of localization performance. A lower standard deviation indicates strong and dependable predictions, whereas more variability implies the model is susceptible to fluctuations in input properties or environmental conditions.
- Ale: ALE is the target variable that the machine learning frameworks should predict. ALE quantifies the accuracy of localization, which is gauged by the deviation of the estimated situation from the actual situation of goal nodes.

## 3.2 Data preprocessing

Before framework training, the database endured preprocessing stages to ensure data quality and compatibility with the machine learning frameworks. This included managing missing values, regulating statistical traits, and encoding explicit variables as necessary.

## 3.3 Database characteristics

The database comprises 107 instances, each characterized by the input traits above and corresponding ALE values. The layered pattern facilitates fine-grained evaluation over various data complexities or network settings. Each layer has a different scenario or level of abstraction, allowing model robustness and generalization to be tested. Such a structure provides insight into where specifically a model performs well or fails, charting future areas of improvement in ALE prediction. Table 2 provides a summary of descriptive statistics, including average, standard deviation, minimum, and maximum, for each feature. Additionally, Fig. 3 presents the bar chart of the input variables.

Table 2: The statistical traits of the input of ALE

| Variables | Indicators | | | |
|---|---|---|---|---|
| | Min | Max | Avg | St. Dev. |
| Anchor_ratio | 10.000 | 30.000 | 20.523 | 6.708 |
| Trans_range | 12.000 | 25.000 | 17.879 | 3.093 |
| Node_density | 100.000 | 300.000 | 159.813 | 70.856 |
| Cycles | 14.000 | 100.000 | 47.888 | 24.553 |
| Sd_ale | 0.003 | 1.092 | 0.266 | 0.183 |
| Ale | 0.394 | 2.268 | 0.981 | 0.396 |

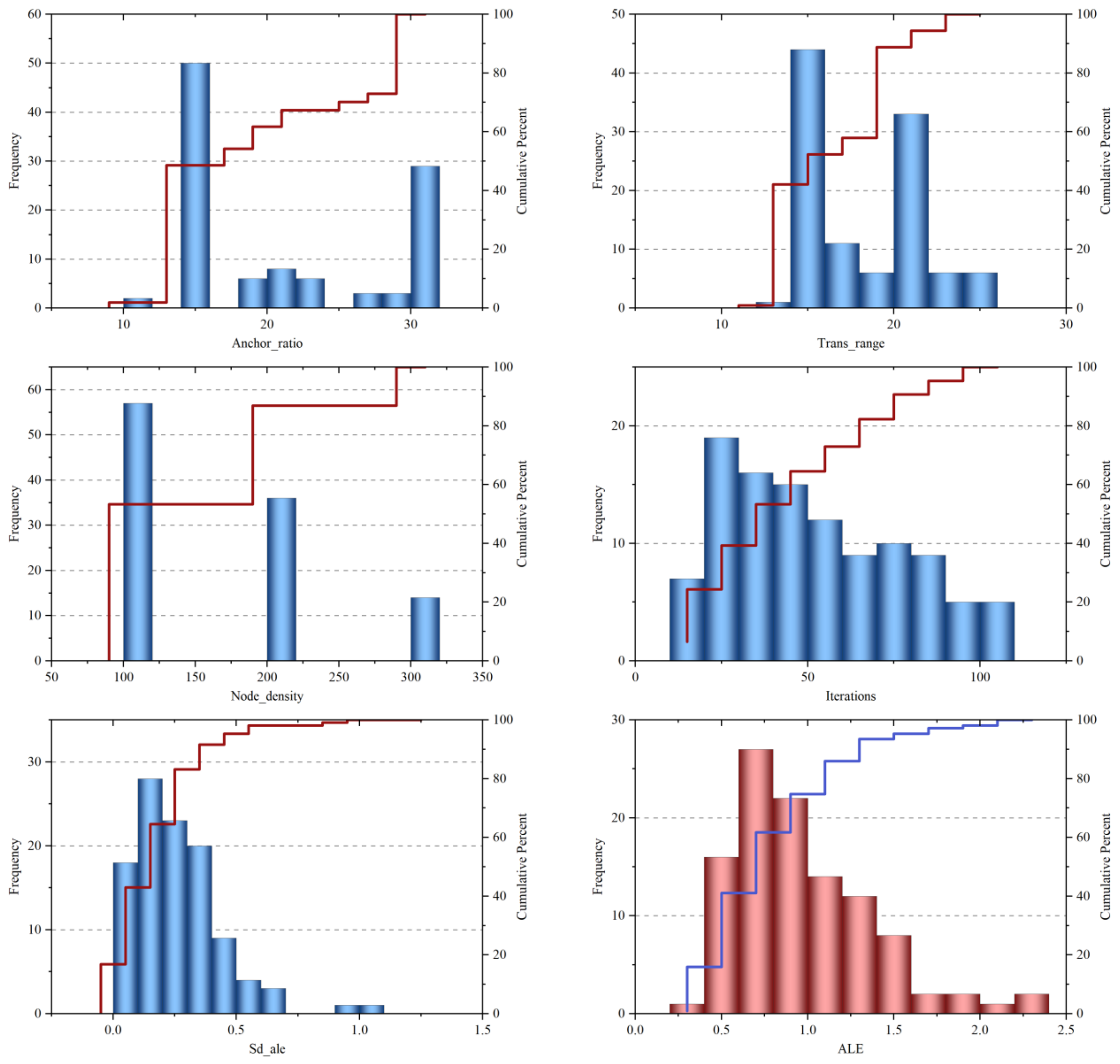


Figure 3: The bar chart for input and output

## 4 Outcomes and consultation

### 4.1 Hyper-parameter tuning

Table 3 depicts the optimum number of neurons obtained through random search for each hidden layer at three test stages for MLGR and MLPO models. MLGR and MLPO represent the Multi-Layer Perceptrons augmented with Gold Rush and Pelican Optimizers, respectively. Each model setup (1, 2, 3) represents one different experimental condition, and each row represents one hidden layer. It is clear from the results that neuron distribution is different across different layers and optimizers, highlighting the adaptivity of the tuning process. MLGR tended to utilize a greater number of neurons in deeper layers than MLPO, potentially impacting learning capability and converging behavior. This layer-by-layer tuning underpins the differences in performance seen in later tests.

Table 3: Optimal number of neurons determined via random search for each hidden layer

| **Hyperparameters** | Models | | | | | |
|---|---|---|---|---|---|---|
| | MLGR (1) | MLPO (1) | MLGR (2) | MLPO (2) | MLGR (3) | MLPO (3) |
| Neuron | 36 | 28 | 37 | 29 | 30 | 25 |
| Neuron | - | - | 33 | 20 | 20 | 26 |
| Neuron | - | - | - | - | 23 | 12 |

## 4.2 Coincidence curvature

Fig. 4 displays a graph that illustrates the time evolution of a repeating enhancement tactic. It shows how the objective function value of the framework alters at every cycle and whether the process is converging toward the best reaction. Fig. 4 shows the convergence behavior of optimization algorithms in different layers. The curves of the objective function evolution in training cycles are depicted. The steepness and earlier flattening of curves signify more efficient convergence and optimization. The MLGR model shows superior convergence properties by reaching a lower objective in fewer cycles, as is evident in its superior performance in the following prediction assessment. When it comes to optimization problems, an algorithm getting close to convergence exhibits a steady decrease or rise in the objective function until it attains a point where more cycles only produce negligible gains. The chart indicates the excellent productivity of the MLGR model over MLPO across all three layers. After 50 cycles, the MLGR model achieves a commendable accuracy of 0.0345, highlighting its excellence in the first layer. In the second layer, the MLGR model attains a high accuracy of 0.0315 after 30 cycles, and in the third layer, it reaches 0.0294 accuracy after 40 cycles. However, even after 60 cycles, the MLPO model demonstrates noticeable improvement in precision across all layers, though it may not have reached the same level of performance as the MLGR model.

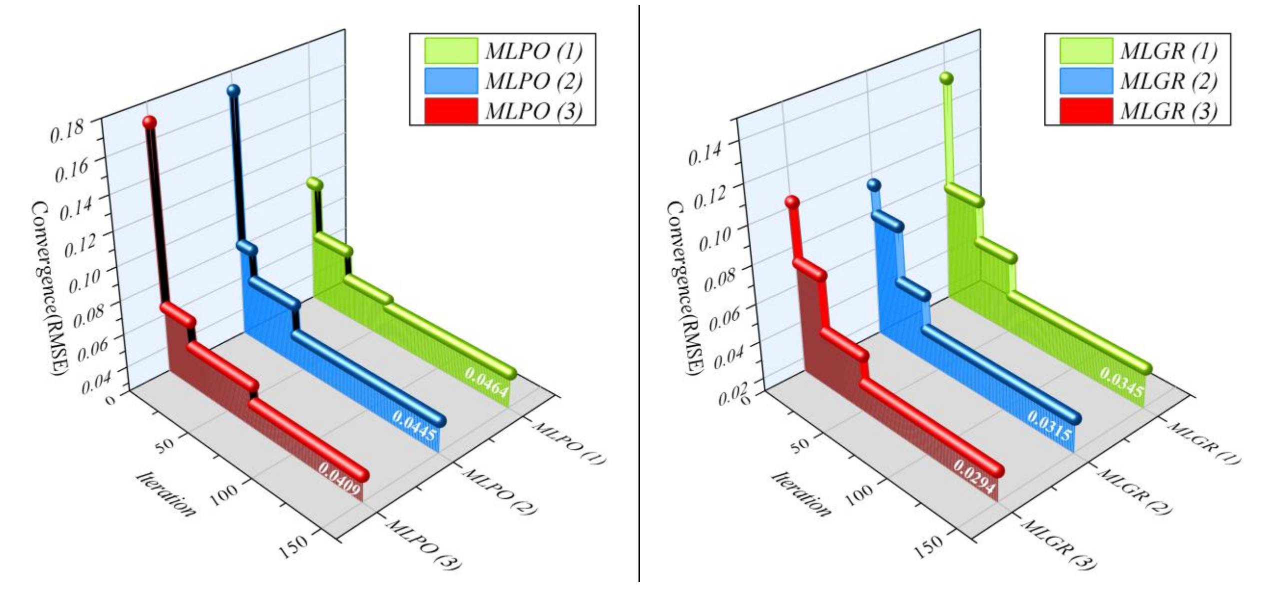


Figure 4: 3D Scatter plot for convergence of compound frameworks

## 4.3 Frameworks comparison

Table 4 displays the outcomes of the built MLPR frameworks, comparing three different stages and five different metrics. The metrics consist of RMSE, R2, MAE, IOA, and MRAE, and the stages include training, validation, and testing. When a framework's R2 and IOA values are close to one, it performs at its best. On the other hand, values close to zero indicate better model performance for RMSE, MAE, and MRAE.

