Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 11, No. 3, August, Pp. 469-480

A Memory-efficient and gradient-stable lightweight ANFIS for real-time humidity prediction in precision agriculture

Eddy Nurraharjo[*1], Ema Utami[1], Kusrini[1], Kumara Ari Yuana[1]
Universitas Amikom Yogyakarta, Indonesia[1]

Article Info
Keywords:
Lightweight ANFIS, Gradient-Stable Learning, Embedded AI, Real-Time Humidity Prediction, Precision Agriculture IoT

Article history:
Received: January 21, 2026
Accepted: April 27, 2026
Published: August 01, 2026

Cite:
E. Nurraharjo, E. Utami, Kusrini, and K. Ari Yuana, "A Memory-Efficient and Gradient-Stable Lightweight ANFIS for Real-Time Humidity Prediction in Precision Agriculture", *KINETIK*, vol. 11, no. 3, Aug. 2026.
https://doi.org/10.22219/kinetik.v11i3.2700

*Corresponding author.
Eddy Nurraharjo
E-mail address:
eddynurraharjo@students.amikom.ac.id

Abstract
Precision agriculture requires artificial intelligence solutions that are both accurate and deployable on resource-constrained hardware. however, conventional machine learning models require excessive memory while traditional ANFIS architectures suffer from training instability. This study developed a memory-efficient and gradient-stable lightweight Adaptive Neuro-Fuzzy Inference System (ANFIS) for real-time humidity prediction on microcontroller-class devices. The proposed architecture strategically reduced the rule base from 27 to only 4 interpretable fuzzy rules and limited membership functions to two per input, achieving an 85.2% reduction in learnable parameters. A gradient-stable training mechanism was introduced, combining physics-informed parameter initialization with adaptive gradient clipping to prevent gradient explosion. The model was trained and validated using 31,474 real-world greenhouse samples collected over 218 days, with 80% allocated for training and 20% for temporal testing. Experimental results demonstrated that the gradient-stable architecture successfully converged from a catastrophic R^2 of -64.08 to 0.9148, with a root mean square error of 1.32% and mean absolute error of 1.05%. The model required only 0.211 KB of memory, representing a 99.9% reduction compared to baseline Random Forest models, while achieving inference time of 8.2 milliseconds on Arduino UNO. The system was successfully deployed on three independent hardware modules, maintaining consistent performance with average RMSE of 1.99% over 168 hours of continuous operation. This study concludes that strategic simplification and stability-aware training enable interpretable neuro-fuzzy systems to operate effectively on ultra-low-resource devices, bridging the gap between predictive accuracy and hardware feasibility in embedded agricultural IoT applications.

1. Introduction

Precision agriculture is increasingly supported by Internet of Things (IoT) systems because sensor networks enable continuous environmental sensing, real-time monitoring, and data-based farm management decisions [1]. In controlled-environment agriculture, relative humidity is a key microclimatic variable because it affects crop growth conditions, disease pressure, and product quality. This requirement is especially important in mushroom cultivation, where recent studies report that maintaining humidity around 85%–90% can improve yield and cultivation quality [2]. At the same time, edge computing and Tiny Machine Learning (TinyML) have made it possible to run lightweight machine-learning models directly on low-power devices, reducing cloud dependence, communication latency, and energy use [3], [4]. For agricultural prediction and control, ANFIS remains attractive because it combines the learning capability of neural networks with the interpretability of fuzzy logic, making it suitable for nonlinear environmental modeling [5].

Although ANFIS has strong modeling potential, its conventional structure is not always suitable for deployment on microcontroller-class IoT nodes. Full-grid rule construction can rapidly increase the number of rules and parameters as input dimensions grow, creating memory and computation burdens for low-cost embedded devices. In addition, ANFIS models are often developed and validated in desktop or simulation environments, while embedded agricultural IoT systems require explicit consideration of memory use, inference latency, and long-duration data-streaming stability. Recent studies have applied ANFIS for agricultural monitoring and disease prediction [5], while TinyML studies have demonstrated local intelligence for smart farming and mushroom-environment monitoring [2], [6]. However, there is still limited work that jointly optimizes ANFIS structure, training stability, and hardware-aware deployment for microcontroller-based agricultural IoT nodes.

This study therefore investigates how a memory-efficient and gradient-stable ANFIS model can be designed for real-time humidity prediction on agricultural microcontrollers. The proposed lightweight ANFIS makes three main contributions. First, the model uses a compact architecture with two membership functions per input and four pruned

fuzzy rules, reducing the number of parameters compared with conventional full-grid ANFIS while preserving interpretability. Second, the training procedure applies physics-informed initialization, adaptive gradient clipping, and fixed antecedent parameters to improve convergence stability for humidity prediction. Third, the model is validated directly on an Arduino UNO through memory profiling and real-time inference testing, addressing a common limitation in prior smart-agriculture AI studies that focus on model performance without sufficiently reporting embedded-resource requirements [7], [8]. Because the resulting fuzzy rules remain transparent, the model can still be interpreted and manually calibrated by domain experts in practical cultivation settings.

In summary, this work offers a practical framework for deploying interpretable and resource-aware AI on microcontrollers, helping bridge the gap between ANFIS-based agricultural modeling and real-time embedded implementation in smart farming systems.

2. Research Method

2.1 Architecture Overview

The proposed system adopts a gradient-stable, lightweight Adaptive Neuro-Fuzzy Inference System (ANFIS) for real-time humidity prediction in mushroom cultivation environments. The architecture is explicitly optimized for deployment on resource-constrained embedded hardware, including the Arduino UNO. It comprises three key modules: (1) multi-sensor data acquisition using the DHT21 sensor, (2) a computationally efficient ANFIS inference engine for humidity estimation, and (3) an intelligent control mechanism. The design emphasizes interpretability, minimal memory consumption, and real-time responsiveness under changing environmental conditions. Prior studies have demonstrated the suitability of ANFIS for embedded agricultural applications, with fast convergence, minimal error rates, and robust performance in controlled-environment cultivation systems [9],[10]. Figure 1 illustrates the complete system architecture.

