Journal of Theoretical and Applied Information Technology
31st January 2025. Vol.103. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FROG LEAP INSPIRED OPTIMIZATION-BASED EXTREME LEARNING MACHINE FOR ACCURATE CLASSIFICATION OF LATENT AUTOIMMUNE DIABETES IN ADULTS (LADA)

B. SUCHITRA[1], J. RAMKUMAR[2], R. KARTHIKEYAN[3]
[1,2] Assistant Professor, Department of Information Technology & Cognitive Systems,
Sri Krishna Arts and Science College, India
[3]Assistant Professor, Department of Computer Technology,
Sri Krishna Adithya College of Arts and Science, India
E-mail: [1]suchiprdaeep@gmail.com, [2]jramkumar1986@gmail.com, [3]karthikeyanrkrish@gmail.com

ABSTRACT

Latent Autoimmune Diabetes in Adults (LADA), also known as type 1.5 diabetes, represents a hybrid form of diabetes with characteristics of both type 1 and type 2 diabetes. Unlike type 1 diabetes, which has a rapid onset, LADA develops gradually, often diagnosed in adults over 30. In contrast to type 2 diabetes, LADA is caused by an autoimmune response that progressively destroys insulin-producing beta cells. This overlap often leads to misdiagnosis, as LADA is commonly mistaken for type 2 diabetes, delaying appropriate treatment. Classification algorithms face challenges in predicting LADA due to overlapping symptoms, high-dimensional health data, and imbalanced datasets. Existing methods lack robustness in accurately differentiating LADA from other diabetes types, leading to frequent misclassifications and treatment inefficiencies. To address these challenges, the Frog Leap Inspired Optimization-based Extreme Learning Machine (FLIO-ELM) has been proposed. FLIO-ELM combines bio-inspired optimization with a single-layer neural network framework. Inspired by frog-leaping behavior, this method enhances input weight and bias optimization, ensuring better feature selection. Frogs in the optimization represent potential solutions, with local and global search phases refining parameters. The Extreme Learning Machine (ELM) framework computes output weights using the least squares method, ensuring fast training and robust generalization. This hybrid mechanism balances exploration and exploitation to improve prediction accuracy. FLIO-ELM achieves significant improvements in performance metrics, with an 81.304% classification accuracy and an 82.093% F-measure. Its false discovery and omission rates are minimized, ensuring reliability in LADA prediction. These results establish FLIO-ELM as an effective diagnostic tool.
Keywords: *LADA, Frog Leap Inspired Optimization, ELM, Diabetes, Bio-Inspired Algorithms*

1. INTRODUCTION

Latent Autoimmune Diabetes in Adults (LADA), often described as "type 1.5 diabetes," represents a distinct form of diabetes blending characteristics from both type 1 and type 2 diabetes[1]. LADA arises from an autoimmune response, where the body's immune system gradually targets and damages the insulin-producing beta cells in the pancreas. Unlike the rapid onset in typical type 1 diabetes, LADA develops slowly, often diagnosed in adults over the age of 30. This gradual decline in insulin production, combined with its slow progression, results in frequent misdiagnoses as type 2 diabetes, leading to treatment challenges. Standard diagnostic tools primarily focused on metabolic markers often fail to identify LADA's autoimmune components. Testing for autoantibodies, such as glutamic acid decarboxylase (GAD) antibodies, can provide a more accurate diagnosis by confirming the autoimmune nature of LADA. Identifying LADA early proves critical, as optimal treatment may involve insulin rather than standard oral medications for type 2 diabetes[2]. Recognizing LADA's unique pathology helps in implementing an appropriate management approach, which may slow beta-cell deterioration and preserve endogenous insulin production. Understanding LADA as a distinct entity within the diabetes spectrum enhances personalized treatment strategies, improving both short-term and long-term outcomes for those affected[3].

Classification algorithms have enabled a significant advancement in diabetes prediction by analyzing complex health data to categorize individuals according to their risk profiles. Through examination of factors such as blood glucose levels, family history, lifestyle habits, and metabolic indicators, these algorithms identify patterns that can

signal early or pre-diabetic stages, allowing for timely intervention[4], [5]. They assess each variable's relevance and weight, refining the prediction of diabetes onset or classification of specific types, which can vary significantly in pathology and treatment needs. Accurate prediction through classification reduces misdiagnosis and guides personalized treatment plans, as specific forms of diabetes require tailored approaches. Early identification through classification models helps healthcare providers implement preventive measures, thereby reducing potential complications linked to delayed diagnosis[6]. By distinguishing among diabetes types with greater precision, classification methods support optimized management strategies, such as lifestyle modifications or timely medication, improving long-term patient outcomes. The ability of classification algorithms to continually adapt with new data ensures their role remains pivotal in the future of diabetes care, equipping healthcare providers with insights to deliver data-driven, proactive treatment[7].

Bio-inspired optimization techniques have proven invaluable in advancing diabetes classification by enhancing the efficiency and accuracy of classification algorithms. These techniques draw inspiration from natural processes—such as the movement of animals, evolutionary patterns, and collective behaviors—replicating these mechanisms to improve algorithm performance[8]. In the classification of diabetes, bio-inspired optimization strengthens algorithms by refining feature selection, identifying which data points—such as glucose levels, age, family history, and lifestyle factors—are most relevant to predicting diabetes risk or distinguishing between different types of diabetes. This focused selection reduces computational demands and improves prediction accuracy, addressing one of the most challenging aspects of diabetes classification. Bio-inspired optimization also addresses issues in handling large, complex datasets, which often contain overlapping characteristics across different diabetes types[9]. By optimizing parameters and tuning the model based on naturally inspired search strategies, these algorithms navigate through complex data to find optimal solutions with fewer resources. The adaptability of bio-inspired optimization allows these classification algorithms to self-adjust and improve continuously as they encounter new data, keeping pace with changing patient information and medical advancements. Through this approach, bio-inspired optimization contributes to creating highly accurate, reliable, and efficient classification models, directly benefiting diabetes prediction and personalized healthcare[10].

1.1 Problem Statement

Diabetes classification remains a challenging area in medical diagnostics due to overlapping symptoms, genetic variability, and environmental factors, which complicate the distinction between types of diabetes. Conventional classification algorithms often face limitations in identifying and categorizing specific diabetes subtypes accurately, leading to frequent misdiagnosis and ineffective treatments. Many current algorithms struggle with high-dimensional data, which includes numerous patient attributes like glucose levels, insulin sensitivity, lifestyle factors, and genetic predispositions. Such complexity creates issues with data redundancy and irrelevant feature selection, reducing prediction accuracy. Additionally, the dynamic nature of patient health data requires adaptive models capable of updating with new information. The absence of precise, adaptive algorithms for diabetes classification impacts early diagnosis and treatment effectiveness, potentially resulting in delayed intervention and increased risk of complications. Improving classification accuracy and adaptability within diabetic datasets remains an urgent need to enhance healthcare outcomes.

1.2. Motivation

The need for accurate diabetes classification has gained importance as diabetes prevalence rises globally, impacting millions of lives and straining healthcare systems. Effective classification is essential not only to ensure appropriate treatments but also to prevent the progression of complications, such as cardiovascular disease, neuropathy, and kidney failure, which can develop from misdiagnosed or poorly managed diabetes. Traditional classification methods often fail to account for the diversity in diabetes presentations, particularly with conditions like Latent Autoimmune Diabetes in Adults (LADA), requiring advanced, adaptive approaches. Bio-inspired optimization provides a promising solution, with its ability to handle large, complex datasets and optimize feature selection. This approach has potential to enhance the predictive accuracy of diabetes classification algorithms, improving patient outcomes. Developing precise and adaptive classification methods addresses a critical gap, fostering early interventions that can ultimately reduce healthcare costs and enhance quality of life for diabetes patients.

1.3. Objective

The main objective of this research focuses on developing a Frog Leap Inspired Optimization-based Extreme Learning Machine (FLIO-ELM) to improve the classification accuracy of Latent Autoimmune Diabetes in Adults (LADA) among other diabetes types. This objective employs frog leap-inspired optimization to enhance feature selection and optimize parameter tuning within the Extreme Learning Machine (ELM) framework. By simulating the adaptive leap strategies of frogs, FLIO-ELM addresses challenges in handling high-dimensional and complex data, improving both speed and predictive accuracy. This optimized approach aims to isolate the most relevant features in diabetic datasets, enabling precise differentiation of LADA cases from type 1 and type 2 diabetes. This refined classification model supports healthcare providers by offering a faster, more reliable diagnostic tool, ultimately contributing to early intervention and personalized treatment strategies for patients with LADA.

2. LITERATURE REVIEW

Hybrid Artificial Neural Network (HANN) is proposed for predicting diabetes mellitus with improved accuracy and reliability. The research investigates the potential of neural network algorithms to analyze patient data and identify patterns indicative of diabetes. The approach employs k-fold cross-validation to ensure unbiased performance evaluation while exploring the influence of hidden layer configurations on prediction accuracy. By utilizing structured data and scalable training methods, the system demonstrates adaptability across varying datasets. The findings provide evidence of significant accuracy improvements, making the model a promising tool for early detection of diabetes. This study reinforces the role of artificial intelligence in advancing healthcare diagnostics and personalized treatment strategies [11]. Efficient Deep Learning Technique (EDLT) is proposed for analyzing and predicting diabetes with improved accuracy using the Indian Diabetes Dataset. The framework integrates advanced neural network architectures to process large-scale medical data effectively. The research emphasizes the importance of feature extraction and optimization strategies to enhance classification outcomes. Rigorous evaluation demonstrates the model's capability in achieving reliable predictions, addressing challenges in chronic disease detection. The study provides a significant contribution to the application of artificial intelligence in healthcare, offering a robust solution for improving diagnostic accuracy and patient care in diabetes management [12].

The Enhanced Machine Learning Approach by Ying et al. (2024) predicts severe proteinuria in IgA nephropathy patients, effectively analyzing clinical data for improved patient stratification. This algorithm integrates robust feature selection, enhancing model accuracy and clinical relevance in nephrology. The study's results underscore the potential for tailored healthcare interventions, aiding in the proactive management of disease progression [13]. The Transparent Machine Learning Algorithm by Musacchio et al. (2024) analyzes HbA1c patterns linked to therapeutic inertia in type 2 diabetes cases where metformin monotherapy fails. By identifying hidden data patterns, this model assists clinicians in overcoming therapeutic inertia, thereby enhancing diabetes management outcomes [14]. The Patient Sentiment Analysis Model by Madan et al. (2024) applies machine learning to assess healthcare sentiment, empowering providers to adapt to patient needs based on real-time feedback. The model has contributed to enhancing patient-centered care by capturing sentiment dynamics, thus facilitating data-driven healthcare improvements [15]. The Predictive Machine Learning Approach by Cichosz et al. (2024) evaluates pancreatic cancer risk among newly diagnosed diabetes patients using biochemical markers. The algorithm identifies risk factors early, underscoring its role in facilitating timely intervention for high-risk individuals, with potential implications in oncology and preventative healthcare [16].

The Machine Learning-Based Model by Nayak et al. (2024) predicts and monitors diabetic kidney disease progression, leveraging retrospective data. This single-center study emphasizes the algorithm's potential in nephrology, where early identification of disease progression enhances patient outcomes and informs therapeutic decisions [17]. The Heart Rate Variability-Based Machine Learning Model by Keng et al. (2025) forecasts sepsis risk, derived from vital sign data. This model validates the prognostic potential of heart rate variability measures in intensive care, highlighting its significance in early sepsis intervention and risk stratification [18]. The Mobile Health Application Adoption Model by Kokila et al. (2024) identifies factors influencing mHealth app adoption using machine learning. This model provides insights into user preferences and potential barriers, contributing to strategies for improving healthcare accessibility through mobile technology [19]. The Web-Based Machine Learning

Model by Rahman et al. (2024) predicts polycystic ovary syndrome (PCOS) early, aiding in proactive health management for women. This approach utilizes accessible clinical data, enhancing early intervention capabilities in reproductive health [20]. The Birth Outcome Predictive Model by Adebanji et al. (2024) analyzes factors affecting birth outcomes through a machine learning perspective. This algorithm supports healthcare providers in identifying high-risk pregnancies, contributing to improved maternal and neonatal healthcare strategies [21].