For instance, in the first layer during training, the MLGR model performs better than the MLPR and MLPO frameworks regarding RMSE, with a value of 0.024. Moreover, with an R2 value of 0.996, the MLGR model beats the MLPO and MLPR frameworks in the same layer and stage. During the training stage, the MLPO model achieved second place with an IOA value of 0.997. Among the three frameworks, the MLPR model has the least promising performance, outperforming the other two.

With a value of 0.040, the MLGR model displays the best performance in MAE during the second layer's validation stage. The MLPO model achieved the second-best performance and is not far behind. The MLPR model performs third with a value of 0.347 in terms of MRAE during the test stage, while the MLPO model comes in second with a value of 0.306. Furthermore, regarding the R2 value during validation, MLGR operates better than MLPO and MLPR, with a value of 0.987.

The test stage's third layer displays that, at 0.991, the MLPR model performs the least robustly in IOA, while MLGR, the best-performing model, achieves a superior value of 0.998. The results for the MAE value in the training stage indicate that MLGR outperforms the others, achieving a value of 0.016. The MLPO model, with a value of 0.030, comes in second place and is followed closely. With values of 0.046 and 0.064, respectively, the MLPO model displays the best performance in the validation stage of RMSE, while MLPR displays the lowest performance. As presented in Table 4, the MLPR model's worst performance occurs in the third layer in the testing phase with an RMSE of 0.080. This suggests a poorer generalization ability in this case than in MLGR and MLPO, given the sensitivity of MLPR to differences in data complexity by layers.

Performance differences between layers are a result of variations in feature distributions as well as data complexity levels. Noisy patterns or less informative patterns in a couple of layers might compromise model generalization. Also, uneven training dynamics lead to weaker tuning of parameters in a particular layer, resulting in increased error rates.

Table 4: The outcome of the created frameworks for the MLPR-based hybrid scheme

| Model | Stage | Index values | | | | |
|---|---|---|---|---|---|---|
| | | RMSE | R2 | MAE | IOA | MRAE |
| MLPR (1) | Train | 0.054 | 0.981 | 0.046 | 0.995 | 0.361 |
| | Validation | 0.056 | 0.981 | 0.048 | 0.995 | 0.409 |
| | Test | 0.080 | 0.965 | 0.074 | 0.990 | 0.408 |
| MLPO (1) | Train | 0.041 | 0.989 | 0.035 | 0.997 | 0.378 |
| | Validation | 0.061 | 0.983 | 0.050 | 0.995 | 0.447 |
| | Test | 0.055 | 0.984 | 0.047 | 0.996 | 0.270 |
| MLGR (1) | Train | 0.024 | 0.996 | 0.022 | 0.999 | 0.300 |
| | Validation | 0.044 | 0.988 | 0.039 | 0.997 | 0.339 |
| | Test | 0.057 | 0.984 | 0.051 | 0.995 | 0.301 |
| MLPR (2) | Train | 0.053 | 0.982 | 0.047 | 0.995 | 0.562 |
| | Validation | 0.081 | 0.983 | 0.070 | 0.991 | 0.490 |
| | Test | 0.066 | 0.975 | 0.056 | 0.994 | 0.347 |
| MLPO (2) | Train | 0.033 | 0.993 | 0.028 | 0.998 | 0.353 |
| | Validation | 0.062 | 0.983 | 0.057 | 0.994 | 0.492 |
| | Test | 0.066 | 0.975 | 0.060 | 0.994 | 0.306 |
| MLGR (2) | Train | 0.023 | 0.996 | 0.020 | 0.999 | 0.154 |
| | Validation | 0.046 | 0.987 | 0.040 | 0.997 | 0.366 |
| | Test | 0.045 | 0.989 | 0.042 | 0.997 | 0.237 |
| MLPR (3) | Train | 0.056 | 0.981 | 0.047 | 0.995 | 0.630 |
| | Validation | 0.064 | 0.978 | 0.049 | 0.994 | 0.312 |
| | Test | 0.080 | 0.966 | 0.071 | 0.991 | 0.372 |
| MLPO (3) | Train | 0.035 | 0.992 | 0.030 | 0.998 | 0.380 |
| | Validation | 0.046 | 0.988 | 0.041 | 0.997 | 0.363 |
| | Test | 0.059 | 0.981 | 0.051 | 0.995 | 0.298 |
| MLGR (3) | Train | 0.018 | 0.998 | 0.016 | 0.999 | 0.176 |
| | Validation | 0.054 | 0.984 | 0.050 | 0.996 | 0.393 |
| | Test | 0.036 | 0.993 | 0.029 | 0.998 | 0.166 |

Table 5 presents the statistical significance of comparisons of model performance using the Mann–Whitney U test for three MLP setups. In the stat column, U test statistics are presented, and in the P value column, the corresponding significance levels are provided. The baseline MLPR, MLPR-GRO, and MLPR-POA are compared. With the majority of the p-values lying above the threshold of 0.05 significance, MLPR-POA is just short of significance in MLP 1 and MLP 3 (p = 0.051), demonstrating potentially significant improvement over the baseline. This finding complements the performance metrics by providing statistical support for the trends noted and confirming the validity of optimizer-driven improvements in MLPR setups.

Table 5: statistical significance results of the model performances across three multi-layer perceptron

| *Models* | *MLP 1* | | *MLP 2* | | *MLP 3* | |
|---|---|---|---|---|---|---|
| | *stat* | *P value* | *stat* | *P value* | *stat* | *P value* |
| *MLGR* | 2522 | 0.254 | 2845 | 0.890 | 2522 | 0.254 |
| *MLPO* | 2261 | 0.051 | 2442 | 0.164 | 2261 | 0.051 |
| *MLPR* | 2673 | 0.607 | 2457 | 0.179 | 2673 | 0.607 |

Fig. 5 displays the Plotting of the spread of expanded compound frameworks for the first layer. The expected value is represented on the Y axis, while the gauge value is depicted on the X axis. A tight grouping of data points near the middle diagonal line, corresponding to the ideal R² value of 1 represents good predictive accuracy. This grouping indicates little deviation between the actual and produced ALE values. Whenever a model's scatter plot resembles clustering near the line in such a way that the model is likely to be picking up the patterns in the available data consistently. Thus, the data points' nearness to the line indicates the model's performance quality and forecasting accuracy in a straightforward way.

Different parts are color-coded to enhance clarity, as illustrated in Fig. 5. When the population falls below the center line, it signifies underestimation, whereas above the core line indicates overestimation. Robust model execution is evident when the linear line closely aligns with the center line and displays no noticeable angle between them. The diagrams in Fig. 5 show that the MLGR model excels, contrasting with the MLPO and MLPR frameworks. Following the MLGR model, the MLPR model demonstrates the second-highest level of performance. The scatter plot verifies model excellence by illustrating the similarity of the predicted values to actual values on the diagonal line. A model that has points closely grouped on that line indicates higher accuracy and lower residual error levels. MLGR plots demonstrate tight grouping, which indicates higher prediction consistency than MLPO and MLPR.