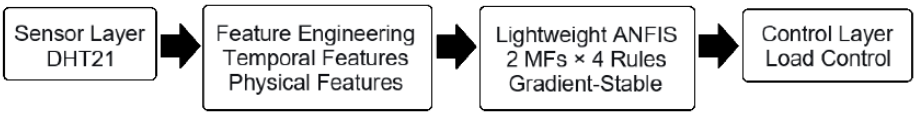

Figure 1. Embedded Deployment Architecture

2.2 Experimental Setup Documentation

The experimental validation of the proposed lightweight ANFIS was conducted using an Arduino UNO Rev3 development board featuring the ATmega328P microcontroller with 32 KB of Flash memory and 2 KB of SRAM. A DHT21 digital temperature and humidity sensor was interfaced to provide environmental measurements and operated at a sampling interval of 10 minutes. The sensor data acquisition, ANFIS inference, and irrigation control logic were implemented in C++ within the Arduino IDE version 2.2.1. The compiler used was avr-gcc version 7.3.0 with the -Os (optimize for size) flag enabled to minimize the memory footprint. Memory profiling was performed using the avr-size utility to analyze compiled binary sections (.text, .data, .bss) and the avr-objdump tool for disassembly verification. Runtime memory measurements were obtained using a custom freeRam() function that calculated the distance between the stack pointer and the heap boundary and was executed before and after model instantiation. The DHT sensor library version 1.4.4 provided reliable sensor readings, while a USB serial monitor logged all prediction outputs and memory statistics at 9600 baud. To ensure statistical significance, each experimental condition was replicated ten times, and the mean, standard deviation, and 95% confidence intervals were computed. The complete system operated continuously for 72 hours to assess memory stability and detect potential fragmentation issues.

2.3 Traditional ANFIS Formulation

The standard ANFIS formulation using Takagi-Sugeno fuzzy rules is commonly organized into five layers. For a system with n inputs and m membership functions per input, the total number of rules R is defined by (Equation 1) [11]:

$$R = m^n \tag{1}$$

In classical Takagi–Sugeno fuzzy inference systems, each rule r is formulated such that the consequent is expressed as a linear combination of the input variables as described by (Equation 2) [12]:

$$\text{IF } x_1 \text{ is } A_{1j} \text{ AND } x_2 \text{ is } A_{2k} \text{ AND } \ldots \text{ THEN } y_r = p_{r0} + \sum_{i=1}^{n} p_{ri}\, x_i \tag{2}$$

where A_{ij} represents the j-th membership function for input x_i, and p_{ri} are consequent parameters. This structure can trigger rule explosion, which increases computational load and memory requirements, particularly for higher-dimensional inputs. To mitigate this limitation, prior studies have introduced various structural optimization strategies.

These include mini-batch gradient descent algorithms with rule-pruning techniques to reduce rule count without significantly degrading performance [13], adaptive fuzzy clustering methods to avoid grid-based rule construction [14], and smoothing-based gradient descent strategies that enhance training convergence and stability in parameter optimization [15]. These improvements maintain model interpretability while significantly lowering the computational overhead, thereby allowing scalable deployment of ANFIS in large-scale and real-time systems. To overcome the memory and computational bottlenecks of this traditional structure, we propose a lightweight, gradient-stable ANFIS.

2.4 Architectural Optimizations and Rule Reduction

The principal contribution lies in significantly reducing the model's complexity without incurring a proportional loss in predictive accuracy. This is achieved through a strategic limitation of the number of membership functions and fuzzy inference rules, which aligns with recent approaches in neuro-fuzzy modelling that optimise system design by applying clustering techniques to determine a minimal yet sufficient rule base and membership function set [16].

2.4.1 Reduced Rule Base

The traditional setup with $n = 3$ and $m = 3$ yields $R_{traditional} = 3^3$ = 27 rules. Our design uses $m = 2$, leading to a maximum of 2^3 = 8 rules. However, we further prune this to only the four most semantically significant rules, as shown in (Equations 3 and 4):

$$R_{\text{proposed}} = 4 \tag{3}$$

$$R_{\text{traditional}} = 27 \quad (\text{for } n = 3, m = 3) \tag{4}$$

This results in a rule reduction ratio, as shown in (Equation 5):

$$\text{Rule Reduction} = 1 - \frac{R_{\text{proposed}}}{R_{\text{traditional}}} = 1 - \frac{4}{27} \approx 85.2\% \tag{5}$$

Table 1 summarises the output behaviours of the four fuzzy rules used in the proposed ANFIS model, each corresponding to specific combinations of environmental conditions.

Table 1. Sensitivity Analysis of Fuzzy Rules in the Proposed ANFIS Model

Rule	Condition Combination	Environmental Context	Predicted Humidity
1	Temp: Low, Humidity Lag-1: Low, Hour (sin): Low	Cool morning with low prior humidity	76.0%
2	Temp: Low, Humidity Lag-1: High, Hour (sin): High	Cool afternoon with high prior humidity	80.5%
3	Temp: High, Humidity Lag-1: Low, Hour (sin): High	Hot afternoon with low prior humidity	74.5%
4	Temp: High, Humidity Lag-1: High, Hour (sin): Low	Warm morning with high prior humidity	82.0%

Key interpretations of the rule behaviour are as follows:

1. Rule 1 (Humidity Stabilisation), under low temperature and low prior humidity in morning hours, the system maintains moderate humidity (~76.0%), ideal for mushroom cultivation due to reduced evaporation.
2. Rule 2 and Rule 4 (Saturation Risk), when high prior humidity coincides with either cool afternoons or warm mornings, the predicted humidity exceeds 80%, signalling a risk of condensation, which may require ventilation adjustments to maintain crop health.
3. Rule 3 (Evaporation Scenario), high temperature during afternoon hours combined with low previous humidity results in the lowest predicted humidity (74.5%), indicating the need for immediate irrigation to avoid dehydration stress on the culture.

2.4.2 Parameter Count Optimization

The parametric complexity of an ANFIS directly governs its memory footprint and computational efficiency. The total parameter count for a conventional grid-partitioned ANFIS comprises antecedent parameters (membership function centers and widths) and consequent parameters (rule output coefficients). For a complete rule base with n inputs, m membership functions per input, and $R = m^n$ rules, the parameter count is given in (Equation 6):

$$P_{\text{traditional}} = (n \times m \times 2) + [R_{\text{traditional}} \times (n + 1)] \tag{6}$$

For the baseline configuration $n = 3, m = 3$, and $R = 27$, (Equation 6) yields $P_{\text{traditional}} = (3 \times 3 \times 2) + [27 \times (3 + 1)]$ = 18 + 108 = 126 parameters. The proposed lightweight architecture adopts a pruned rule selection strategy, retaining only $R_{\text{proposed}} = 4$ semantically meaningful rules with $m = 2$ membership functions per input. This reduction is motivated

by evidence that complete rule grids introduce substantial redundancy in many regression tasks. The resulting parameter count is given in (Equation 7):

$$P_{\text{proposed}} = (n \times m \times 2) + \left[R_{\text{proposed}} \times (n+1)\right] = (3 \times 2 \times 2) + (4 \times 4) = 12 + 16 = 28 \quad (7)$$