The Cancer-Related Fatigue Prediction Model by Wang et al. (2024) assesses fatigue risks in lymphoma survivors. Through machine learning, this model supports patient quality-of-life improvements by anticipating fatigue symptoms, offering a valuable tool for personalized survivorship care [22]. Explainable Deep Learning Approach by Tanim et al. (2025) utilizes DeepNetX2 to enhance diabetes diagnosis, integrating explainability in model decisions for healthcare applications. DeepNetX2 applies layered feature extraction, combining convolutional layers with fully connected ones, yielding accurate diagnostic outcomes. By emphasizing interpretability, the model enables clinicians to understand decision pathways, thus aligning deep learning outputs with clinical reasoning in diabetes care [23]. Subspace Learning Machine (SLM) by Fu et al. (2024) introduces a novel methodology for subspace learning by extracting low-dimensional representations from high-dimensional data. SLM employs geometric and statistical principles, enhancing learning efficiency while retaining essential data features. Performance evaluations reveal its robustness across diverse tasks, establishing SLM as a powerful tool in visual data representation [24].

Chronic Kidney Disease Diagnostic Model by Dharmarathne et al. (2024) implements a machine learning-based interface integrated with explainable AI to diagnose chronic kidney disease. The interface combines decision tree-based predictions with transparency tools that elucidate the reasoning behind model outputs, providing clinicians with insights for improved diagnostic accuracy in nephrology [25]. AI-aided Cardiovascular Diagnosis in Cattle by Cihan et al. (2024) compares machine learning and deep learning for detecting cardiovascular disease in cattle using retinal imaging. By leveraging image processing techniques, the study demonstrates deep learning's superiority in extracting nuanced retinal features, showing promise in animal health management and early diagnosis [26]. Diverse Ensemble Learning Classifiers by Kawarkhe and Kaur (2024) for diabetes prediction utilize a mix of ensemble classifiers, including Random Forest, AdaBoost, and Gradient Boosting, to achieve high diagnostic accuracy. The model integrates multiple weak learners, enhancing predictive precision in diabetes datasets and underscoring ensemble methods' robustness in medical diagnostics [27]. Machine Learning Model for Knee Arthroplasty by Mittal et al. (2024) predicts prolonged hospital stays post-revision knee arthroplasty, utilizing a national dataset. Through regression-based techniques, the model identifies key predictors for length of stay, informing post-operative management strategies that optimize healthcare resources and improve patient outcomes [28].

Atrial Fibrillation Mortality Prediction Model by Luo et al. (2024) applies machine learning for in-hospital mortality prediction in critically ill atrial fibrillation patients. This model integrates demographic, clinical, and lab data, employing gradient boosting and logistic regression to stratify risk levels accurately, thus aiding in critical care decision-making [29]. Big Data Classification Algorithms by Singh et al. (2025) explores various machine learning algorithms in classifying big data, evaluating methods such as Decision Trees, SVM, and Neural Networks. This comparative study analyzes algorithmic efficiency and accuracy, establishing criteria for selecting optimal classifiers in large-scale datasets, crucial in data-intensive applications [30]. Red Blood Cell Demand Prediction Model by Hur et al. (2024) uses machine learning for predicting personalized red blood cell needs in thoracic surgery. Validated through clinical datasets, this model incorporates patient-specific parameters to forecast transfusion requirements, contributing to precision in resource allocation and patient safety in surgical care [31]. Differentiable L-1 Norm in Pattern Recognition by Zhang et al. (2024) introduces a novel L-1 norm, enhancing gradient-based optimization in pattern recognition. Designed for differentiability, this norm improves convergence in machine learning models, optimizing feature selection and classification accuracy, pivotal for complex recognition tasks [32].

Bio-inspired optimization algorithms consistently outperform conventional optimization algorithms by mimicking natural processes and behaviors, allowing for dynamic adaptability, global

search capabilities, and robustness against local optima [33]-[60]. These algorithms excel in handling complex, high-dimensional, and non-linear optimization problems with improved accuracy and efficiency.

3. Frog Leap Inspired Optimization-based Extreme Learning Machine (FLIO-ELM)

Frog Leap Inspired Optimization-based Extreme Learning Machine (FLIO-ELM) integrates the exploration capabilities of Frog Leap Inspired Optimization (FLIO) with the learning efficiency of Extreme Learning Machine (ELM). In this hybrid model, FLIO optimizes ELM's input weights and biases by simulating frogs' social foraging behavior, which enhances search and convergence. The ELM component uses a single-layer neural network with randomly assigned hidden layer weights, which are further refined by FLIO. This combination accelerates training, improves accuracy, and ensures robust model generalization, making FLIO-ELM suitable for complex tasks in large datasets, including classification, regression, and real-time decision-making applications.

3.1. Problem Formulation

Define the dataset where represents the input features and is the target variable for a regression task or for a classification task. The objective in the step involves determining the input matrix and target vector that will be used to train the ELM model. Let be the input matrix, where each row corresponds to an input vector and is the target vector. The aim is to feed these inputs to the hidden layer of the ELM, which can be expressed mathematically as:

$$H = g(XW + b) \quad \textbf{(1)}$$

where is the hidden layer matrix, represents the weight matrix between the input and hidden layer, is the bias vector, and is the activation function applied element-wise. The hidden layer output is calculated based on the input matrix the weights and the bias

Once has been computed, the next step involves determining the output weights This can be done using the following least squares solution**:**

$$\beta = (H^T H)^{-1} H^T Y \quad \textbf{(2)}$$

The final output of the ELM model can be represented as:

$$Y - H\beta \quad (3)$$

where $\hat{Y} \in R^{n\times 1}$ represents the predicted output from the model. To measure the error between predicted and actual values, calculate the error vector $E \in R^{n\times 1}$ as:

$$E = Y - Y \quad (4)$$

3.2. Initialize ELM Parameters

In this step, initialize the ELM parameters, including the weights and biases of the hidden layer. Let the weight matrix for the hidden layer be represented as $W \in R^{m\times L}$, where m denotes the number of input features, and L represents the number of neurons in the hidden layer. The bias vector $b \in R^{1\times L}$ is also randomly initialized. The hidden layer output $H \in R^{n\times L}$ can be computed as:

$$H = g(XW + b) \quad \textbf{(5)}$$

where $X \in R^{n\times m}$ represents the input matrix and $g(.)$ denotes the activation function applied element-wise. Depending on the configuration, the activation function could be a sigmoid, hyperbolic tangent, or ReLU.

Initialize these parameters using a uniform distribution over a specific range. This can be mathematically expressed as:

$$W_{ij} \sim U(-\alpha, \alpha), b_j \sim U(-\beta, \beta) \quad (6)$$

where α and β are small positive constants, and $U(-\alpha, \alpha)$ denotes a uniform distribution within the range $[-\alpha, \alpha]$.

The next step involves calculating the output weights $\beta \in R^{L\times 1}$ that connect the hidden layer output $H$to the final predicted output $\hat{Y}$. Using the least squares solution determine β as:

$$\beta = (H^T H)^{-1} {}_1 H^T Y$$

where $Y \in R^{n\times 1}$ is the target vector. The predicted output $\hat{Y} \in R^{n\times 1}$ is the calculated by:

$$\hat{Y} = H\beta \quad (8)$$

The prediction error is computed by:

$$E = Y - \hat{Y} \quad (9)$$

This initializes the ELM and prepares it for further optimization in subsequent steps.

3.3. Frog Population Initialization for ELM Optimization

In this step, the frog population is initialized for the FLIO to optimize the ELM parameters. Each frog in the population represents a potential solution

corresponding to a set of ELM input weights and biases. Let the population consist of P frogs, where each frog is represented by a solution vector. S_i for $i = 1,2, \dots, P$. Each solution vector S_i consists of the ELM parameters, i.e., the input weight matrix $W \in R^{m \times L}$ and bias vector $b \in R^{1 \times L}$. Hence, each frog's solution vector can be expressed as:

$$Si=Wi,bi,\ \ Wi \in Rm \times L, bi \in R1 \times L \quad (10)$$

The initial population of frogs S_i is generated randomly within predefined bounds. For each frog, initialize the input weights and biases for the ELM model using random values from a uniform distribution, as done in the previous step:

$$W_{ij} \sim U(-\alpha, \alpha), \qquad b_j \sim U(-\beta, \beta) \quad (11)$$

where α and β are small positive constants, and $U(-\alpha, \alpha)$ denotes the uniform distribution over the interval $[-\alpha, \alpha]$.

The fitness of each frog S_i is evaluated based on the performance of the corresponding ELM model. For each frog, compute the hidden layer output $H \in R^{n \times L}$ using the current weights W_i and biases b_i as follows:

$$H_i = g(XW_i + b_i) \quad (12)$$

where $X \in R^{n \times m}$ is the input matrix, and $g(.)$ is the activation function applied element-wise.

The output weights $\beta_i \in R^{L \times 1}$ for the ELM are computed using the least squares method:

$$\beta_i = (H_i^T H_i)^{-1} H_i^T Y \quad (13)$$

where $Y \in R^{n \times 1}$ is the target vector. The predicted predicted output $\hat{Y}_i \in R^{n \times 1}$ for the ith frog's ELM model is given by:

$$\hat{Y}_i = H_i \beta_i \quad (14)$$

To evaluate the performance of the i-th frog, compute the error between the predicted output. $\hat{Y}_i$ and the actual target Y using the following error metric, such as the mean squared error (MSE):

$$E_i = \frac{1}{n} \sum_{j=1}^{n} \left(y_j - \hat{y}_{ij}\right)^2 \quad (15)$$

where y_j and $\hat{y}_{ij}$ are the actual and predicted values for the jth sample, respectively. The fitness F_i of the ith frog is inversely proportional to the error:

$$Fi=1Ei \quad (16)$$

A lower error corresponds to higher fitness, so the optimization process aims to improve each frog's fitness by adjusting the ELM model's corresponding input weights and biases. After computing the fitness for all frogs in the population, sort them in descending order based on their fitness values. The frogs with higher fitness (better-performing ELM models) are positioned at the top, while the lower fitness frogs are placed at the bottom. Divide the frog population into M memeplexes, each containing a subset of frogs, where each memeplex performs independent local searches.

This initialization of the frog population and fitness evaluation prepares the system for further optimization of the ELM model using FLIO, focusing on improving the input weights and biases to enhance the overall model performance. Each frog represents a potential solution that will evolve in subsequent steps based on the frog leaping rules and the optimization strategy.

3.4. Fitness Evaluation Based on ELM Performance

In this step, the frog population will be divided into memeplexes for local optimization. Let P represent the total number of frogs in the population, and M denote the number of memeplexes. Each memeplex will consist of N frogs, where $N = \frac{P}{M}$. Therefore, divide the frogs into M distinct memeplexes, $M_1, M_2, \dots, M_M$, where each memeplex contains N frogs. The sorting of frogs based on fitness, performed in the previous step, ensures that the frogs with higher fitness occupy higher positions in the memeplexes. Denote the sorted frogs as $S_1, S_2, \dots, S_P$, where S_1 has the highest fitness and S_P has the lowest fitness. Divide these frogs across memeplexes in a round-robin fashion.