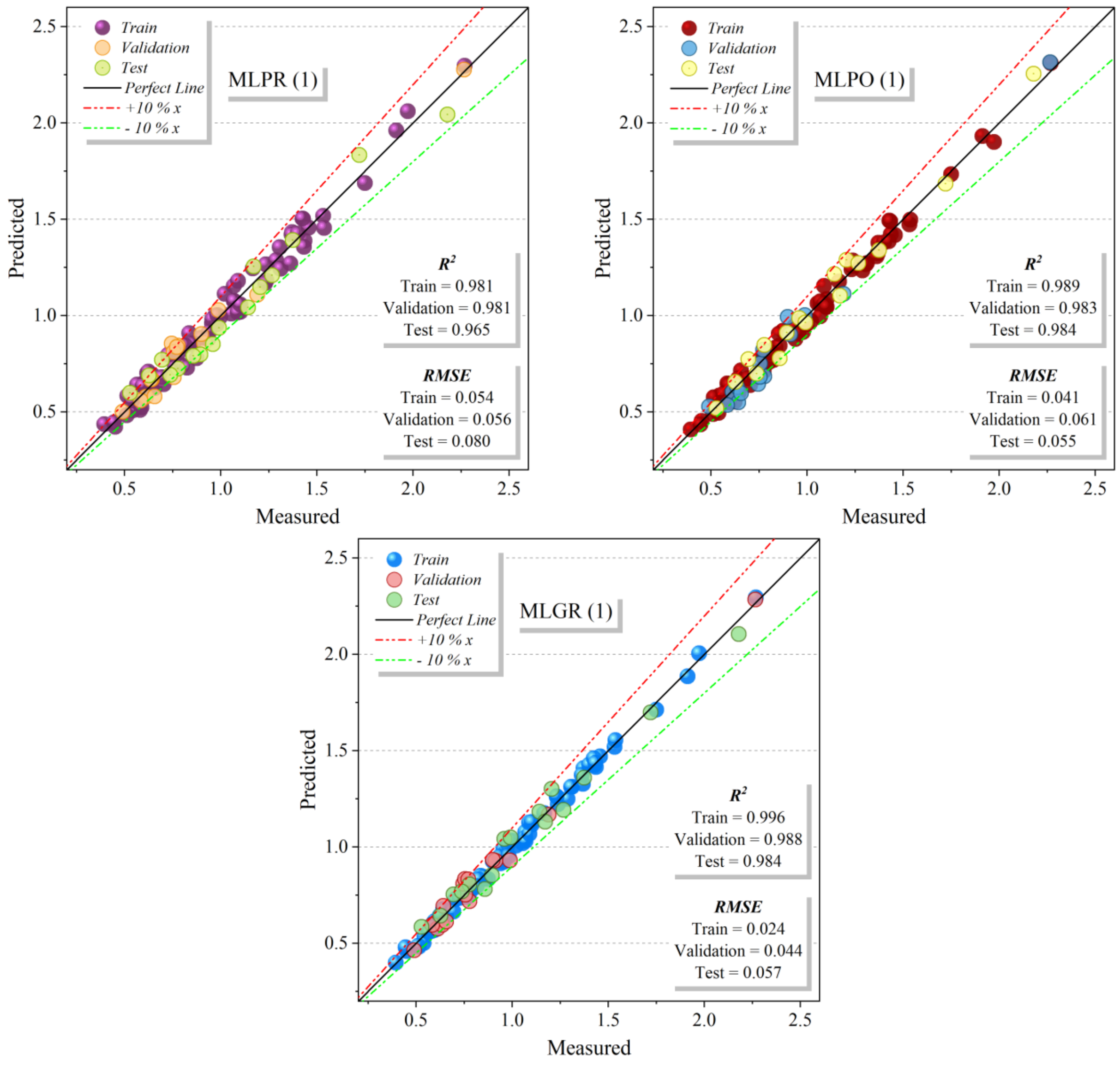


Figure 5: Plotting the spread of expanded compound frameworks

Fig. 6 plots the spread of evolved compound frameworks for the second layer. The expected value is represented on the Y axis, while the gauge value is depicted on the X axis. The population surrounding the middle line, denoting $R^2$, becomes filled to indicate that the framework with optimal execution is at the core. Different portions are color-coded to enhance clarity, as illustrated in Fig. 6. When the population falls below the center line, it signifies underestimation, whereas above the center line indicates overestimation. A strong model performance is evident when the linear line closely aligns with the center line and displays no noticeable angle between them. The diagrams in Fig. 6 show that the MLGR model excels, contrasting with the MLPO and MLPR frameworks. Following the MLGR model, the MLPR model demonstrates the second-highest level of performance.

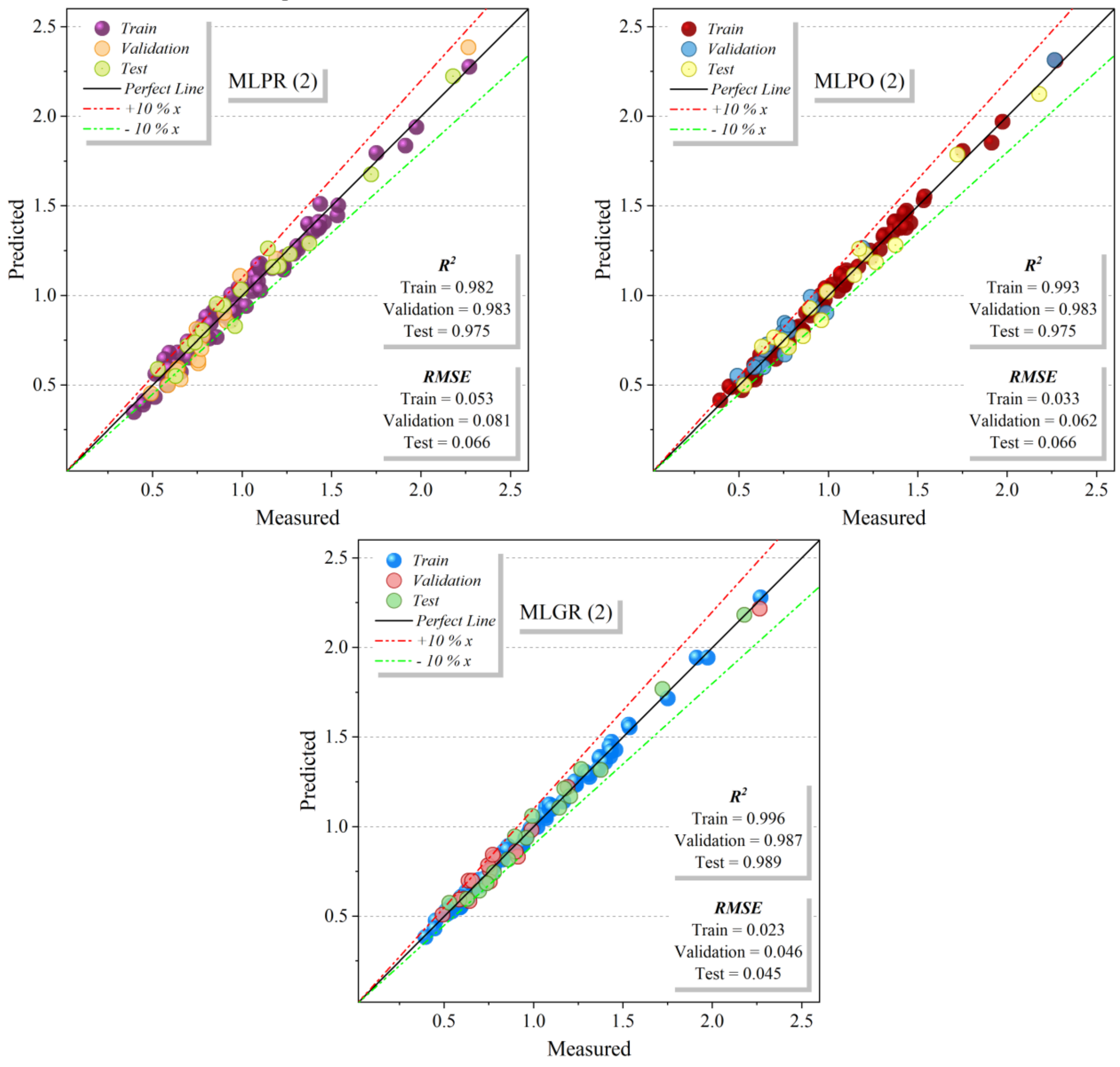


Figure 6: Plotting the spread of expanded compound frameworks

Fig. 7 plots the spread of evolved compound frameworks for the third layer. The expected value is represented on the Y axis, while the gauge value is depicted on the X axis. The population surrounding the middle line, denoting $R^2$, becomes filled to indicate that the framework with optimal execution is situated at the center. Different parts are color-coded to enhance clarity, as illustrated in Fig. 7. When the population falls below the center line, it signifies underestimation, whereas above the center line indicates overestimation. Robust model execution is evident when the linear line closely aligns with the center line and displays no noticeable angle between them. The diagrams in Fig. 7 show that the MLGR model excels, contrasting with the MLPO and MLPR frameworks. Following the MLGR model, the MLPR model demonstrates the second-highest level of performance.