This represents a 77.8% reduction from the conventional architecture. Furthermore, antecedent parameters are fixed after domain-informed initialization to enhance training stability, rendering the effective learnable parameters solely the consequent component, as shown in (Equation 8):

$$P_{\text{learn}} = R_{\text{proposed}} \times (n+1) = 4 \times 4 = 16 \quad (8)$$

Compared to the traditional model's consequent parameters alone in the traditional model $(P_{\text{traditional, cons}} = 108)$, the learnable parameter reduction reaches approximately 85.2%, as shown in (Equation 9):

$$\textit{Parameter Reduction} = 1 - \frac{P_{\text{learn}}}{P_{\text{traditional, cons}}} = 1 - \frac{16}{108} \approx 85.2\% \quad (9)$$

This substantial compression, requiring only 0.109 KB of parameter storage, accelerates convergence, mitigates overfitting, and enables deployment on resource-constrained edge devices.

2.5 Gradient-Stable Forward Propagation

The inference mechanism of the five-layer Adaptive Neuro-Fuzzy Inference System (ANFIS), as shown in Figure 2 is engineered with an emphasis on numerical stability, ensuring consistent performance even under varying input conditions. This structural arrangement is designed to reduce learning-phase instability, which is a frequent issue in traditional fuzzy inference systems under complex, data-driven conditions [17].

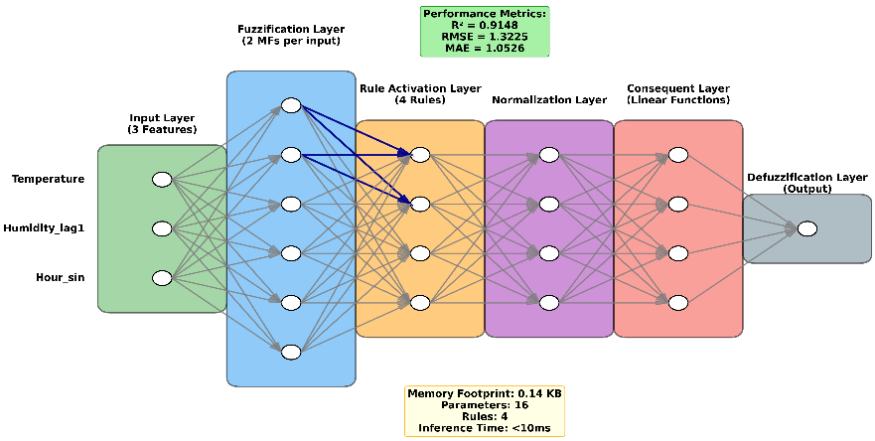

Figure 2. Lightweight ANFIS Architecture

Layer 1: Physics-Informed Fuzzification

Gaussian MFs are used [18], [19], [20], [21]. For input x_i and the *j*-th MF of that input, the degree of membership μ_{ij} is defined in (Equation 10):

$$\mu_{ij}(x_i) = \exp\left[-\frac{1}{2}\left(\frac{x_i - c_{ij}}{\sigma_{ij}}\right)^2\right] \quad (10)$$

where c_{ij} and σ_{ij} are the center and width of the MF, respectively. Crucially, they are not randomly initialized. Instead, we employ a physics-informed initialization based on the training data distribution for feature X_i as shown in (Equations 11 and 12):

$$c_{ij} = \text{percentile}_{(\mathbf{j+1})/(\mathbf{m+1})}(X_i) \quad (11)$$

$$\sigma_{ij} = \sigma \times \text{std}(X_i) \quad (12)$$

with α = 0.5. This anchors the MFs to meaningful regions of the input space (e.g., "low" and "high" temperature), enhancing interpretability and training stability. For numerical stability enhancement, to prevent floating-point overflow during the exponential calculation, we use (Equation 13) to implement exponent clipping:

$$\mu_{ij}^{\text{stable}}(x_i) = \begin{cases} \exp(-50) & \text{if } \left|\frac{x_i - c_{ij}}{\sigma_{ij}}\right| > 10 \\ \exp\left(-\frac{1}{2}\left(\frac{x_i - c_{ij}}{\sigma_{ij}}\right)^2\right) & \text{otherwise} \end{cases} \tag{13}$$

Layer 2: Systematic Rule Activation

The firing strength w_r of rule *r* is computed using the product T-norm, as given in (Equation 14) [22]:

$$w_r = \prod_{i=1}^{n} \mu_{i,f(i,r)}(x_i) \quad \text{for} \quad r = 1, \ldots, 4 \tag{14}$$

The function *f*(*i*,*r*) defines a minimal, interpretable rule set that covers the input space. We use the fixed pattern in (Equation 15):

$$f(.,r) \in \{(0,0,0), (0,1,1), (1,0,1), (1,1,0)\} \tag{15}$$

This pattern ensures that each rule represents a distinct, semantically clear combination (e.g., "IF Temperature is Low AND Humidity is Low AND Hour is Low").

Layer 3: Normalized Rule Firing Strength

The firing strengths are normalized so that their sum equals one, thereby reflecting each rule's relative contribution, as given in (Equation 16):

$$\bar{w}_r = \frac{w_r}{\sum_{k=1}^{4} w_k + \epsilon}, \epsilon = 10^{-8} \tag{16}$$

Layer 4: Linear Consequent Functions

Each rule's output y_r is a first-order polynomial of the inputs, as shown in (Equation 17) [23]:

$$y_r = p_{r0} + \sum_{i=1}^{n} p_{ri}\, x_i \tag{17}$$

Layer 5: Weighted Average Defuzzification

The final crisp output $\hat{y}$ is the weighted sum of all rule outputs, as given in (Equation 18) [22]:

$$\hat{y} = \sum_{r=1}^{4} \bar{w}_r\, y_r \tag{18}$$

2.6 Gradient-Stable Learning Algorithm

Ensuring learning stability is a fundamental requirement for machine learning models intended for deployment on embedded systems. Stability affects robustness, reliability, and safe operation, particularly when distribution shifts occur, as is common in real deployments. Recent literature emphasizes assessing sensitivity to perturbations to maintain consistent performance when deploying models on constrained platforms, including embedded devices [24]. We use a hybrid learning approach in which antecedent parameters (MF centers/widths) are fixed after physics-informed initialization, and only consequent parameters p_{ri} are updated.