For each memeplex M_k, where $k \in \{1,2, \dots, M\}$, assign frogs $S_k, S_{k+M}, S_{k+2M}, \dots, S_{k+(N-1)M}$. This division ensures a balanced distribution of frogs across memeplexes with a mixture of high, medium, and low fitness solutions. Once memeplexes are established, apply the local search within each memeplex. For a given memeplex M_k, identify the frog with the highest fitness, denoted as S_{best}^k, and the frog with the lowest fitness, denoted as S_{worst}^k.

The goal of the local search is to improve the fitness of S_{worst}^k by moving it closer to S_{best}^k within the memeplex. The update rule for the frog leaping strategy is expressed as:

$$S_{worst}^{k} = S_{worst}^{k} + r.\left(S_{best}^{k} - S_{worst}^{k}\right) \quad (17)$$

where $r \in [0,1]$ is a random number drawn from a uniform distribution. This equation moves the worst frog towards the best-performing frog within the memeplex, improving its position in the solution space. The new solution S_{worst}^{k} contains updated ELIM parameters (input weights W and biases b).

For the updated solution, recompute the hidden layer output matrix. H_{worst}^{k} for the corresponding ELM using the updated weights W_{worst}^{k} and biases b_{worst}^{k}.

$$H_{worst}^{k} = g(XW_{worst}^{k} - b_{worst}^{k}) \quad (18)$$

Next, update the output weights β_{worst}^{k} by solving the least squares equation:

$$\beta_{worst}^{k} = \left({H_{worst}^{k}}^{T} H_{worst}^{k}\right)^{-1} {H_{worst}^{k}}^{T} Y \quad (19)$$

where Y is the target vector, and the predicted output $\hat{Y}_{worst}^{k}$ is:

$$\hat{Y}_{worst}^{k} = H_{worst}^{k}\beta_{worst}^{k} \quad (20)$$

To evaluate the fitness of the updated worst frog S_{worst}^{k}, calculate the error between the predicted output $\hat{Y}_{worst}^{k}$ and the actual target values Y using the mean squared error (MSE):

$$E_{worst}^{k} = \frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_{worst,j}^{k}\right)^2 \quad (21)$$

The new fitness F_{worst}^{k} is:

$$F_{worst}^{k} = \frac{1}{E_{worst}^{k}} \quad (22)$$

It the fitness of the updated frog F_{worst}^{k} improves compared to its previous fitness, retain the new solution; otherwise, restore the previous solution. Perform the local search for each memeplex M_k independently. After completing the local search for all memeplexes, regroup the entire frog population for the global search phase in subsequent steps. This process ensures that each memeplex undergoes localized optimization while balancing exploration and exploitation of the solution space.

3.5. Adaptive Frog Leaping Strategy

In this step, proceed with the global search phase by regrouping all frogs from the memeplexes after completing the local search. The global search allows information exchange among the memeplexes, enhancing exploration in the overall solution space. The entire frog population, consisting of P frogs, is reassembled after local searches to evaluate their performance across the entire population. Let the reassembled population be represented by $S_1, S_2, \ldots, S_P$, where each frog S_i is defined by a solution vector containing the optimized ELM parameters $W_i \in R^{m\times L}$ (weights) and $b_i \in R^{1\times L}$ (biases). The goal of the global search is to perform a broader exploration of the solution space and improve the global fitness of the population.

The fitness F_i of each frog is calculated again after the local search using the mean squared error (MSE) of the corresponding ELM model. For each frog, compute the hidden layer output matrix as:

$$H_i = g(XW_i + b_i) \quad (23)$$

where $g(.)$ is the activation function, $X \in R^{n\times m}$ is the input matrix, and W_i and b_i are the frog's current weights and biases, respectively. The output weights $\beta_i \in R^{L\times 1}$ for each frog are updated using the least squares method:

$$\beta_i = (H_i^T H_i)^{-1} H_i^T Y \quad (24)$$

where $Y \in R^{n\times 1}$ is the target vector. The predicted output $\hat{Y}_i \in R^{n\times 1}$ is given by:

$$\hat{Y}_i = H_i\beta_i \quad (25)$$

The error E_i for each frog is computed as:

$$E_i = \frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_{ij}\right)^2 \quad (26)$$

where y_j represents the actual target values, and $\hat{y}_{ij}$ are the predicted values from the ELM model of frog S_i. The fitness F_i of each frog is updated as:

$$F_i = \frac{1}{E_i} \quad (27)$$

Following the fitness evaluation, identify the frog with the highest global fitness, denoted as S_{best}, and the frog with the lowest global fitness, denoted as S_{worst}. The global search then moves the worst-performing frog towards the best-performing frog across all memeplexes. The global update rule for S_{worst} is given by:

$$S_{worst} = S_{worst} + r.(S_{best} - S_{worst}) \quad (28)$$

where $r \in [0,1]$ is a random number from a uniform distribution. This equation adjusts the worst frog's

ELM parameters (input weights and biases) to improve its position based on the best-performing frog in the global population.

After updating S_{worst}, compute the new hidden layer output H_{worst} using the updated weights S_{worst} and biases b_{worst}:

$$H_{worst} = g(XW_{worst} + b_{worst}) \quad (29)$$

Update the output weights β_{worst} as:

$$\beta_{worst} = (H_{worst}^T H_{worst})^{-1} H_{worst}^T Y \quad (30)$$

Calculate the new predicted output $\hat{Y}_{worst}$ as:

$$\hat{Y}_{worst} = H_{worst}\beta_{worst} \quad (31)$$

Evaluate the new error E_{worst} and fitness F_{worst} for the updated frog. If the fitness improves, retain the new solution; otherwise, revert to the previous solution. Repeat this process until all frogs in the population have been updated based on the global search phase. This step ensures that the worst-performing frogs benefit from the global best solution, promoting convergence towards better solutions across the population.

3.6. Dynamic Hidden Neuron and Activation Function Configuration

In this step, employ an adaptive frog leaping strategy to control the intensity of the search process. The adaptive leaping mechanism adjusts the leap size based on the performance improvements in the fitness values of the frogs, ensuring efficient exploration and exploitation during optimization. Let S_i represent the solution vector of the ith frog, where each solution vector consists of the input weight matrix $W_i \in R^{m\times L}$ and bias vector $b_i \in R^{1\times L}$. The leap size d_i for the ith frog is calculated based on its fitness improvement ΔF_i between two consecutive iterations. If the fitness improvement is significant, the leap size is reduced to fine-tune the solution, whereas a smaller fitness improvement results in larger leaps to explore other regions of the solution space. The adaptive leap size d_i can be defined as:

$$d_i = r.\left(\frac{\Delta F_i}{F_{best}}\right).(S_{best} - S_i) \quad (32)$$

where $r \in [0,1]$ is a random number drawn from a uniform distribution, F_{best} is the fitness of the best-performing frog in the population, and ΔF_i is the change in fitness of the ith frog from the previous iteration, calculated as:

$$\Delta F_i = F_i^{(t)} - F_i^{(t-1)} \quad (33)$$

where $F_i^{(t)}$ and $F_i^{(t-1)}$ denote the fitness values of the ith frog at the current and previous iterations, respectively. If ΔF_i is large, indicating significant improvement, the leap size d_i is reduced, allowing fine adjustments to the frog's position. Conversely, a smaller ΔF_i increases the leap size, promoting exploration.

Update the solution vector S_i for the ith frog by applying the adaptive leap size d_i:

$$S_i^{(t+1)} = S_i^{(t)} + d_i \quad (34)$$

where $S_i^{(t+1)}$ represents the updated solution for the next iteration. The new input weights $W_i^{(t+1)}$ and biases $b_i^{(t+1)}$are extracted from the updated solution vector $S_i^{(t+1)}$.

For the updated frog, recompute the hidden layer output matrix H_i of the ELM using the updated weights $W_i^{(t+1)}$ and biases $b_i^{(t+1)}$.

$$H_i^{(t+1)} = g\left(XW_i^{(t+1)} + b_i^{(t+1)}\right) \quad (35)$$

Update the output weights $\beta_i^{(t+1)}$ for the ELM model by solving the least squares equation:

$$\beta_i^{(t+1)} = \left(H_i^{(t+1)T} H_i^{(t+1)}\right)^{-1} H_i^{(t+1)T} Y \quad (36)$$

where Y is the target vector. Calculate the predicted output $\hat{Y}_i^{(t+1)}$ for the updated frog:

$$\hat{Y}_i^{(t+1)} = H_i^{(t+1)}\beta_i^{(t+1)} \quad (37)$$

To evaluate the performance of the updated frog, calculate the error $E_i^{(t+1)}$ between the predicted output $\hat{Y}_i^{(t+1)}$ and the actual target values Y using the mean squared error (MSE):

$$E_i^{(t+1)} = \frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_{ij}^{(t+1)}\right)^2 \quad (38)$$

Finally, update the fitness $F_i^{(t+1)}$ of the frog based on the computed error:

$$F_i^{(t+1)} = \frac{1}{E_i^{(t+1)}} \quad (39)$$

Repeat the adaptive frog leaping process until the performance of the ELM model converges or the maximum number of iterations is reached. The adaptive strategy ensures that the search intensity adjusts dynamically based on fitness improvements, allowing for both exploration of new solutions and exploitation of near-optimal solutions.

3.7. Local Search in Memeplexes (Exploitation)

In this step, the mutation mechanism is incorporated to introduce diversity into the frog population and prevent premature convergence during the optimization of the ELM. Mutation randomly perturbs the parameters of a subset of frogs, enhancing exploration and enabling the population to escape local optima. Let P represent the total population of frogs, where each frog S_i is represented by a solution vector containing the input weight matrix $W_i \in R^{m\times L}$ and bias vector $b_i \in R^{1\times L}$. Apply mutation to a subset of the population, denoted as P_{mut}, where $P_{mut} \subseteq P$ and the size of the subset is $|P_{mut}|$.

For each frog $S_i \in P_{mut}$, randomly perturb its input weights and biases. Let the mutation rate $\mu \in [0,1]$ determine the probability of perturbation for each element of the weight matrix W_i and bias vector b_i. For each element W_{ij} in the weight matrix W_i, mutate as follows:

$$W_{ij}^{(mut)} = W_{ij}^{(t)} + \delta.\sigma_{ij} \quad (40)$$

where δ is a small random perturbation drawn from a uniform distribution $U(-\gamma,\gamma)$, and σ_{ij} is a binary indicator that equals 1 if mutation occurs, determined by the mutation rate μ, and otherwise. Similarly, for each element b_j in the bias vector b_i, apply the mutation rule:

$$b_j^{(mut)} = b_j^{(t)} + \delta.\sigma_{ij} \quad (41)$$

where δ follows the same distribution $U(-\gamma,\gamma)$, and σ_{ij} is the binary indicator for mutation. The mutation process ensures that random perturbations are applied to the weights and biases of the selected frogs, introducing new potential solutions into the population.

After mutation, the updated solution vector $S_i^{(mut)} = \left\{W_i^{(mut)}, b_i^{(mut)}\right\}$ represents the mutated ELM parameters for frog S_i. For each mutated frog, recompute the hidden layer output $H_i^{(mut)}$ using the mutated weights and biases:

$$H_i^{(mut)} = g\left(XW_i^{(mut)} + b_i^{(mut)}\right) \quad (42)$$

Update the output weights $\beta_i^{(mut)}$ by solving the least squares problem:

$$\beta_i^{(mut)} = \left(H_i^{(mut)T} H_i^{(mut)}\right)^{-1} H_i^{(mut)Y} Y \quad (43)$$

where $Y \in R^{n\times 1}$ is the target vector. Calculate the predicted output $\hat{Y}_i^{(mut)}$ as:

$$\hat{Y}_i^{(mut)} = H_i^{(mut)}\beta_i^{(mut)} \quad (44)$$

To evaluate the fitness of each mutated frog, compute the error $E_i^{(mut)}$ between the predicted output $\hat{Y}_i^{(mut)}$ and the actual target values Y using the mean squared error (MSE):

$$E_i^{(mut)} = \frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_{ij}^{(mut)}\right)^2 \quad (45)$$

The new fitness $F_i^{(mut)}$ of the mutated frog is then calculated as:

$$F_i^{(mut)} = \frac{1}{E_i^{(mut)}} \quad (46)$$

The mutation process introduces variability into the population by creating new potential solutions that differ from the original set of frogs. If the fitness $F_i^{(mut)}$ of the mutated frog improves compare to its previous fitness $F_i^{(t)}$, retain the mutated solution; otherwise, revert to the original solution.