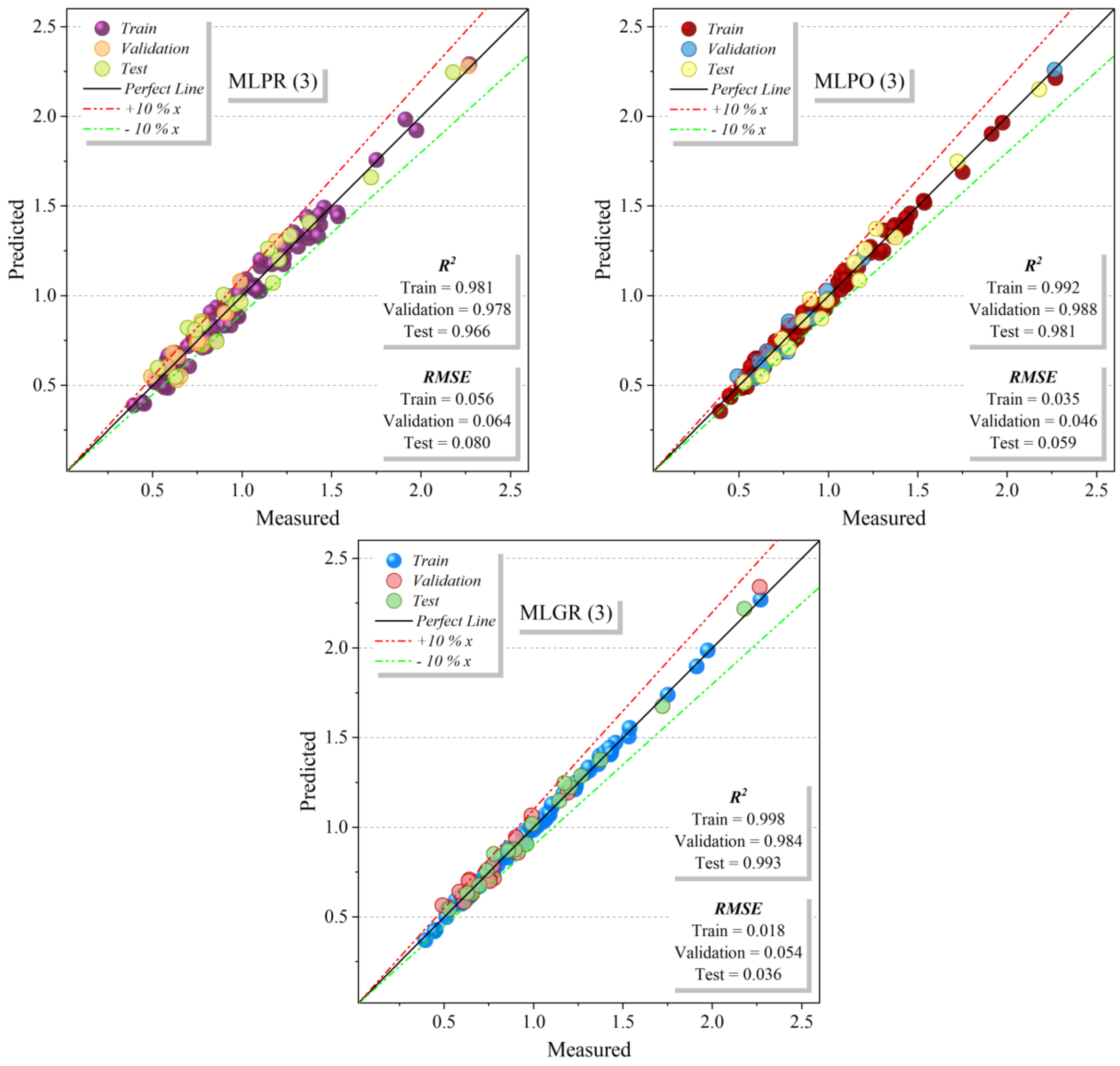
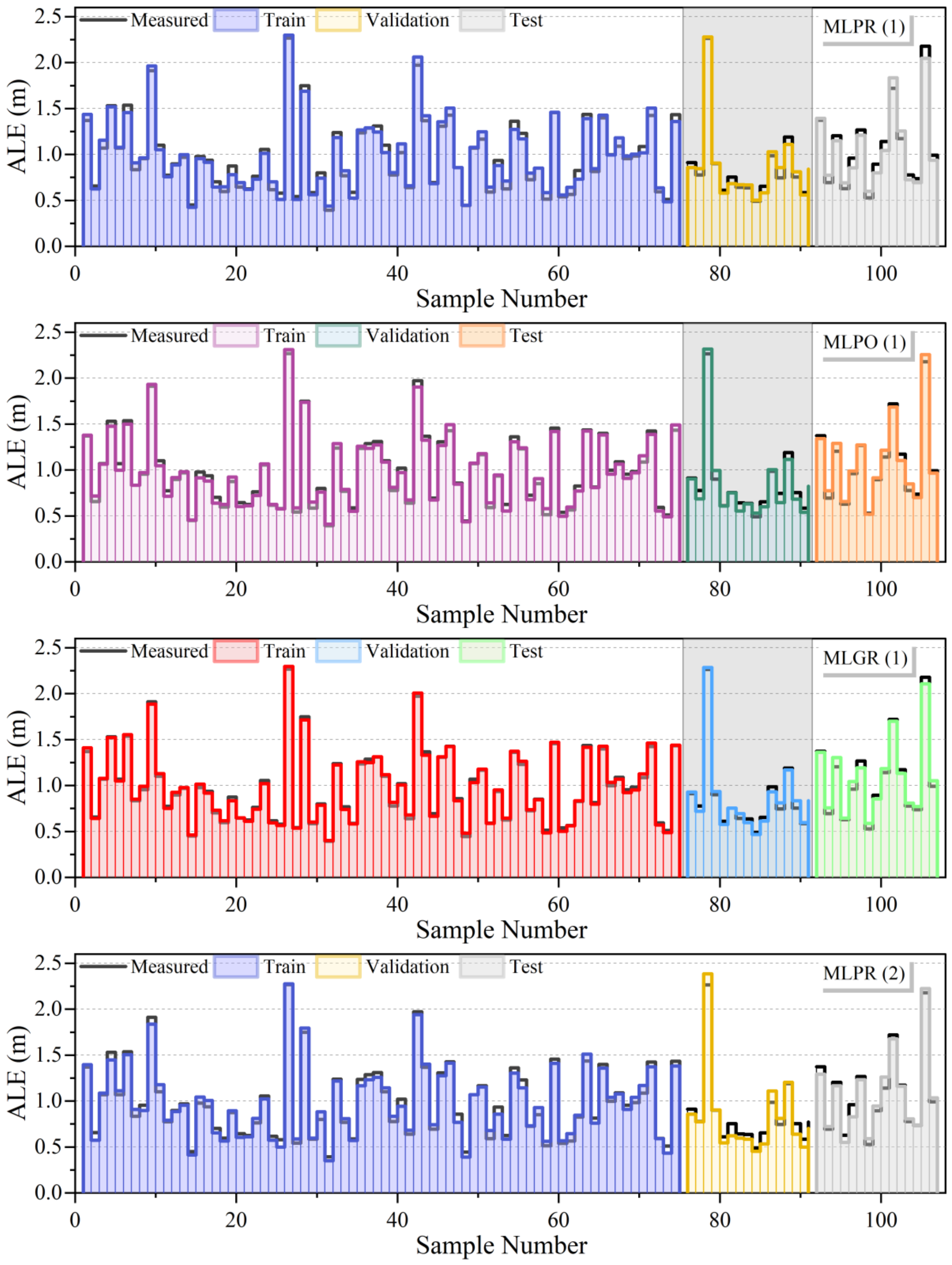


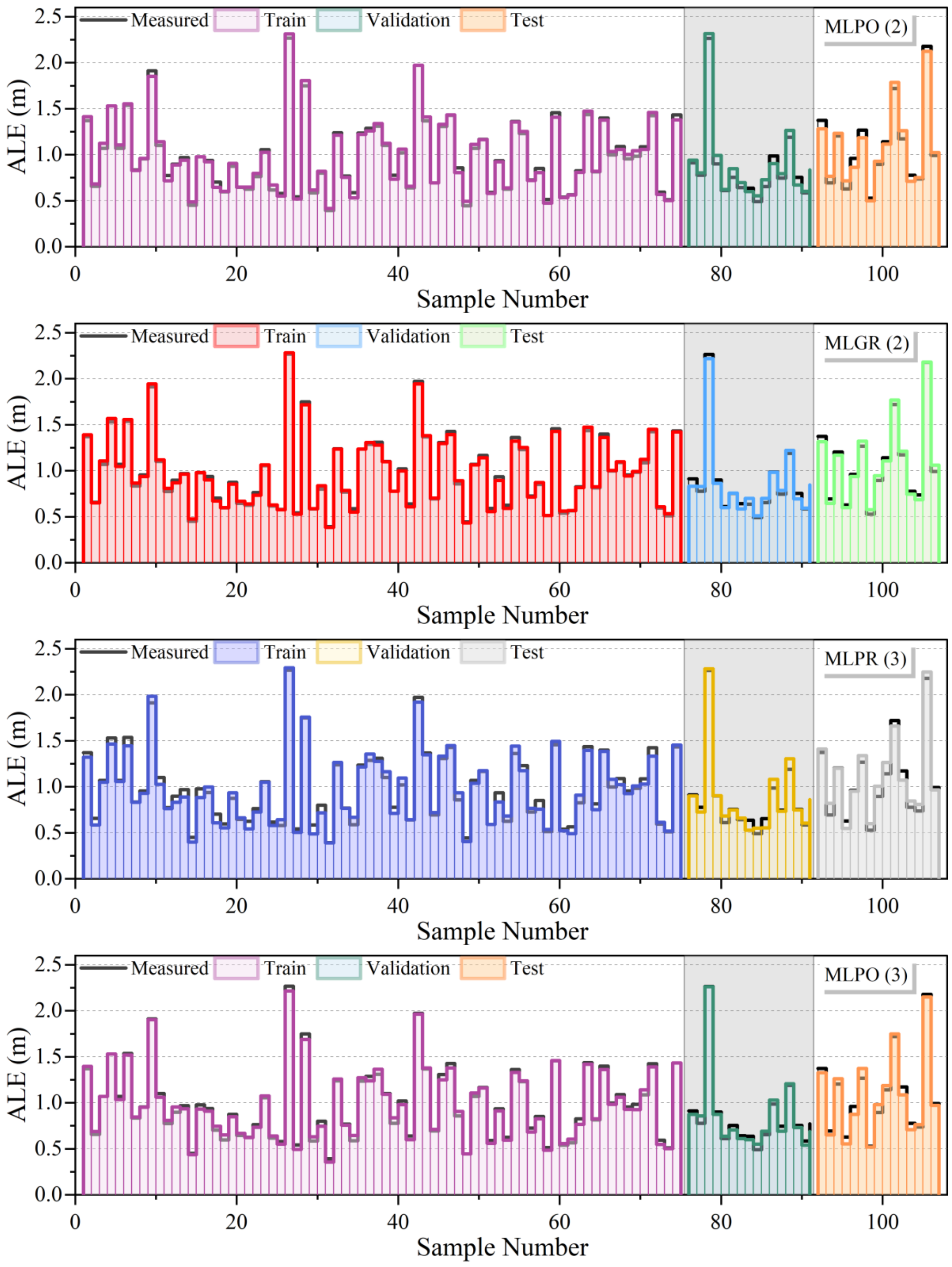
Figure 7: Plotting the spread of evolved compound frameworks

Fig. 8 displays the contrast of projected and gauged values. The alignment between the gauged line and the projected columns indicates the framework's anticipation precision. A close correspondence between the gauged line and the expected column reflects a peak level of precision, while deviations suggest a lower level of performance.