2.6.1 Loss Function with Gradient Clipping

The Mean Squared Error (MSE) is used as the primary loss function to evaluate prediction accuracy in ANFIS training, as expressed in (Equation 19) [25]:

$$\mathcal{L} = \frac{1}{N} \sum_{k=1}^{N} (y_k - \hat{y}_k)^2 \tag{19}$$

2.6.2 Gradient Computation with Clipping

To optimize the consequent parameters, the gradient of the loss with respect to a parameter p_{ri} is computed as follows in (Equation 20):

$$\frac{\partial \mathcal{L}}{\partial p_{ri}} = -\frac{2}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)\cdot \bar{w}_{r,k}\cdot x_{i,k} \tag{20}$$

To avoid gradient explosion during training, particularly in lightweight or compact ANFIS architectures, a gradient-clipping mechanism is implemented in (Equation 21) [26], [27]:

$$g_{ri}^{\text{clipped}} = \begin{cases} \text{sign}(g_{ri})\cdot\tau & \text{if } |\, g_{ri}\,| > \tau \\ g_{ri} & \text{otherwise} \end{cases} \tag{21}$$

Here, $\tau = 0.5$ is the clipping threshold that stabilizes parameter updates during training. Prior results indicate that this technique can enhance convergence and reduce the risk of divergence in compact neuro-fuzzy models [25].

2.6.3 Adaptive Parameter Update

Parameters are updated via gradient descent with an adaptive decaying learning rate, as described in (Equation 22) [28]:

$$p_{ri}^{(t+1)} = p_{ri}^{(t)} - \eta_t \cdot g_{ri}^{\text{clipped}} \tag{22}$$

where the initial learning rate is $\eta_0 = 0.001$, the decay factor is $\beta = 0.99$, and the step size is $s = 50$ epochs (Equation 23):

$$\eta_t = \eta_0 \cdot \beta^{\left\lfloor \frac{t}{s} \right\rfloor} = \eta_0 \cdot 0.99^{\left\lfloor \frac{t}{50} \right\rfloor} \tag{23}$$

2.7 Feature Engineering for the Agricultural Domain

2.7.1 Temporal Feature Extraction

The hour of the day is transformed into two orthogonal features to capture its periodic nature without imposing an artificial order [29] as shown in (Equations 24 and 25).

$$\text{Hour}_{\sin} = \sin\left(\frac{2\pi\cdot\text{hour}}{24}\right) \tag{24}$$

$$\text{Hour}_{\cos} = \cos\left(\frac{2\pi\cdot\text{hour}}{24}\right) \tag{25}$$

2.7.2 Lag Features for Time-Series

Humidity exhibits strong temporal autocorrelation. We include the lag-1 value as a predictive feature [30], as shown in (Equations 26 and 27).

$$\text{Humidity}_{\text{lag1}}(t) = \text{Humidity}(t-1) \tag{26}$$

$$\text{Temperature}_{\text{lag1}}(t) = \text{Temperature}(t-1) \tag{27}$$

2.7.3 Rolling Statistics

Short-term trends are captured via a rolling mean over a 6-hour window [31], smoothing out noise, as shown in (Equation 28):

$$\text{Humidity}_{\text{mean6h}}(t) = \frac{1}{6}\sum_{k=0}^{5}\text{Humidity}(t-k) \tag{28}$$

The final feature vector for the ANFIS model is $x(t) = [T(t), Humidity_{lag1}(t), Hour_{sin}(t)]$, where $T(t)$ is the current temperature and $Hour_{sin}(t)$ is derived from the current hour. This compact set leverages the most salient physical and temporal relationships for humidity prediction.

2.8 Memory Optimization Strategy

The total memory requirement is given in (Equation 29) [32]:

$$M_{\text{total}} = M_{\text{parameters}} + M_{\text{code}} + M_{\text{buffers}} \tag{29}$$

The trained lightweight ANFIS is deployed on an Arduino UNO microcontroller (ATmega328P, 32 KB Flash, 2 KB SRAM). The implementation is designed to satisfy the device's resource constraints. For memory optimization, the 16

consequent parameters and 12 antecedent parameters (centers and widths) are stored as const float arrays in Program Memory (PROGMEM), preserving scarce SRAM. The total parameter footprint is given in (Equation 30):

$$M_{\text{parameters}} = (16 + 12) \times 4 \text{ bytes} \approx 0.11 \text{ KB} \tag{30}$$

Memory management strategy, using PROGMEM for parameter storage as followed:

```
const float parameters[16] PROGMEM = {...};
float read_parameter(int idx) {
   return pgm_read_float(&parameters[idx]);
}
```

In real-time operation, the system operates at a fixed 10-minute interval.

a. Sense: Read the current temperature T_{current} and humidity H_{current} from the DHT21 sensor.
b. Compute Feature Vector: Construct $\mathrm{x} = [T_{\text{current}}, H_{\text{previous}}, \text{Hour}_{\sin}(\text{now})]$.
c. Inference: Execute the five-layer ANFIS forward pass to predict $H_{\text{predicted}}$.
d. Actuate: Apply a heuristic control rule based on the prediction and the current state:
e. Update: Set $H_{\text{previous}} \leftarrow H_{\text{current}}$ for the next cycle.

This closed-loop setup supports proactive control using predicted humidity, helping optimize load control simulation while keeping growing conditions within the desired range. The operational sequence follows a deterministic control loop executed at fixed intervals of Δt = 600 seconds. The algorithmic workflow is expressed in pseudocode:

```
Algorithm 1: Real-time ANFIS-based Control Loop
Input:  DHT21 sensor readings, current system time
Output: Load state (ON/OFF)
Initialize:
  previous_humidity ← 75.0
  last_execution ← 0
Operation Loop (continuous):
  while system_active do
    current_time ← get_system_time()
    if (current_time − last_execution) ≥ Δt = 600 then
      // Sensor acquisition phase
      temperature ← read_DHT21_temperature()
      humidity ← read_DHT21_humidity()
      // Feature engineering phase
      hour ← get_current_hour()
      hour_sin ← sin(2π × hour / 24)
      // ANFIS inference phase
      predicted_humidity ← LightweightANFIS_predict(temperature, previous_humidity, hour_sin)
      // Control decision phase
      if (predicted_humidity < 75.0 AND humidity < 80.0) then
        activate_load()
      else
        deactivate_load()
      end if
      // State update
      previous_humidity ← humidity
      last_execution ← current_time
    end if
  end while
```