Repeat the mutation process for all selected frogs in P_{mut} ensuring that diversity is maintained throughout the population. This step helps to balance exploration and exploitation, enhancing the overall performance of the FLIO-ELM model by introducing random perturbations and allowing for the discovery of new, potentially better solutions.

3.8. Mutation Mechanism for Diversity

In this step, a penalty function will be implemented to discourage the creation of overly complex ELM models during the optimization process. The goal is to maintain the efficiency of the models by penalizing solutions that use too many hidden neurons or large parameter values, promoting simpler models with better generalization capabilities. Let S_i represent the solution vector for the ith frog, consisting of the input weight matrix $W_i \in R^{m\times L}$ and the bias vector $b_i \in R^{1\times L}$, where L denotes the number of hidden neurons. A penalty

term $P(S_i)$ is added to the frog's fitness function to prevent the optimization process from favoring overly complex solutions.

Define the penalty function $P(S_i)$ as a combination of two terms: one penalizing the number of hidden neurons L and another penalizing the magnitude of the input weights W_i and biases b_i. The penalty function is expressed as:

$$P(S_i) = \lambda_1 . L + \lambda_2 . \left(\frac{1}{mL} \sum_{j=1}^{m} \sum_{k=1}^{L} W_{jk}^2 + \frac{1}{L} \sum_{k=1}^{L} b_k^2 \right) \quad (47)$$

where λ_1 and λ_2 are regularization parameters controlling the strength of the penalties, L is the number of hidden neurons, and the terms W_{jk}^2 and b_k^2 represent the squared magnitudes of the input weights and biases, respectively. The first term penalizes models with a larger number of hidden neurons, while the second term discourages large weight and bias values.

The modified fitness function for the i-th frog, incorporating the penalty term, is given by:

$$F_i^{(pen)} = \frac{1}{E_i} - P(S_i) \quad (48a)$$

where E_i is the mean squared error (MSE) between the predicted output $\hat{Y}_i$ and the actual target values Y, computed.

$$E_i = \frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_{ij})^2 \quad (48b)$$

The fitness $F_i^{(pen)}$ decreases when the penalty function $P(S_i)$ increases, which encourages the optimization process to favor simpler models with fewer hidden neurons and smaller parameter magnitudes. This penalization approach balances model complexity with accuracy. After applying the penalty function, the optimization continues by updating the solution vector S_i, which contains the input weights W_i and biases b_i, using the frog leaping strategy or other optimization mechanisms described in the previous steps. For the updated solution, recalculate the hidden layer output H_i using the updated parameters:

$$H_i = g(XW_i + b_i) \quad (49)$$

where $g(.)$ is the activation function, and $X \in R^{n \times m}$ is the input matrix. The output weights $\beta_i \in R^{L \times 1}$ are then recalculated using the least squares method:

$$\beta_i = (H_i^T H_i)^{-1} H_i^T Y \quad (50)$$

The predicted output $\hat{Y}_i \in R^{n \times 1}$ is obtained as:

$$\hat{Y}_i = H_i \beta_i \quad (51)$$

The error E_i is updated based on the new predicted output, and the fitness $F_i^{(pen)}$ is recalculated by incorporating the updated penalty term. The penalty-based fitness function is then used to guide the optimization process toward more efficient solutions that generalize well without excessive complexity. Introducing this penalty function controls the model's complexity while still optimizing for performance. The trade-off between accuracy and simplicity ensures that the ELM models remain computationally efficient and robust against overfitting, which is critical for achieving generalization in real-world applications.

3.9. Global Search Across Memeplexes (Exploration)

In this step, the hybrid learning approach will be integrated by combining the FLIO improvements with traditional ELM output weight adjustments. This hybrid learning step ensures that the FLIO optimization process effectively refines the input weights and biases, while ELM's least squares method is used to compute the final output weights, maintaining computational efficiency. Let S_i represent the optimized solution vector of the ith frog, where the vector consists of the input weight matrix $W_i \in R^{m \times L}$ and bias vector $b_i \in R^{1 \times L}$. For each optimized frog S_i, use the refined input weights and biases obtained from the FLIO optimization to calculate the hidden layer output H_i:

$$H_i = g(XW_i + b_i) \quad (52)$$

where $g(.)$ is the activation function, and $X \in R^{n \times m}$ is the input matrix. The output weights $\beta_i \in R^{L \times 1}$ are then updated using the traditional least squares method, which minimizes the error between the predicted and actual outputs.

The least squares solution for β_i is given by:

$$\beta_i = (H_i^T H_i)^{-1} H_i^T Y \quad (53)$$

The least squares solution for β_i is given by:

$$\beta_i = (H_i^T H_i)^{-1} H_i^T Y \quad (54)$$

where $Y \in R^{n\times 1}$ is the target vector. The predicted output $\hat{Y}_i \in R^{n\times 1}$ for the optimized frog is then calculated as:

$$\hat{Y}_i = H_i\beta_i \quad (55)$$

Next, calculate the error E_i between the predicted output $\hat{Y}_i$ and the actual target values Y using the mean squared error (MSE) as the error metric:

$$E_i = \frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_{ij}\right)^2 \quad (56)$$

The error $E_i$measures the performance of the ELM model after applying both FLIO optimization and least squares output weight adjustments. The fitness F_i for each frog is updated based on the calculated error as:

$$F_i = \frac{1}{E_i} \quad (57)$$

This hybrid learning step ensures that the refined input weights and biases, obtained through FLIO, are effectively combined with the computationally efficient least squares method for adjusting the output weights. This approach leverages FLIO's strength in optimizing the hidden layer and allows the least squares method to handle the output layer efficiently, ensuring faster convergence.

After updating the output weights and calculating the fitness for each frog, identify the frog with the highest fitness, denoted as S_{best}. The global search or mutation mechanism can further refine the solution, as described in the previous steps. The hybrid learning approach ensures that the optimization process benefits from the exploratory capabilities of FLIO while preserving the ELM's computational efficiency. The least squares method provides a closed-form solution for the output weights, reducing the complexity of simultaneously optimizing both the input weights and output weights.

To ensure generalization, this step incorporates cross-validation on the updated ELM model for each frog. Divide the dataset into training and validation sets, and evaluate the performance of the ELM model on the validation set to check for overfitting. The cross-validation process is expressed as:

$$E_{val} = \frac{1}{n_{val}}\sum_{j=1}^{n_{val}}\left(y_{val,j} - \hat{y}_{val,j}\right)^2 \quad (58)$$

where E_{val} represents the validation error, n_{val} is the number of validation samples $y_{val,j}$ represents the actual target values for the validation set, and $\hat{y}_{val,j}$ represents the predicted values for the validation set.

This hybrid learning step, by combining FLIO for input weight optimization and the least squares method for output weights, ensures that the ELM model achieves accuracy and computational efficiency. The resulting model exhibits improved generalization and performance across diverse datasets.

3.10. Hybrid Learning Step

In this step, a convergence check should be implemented to determine whether the optimization process should stop or continue. The stopping criteria are based on the fitness values, error reduction, and the number of iterations completed. The goal is to ensure that the FLIO-ELM model has reached an optimal or near-optimal solution without overfitting or wasting computational resources. Let $F_i^{(t)}$ represent the fitness of the ith frog at iteration t, and let $E_i^{(t)}$ represent the corresponding error at that iteration. A typical convergence check involves monitoring the change in fitness $\Delta F_i^{(t)}$ or the error $\Delta E_i^{(t)}$ between consecutive iterations. Define the change in fitness as:

$$\Delta Fi(t)=Fi(t)-Fi(t-1) \quad (59)$$

Similarly, the change in error is defined as:

$$\Delta Ei(t)=Ei(t-1)-Ei(t) \quad (60)$$

The optimization process can stop if the absolute change in fitness $\Delta F_i^{(t)}$ or error $\Delta E_i^{(t)}$ falls below a predefined threshold ϵ. Mathematically, the stopping condition can be expressed as:

$$\left|\Delta F_i^{(t)}\right| < \epsilon \text{ or } \left|\Delta E_i^{(t)}\right| < \epsilon \quad (61)$$

where ϵ is a small positive constant that defines the tolerance for convergence. If either of these conditions is satisfied, the optimization for the corresponding frog S_i is considered converged. This ensures that further iterations will not yield significant fitness or error reduction improvements.

Another stopping criterion involves monitoring the maximum number of iterations T_{max}. If the optimization reaches T_{max}iterations, the process stops to prevent excessive computation. This condition is expressed as:

$$t \geq Tmax \quad (62)$$

Once the convergence check is performed for all frogs in the population, those frogs that have met the stopping criteria will no longer undergo further updates. The remaining frogs, which have not yet converged, continue with the optimization process in subsequent iterations. This selective convergence approach ensures that only non-converged frogs participate in further searches, reducing computational effort.

To further enhance the optimization process, introduce a dynamic adjustment of the mutation rate μ based on the convergence behavior of the population. If a large proportion of the population has converged, reduce μ to focus more on fine-tuning the remaining frogs. Conversely, if most frogs have not converged, increase μ to encourage exploration. The dynamic mutation rate is defined as:

$$\mu^{(t+1)} = \mu^{(t)} \cdot \left(\frac{|P_{conv}|}{P} \right) \quad (63)$$

where $\mu^{(t)}$ is the mutation rate at iteration t, $|P_{conv}|$ represents the number of converged frogs, and P is the total number of frogs. This dynamic adjustment helps balance exploration and exploitation during the optimization process.

To ensure the robustness of the FLIO-ELM model, cross-validation must be applied during the convergence check. Evaluate the performance of the model on a validation set at each iteration. If the validation error $E_{val}^{(t)}$ starts to increase while the training error $E_i^{(t)}$ decreases, the model may be overfitting. The validation error is calculated as:

$$E_{val}^{(t)} = \frac{1}{n_{val}} \sum_{j=1}^{n_{val}} \left(y_{val,j} - \hat{y}_{val,j}^{(t)} \right)^2 \quad (64)$$

where n_{val} represents the number of validation samples, and $y_{val,j}$ and $\hat{y}_{val,j}^{(t)}$ represent the actual and predicted values for the validation set, respectively. If overfitting is detected, reduce the complexity of the model by adjusting the number of hidden neurons L, or applying regularization to the input weights W and biases b.

Through these convergence checks and dynamic adjustments, the FLIO-ELM model ensures efficient optimization, preventing unnecessary iterations and maintaining generalization capabilities across various datasets.