In layer one, the MLGR model demonstrated strong performance in the training stage, with minimal gauged lines exhibiting distances more significant than the projected column. The validation segment of this framework outperforms the test section, showcasing a close alignment between the MLGR framework's anticipations and the observed data. MLPO exhibited poorer performance in the training stage than the MLGR model, with fewer projected columns with distances from gauged lines. However, in both the test and validation parts, MLPO performed excellently.

In the test stage, the MLPR model performs less effectively than the MLPO and MLGR frameworks, with a notable gap between the gauged line and the expected column. Nevertheless, the MLPR model performs satisfactorily in the validation and training stages. This pattern persists across all layers.


Measured
Train
Validation
Test
MLPR (1)
MLPO (1)
MLGR (1)
MLPR (2)
ALE (m)
Sample Number


Measured
Train
Validation
Test
MLPO (2)
MLGR (2)
MLPR (3)
MLPO (3)
ALE (m)
Sample Number
0.0
0.5
1.0
1.5
2.0
2.5
0
20
40
60
80
100

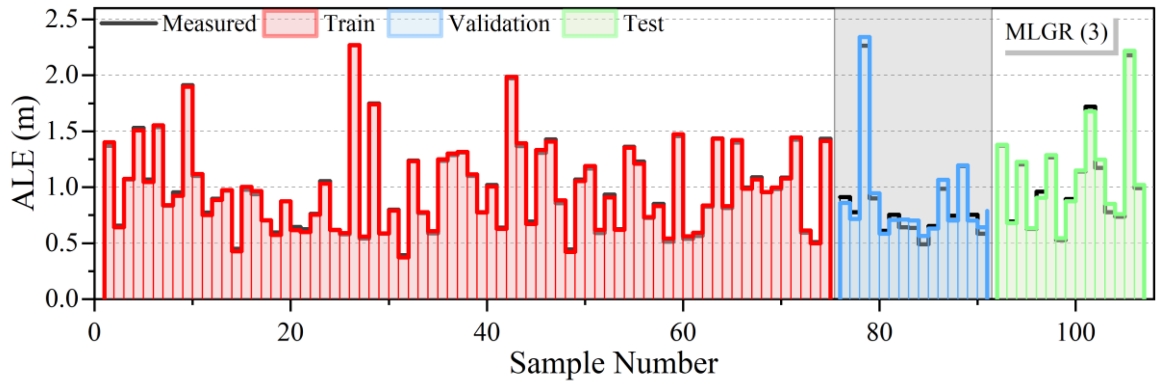
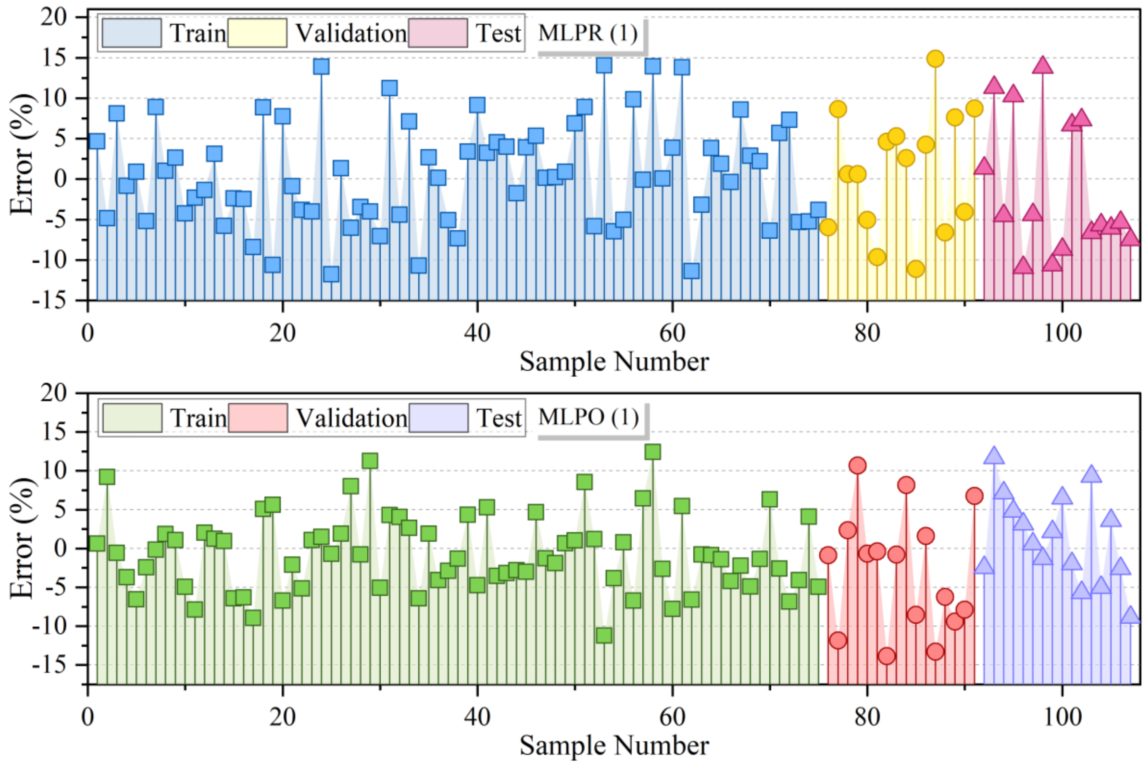


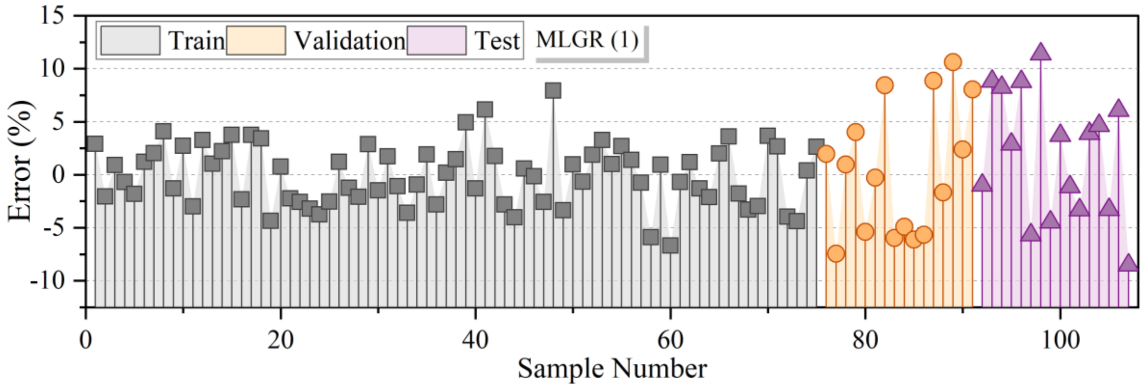
Figure 8: The contrast between projected and gauged values

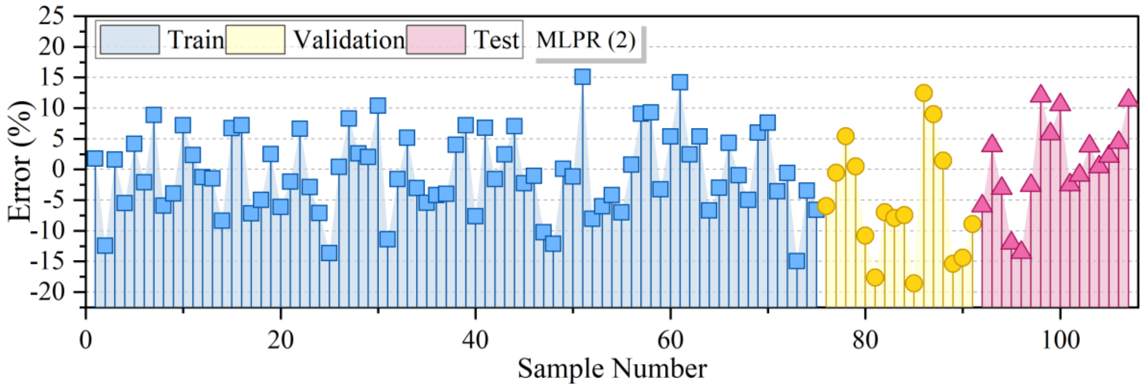
Fig. 9 illustrates the error percentages of the frameworks based on the Scatter plot. Optimal model performance is indicated when the error proportion approaches zero. For instance, in layer one of MLPR, the peak error proportion during the training stage is approximately 14%. Compared to the other two frameworks, it displays the peak error proportion, reaching about 16%.