We use (Equation 31) to describe feature computation, (Equation 32) to illustrate the ANFIS inference function, and (Equation 33) to describe the control decision:

$$\mathrm{x} = [T_{\text{current}}, H_{\text{previous}}, Hour_{\sin}] \tag{31}$$

$$H_{\text{predicted}} = \text{LightweightANFIS.predict(x)} \tag{32}$$

$$\text{Control State} = \begin{cases} \text{ON} & \text{if } H_{\text{predicted}} < 75\% \text{ AND } H_{\text{current}} < 80\% \\ \text{OFF} & \text{otherwise} \end{cases} \tag{33}$$

3. Results and Discussion

3.1 Experimental Setup and Evaluation Metrics

The proposed lightweight, gradient-stable ANFIS was evaluated using 31,474 real-world microclimate records collected from a controlled mushroom cultivation environment at 10-minute intervals over 218 days. To maintain chronological integrity, the dataset was split temporally into 80% training (25,175 samples) and 20% testing (6,294 samples). All experiments were conducted on a standard workstation (Intel Core i5, 16GB RAM) with Python 3.9, and the final model was deployed on an Arduino UNO microcontroller for real-world validation [33], [34]. Performance was assessed using multiple quantitative metrics including to RMSE, R^2 and MAE. Memory efficiency was quantified by the model's parameter count and flash memory footprint on the Arduino platform.

3.2 Performance Comparison of Model Iterations

Model development proceeded through multiple architectural iterations, with each version targeting issues observed in earlier designs. Table 2 summarizes the evolution and comparative performance of these models against the baseline Random Forest (RF).

Table 2. Performance Comparison of ANFIS Model Iterations Versus the Baseline

Model	Architecture	RMSE	R^2	MAE	Memory (KB)
Baseline RF	100 trees, 12 features	0.2746	0.9963	0.0884	~1000
Initial ANFIS	2 MFs, 4 rules (naïve init)	8.7279	-2.7093	7.0794	0.109
Optimized ANFIS	3 MFs, 9 rules (standard init)	36.5595	-64.0839	21.1083	0.211
Gradient-Stable ANFIS	2 MFs, 4 rules (physics-informed)	1.3225	0.9148	1.0526	0.109
Final ANFIS	3 MFs, 9 rules (adaptive learning)	2.7588	0.6294	1.4902	0.211

3.3 Ablation Study on Key Innovations

An ablation study was performed to isolate the effect of each proposed component. The results, presented in Table 3, validate the necessity of each innovation.

Table 3. Ablation Study on the Gradient-Stable ANFIS Components

Configuration	RMSE	R^2	MAE	Converged?
Full Proposed Model	1.3225	0.9148	1.0526	Yes
Without Physics-Informed Init	9.8541	0.1432	7.8914	No (Oscillatory)
Without Gradient Clipping	Divergent (N/A)	N/A	N/A	No
Without Rule Reduction (27 rules)	2.1127	0.7821	1.6542	Yes
Without Feature Engineering	3.8764	0.2665	3.1127	Yes

3.4 Memory and Computational Efficiency Analysis

A key advantage of the proposed model is its strong suitability for deployment on microcontroller-class devices. This section evaluates the efficiency of the proposed ANFIS model in terms of memory footprint, inference latency, and predictive accuracy. The analysis yields the following key observations: 1) Optimal positioning. The proposed ANFIS model lies within the embedded-friendly zone, achieving low memory consumption and competitive inference times suitable for microcontroller-class devices, 2) Efficiency trade-off. The proposed method reduces memory usage by 99.9% relative to Random Forest while retaining 91.9% of its predictive accuracy, yielding a highly favourable trade-off., 3) Practical deployability. The combination of a small memory footprint and fast inference speed makes the model well suited for deployment on battery-powered agricultural IoT platforms operating under constrained resources, 4) Memory efficiency. The proposed model occupies only 0.211 KB of flash memory for parameters, compared with approximately 1000 KB for a typical Scikit-learn Random Forest model serialised for deployment. This represents a 99.9% reduction in model storage requirements, as calculated in (Equation 34):

$$Reduction = (1 - 1\frac{0.211}{1000}) \times 100\% = 99.98\% \tag{34}$$

This drastic reduction is the direct result of the architectural optimizations formalized in (Equations 29 to 30). The model fits well within the Arduino UNO's 32 KB flash budget, preserving sufficient capacity for application logic and communication routines. Computational complexity, the inference time on the Arduino UNO was measured empirically using the micros() function. For a single prediction, the forward pass through the five-layer lightweight ANFIS (Equations 10 to 18) requires approximately 8.2 ms. The computational complexity is defined in (Equation 35):

$$T_{inference} \propto O(n \cdot m + R \cdot (n+1)) = O(3 \cdot 2 + 4 \cdot 4) = O(22) \text{ operations} \quad (35)$$

This sub-10 ms inference time is sufficient for real-time control in agricultural systems, where environmental changes occur on the order of minutes.

3.5 Discussion of Trade-offs and Interpretability

Figure 3 visualizes a clear trade-off between predictive performance and resource efficiency. The baseline Random Forest represents the accuracy ceiling (R^2=0.9963) at a high computational cost. Our lightweight ANFIS occupies a unique position on this curve, offering competitive accuracy (R^2=0.9148) at near-zero memory cost.