3.11. Penalty Function for Efficient ELM Models

In this step, evaluate the final FLIO-ELM model after completing the convergence checks and optimization. The objective involves analyzing the model's performance on the test dataset and ensuring that the learned parameters (weights and biases) generalize well to unseen data. Performance evaluation is based on calculating various metrics, such as accuracy, mean squared error (MSE), precision, recall, and F1-score. Let the final set of optimized weights and biases for the ith frog be denoted by $W_i^* \in R^{m \times L}$ and $b_i^* \in R^{1 \times L}$, respectively, after the optimization process. For each test input $X_{test} \in R^{n_{test} \times m}$, where n_{test} represents the number of test samples, compute the hidden layer output H_i^* using the optimized weights and biases:

$$H_i^* = g(X_{test} W_i^* + b_i^*) \quad (65)$$

where $g(.)$ is the activation function applied element-wise. The output weights $\beta_i^* \in R^{1 \times L}$ obtained from the previous optimization are used to predict the output $\hat{Y}_{test} \in R^{n_{test} \times 1}$.

$$\hat{Y}_{test} = H_i^* \beta_i^* \quad (66)$$

To assess the accuracy of the FLIO-ELM model, calculate the error E_{test} between the predicted output $\hat{Y}_{test}$ and the actual target values $Y_{test} \in R^{n_{test} \times 1}$. The mean squared error (MSE) for the test dataset is expressed as:

$$E_{test} = \frac{1}{n_{test}} \sum_{j=1}^{n_{test}} \left(y_{test,j} - \hat{y}_{test,j} \right)^2 \quad (67)$$

where $y_{test,j}$ represents the actual target values, $\hat{y}_{test,j}$ are the predicted values for each test sample. A lower E_{test} value indicates better performance of the model on the test set.

Additional evaluation metrics such as accuracy, precision, recall, and F1-score are calculated for classification tasks. Accuracy measures the percentage of correctly predicted labels out of all test samples:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (68)$$

where TP represents true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives. Precision evaluates the proportion of correctly predicted positive samples out of all predicted positives:

$$Precision = \frac{TP}{TP + FP} \quad (69)$$

Recall (or sensitivity) measures the proportion of correctly predicted positive samples out of all actual positives:

$$Recall = \frac{TP}{TP + FN} \quad (70)$$

The F1-score provides a balance between precision and recall and is calculated as:

$$F1 - score = 2.\frac{Precision.Recall}{Precision + Recall} \quad (71)$$

These evaluation metrics are used to assess the performance of the final FLIO-ELM model on the test dataset. If the model performs well on the test set with low MSE and high accuracy, precision, recall, and F1-score, it indicates successful generalization of the learned parameters to unseen data. To further validate the robustness of the FLIO-ELM model, perform k-fold cross-validation on the entire dataset, including training and testing sets. Divide the data into k equal-sized subsets and use $k-1$ subsets for training and the remaining subset for testing. Repeat this process k times, with each subset serving as the test set exactly once. The cross-validation error E_{cv} is calculated as the average error across all folds:

$$E_{cv} = \frac{1}{k}\sum_{i=1}^{k} E_{test,i} \quad (72)$$

where $E_{test,i}$ represents the error for the i-th fold. This step helps assess the model's ability to generalize across different splits of the data, reducing the likelihood of overfitting.

By thoroughly evaluating the performance of the final FLIO-ELM model using these metrics and validation techniques, the reliability and generalization capabilities of the optimized model are ensured.

3.12. Regularization for Stability

In this step, a final refinement of the FLIO-ELM model is performed by adjusting its hyperparameters to maximize performance. Hyperparameters such as the number of hidden neurons L, the regularization parameters λ_1 and λ_2, the mutation rate μ, and the learning rate η require fine-tuning to optimize the model's accuracy and generalization capability. Let the number of hidden neurons L be initially set based on the dataset's complexity. A search process can be implemented to find the optimal number of hidden neurons that minimizes the error on the validation dataset. Let the set of candidate values for L be denoted as $L = \{L_1, L_2, \dots, L_m\}$, where each $L_j \in L$ represents a potential value for the number of hidden neurons. For each L_j, compute the hidden layer output H_j using the input weights $W_j \in R^{m\times L_j}$ and bias vector $b_j \in R^{1\times L_j}$:

$$H_j = g(XW_j + b_j) \quad (73)$$

For each candidate L_j update the output weights $\beta_j \in R^{L_j\times 1}$ using the least squares solution:

$$\beta_j = \left(H_j^T H_j\right)^{-1} H_j^T Y \quad (74)$$

Compute the predicted output $\hat{Y}_j$ for each candidate L_j:

$$\hat{Y}_j = H_j\beta_j \quad (75)$$

Evaluate the error E_j for each L_j using the mean squared error (MSE) on the validation set:

$$E_j = \frac{1}{n_{val}}\sum_{i=1}^{n_{val}}\left(y_{val,i} - \hat{y}_{j,i}\right)^2 \quad (76)$$

Select the value $L_{opt} \in L$ that minimizes E_j:

$$L_{opt} = \arg\min_{L_j\in L} E_j \quad (77)$$

After selecting L_{opt}, update the model's input weights and biases using the refined number of hidden neurons. Next, optimize the regularization parameters. λ_1 and λ_2, which control the penalty on the number of neurons and the magnitude of the weights and biases, respectively. Let $R = \{(\lambda_1, \lambda_2)\}$ represent the set of candidate regularization parameter pairs. For each pair $(\lambda_1, \lambda_2) \in R$, compute the penalty function $P(S_j)$ for the corresponding frog solution S_j as:

$$P(S_j) = \lambda_1.L_j + \lambda_2.\left(\frac{1}{mL_j}\sum_{k=1}^{m}\sum_{l=1}^{L_j} W_{kl}^2 + \frac{1}{L_j}\sum_{l=1}^{L_j} b_l^2\right) \quad (78)$$

Update the fitness $F_j^{(pen)}$ for each candidate pair based on the penalty function:

$$F_j^{(pen)} = \frac{1}{E_j} - P(S_j) \quad (79)$$

Select the optimal regularization parameters $(\lambda_1^{opt}, \lambda_2^{opt})$ that maximize the fitness $F_j^{(pen)}$.

$$(\lambda_1^{opt}, \lambda_2^{opt}) = arg \max_{(\lambda_1,\lambda_2)\in R} R_j^{(pen)} \quad (80)$$

Once the optimal hyperparameters for L, λ_1, and $\lambda_2$have been determined, proceed to refine the mutation rate μ and learning rate η to further enhance the optimization process. For the mutation rate μ, perform a grid search over a set of candidate values $M = \{\mu_1, \mu_2, \dots, \mu_k\}$. For each $\mu_l \in M$, evaluate the performance of the model by computing the validation error and fitness. Select the mutation rate μ_{opt} that yields the best performance. Finally, refine the learning rate η by performing a search over a range of candidate values. The learning rate controls the magnitude of updates to the weights and biases during the optimization process. Let $\eta \in \epsilon = \{\eta_1, \eta_2, \dots, \eta_r\}$ represent the set of candidate learning rates. For each $\eta_s \in \epsilon$, evaluate the model's performance and select the optimal learning rate η_{opt} based on the fitness and error reduction.

This final refinement ensures that the FLIO-ELM model operates at peak efficiency by selecting the optimal hyperparameters that minimize error and maximize fitness across various validation datasets.

3.13 Model Stability and Robustness Analysis

In this step, a comprehensive analysis of the FLIO-ELM model's stability and robustness will be performed to ensure consistent performance across various datasets and conditions. The objective is to evaluate how the optimized model responds to the data perturbations and assess its generalization capability under different scenarios. First, introduce noise into the input data X to test the model's sensitivity. Create a perturbed dataset $X_{perturbed}$ by adding Gaussian noise ϵ to the original input data:

$$X_{perturbed} = X + \epsilon \quad (81)$$

where $\epsilon \sim N(0, \sigma^2)$ and σ represents the standard deviation of the noise. Vary σ to simulate different levels of noise intensity.

Compute the hidden layer output $H_{perturbed}$ using the optimized input weights W^* and biases b^*.

$$X_{perturbed} = g(X_{perturbed} W^* + b^*) \quad (82)$$

Calculate the predicted output $\hat{Y}_{perturbed}$ for the perturbed data:

$$\hat{Y}_{perturbed} = H_{perturbed}\beta^* \quad (83)$$

Evaluate the error $E_{perturbed}$ between the predicted output $\hat{Y}_{perturbed}$ and the actual target values Y:

$$E_{perturbed} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{Y}_{perturbed,i})^2 \quad (84)$$

Compare $E_{perturbed}$ with the original error $E_{original}$ calculated on the unperturbed data to assess the model's robustness to noise. Next, perform a sensitivity analysis by varying the input features to determine the model's dependence on specific variables. For each input feature x_j, where $j = 1,2, \dots, m$, introduce a small perturbation δ_j and compute the change in the output:

$$\delta\hat{Y}_j = H(W_j^* + \delta_j)\beta^* - HW_j^*\beta^* \quad (85)$$

where W_j^* is the jth column of the weight matrix W^*, and δ_j is a small change applied to that column.

Compute the sensitivity S_j of the model to each input feature:

$$S_j = \frac{\|\delta\hat{Y}_j\|}{\|\delta_j\|} \quad (86)$$

where $\|.\|$ denotes the Euclidean norm. A higher S_j indicates greater sensitivity to the corresponding input feature.

Test the model's stability by subjecting it to K-fold cross-validation with different random seeds. For each fold k, train the model on a training set $D_{train}^{(k)}$ and evaluate it on a validation set $D_{val}^{(k)}$. Compute the validation error $E_{val}^{(k)}$:

$$E_{val}^{(k)} = \frac{1}{n_{val}}\sum_{i=1}^{n_{val}}(y_{val,i}^{(k)} - \hat{y}_{val,i}^{(k)})^2 \quad (87)$$

Calculate the mean validation error $\bar{E}_{val}$ across all folds:

$$\bar{E}_{val} = \frac{1}{K}\sum_{k=1}^{K} E_{val}^{(k)} \quad (88)$$

Compute the standard deviation σ_E of the validation errors to assess the model's stability:

$$\sigma_E = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(E_{val}^{(k)} - \bar{E}_{val})^2} \quad (89)$$

A lower σ_E indicates that the model's performance is consistent across different data splits, demonstrating stability. Furthermore, evaluate the model's robustness to adversarial attacks by introducing adversarial examples X_{adv} generated using the Fast Gradient Sign Method (FGSM). Compute X_{adv} as:

$$X_{adv} = X + \epsilon . sign(\nabla_X E) \quad (90)$$

where ϵ is a small perturbation parameter, $\nabla_X E$ represents the gradient of the error with respect to the input X, and sign(.) Denotes the sign function applied element-wise.

Compute the hidden layer output for the adversarial examples:

$$H_{adv} = g(X_{adv} W^* + b^*) \quad (91)$$

Calculate the predicted output $\hat{Y}_{adv}$ for the adversarial data:

$$\hat{Y}_{adv} = H_{adv} \beta^* \quad (92)$$

Evaluate the adversarial error E_{adv}:

$$E_{adv} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{y}_{adv,i}\right)^2 \quad (93)$$

Comparing E_{adv} with $E_{original}$ assesses the model's robustness to adversarial perturbations. Finally, analyze the model's response to different initializations of weights and biases. Repeat the entire optimization process with varying initial weights W_{int} and biases b_{init}:

$$W_{init}^{(l)} \sim U(-\alpha, \alpha), \qquad b_{init}^{(l)} \sim U(-\beta, \beta) \quad (94)$$

For $l = 1,2, \dots, L_{init}$, where L_{init} is the number of different initializations, and $U(-\alpha, \alpha)$ denotes a uniform distribution. Evaluate the performance of each model instance and compute the variance in errors:

$$\sigma_{E_{init}} = \sqrt{\frac{1}{L_{init}} \sum_{l=1}^{L_{init}} (E^{(l)} - \bar{E}_{init})^2} \quad (95)$$

where $\bar{E}_{init}$ is the mean error across all initializations. A low $\sigma_{E_{init}}$ indicates that the model's performance is not heavily dependent on the initial parameter values, demonstrating robustness.