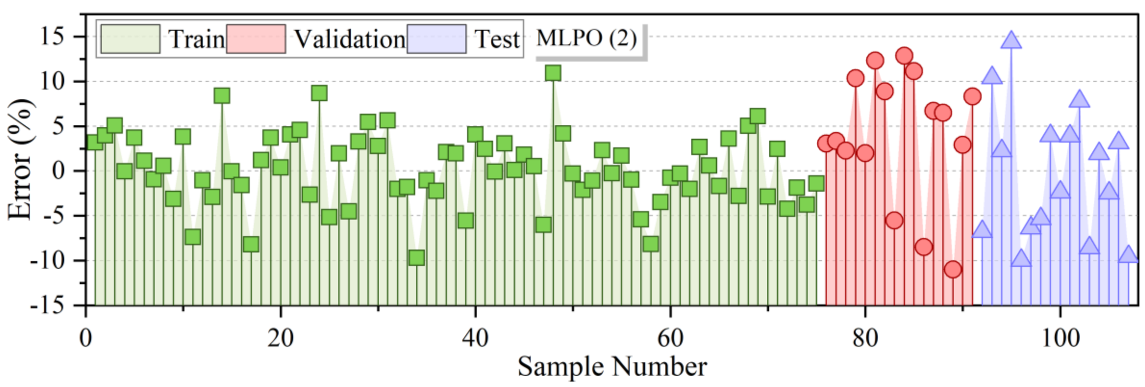
The highest error proportion in MLPO is around 14%, lower than the error proportion in MLPR. The training stage of MLPO entails the peak error proportion, and in this exploration, this framework operates the second-best. In contrast, the peak error proportion in MLGR is 12%, the lowest among the three frameworks. The test stage of this framework displays the highest fault proportion, while the validation stage exhibits the lowest error proportion.

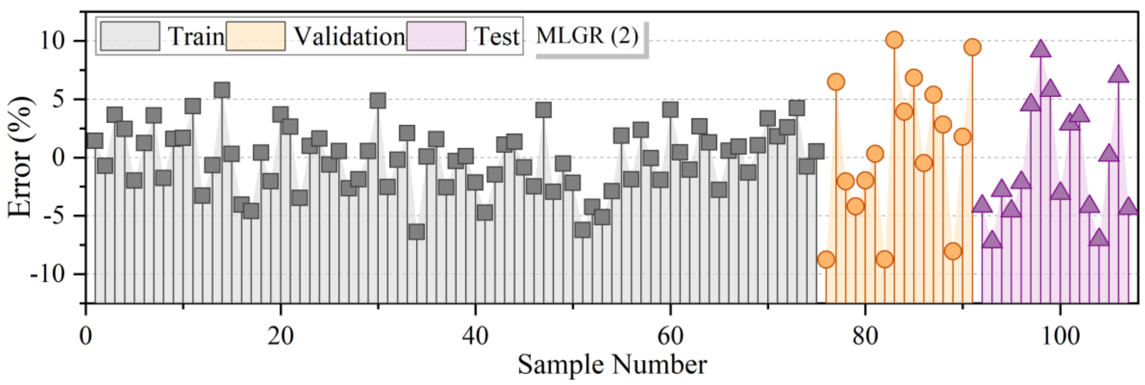
Moving to layer two in MLPR, the peak error proportion is approximately 17%, occurring in the training stage. The validation stage has a lower error proportion than the training and test stages. The error proportion varies from -18 to 17 in MLPR. In MLPO, 15% is the highest error proportion, which takes place in the test stage and is the second best during this study. At layer three, the MLGR model has a maximum of about 17% error proportions in the validation stage, while the MLPO model here performs the best in the meantime with an error proportion of 13%. Distributions of error percentage reflect the stability and consistency of predictive performance of the models. Tighter distributions with lower peak error reflect stable performance, whereas wider spreads reflect variable performance and overfitting and under fitting risks. In the research conducted here, MLGR consistently has tighter and lower error distributions, reiterating its stability over layers against MLPO and MLPR.




Train
Validation
Test
MLGR (1)
Error (%)
Sample Number


Train
Validation
Test
MLPR (2)
Error (%)
Sample Number


Train
Validation
Test
MLPO (2)
Error (%)
Sample Number


Train
Validation
Test
MLGR (2)
Error (%)
Sample Number

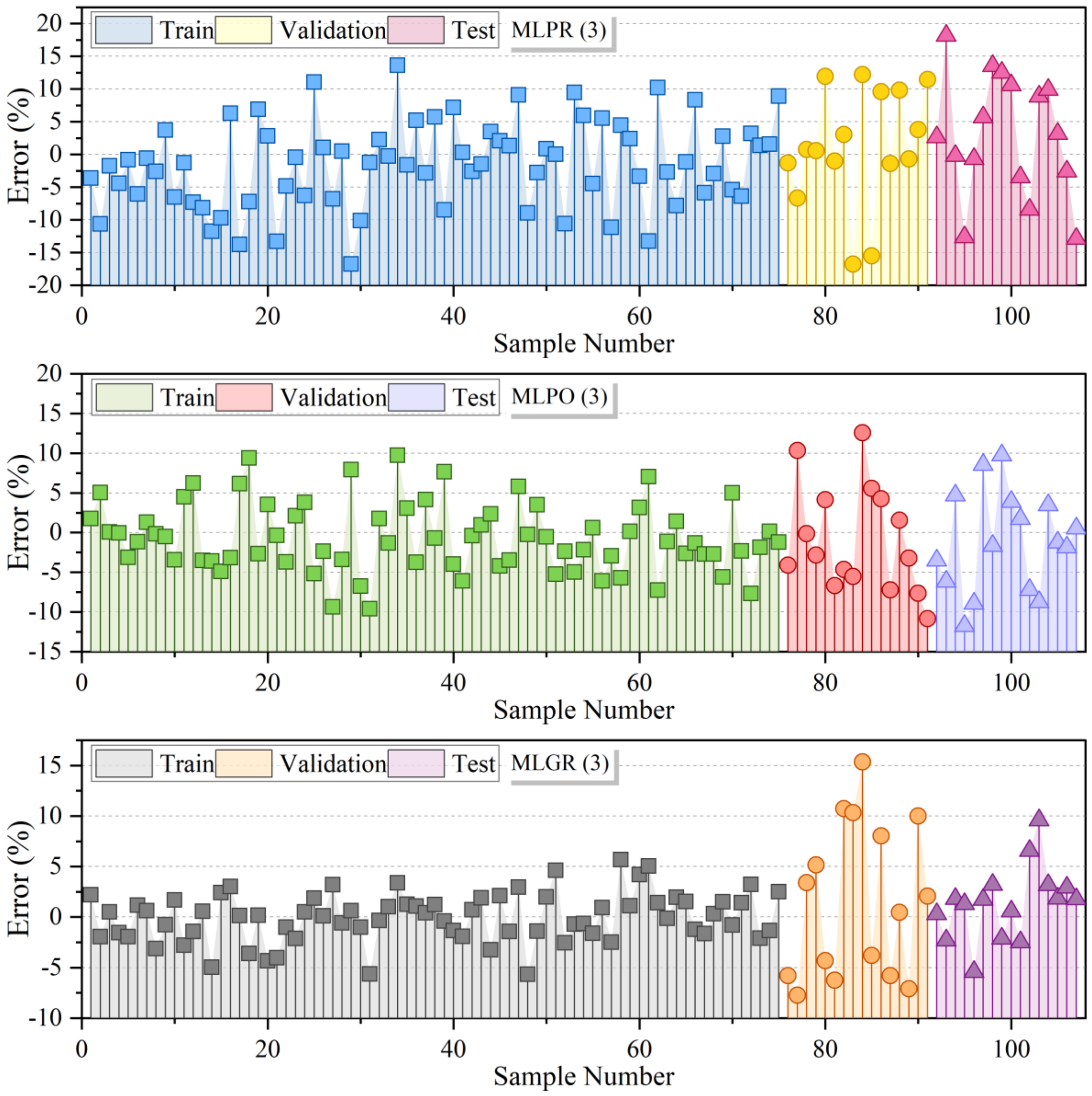


Figure 9: The error proportion of the frameworks in the scatter plot

Fig. 10 displays the half-box plot errors for the recommended frameworks so that their performance attributes can be visually represented. When the median line is closer to zero, it indicates a smaller error proportion in a model. Additionally, the data distribution within the IQR displays the model's effectiveness; the denser the population within this range, the better the efficacy. Looking at the MLGR model in the third layer of the training stage, one finds that the population is indeed closely clustered around the range of IQR. This close grouping around the IQR range represents the model's performance strength. The validation stage confirms this by showing that the median line of this framework hovers nearer to zero.

Conversely, MLPO is widespread in the IQR range, with the median line keeping a distance from zero, especially during the test stage. While the training stage displays a moderately concentrated population around the IQR range, the wider IQR indicates a potential spread in the model's performance. In the case of the MLPR model, the median line approaches zero during the test stage, which means lower error. However, the data distribution around the IQR range is less concentrated, with one data point diverging from this range, implying a nuanced performance in this stage.

This framework reveals a population at the training stage so closely clustered around the IQR range in the second layer of MLGR. In the test stage, the median line has a distance from zero, which indicates that this framework performed poorly at this stage.

Conversely, the MLPO model performs weaker in the validation stage than the MLGR model. The median line's distance from zero is too great, as in the test stage. However, this framework's performance is acceptable in the training stage. The MLPR model performs the weakest in all stages compared to these two frameworks. The population around the IQR range is not clustered, and the median line is too far from the zero point.