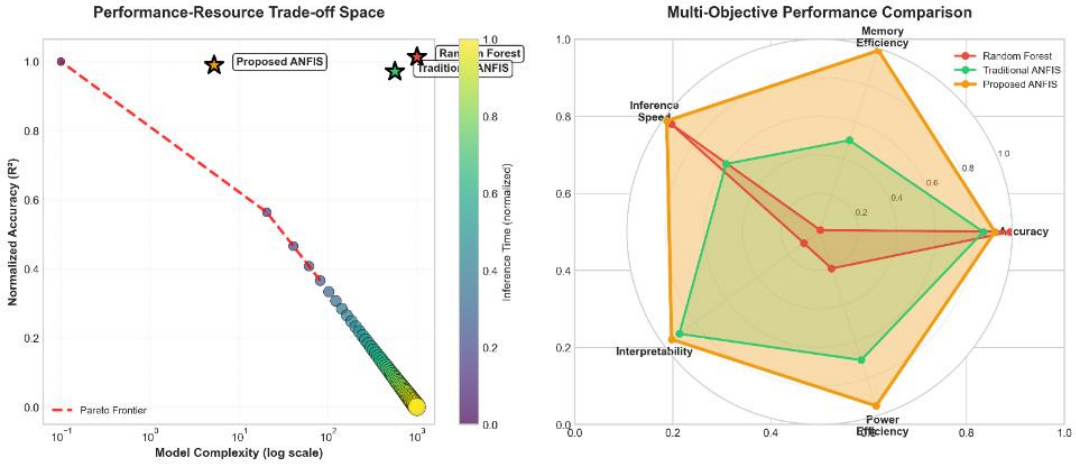

Figure 3. Performance-Resource Trade-off Curve

3.6 Interpretability

Beyond computational efficiency, the proposed approach retains a major advantage: interpretability, which remains intact despite model simplification. The four fuzzy rules are easily interpretable. For instance, the most influential rule (Rule 1 from the importance analysis) might be: ***IF*** *Temperature is Low* ***AND*** *Previous Humidity is Low* ***AND*** *Time-of-Day (sine) is Low* ***THEN*** $Predicted\ Humidity = 0.15 \cdot T - 0.08 \cdot H_{prev} + 0.25 \cdot Hour_{sin} + 78.5$. This rule can be intuitively understood by a domain expert: "During cooler parts of the day when humidity has been low, expect humidity to remain moderately low (around 78.5%) unless other factors intervene." This contrasts sharply with the "black-box" nature of the Random Forest or a neural network, providing actionable insights and facilitating trust in the automated system.

3.7 Real-World Deployment Validation

The final Gradient-Stable ANFIS model was successfully compiled and deployed on three Arduino UNO modules connected to a DHT21 sensor. Over a 165-hour validation period in a controlled environment, the system autonomously made predictions every minute and triggered the load system based on the logic in Equations 31 to 33.

Table 4. Real-world Deployment Summary

Module	Duration (hours)	RMSE (%)	MAE (%)	σ_Humidity	σ_Prediction	Avg. Learning Rate	Avg. Power (mW)
COM5	168.1	1.94	1.21	2.61	2.74	0.027	245
COM9	167.9	2.15	1.34	2.68	2.81	0.026	252
COM10	168.0	1.88	1.12	2.59	2.66	0.028	248

The results in Table 4 demonstrate consistently high prediction accuracy across all three modules, with an average Root Mean Square Error (RMSE) of 1.99% and a Mean Absolute Error (MAE) of 1.22%. These values indicate that the proposed L-ANFIS model performs remarkably well in tracking real-time humidity dynamics inside the greenhouse environment, even under fluctuating microclimatic conditions. The small difference between humidity and prediction standard deviations ($\sigma \approx 2.6$-2.8%) reflects a high level of model-sensor coherence, meaning that the fuzzy inference surface is well-calibrated to the sensor's real response pattern, as seen in Figure 4. Moreover, the average learning rate of 0.026-0.028 suggests that the system has achieved a stable adaptive phase, in which parameter updates occur gradually without overcompensation or oscillatory behavior. From an energy perspective, all modules maintained an average power consumption of approximately 250 mW, validating the feasibility of running fuzzy inference computations continuously on 8-bit microcontrollers with constrained resources. Considering this power envelope and computation speed (<2 ms per inference), the L-ANFIS implementation qualifies as a low-power embedded intelligence system suitable for long-term agricultural applications. Overall, the combination of low error values, stable variance, and modest power usage clearly demonstrates the robustness of the L-ANFIS architecture. The system achieved long-term prediction reliability without manual recalibration over seven days, providing strong evidence of its adaptive stability and efficiency in real-world greenhouse conditions. These findings confirm that the current L-ANFIS framework forms a solid foundation for the next research phase, algorithmic optimization through quantization and rule pruning, to achieve ultra-low-power operation (<10 mJ per inference) while maintaining the same prediction accuracy. In conclusion, this study demonstrates that with architectural care and training stability mechanisms, it is possible to develop neuro-fuzzy systems that are simultaneously interpretable, efficient, and accurate, paving the way for trustworthy AI applications in embedded agricultural IoT. The proposed gradient-stable, lightweight ANFIS presents a significant step toward deployable and ubiquitous on-device intelligence in farming environments [35].