By conducting these analyses, the stability and robustness of the FLIO-ELM model are thoroughly evaluated, ensuring reliable performance in practical applications.

3.14. Convergence Check with Error Feedback

This step assesses the computational complexity and scalability of the FLIO-ELM model are assessed. The goal is to evaluate the time complexity and memory usage, ensuring the model remains efficient even as the dataset size and the number of hidden neurons increase. Let n represent the number of data samples, m represent the number of input features, and L denote the number of hidden neurons in the ELM model. The computational complexity for calculating the hidden layer output H is:

$$O(n.m.L) \quad (96)$$

Since the matrix multiplication between the input data $X \in R^{n \times m}$ and the input weights $W \in R^{m \times L}$ requires $n.m.L$ operations. The calculation of the output weights β using the least squares solution involves solving the following system:

$$\beta = (H^T H)^{-1} H^T Y \quad (97)$$

The matrix inversion dominates the computational complexity for solving this system. $(H^T H)^{-1}$, which has a complexity of $O(L^3)$.Next, analyze the memory requirements. The memory needed for storing the input weights $W \in R^{m \times L}$, biases $b \in R^{1 \times L}$, and output weights $\beta \in R^{L \times 1}$ is:

$$O(m.L + L + L) \quad (98)$$

This expression simplifies to $O(m \cdot L)$, which grows linearly with the number of features m and hidden neurons L. Additionally, the model's scalability is tested by increasing the dataset size and observing the impact on the training time and memory usage. As the number of samples n increases, the complexity for computing H scales linearly with n, while the least squares solution remains dependent on L. Thus, evaluating both time and space complexity ensures the model's efficiency when scaling to larger datasets or more complex architectures.

3.15. Final ELM Model Output

In this step, the scalability and performance of the FLIO-ELM model in a distributed computing environment will be tested. This ensures the model can handle larger datasets and more complex optimization tasks by distributing the computational workload across multiple processors. First, partition the dataset $D = \{(X, Y)\}$, where $X \in R^{n \times m}$ represents the input features and $Y \in R^{n \times 1}$ is the target vector, into smaller subsets $D_p = \{(X_p, Y_p)\}$,

where $p = 1,2,\dots,P$ represents the number of partitions, and P is the total number of processors available. Each subset D_p contains approximately $\frac{n}{p}$ samples. The computation of the hidden layer output H_p for each partition is performed in parallel on each processor. For each partition D_p, compute H_p as:

$$H_p = g(X_P W + b) \quad (99)$$

where $W \in R^{m \times L}$ and $b \in R^{m \times L}$ are the input weights and biases shared across all processors, and $g(.)$ is the activation function applied element-wise. The complexity for computing H_p on each processor is reduced to $O\left(\frac{n}{P}.m.L\right)$, ensuring faster execution.

After computing the hidden layer output on all partitions, the outputs H_p from each processor are aggregated to form the global hidden layer matrix $H \in R^{n \times L}$. The aggregation step can be performed as:

$$H = \bigcup_{p=1}^{P} H_p \quad (100)$$

Once H is constructed, the output weights $\beta \in R^{L \times 1}$ are computed using the least squares method, which minimizes the error between the predicted output $\hat{Y} \in R^{n \times 1}$ and the actual target values Y. The solution for β is obtained by solving:

$$\beta = (H^T H)^{-1} H^T Y \quad (101)$$

The complexity for solving this system remains $O(L^3)$, but since the hidden layer output H is computed in parallel, the overall computational time for calculating H and aggregating the results is significantly reduced. Next, evaluate the performance of the FLIO-ELM model on a large dataset by calculating the predicted output $\hat{Y}$ on the full dataset:

$$\hat{Y} = H\beta \quad (102)$$

Compute the error E using the mean squared error (MSE) metric:

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (103)$$

To further improve scalability, distribute the optimization steps of the FLIO algorithm across multiple processors. Partition the frog population $S = \{S_1, S_2, \dots, S_P\}$ into subpopulations S_p, where each subpopulation is optimized in parallel. For each frog S_i in the subpopulation, the fitness is computed based on the performance of the corresponding ELM model.

The fitness F_i for each frog is computed as:

$$F_i = \frac{1}{E_i} \quad (104)$$

where E_i represents the error for the ith frog's ELM model. The parallel execution of the fitness evaluation and optimization steps ensures that the FLIO algorithm can efficiently handle larger populations and search spaces.

After each parallel optimization iteration, the best-performing frogs from each subpopulation are exchanged across processors to ensure diversity and avoid local optima. This exchange can be modeled as:

$$S_{global} = best\left(\bigcup_{p=1}^{P} S_P\right) \quad (105)$$

where S_{global} represents the set of globally best-performing frogs.

Finally, a distributed evaluation of the model's generalization capability will be performed by conducting k-fold cross-validation across processors. Each fold is processed independently on separate processors, with the validation error for each fold $E_{val,k}$ computed as:

$$E_{val,k} = \frac{1}{n_{val}}\sum_{i=1}^{n_{val}}\left(y_{val,i} - \hat{y}_{val,i}\right)^2 \quad (106)$$

The average validation error $\bar{E}_{val}$ is the computed as:

$$\bar{E}_{val} = \frac{1}{K}\sum_{k=1}^{K} E_{val,k} \quad (107)$$

This distributed computing approach allows the FLIO-ELM model to scale efficiently with increasing data size and complexity, ensuring fast execution and improved generalization performance.

Algorithm: FLIO-ELM

Input:

- Dataset $D = \{(X, Y)\}$ where X represents input features and Y represents target values
- Parameters: number of frogs P, number of memeplexes M, number of hidden neurons L, mutation rate μ, regularization parameters λ_1 and λ_2

Output:

- Optimized ELM model with refined weights, biases, and output weights

Procedure:

1. Problem Formulation: Define the problem to be solved using ELM, including the input and target datasets.
2. Initialize ELM Parameters: Randomly initialize input weights, biases, and the number of hidden neurons for the ELM model.
3. Frog Population Initialization for ELM Optimization: Initialize a population of frogs where each frog represents an ELM model with a unique combination of weights and biases.
4. The division into Memeplexes: Divide the frog population into memeplexes based on fitness rankings for local optimization.
5. Local Search within Memeplexes: Perform a local search in each memeplex by moving the worst-performing frogs toward the best-performing frogs.
6. Adaptive Frog Leaping Strategy: Adjust the leap size dynamically based on fitness improvement in each memeplex.
7. Mutation Mechanism for Diversity: Apply mutation to a subset of frogs to introduce random variations in their weights and biases.
8. Penalty Function for Efficient ELM Models: Apply a penalty function to discourage overly complex models and favor efficient configurations.
9. Hybrid Learning Step: FLIO and Least Squares Integration: Refine the ELM model by combining FLIO-optimized weights with most minuscule squares adjustments for output weights.
10. Convergence Check and Stopping Criteria: Monitor the change in fitness or error and stop the process if predefined convergence criteria are met.
11. Performance Evaluation of the Final FLIO-ELM Model: Evaluate the performance of the optimized model on a test dataset using error metrics and other evaluation criteria.
12. Final Refinement and Hyperparameter Optimization: Fine-tune hyperparameters such as the number of hidden neurons, mutation rate, and regularization parameters.
13. Model Stability and Robustness Analysis: Test the stability and robustness of the model by introducing noise, performing cross-validation, and assessing sensitivity to input features.
14. Computational Complexity and Scalability Assessment: Analyze the time and space complexity of the model to assess its efficiency for larger datasets.
15. Distributed Scalability and Performance in FLIO-ELM: Distribute the computation across multiple processors to handle larger datasets and optimize the model in a parallel computing environment.

4. DATASET

The Gestational Diabetes Mellitus (GDM) dataset encompasses detailed records from 3,525 cases with 17 critical variables. This dataset serves as a resource for analyzing factors contributing to gestational diabetes. Key features include demographic data such as age and number of pregnancies, medical indicators like BMI, HDL levels, and blood pressure, and specific risk factors such as family history, sedentary lifestyle, and PCOS. Unique attributes like unexplained prenatal loss and large child or birth defect indicators provide insights into prenatal complications. The dataset also tracks clinical test results, including oral glucose tolerance test (OGTT) and hemoglobin levels, which are essential for diagnosing and monitoring prediabetes. Focused on diabetes types, blurred vision, and autoimmune disorders, it captures a comprehensive view of health outcomes related to gestational diabetes. This dataset aids in predictive modeling, enhancing early diagnosis and personalized intervention strategies.

Table 1: GDM Dataset Features Description

Feature Name	Description
Case Number	Unique identifier for each patient in the dataset.
Age	Age of the individual at the time of data collection.
Number of Pregnancies	Total number of pregnancies experienced by the individual.
Gestation	Duration of pregnancy in weeks.
BMI	Body Mass Index, a measure of body fat based on height and weight.
HDL	High-Density Lipoprotein cholesterol levels, indicating heart health.
Family History	Indicator of a family history of diabetes or related conditions.
Unexplained Prenatal Loss	History of unexplained miscarriage or pregnancy loss.
Large Child or Birth Defect	Record of delivering a child with high birth weight or congenital anomalies.

PCOS	Presence of Polycystic Ovary Syndrome, a risk factor for diabetes.
Systolic BP	Systolic blood pressure, measured in mmHg.
Diastolic BP	Diastolic blood pressure, measured in mmHg.
OGTT	Oral Glucose Tolerance Test results, used for diabetes diagnosis.
Hemoglobin	Hemoglobin levels, reflecting the individual's blood health.
Sedentary Lifestyle	Indicator of a lack of physical activity.
Prediabetics	Status indicating a prediabetic condition.

5. RESULTS AND DISCUSSION

5.1. Classification Accuracy and F-Measure Analysis

Classification accuracy measures the proportion of correctly classified instances among all predictions, reflecting the overall reliability of a model. The F-measure combines precision and recall, providing a balanced evaluation of a model's ability to handle false positives and false negatives. These metrics are critical in LADA prediction, where accurate and balanced classification ensures effective diagnosis and early intervention.

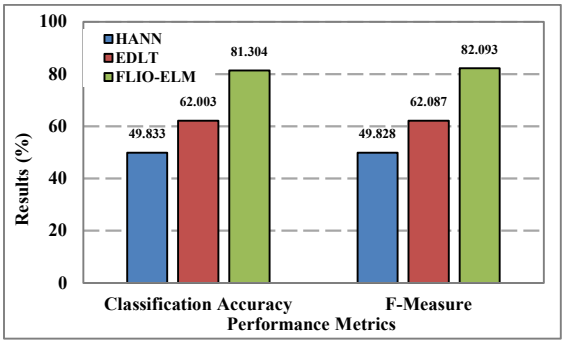

Figure 1: Classification Accuracy and F-Measure

Figure 1 illustrates the performance comparison of three models—HANN, EDLT, and FLIO-ELM—using classification accuracy and F-measure metrics. HANN records the lowest classification accuracy at 49.833%, highlighting limitations in correctly classifying LADA instances. EDLT improves upon this, achieving an accuracy of 62.003%, demonstrating a better capability to handle LADA prediction. FLIO-ELM outperforms both, with an accuracy of 81.304%, showcasing its robust design and optimization efficiency. The F-measure analysis aligns with the accuracy trends. HANN shows limited performance with a value of 49.828%, indicating challenges in balancing precision and recall. EDLT improves this balance, attaining a value of 62.087%. FLIO-ELM achieves the highest F-measure of 82.093%, highlighting its ability to reduce false predictions effectively. The results establish FLIO-ELM as a superior approach in LADA prediction, demonstrating its efficiency in addressing both classification and balance challenges.