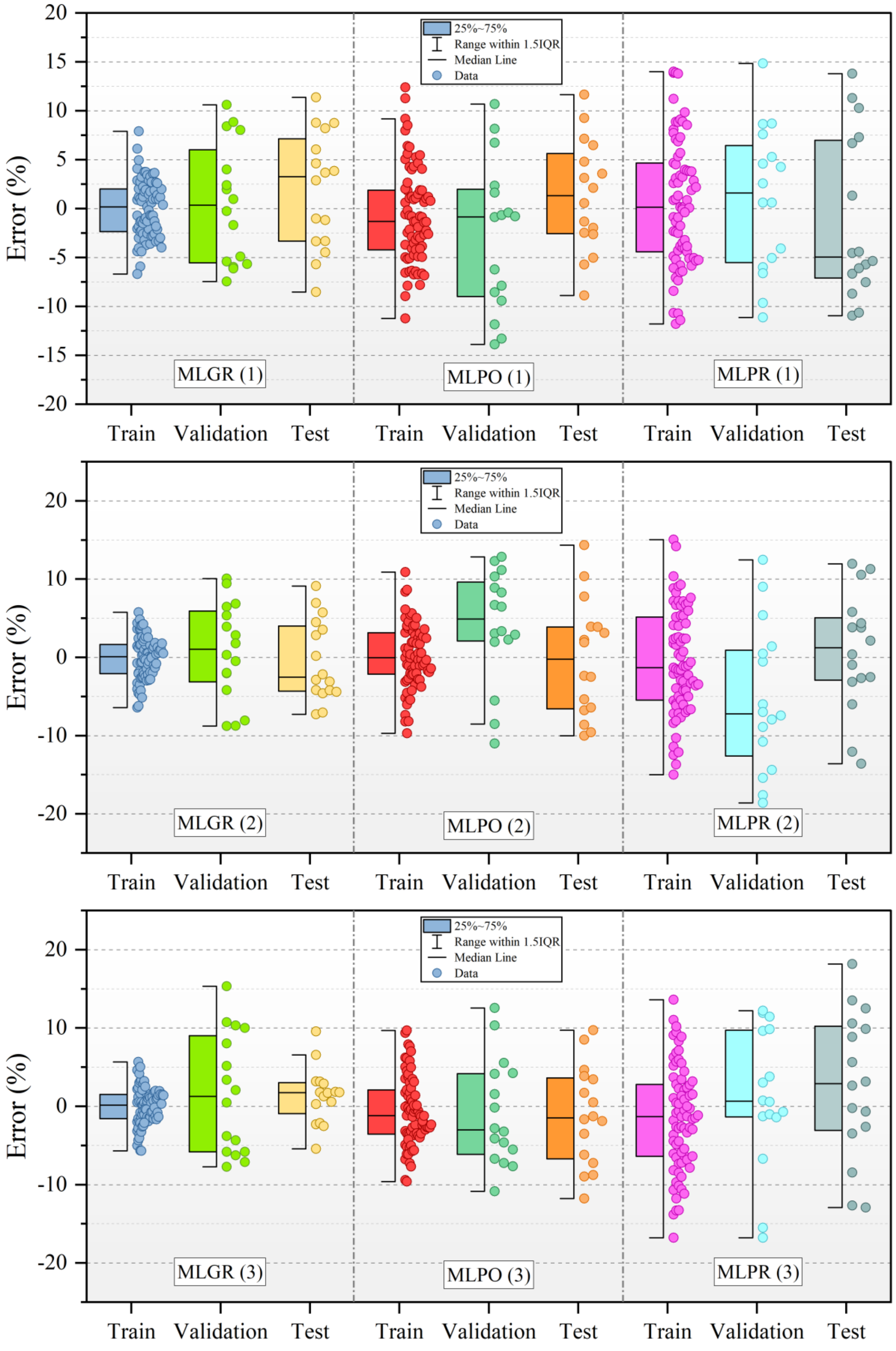


Figure 10: The half box has symbol plot errors for recommended frameworks

The consistently higher performance of the MLGR approach over both MLPO and MLPR can be explained by a number of key benefits inherent in the optimization strategy of GRO and its harmonization with MLPR. First, GRO provides a human-inspired optimization technique that benefits over the biologically inspired strategy of PO by utilizing group adaptation decision-making and dynamic movement towards regions of optimality. Such an ability allows MLGR to evade premature convergence and avoid being trapped by local optima two typical drawbacks seen in both MLPR and, to a lesser degree, MLPO. Second, selective pausing by GRO through fitness-distance correlation enables a better exploitation/exploration trade-off. It leads to more targeted convergence behavior and more rapid identification of optimal solutions in ALE task prediction, while PO heavily depends on random exploration, potentially slowing down convergence. Third, comparative results in all three layers and evaluation phases validate MLGR's dominance. When tested using the third layer, MLGR recorded an RMSE of 0.036, $R^2$ of 0.993, and MRAE of 0.166, surpassing MLPO (RMSE: 0.059, $R^2$: 0.981, MRAE: 0.298) and MLPR (RMSE: 0.080, $R^2$: 0.966, MRAE: 0.372) performance. Ultimately, MLGR benefits from superior generalization to novel data through GRO's progressive correction mechanism. Combined, these aspects increased convergence, enhanced optimization depth, and higher predictive accuracy render MLGR the best framework to make ALE predictions in this work. The hybrid strategy does have increased training time as a result of additional computational cost from GRO and PO, but this is counterbalanced by enhanced model accuracy and convergence. Inference, on the other hand, is still efficient because optimization happens only at training time, hence making it appropriate for offline training and real-time inference applications.

## 5 Conclusion

This exploration examined the potential of ML to boost indoor localization in WSNs. The limitations of GPS in indoor environments were addressed, emphasizing the need for alternative methods such as reference nodes and localization frameworks. Existing range-based and range-free frameworks were reviewed alongside recent advancements in machine-learning approaches for WSN localization. In this study, the recommended approach utilizes MLPR to predict ALE. To potentially enhance MLPR's performance, the study investigates the use of optimizers containing GRO and POA. The hybridized frameworks' performance was compared to identify the most accurate and precise solution for ALE prediction. The developed frameworks were evaluated in three layers and stages: Training, Validation, and Testing. The MLPO model achieved the best performance in the first laye. with a low RMSE of 0.055, followed closely by the MLGR model at 0.057. However, the MLPR model showed the weakest performance here, reaching an RMSE of 0.080. In the second layer, the focus shifted to R-squared ($R^2$), where the MLGR model shone with a value of 0.989. Interestingly, the MLPO and MLPR frameworks achieved similar results in this layer, sharing the second-best $R^2$ of 0.975. Finally, the third layer assessed with MRAE cemented the MLGR framework's dominance. It achieved the lowest MRAE of 0.166, indicating the most accurate predictions. The MLPO model followed with a value of 0.289, while MLPR revealed the peak MRAE (0.372), suggesting the least accurate predictions in the final layer. This analysis highlighted the consistent superiority of the MLGR model across most metrics, showcasing its effectiveness in predicting ALE. While the MLPO model showed promise in the first layer, its performance lagged in the later stages. The MLPR model consistently had the weakest performance, underlining the potential benefits observed in this study when using optimizers contained GRO with the MLPR model. Even in strong performance, the suggested approach needs greater computational power through hybrid optimization, making its applicability to real-time or resource-limited settings narrow. The model's reliance upon well-processed input data might also lessen robustness during uncertain or incomplete data, as opposed to probabilistic or ensemble approach methods that are more flexible when dealing with uncertainty. This approach has demonstrably enhanced the framework's predictive power for ALE estimation. Therefore, this research advances ML strategies for achieving efficient and reliable indoor localization within WSNs.

## Competing Interests

The scholars claim no competing interests.

## Authorship Contribution Statement

Jian SUN: Writing-Original draft preparation Conceptualization, Supervision, Project administration.

Jia LIU: Methodology, Software

## Data Availability

The scholars will make the raw data supporting this article's conclusions available without undue reservation.

## Declarations

Not applicable.

## Conflicts of Interest

The scholars claimed no conflicts of interest considering this investigation.

## Author Statement

The manuscript has been read and approved by all the authors, the requirements for authorship, as stated earlier in this document, have been met, and each author believes that the manuscript displays honest work.

## Funding

Not applicable.

## Ethical Approval

All scholars have been personally and actively involved in substantial work leading to the paper and will take public responsibility for its content.

## References

[1] Khan, I., F. Belqasmi, R. Glitho, N. Crespi, M. Morrow and P. Polakos (2015). Wireless sensor network virtualization: A survey. *IEEE Communications Surveys & Tutorials*, IEEE, 18(1), pp. 553–576. https://doi.org/10.1109/COMST.2015.2412971.

[2] Jouhari, M., K. Ibrahimi, H. Tembine and J. Ben-Othman (2019). Underwater wireless sensor networks: A survey on enabling technologies, localization protocols, and internet of underwater things. *IEEE Access*, IEEE, 7, pp. 96879–96899. https://doi.org/10.1109/ACCESS.2019.2928876.

[3] Xiong, H. and M.L. Sichitiu (2019). A lightweight localization solution for small, low resources WSNs. *Journal of Sensor and Actuator Networks*, MDPI, 8(2), pp. 26. https://doi.org/10.3390/jsan8020026.

[4] Zheng, J. and A. Dehghani (2012). Range-free localization in wireless sensor networks with neural network ensembles. *Journal of Sensor and Actuator Networks*, MDPI, 1(3), pp. 254–271. https://doi.org/10.3390/jsan1030254.

[5] Wang, Y., Z. Chen, T. Zhu, J. Liu and X. Du (2025). Intelligent Detection and Localization of Cable Faults Using Advanced Discharge Analysis Techniques. *Informatica*, Slovenian Society Informatika, 49(9). https://doi.org/10.31449/inf.v49i9.5468.

[6] Yan, C (2024). Application of a Graphical Image Pre-retrieval Method Based on Compatible Rough Sets to the Self-localization Method of Mobile Robots. *Informatica*, Slovenian Society Informatika, 48(11). https://doi.org/10.31449/inf.v48i11.5508.

[7] Singh, A., S. Sharma, J. Singh and R. Kumar (2019). Mathematical modelling for reducing the sensing of redundant information in WSNs based on biologically inspired techniques. *Journal of Intelligent & Fuzzy Systems*, Sage Publications, 37(5), pp. 6829–6839. https://doi.org/10.3233/JIFS-190605.

[8] Amutha, J., S. Sharma and J. Nagar (2020). WSN strategies based on sensors, deployment, sensing models, coverage and energy efficiency: Review, approaches and open issues. *Wireless Personal Communications*, Springer Nature, 111, pp. 1089–1115. https://doi.org/10.1007/s11277-019-06903-z.