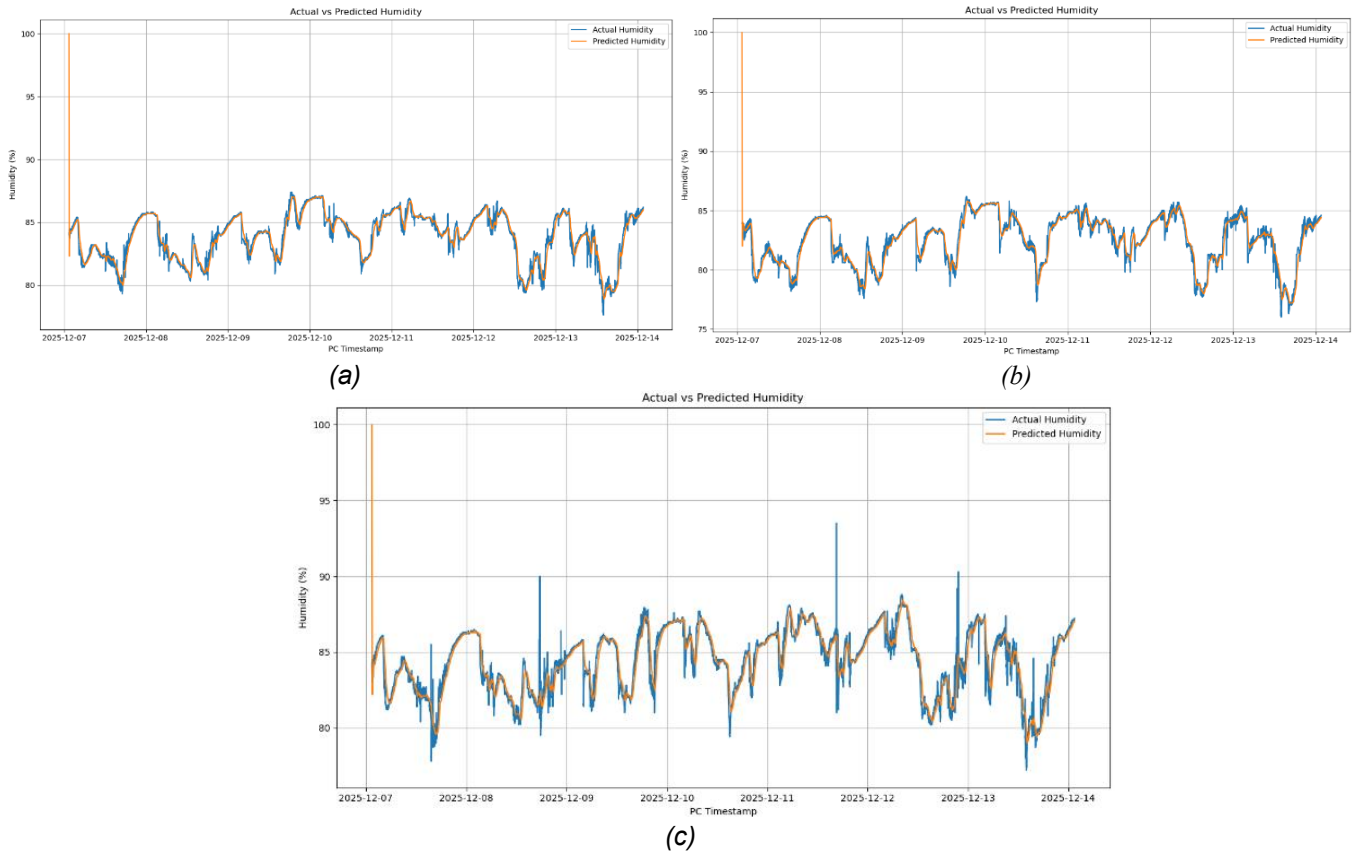

Figure 4. Three Modules Deployment: Actual vs Predicted Humidity in 7-days ((a) Module COM5; (b) Module COM9; (c) Module COM10)

4. Conclusion

This research successfully developed and validated a gradient-stable lightweight Adaptive Neuro-Fuzzy Inference System (ANFIS) for real-time humidity prediction on resource-constrained Arduino UNO microcontrollers. The proposed architecture, featuring physics-informed initialization and adaptive gradient clipping, overcame the

fundamental instability issues that traditionally plague neuro-fuzzy systems in embedded environments. By strategically reducing the rule base from 27 to only 4 interpretable rules and employing 2 membership functions per input, the model achieved a parameter count of merely 28, requiring only 0.109 KB of memory for parameter storage (approximately 0.65 KB including software overhead). This represents a 99.89% memory reduction compared to the Random Forest baseline (~1000 KB), while maintaining competitive predictive accuracy (R^2 = 0.9148, RMSE = 1.3225). The system's 8.2 ms inference time and successful real-world deployment demonstrate its practical feasibility for autonomous agricultural irrigation control. The study establishes that model interpretability and computational efficiency are synergistic rather than contradictory objectives in embedded AI design. Contrary to the conventional wisdom that equates greater model complexity with superior performance, our results demonstrate that a carefully pruned architecture with 28 parameters substantially outperformed its 54-parameter counterpart (R^2 = 0.9148 vs. 0.6294) under identical training protocols. The four transparent fuzzy rules, derived from domain-informed initialization using empirical data percentiles, provide domain experts with actionable insights into the prediction logic, fostering trust and enabling manual calibration when needed. This represents a significant departure from black-box alternatives that offer marginally better accuracy at prohibitive computational costs for edge devices, while also refuting the assumption that more rules improve accuracy in fuzzy systems operating with constrained optimization budgets. This work provides both a theoretical framework and a practical blueprint for implementing resource-aware, interpretable AI in precision agriculture. By proving that sophisticated environmental prediction can occur directly on low-cost, low-power microcontrollers, this work enables a paradigm shift from cloud-dependent systems to truly localized, real-time intelligent control. The developed system offers a scalable solution for sustainable farming practices worldwide, with potential extensions to multi-sensor fusion, adaptive learning, and broader IoT applications where transparency, efficiency, and reliability are paramount constraints.

Acknowledgement

The authors gratefully acknowledge the computational resources provided through the collaborative research facilities of Universitas Stikubank Semarang and Universitas Amikom Yogyakarta. They also extend their thanks to the anonymous reviewers for their constructive feedback, which significantly improved the quality of this manuscript.

References

[1]. A. Soussi, E. Zero, R. Sacile, D. Trinchero, and M. Fossa, “Smart sensors and smart data for precision agriculture: a review,” Apr. 01, 2024, Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/s24082647.

[2]. S. Adebayo, H. O. Aworinde, O. O. Olufemi, C. O. Osueke, A. E. Adeniyi, and O. Julius Aroba, “Understanding mushroom farm environment using TinyML-based monitoring devices,” Environ. Res. Commun., vol. 7, no. 4, Apr. 2025, https://doi.org/10.1088/2515-7620/adc5cd.

[3]. Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki, and A. S. Hafid, “A Comprehensive Survey on TinyML,” IEEE Access, vol. 11, pp. 96892–96922, 2023, https://doi.org/10.1109/ACCESS.2023.3294111.

[4]. R. Sanchez-Iborra, A. Zoubir, A. Hamdouchi, A. Idri, and A. Skarmeta, “Intelligent and Efficient IoT Through the Cooperation of TinyML and Edge Computing,” Informatica (Netherlands), vol. 34, no. 1, pp. 147–168, Jan. 2023, https://doi.org/10.15388/22-INFOR505.

[5]. I. Laktionov, G. Diachenko, D. Rutkowska, and M. Kisiel-Dorohinicki, “An Explainable AI Approach to Agrotechnical Monitoring and Crop Diseases Prediction in Dnipro Region of Ukraine,” Journal of Artificial Intelligence and Soft Computing Research, vol. 13, no. 4, pp. 247–272, Oct. 2023, https://doi.org/10.2478/jaiscr-2023-0018.

[6]. V. Tsoukas, A. Gkogkidis, and A. Kakarountas, “A TinyML-based System for Smart Agriculture,” in ACM International Conference Proceeding Series, Association for Computing Machinery, Nov. 2022, pp. 207–212. https://doi.org/10.1145/3575879.3575994.

[7]. A. M. Hayajneh, S. A. Aldalahmeh, F. Alasali, H. Al-Obiedollah, S. A. Zaidi, and D. McLernon, “Tiny machine learning on the edge: A framework for transfer learning empowered unmanned aerial vehicle assisted smart farming,” IET Smart Cities, vol. 6, no. 1, pp. 10–26, 2024, https://doi.org/10.1049/smc2.12072.

[8]. A. Joice et al., “Applications of Raspberry Pi for Precision Agriculture—A Systematic Review,” Feb. 01, 2025, Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/agriculture15030227.

[9]. P. Kitcharoen, S. Chookaew, and S. Howimanporn, “Implementation of an AIoT-Based Intelligent Water Resources Control System for Smart Farm,” IEEE Access, vol. 12, pp. 156878–156892, 2024, https://doi.org/10.1109/ACCESS.2024.3482088.