5.2. False Discovery and Omission Analysis

False discovery rate (FDR) quantifies the proportion of false positive predictions out of all positive predictions, providing a measure of the model's precision. A high FDR indicates a larger number of incorrect positive classifications, which can lead to unnecessary interventions or misdiagnoses. False omission rate (FOR) represents the proportion of false negatives among all negative predictions, measuring the extent of missed true cases. A high FOR signifies that the model fails to detect a significant number of actual positive cases, which can delay critical treatment. In the context of LADA, minimizing both FDR and FOR is essential to ensure accurate and timely diagnoses, balancing precision and recall to achieve reliable outcomes in clinical applications.

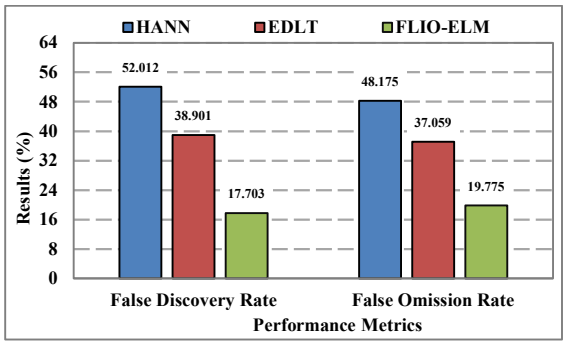

Figure 2 : False Discovery and False Omission

Figure 2 demonstrates the comparative performance of HANN, EDLT, and FLIO-ELM models with respect to FDR and FOR. HANN exhibits the highest FDR at 52.012%, suggesting considerable inaccuracies in positive predictions. EDLT reduces this rate to 38.901%, indicating a moderate improvement in precision over HANN. FLIO-ELM records the lowest FDR of 17.703%, reflecting its ability to minimize false positive rates effectively. In LADA prediction, FLIO-ELM's performance underscores its reliability in correctly identifying cases with minimal misclassifications, thereby enhancing the precision of diagnostic decisions. These results demonstrate the limitations of HANN and highlight EDLT's marginal

improvements, while establishing FLIO-ELM as a superior approach in managing positive classification errors. The FOR analysis presents a similar trend. HANN reports the highest value at 48.175%, indicating significant challenges in detecting true negative cases. EDLT shows improvement, achieving a FOR of 37.059%, demonstrating enhanced reliability in reducing missed diagnoses. FLIO-ELM records the lowest FOR at 19.775%, effectively minimizing missed positive cases. These findings highlight FLIO-ELM's effectiveness in balancing prediction errors, reinforcing its efficiency in LADA prediction tasks.

5.3. Positive and Negative Likelihood Analysis

Positive likelihood ratio (PLR) measures how much a positive result increases the probability of having a condition, with higher values indicating better diagnostic capability in identifying true positives. Negative likelihood ratio (NLR) assesses how much a negative result decreases the probability of having the condition, with lower values indicating fewer missed cases. These metrics are essential in evaluating model effectiveness for predictive healthcare applications such as LADA diagnosis.

Table 2. Positive and Negative Likelihood

Algorithms	Positive Likelihood Ratio (%)	Negative Likelihood Ratio (%)
HANN	0.996	1.004
EDLT	1.615	0.605
FLIO-ELM	4.234	0.225

Table 2 compares PLR and NLR for HANN, EDLT, and FLIO-ELM models. HANN shows a PLR of 0.996, indicating minimal capability to differentiate true positives from false positives. The NLR for HANN is 1.004, highlighting significant limitations in reducing false negatives, thereby compromising reliability for accurate LADA prediction. EDLT improves PLR to 1.615, demonstrating better effectiveness in identifying true positives. The NLR for EDLT is reduced to 0.605, indicating moderate improvement in minimizing false negatives. These results suggest EDLT has an enhanced but limited diagnostic performance compared to HANN. FLIO-ELM achieves a PLR of 4.234, representing a substantial improvement in identifying true positives with high diagnostic accuracy. Its NLR of 0.225 reflects a significant reduction in false negatives, showcasing its robustness in ensuring comprehensive detection. These results position FLIO-ELM as a reliable model for accurate and efficient LADA prediction. The comparative analysis demonstrates that FLIO-ELM significantly outperforms both HANN and EDLT across PLR and NLR metrics. Its strong diagnostic reliability makes it suitable for real-world healthcare applications requiring precise and consistent detection capabilities.

6. CONCLUSION

This research addresses the challenges in accurately diagnosing Latent Autoimmune Diabetes in Adults (LADA), a form of diabetes blending characteristics of type 1 and type 2. Misclassification of LADA as type 2 diabetes remains a critical issue, often delaying appropriate interventions and compromising patient outcomes. Existing classification algorithms struggle with overlapping symptoms, high-dimensional data, and the dynamic nature of LADA's progression, limiting their ability to distinguish this subtype accurately. To resolve these challenges, the Frog Leap Inspired Optimization-based Extreme Learning Machine (FLIO-ELM) was proposed. This hybrid model integrates bio-inspired optimization with the computational efficiency of Extreme Learning Machines to enhance classification performance. FLIO-ELM optimizes input weights and biases by simulating frog leap strategies, ensuring better convergence and feature selection. The adaptive leap mechanism balances exploration and exploitation, refining predictions by effectively handling complex datasets and reducing false discovery and omission rates. The experimental analysis demonstrates FLIO-ELM's superior performance compared to existing models. It achieved an accuracy of 81.304% while significantly minimizing false positive and false negative rates, as evidenced by improved precision, recall, and F-measure scores. These results validate FLIO-ELM as a robust, efficient, and reliable model for LADA prediction, offering substantial potential for clinical applications and personalized treatment strategies.

REFERENCES:

[1]. R. M. Wasserman, S. R. Patton, M. A. Clements, D. Guffey, D. D. Schwartz, and B. J. Anderson, "Risky self-management behaviors in adolescents with type 1 diabetes: Measurement validation for the Diabetes-Specific Risk-Taking Inventory," *Pediatr Diabetes*, vol. 23, no. 7, pp. 1113–1121, 2022, doi: 10.1111/pedi.13387.

[2]. N. Nirala, R. Periyasamy, B. K. Singh, and A. Kumar, "Detection of type-2 diabetes using characteristics of toe photoplethysmogram by applying support vector machine," *Biocybern Biomed Eng*, vol. 39, no. 1, pp. 38–51, Jan. 2019, doi: 10.1016/j.bbe.2018.09.007.

[3]. J. A. Carter, C. S. Long, B. P. Smith, T. L. Smith, and G. L. Donati, "Combining elemental analysis of toenails and machine learning techniques as a non-invasive diagnostic tool for the robust classification of type-2 diabetes," *Expert Syst Appl*, vol. 115, pp. 245–255, Jan. 2019, doi: 10.1016/j.eswa.2018.08.002.

[4]. S. P. Chatrati*et al.*, "Smart home health monitoring system for predicting type 2 diabetes and hypertension," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 3, pp. 862–870, Mar. 2022, doi: 10.1016/j.jksuci.2020.01.010.

[5]. K. Kannadasan, D. R. Edla, and V. Kuppili, "Type 2 diabetes data classification using stacked autoencoders in deep neural networks," Clin Epidemiol Glob Health, vol. 7, no. 4, pp. 530–535, Dec. 2019, doi: 10.1016/j.cegh.2018.12.004.

[6]. A. Talaei-Khoei and J. M. Wilson, "Identifying people at risk of developing type 2 diabetes: A comparison of predictive analytics techniques and predictor variables," *Int J Med Inform*, vol. 119, pp. 22–38, 2018, doi: 10.1016/j.ijmedinf.2018.08.008.

[7]. B. P. Nguyen *et al.*, "Predicting the onset of type 2 diabetes using wide and deep learning with electronic health records," *Comput Methods Programs Biomed*, vol. 182, p. 105055, Dec. 2019, doi: 10.1016/j.cmpb.2019.105055.

[8]. J. Ramkumar and R. Vadivel, "Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN)," World Journal of Engineering, vol. 15, no. 2, pp. 306–311, 2018, doi: 10.1108/WJE-08-2017-0260.

[9]. J. Ramkumar and R. Vadivel, CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks, vol. 556. 2017. doi: 10.1007/978-981-10-3874-7_14.

[10]. D. Jayaraj, J. Ramkumar, M. Lingaraj, and B. Sureshkumar, "AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network," International Journal of Computer Networks and Applications, vol. 10, no. 1, pp. 119–129, 2023, doi: 10.22247/ijcna/2023/218516.

[11]. B. Paul and B. Karn, "Diabetes Mellitus Prediction using Hybrid Artificial Neural Network," in 2021 IEEE Bombay Section Signature Conference (IBSSC), 2021, pp. 1–5. doi: 10.1109/IBSSC53889.2021.9673397.

[12]. P. Singh, S. Silakari, and S. Agrawal, "An Efficient Deep Learning Technique for Diabetes Classification and Prediction Based on Indian Diabetes Dataset," in 2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS), 2023, pp. 487–491. doi: 10.1109/ICTACS59847.2023.10390518.

[13]. Y. Ying et al., "An enhanced machine learning approach for effective prediction of IgA nephropathy patients with severe proteinuria based on clinical data," ComputBiol Med, vol. 173, p. 108341, 2024, doi: https://doi.org/10.1016/j.compbiomed.2024.108341.

[14]. N. Musacchio et al., "A transparent machine learning algorithm uncovers HbA1c patterns associated with therapeutic inertia in patients with type 2 diabetes and failure of metformin monotherapy," Int J Med Inform, vol. 190, p. 105550, 2024, doi: https://doi.org/10.1016/j.ijmedinf.2024.105550.

[15]. Prof. M. Madan, Ms. R. Madan, and D. P. Thakur, "Analysing The Patient Sentiments in Healthcare Domain Using Machine Learning," Procedia Comput Sci, vol. 238, pp. 683–690, 2024, doi: https://doi.org/10.1016/j.procs.2024.06.077.

[16]. S. L. Cichosz, M. H. Jensen, O. Hejlesen, S. D. Henriksen, A. M. Drewes, and S. S. Olesen, "Prediction of pancreatic cancer risk in patients with new-onset diabetes using a machine learning approach based on routine biochemical parameters," Comput Methods Programs Biomed, vol. 244, p. 107965, 2024, doi: https://doi.org/10.1016/j.cmpb.2023.107965.

[17]. S. Nayak et al., "Development of a machine learning-based model for the prediction and progression of diabetic kidney disease: A single centred retrospective study," Int J Med Inform, vol. 190, p. 105546, 2024, doi: https://doi.org/10.1016/j.ijmedinf.2024.105546.

[18]. H.-T. Keng et al., "Derivation and validation of heart rate variability based Machine

learning prognostic models for patients with suspected sepsis," Biomed Signal Process Control, vol. 99, p. 106854, 2025, doi: https://doi.org/10.1016/j.bspc.2024.106854.

[19]. Kokila, R. Jain, A. Munde, and Z. A. Ansari, "Determinants of Adoption of Mobile Health Applications: A Machine Learning Approach," Procedia Comput Sci, vol. 235, pp. 1568–1576, 2024, doi: https://doi.org/10.1016/j.procs.2024.04.148.

[20]. M. M. Rahman et al., "Empowering early detection: A web-based machine learning approach for PCOS prediction," Inform Med Unlocked, vol. 47, p. 101500, 2024, doi: https://doi.org/10.1016/j.imu.2024.101500.

[21]. A. O. Adebanji, C. Asare, and S. A. Gyamerah, "Predictive analysis on the factors associated with birth Outcomes: A machine learning perspective," Int J Med Inform, vol. 189, p. 105529, 2024, doi: https://doi.org/10.1016/j.ijmedinf.2024.105529.

[22]. Y. Wang et al., "Development and validation of machine learning models for predicting cancer-related fatigue in lymphoma survivors," Int J Med Inform, vol. 192, p. 105630, 2024, doi: https://doi.org/10.1016/j.ijmedinf.2024.105630.