[9] Khelifi, M., S. Moussaoui, S. Silmi and I. Benyahia (2015). Localisation algorithms for wireless sensor networks: A review. *International Journal of Sensor Networks*, Inderscience, 19(2), pp. 114–129. https://doi.org/10.1504/IJSNET.2015.071632.

[10] Tarrío, P., A.M. Bernardos and J.R. Casar (2011). Weighted least squares techniques for improved received signal strength based localization. *Sensors*, MDPI, 11(9), pp. 8569–8592. https://doi.org/10.3390/s110908569.

[11] Whitehouse, K (2002). *The design of calamari: an ad-hoc localization system for sensor networks*.

[12] Wen, C.-Y. and F.-K. Chan (2010). Adaptive AOA-aided TOA self-positioning for mobile wireless sensor networks. *Sensors*, MDPI, 10(11), pp. 9742–9770. https://doi.org/10.3390/s101109742.

[13] Bulusu, N., J. Heidemann and D. Estrin (2000). GPS-less low-cost outdoor localization for very small devices. *IEEE Personal Communications*, IEEE, 7(5), pp. 28–34. https://doi.org/10.1109/98.878533.

[14] Niculescu, D. and B. Nath (2001). Ad hoc positioning system (APS), In *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270)*, IEEE, San Antonio, TX, USA, pp. 2926–2931. https://doi.org/10.1109/GLOCOM.2001.965964.

[15] Waadt, A.E., C. Kocks, S. Wang, G.H. Bruck and P. Jung (2010). Maximum likelihood localization estimation based on received signal strength, In *2010 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2010)*, IEEE, Rome, Italy, pp. 1–5. https://doi.org/10.1109/ISABEL.2010.5702817.

[16] Coluccia, A. and F. Ricciato (2014). RSS-based localization via Bayesian ranging and iterative least squares positioning. *IEEE Communications Letters*, IEEE, 18(5), pp. 873–876. https://doi.org/10.1109/LCOMM.2014.040214.132781.

[17] Coluccia, A. and A. Fascista (2019). Hybrid TOA/RSS range-based localization with self-calibration in asynchronous wireless networks. *Journal of Sensor and Actuator Networks*, MDPI, 8(2), p. 31. https://doi.org/10.3390/jsan8020031.

[18] Kulkarni, V.R., V. Desai and R. V Kulkarni (2019). A comparative investigation of deterministic and metaheuristic algorithms for node localization in wireless sensor networks. *Wireless Networks*, Springer Nature, 25, pp. 2789–2803. https://doi.org/10.1007/s11276-019-01994-9.

[19] Kennedy, J. and R. Eberhart (1995). Particle swarm optimization, In *Proceedings of ICNN'95-International Conference on Neural Networks*, IEEE, Perth, WA, Australia, pp. 1942–1948. https://doi.org/10.1109/ICNN.1995.488968.

[20] Gopakumar, A. and L. Jacob (2008). Localization in wireless sensor networks using particle swarm optimization, In *2008 IET International Conference on Wireless, Mobile and Multimedia Networks*, IET, Beijing, China, pp. 227–230.

[21] Goyal, S. and M.S. Patterh (2014). Wireless sensor network localization based on cuckoo search algorithm. *Wireless Personal Communications*, Springer Nature, 79, pp. 223–234. https://doi.org/10.1007/s11277-014-1850-8.

[22] Cheng, J. and L. Xia (2016). An effective cuckoo search algorithm for node localization in wireless sensor network. *Sensors*, MDPI, 16(9), pp. 1390. https://doi.org/10.3390/s16091390.

[23] Liu, T. and Z. Zhang (2024). The Application Effect of Improved CS-RBF Neural Network in Industrial Internet of Things Node Localization. *Informatica*, Slovenian Society Informatika, 48(13). https://doi.org/10.31449/inf.v48i13.6004.

[24] Tan, C. and P. Li (2025). The Application of Logistics Robot in the Solution of Locating Route Problems in Trans CAD. *Informatica*, Slovenian Society Informatika, 49(11). https://doi.org/10.31449/inf.v49i11.6602.

[25] Morelande, M.R., B. Moran and M. Brazil (2008). Bayesian node localisation in wireless sensor networks, In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, Las Vegas, NV, USA, pp. 2545–2548. https://doi.org/10.1109/ICASSP.2008.4518167.

[26] Musso, C., N. Oudjane and F. Le Gland (2001). Improving regularised particle filters, In *Sequential Monte Carlo Methods in Practice*, Springer, New York, NY, USA, pp. 247–271. https://doi.org/10.1007/978-1-4757-3437-9_12.

[27] Gharghan, S.K., R. Nordin and M. Ismail (2016). A wireless sensor network with soft computing localization techniques for track cycling applications. *Sensors*, MDPI, 16(8), pp. 1043. https://doi.org/10.3390/s16081043.

[28] Ahmadi, H. and R. Bouallegue (2017). Exploiting machine learning strategies and RSSI for localization in wireless sensor networks: A survey, In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, Valencia, Spain, pp. 1150–1154. https://doi.org/10.1109/IWCMC.2017.7986447.

[29] Makhlouf, A., A. Benmachiche and I. Boutabia (2024). Enhanced Autonomous Mobile Robot Navigation Using a Hybrid BFO/PSO Algorithm for Dynamic Obstacle Avoidance. *Informatica*, Slovenian Society Informatika, 48(17). https://doi.org/10.31449/inf.v48i17.6716.

[30] Zhong, C. and G. Yang (2025). Design and Application of Improved Genetic Algorithm for Optimizing the Location of Computer Network Nodes. *Informatica*, Slovenian Society Informatika, 49(16). https://doi.org/10.31449/inf.v49i16.7201.

[31] Li, J. and Z. Tian (2024). Construction and Optimization of a Precise Positioning Model for Logistics Vehicles Based on Sustainable Operation. *Informatica*, Slovenian Society Informatika, 48(17). https://doi.org/10.31449/inf.v48i17.6393.

[32] Bhatti, M.A., R. Riaz, S.S. Rizvi, S. Shokat, F. Riaz and S.J. Kwon (2020). Outlier detection in indoor localization and Internet of Things (IoT) using machine learning. *Journal of Communications and Networks*, IEEE, 22(3), pp. 236–243. https://doi.org/10.1109/JCN.2020.000018.

[33] Wang, L., M.J. Er and S. Zhang (2020). A kernel extreme learning machines algorithm for node localization in wireless sensor networks. *IEEE Communications Letters*, IEEE, 24(7), pp. 1433–1436. https://doi.org/10.1109/LCOMM.2020.2986676.

[34] Gharghan, S.K., R. Nordin and M. Ismail (2016). A wireless sensor network with soft computing localization techniques for track cycling applications. *Sensors*, MDPI, 16(8), pp. 1043. https://doi.org/10.3390/s16081043.

[35] Goyal, S. and M.S. Patterh (2014). Wireless sensor network localization based on cuckoo search algorithm. *Wireless Personal Communications*, Springer Nature, 79, pp. 223–234. https://doi.org/10.1007/s11277-014-1850-8.

[36] Morelande, M.R., B. Moran and M. Brazil (2008). Bayesian node localisation in wireless sensor networks, In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, Las Vegas, NV, USA, pp. 2545–2548. https://doi.org/10.1109/ICASSP.2008.4518167.

[37] Noriega, L (2005). Multilayer perceptron tutorial. School of Computing. *Staffordshire University*, 4, p. 5.

[38] Ramchoun, H., Y. Ghanou, M. Ettaouil and M.A. Janati Idrissi (2016). Multilayer perceptron: Architecture optimization and training. International *Journal of Interactive Multimedia and Artificial Intelligence*, Universidad Internacional de La Rioja (UNIR), http://doi.org/10.9781/ijimai.2016.415.

[39] Murtagh, F (1991). Multilayer perceptrons for classification and regression. *Neurocomputing*, Elsevier, 2(5–6), pp. 183–197. https://doi.org/10.1016/0925-2312(91)90023-5.

[40] Bishop, C.M (1995). *Neural networks for pattern recognition*. Oxford university press.

[41] Trojovský, P. and M. Dehghani (2022). Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications. *Sensors*, MDPI, 22(3), pp. 855. https://doi.org/10.3390/s22030855.

[42] Alamir, N., S. Kamel, T.F. Megahed, M. Hori and S.M. Abdelkader (2023). Developing hybrid demand response technique for energy management in microgrid based on pelican optimization algorithm. *Electric Power Systems Research*, Elsevier, 214, pp. 108905. https://doi.org/10.1016/j.epsr.2022.108905.

[43] Tuerxun, W., C. Xu, M. Haderbieke, L. Guo and Z. Cheng (2022). A wind turbine fault classification model using broad learning system optimized by improved pelican optimization algorithm. *Machines*, MDPI, 10(5), pp. 407. https://doi.org/10.3390/machines10050407.

[44] Al-Wesabi, F.N., H.A. Mengash, R. Marzouk, N. Alruwais, R. Allafi, R. Alabdan, M. Alharbi and D. Gupta (2023). Pelican Optimization Algorithm with federated learning driven attack detection model in Internet of Things environment. *Future Generation Computer Systems*, Elsevier. https://doi.org/10.1016/j.future.2023.05.029.

[45] Singh, A., V. Kotiyal, S. Sharma, J. Nagar and C.-C. Lee (2020). A machine learning approach to predict the average localization error with applications to wireless sensor networks. *IEEE Access*, IEEE, 8, pp. 208253–208263. https://doi.org/10.1109/ACCESS.2020.3038645.