[23]. S. A. Tanim, A. R. Aurnob, T. E. Shrestha, M. D. R. I. Emon, M. F. Mridha, and M. S. U. Miah, "Explainable deep learning for diabetes diagnosis with DeepNetX2," Biomed Signal Process Control, vol. 99, p. 106902, 2025, doi: https://doi.org/10.1016/j.bspc.2024.106902.

[24]. H. Fu, Y. Yang, V. K. Mishra, and C.-C. J. Kuo, "Subspace learning machine (SLM): Methodology and performance evaluation," J Vis Commun Image Represent, vol. 98, p. 104058, 2024, doi: https://doi.org/10.1016/j.jvcir.2024.104058.

[25]. G. Dharmarathne, M. Bogahawaththa, M. McAfee, U. Rathnayake, and D. P. P. Meddage, "On the diagnosis of chronic kidney disease using a machine learning-based interface with explainable artificial intelligence," Intelligent Systems with Applications, vol. 22, p. 200397, 2024, doi: https://doi.org/10.1016/j.iswa.2024.200397.

[26]. P. Cihan, A. Saygılı, C. Şahin Ermutlu, U. Aydın, and Ö. Aksoy, "AI-aided cardiovascular disease diagnosis in cattle from retinal images: Machine learning vs. deep learning models," Comput Electron Agric, vol. 226, p. 109391, 2024, doi: https://doi.org/10.1016/j.compag.2024.109391.

[27]. M. Kawarkhe and P. Kaur, "Prediction of Diabetes Using Diverse Ensemble Learning Classifiers," Procedia Comput Sci, vol. 235, pp. 403–413, 2024, doi: https://doi.org/10.1016/j.procs.2024.04.040.

[28]. A. Mittal et al., "Predicting prolonged length of stay following revision total knee arthroplasty: A national database analysis using machine learning models," Int J Med Inform, vol. 192, p. 105634, 2024, doi: https://doi.org/10.1016/j.ijmedinf.2024.105634.

[29]. Y. Luo, R. Dong, J. Liu, and B. Wu, "A machine learning-based predictive model for the in-hospital mortality of critically ill patients with atrial fibrillation," Int J Med Inform, vol. 191, p. 105585, 2024, doi: https://doi.org/10.1016/j.ijmedinf.2024.105585.

[30]. B. Singh, S. Indu, and S. Majumdar, "Comparison of machine learning algorithms for classification of Big Data sets," Theor Comput Sci, vol. 1024, p. 114938, 2025, doi: https://doi.org/10.1016/j.tcs.2024.114938.

[31]. S. Hur et al., "Development, validation, and usability evaluation of machine learning algorithms for predicting personalized red blood cell demand among thoracic surgery patients," Int J Med Inform, vol. 191, p. 105543, 2024, doi: https://doi.org/10.1016/j.ijmedinf.2024.105543.

[32]. M. Zhang et al., "Design of a differentiable L-1 norm for pattern recognition and machine learning," Pattern Recognit Lett, vol. 186, pp. 126–132,2024,doi: https://doi.org/10.1016/j.patrec.2024.09.020.

[33]. N. K. Ojha, A. Pandita, and J. Ramkumar, "Cyber security challenges and dark side of AI: Review and current status," in Demystifying the Dark Side of AI in Business, 2024, pp. 117–137. doi: 10.4018/979-8-3693-0724-3.ch007.

[34]. M. P. Swapna and J. Ramkumar, "Multiple Memory Image Instances Stratagem to Detect Fileless Malware," in Communications in Computer and Information Science, S. Rajagopal, K. Popat, D. Meva, and S. Bajeja, Eds., Cham: Springer Nature Switzerland, 2024, pp. 131–140. doi: 10.1007/978-3-031-59100-6_11.

[35]. J. Ramkumar, S. S. Dinakaran, M. Lingaraj, S. Boopalan, and B. Narasimhan, "IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion," in Lecture Notes in Electrical Engineering, K. Murari, N. Prasad Padhy, and S. Kamalasadan, Eds., Singapore: Springer Nature Singapore, 2023, pp. 17–27. doi: 10.1007/978-981-19-8353-5_2.

[36]. R. Jaganathan and V. Ramasamy, "Performance modeling of bio-inspired routing protocols in Cognitive Radio Ad Hoc Network to reduce end-to-end delay," Int. J. Intell. Eng. Syst., vol. 12, no. 1, pp. 221–231, 2019, doi: 10.22266/IJIES2019.0228.22.

[37]. L. Mani, S. Arumugam, and R. Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," ACM Int. Conf. Proceeding Ser., pp. 1–5, Dec. 2022, doi: 10.1145/3590837.3590907.

[38]. J. Ramkumar, R. Vadivel, B. Narasimhan, S. Boopalan, and B. Surendren, "Gallant Ant Colony Optimized Machine Learning Framework (GACO-MLF) for Quality of Service Enhancement in Internet of Things-Based Public Cloud Networking," in Data Science and Communication. ICTDsC 2023. Studies in Autonomic, Data-driven and Industrial Computing, J. M. R. S. Tavares, J. J. P. C. Rodrigues, D. Misra, and D. Bhattacherjee, Eds., Singapore: Springer Nature Singapore, 2024, pp. 425–438. doi: 10.1007/978-981-99-5435-3_30.

[39]. J. Ramkumar, K. S. Jeen Marseline, and D. R. Medhunhashini, "Relentless Firefly Optimization-Based Routing Protocol (RFORP) for Securing Fintech Data in IoT-Based Ad-Hoc Networks," Int. J. Comput. Networks Appl., vol. 10, no. 4, pp. 668–687, 2023, doi: 10.22247/ijcna/2023/223319.

a. Senthilkumar, J. Ramkumar, M. Lingaraj, D. Jayaraj, and B. Sureshkumar, "Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing," Int. J. Comput. Networks Appl., vol. 10, no. 2, pp. 217–230, 2023, doi: 10.22247/ijcna/2023/220737.

[40]. J. Ramkumar and R. Vadivel, "Improved Wolf prey inspired protocol for routing in cognitive radio Ad Hoc networks," Int. J. Comput. Networks Appl., vol. 7, no. 5, pp. 126–136, 2020, doi: 10.22247/ijcna/2020/202977.

[41]. D. Jayaraj, J. Ramkumar, M. Lingaraj, and B. Sureshkumar, "AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network," Int. J. Comput. Networks Appl., vol. 10, no. 1, pp. 119–129, Jan. 2023, doi: 10.22247/ijcna/2023/218516.

[42]. R. Vadivel and J. Ramkumar, "QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications," Inc. Internet Things Healthc. Appl. Wearable Devices, pp. 109–121, 2019, doi: 10.4018/978-1-7998-1090-2.ch006.

[43]. P. Menakadevi and J. Ramkumar, "Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data," 2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022, pp. 1–5, Mar. 2022, doi: 10.1109/ICACTA54488.2022.9753203.

[44]. J. Ramkumar, A. Senthilkumar, M. Lingaraj, R. Karthikeyan, and L. Santhi, "Optimal Approach for Minimizing Delays in Iot-Based Quantum Wireless Sensor Networks Using Nm-Leach Routing Protocol," J. Theor. Appl. Inf. Technol., vol. 102, no. 3, pp. 1099–1111, 2024, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85185481011&partnerID=40&md5=bf0ff974ceabc0ad58e589b28797c684

[45]. M. Lingaraj, T. N. Sugumar, C. S. Felix, and J. Ramkumar, "Query aware routing protocol for mobility enabled wireless sensor network," Int. J. Comput. Networks Appl., vol. 8, no. 3, pp. 258–267, 2021, doi: 10.22247/ijcna/2021/209192.

[46]. R. Jaganathan and R. Vadivel, "Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks," Int. J. Comput. Digit. Syst., vol. 10, no. 1, pp. 1063–1074, 2021, doi: 10.12785/ijcds/100196.

[47]. J. Ramkumar and R. Vadivel, "Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN)," World J. Eng., vol. 15, no. 2, pp. 306–311, 2018, doi: 10.1108/WJE-08-2017-0260.

[48]. J. Ramkumar and R. Vadivel, "Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks," Wirel. Pers. Commun., vol. 120, no. 2, pp. 887–909, Apr. 2021, doi: 10.1007/s11277-021-08495-z.

[49]. J. Ramkumar, R. Vadivel, and B. Narasimhan, "Constrained Cuckoo Search Optimization Based Protocol for Routing in Cloud Network," Int. J. Comput. Networks Appl.,

vol. 8, no. 6, pp. 795–803, 2021, doi: 10.22247/ijcna/2021/210727.

[50]. [19] J. Ramkumar and R. Vadivel, "Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network," Int. J. Comput. Networks Appl., vol. 8, no. 4, pp. 455–464, 2021, doi: 10.22247/ijcna/2021/209711.

[51]. J. Ramkumar and R. Vadivel, "CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks," in Advances in Intelligent Systems and Computing, Springer Verlag, 2017, pp. 145–153. doi: 10.1007/978-981-10-3874-7_14.

[52]. J. Ramkumar, C. Kumuthini, B. Narasimhan, and S. Boopalan, "Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol," in 2022 International Conference on Advanced Computing Technologies and Applications, ICACTA 2022, 2022. doi: 10.1109/ICACTA54488.2022.9752899.

[53]. S. P. Geetha, N. M. S. Sundari, J. Ramkumar, and R. Karthikeyan, "Energy Efficient Routing in Quantum Flying Ad Hoc Network (Q-Fanet) Using Mamdani Fuzzy Inference Enhanced Dijkstra ' S Algorithm (Mfi-Eda)," J. Theor. Appl. Inf. Technol., vol. 102, no. 9, pp. 3708–3724, 2024.

[54]. M. P. Swapna, J. Ramkumar, and R. Karthikeyan, "Energy-Aware Reliable Routing with Blockchain Security for Heterogeneous Wireless Sensor Networks BT - Advances in Information Communication Technology and Computing," V. Goar, M. Kuri, R. Kumar, and T. Senjyu, Eds., Singapore: Springer Nature Singapore, 2025, pp. 713–723.

[55]. Bio-inspired intelligence for smart decision-making. Sri Krishna Arts and Science College, India; Auckland University of Technology, New Zealand; Mata Sundri University Girls College, Mansa, India: IGI Global, 2024. doi: 10.4018/979-8-3693-5276-2.

[56]. S. P. Priyadharshini and J. Ramkumar, "Mappings Of Plithogenic Cubic Sets," Neutrosophic Sets Syst., vol. 79, pp. 669–685, 2025, doi: 10.5281/zenodo.14607210.

[57]. R. Karthikeyan and R. Vadivel, "Proficient Dazzling Crow Optimization Routing Protocol (PDCORP) for Effective Energy Administration in Wireless Sensor Networks," in IEEE International Conference on Electrical, Electronics, Communication and Computers, ELEXCOM 2023, 2023, pp. 1–6. doi: 10.1109/ELEXCOM58812.2023.10370559.

[58]. R. Karthikeyan and R. Vadivel, "Boosted Mutated Corona Virus Optimization Routing Protocol (BMCVORP) for Reliable Data Transmission with Efficient Energy Utilization," Wirel. Pers. Commun., vol. 135, no. 4, pp. 2281–2301, 2024, doi: 10.1007/s11277-024-11155-7.

[59]. R. Karthikeyan and R. Vadivel, "Boosted Mutated Corona Virus Optimization Routing Protocol (BMCVORP) for Reliable Data Transmission with Efficient Energy Utilization," Wirel. Pers. Commun., vol. 135, no. 4, pp. 2281–2301, 2024, doi: 10.1007/s11277-024-11155-7.