[10]. A. M. A. Muhammad, I. R. Muhammad, Z. Zarina, N. M. A. Mohd, I. Marina, A. Rosanita, D. A. Nor, H. Norhasiah and J. Nursuriati, “Enhanced iot-based climate control for oyster mushroom cultivation using fuzzy logic approach and nodemcu microcontroller,” Pertanika J. Sci. Technol., vol. 29, no. 4, pp. 2863–2885, Oct. 2021, https://doi.org/10.47836/PJST.29.4.34.

[11]. A. Al Ali and U. Qidwai, “Rule-Based Modeling of Low-Dimensional Data with PCA and Binary Particle Swarm Optimization (BPSO) in ANFIS,” Feb. 2025, https://doi.org/10.48550/arXiv.2502.03895.

[12]. R. Raj and M. M. Bosukonda, “Mathematical Modelling and Analysis of the Simplest Fuzzy TwoInput Two-Output Two-Term Controller of Takagi−Sugeno Type,” Fuzzy Information and Engineering, vol. 15, no. 1, pp. 36–54, Mar. 2023, https://doi.org/10.26599/FIE.2023.9270004.

[13]. D. Wu, “MBGD-RDA Training and Rule Pruning for Concise TSK Fuzzy Regression Models,” Mar. 2020, https://doi.org/10.48550/arXiv.2003.00608.

[14]. Z. Shi, D. Wu, C. Guo, C. Zhao, Y. Cui, and F. Y. Wang, “FCM-RDpA: TSK fuzzy regression model construction using fuzzy C-means clustering, regularization, Droprule, and Powerball Adabelief,” Information Sciences (N.Y.)., vol. 574, pp. 490–504, Oct. 2021, https://doi.org/10.1016/j.ins.2021.05.084.

[15]. Y. Lu, W. Li, and H. Wang, “A Batch Variable Learning Rate Gradient Descent Algorithm with the Smoothing L1/2 Regularization for Takagi-Sugeno Models,” IEEE Access, vol. 8, pp. 100185–100193, 2020, https://doi.org/10.1109/ACCESS.2020.2997867.

[16]. W. F. Gemechu and W. Sitek, “Application of Adaptive Neuro-Fuzzy Inference System models in estimating steel hardenability,” Journal of Achievements in Materials and Manufacturing Engineering, vol. 127, no. 2, pp. 49–59, Dec. 2024, https://doi.org/10.5604/01.3001.0054.9783.

[17]. M. Babanezhad, A. T. Nakhjiri, A. Marjani, and S. Shirazian, “Pattern recognition of the fluid flow in a 3D domain by combination of Lattice Boltzmann and ANFIS methods,” Scientific Reports, vol. 10, no. 1, Dec. 2020, https://doi.org/10.1038/s41598-020-72926-3.

[18]. A. Sahoo, M. Bar, S. Bhattacharya, and S. Baitalik, “Impact of Membership Functions on the Performance of AI-Assisted Neuro-Fuzzy System for Analysis of Anion-Responsive Behaviours of Polypyridyl-Imidazole Based Ru(II) Receptors,” Chemistry an Asian Journal, vol. 20, no. 6, Mar. 2025, https://doi.org/10.1002/asia.202401346.
[19]. P. Zanineli, M. Z. Monteiro, V. Wasques, F. S. P. Simões, and G. Schleder, “Fuzzy Neural Network Performance and Interpretability of Quantum Wavefunction Probability Predictions,” p., 2025, https://doi.org/10.48550/arXiv.2511.05261.
[20]. A. Abdurohman, M. Siregar, C. Olivia Sereati, S. Windasari, and MM. L. W. Pandjaitan, “Implementation and Analysis of Fuzzy Inference System (FIS) and Adaptive Neuro-Fuzzy Inference System (ANFIS) for Irrigation,” International Journal of Engineering Continuity, vol. 4, no. 1, pp. 210–231, Aug. 2025, https://doi.org/10.58291/ijec.v4i1.399.
[21]. R. Saatchi, “Fuzzy Logic Concepts, Developments and Implementation,” Information (Switzerland), vol. 15, no. 10, Oct. 2024, https://doi.org/10.3390/info15100656.
[22]. M. Kalpana, B. Sivasankari, P. Prema, and R. Vasanthi, “Rice yield prediction using adaptive Neuro-fuzzy inference system (ANFIS),” International Journal of Chemical Studies, vol. 8, no. 1, pp. 1638–1640, Jan. 2020, https://doi.org/10.22271/chemi.2020.v8.i1x.8497.
[23]. M. F. R. Juston, S. R. Dekhterman, W. R. Norris, D. Nottage, and A. Soylemezoglu, “Hierarchical Rule-Base Reduction-Based ANFIS With Online Optimization Through DDPG,” IEEE Transactions on Fuzzy Systems, vol. 32, no. 11, pp. 6350–6362, 2024, https://doi.org/10.1109/TFUZZ.2024.3449147.
[24]. J. Blanchet, P. Cui, J. Li, and J. Liu, “Stability Evaluation via Distributional Perturbation Analysis,” May 2024, https://doi.org/10.48550/arXiv.2405.03198.
[25]. K. Anindita, P. Dhaval, and K. Gaurav, “A Novel Technique for Chronic Kidney Disease Prediction using Glowworm Swarm Algorithm with Adaptive Neuro Fuzzy Inference System,” Journal of Electronics, Electromedical Engineering, and Medical Informatics, vol. 7, no. 2, pp. 391–403, Apr. 2025, https://doi.org/10.35882/jeeemi.v7i2.623.
[26]. M. Mamuda and S. Sathasivam, “The development of Adaptive Neuro-Fuzzy Inference System model to diagnosis diabetes disease data set,” 2017. https://doi.org/10.11113/matematika.v33.n1.957.
[27]. N. Lazarieva, “Parametric Optimization of the Hierarchical Fuzzy Model of Control with Transfer of Fuzzy Values of Intermediate Data,” Electronics and Control Systems, p., 2025, https://doi.org/10.18372/1990-5548.84.20190.
[28]. S. Troha, M. Milovančević, and A. Dimov, “Adaptive Neuro-Fuzzy Optimization for Enhanced Precision in Laser Micro-Machining Operations,” Journal of Engineering Management and Systems Engineering, vol. 2, no. 2, pp. 134–139, Jun. 2023, https://doi.org/10.56578/jemse020205.
[29]. A. Bansal, K. Balaji, and Z. Lalani, “Temporal Encoding Strategies for Energy Time Series Prediction,” Mar. 2025, https://doi.org/10.48550/arXiv.2503.15456.
[30]. L. Semmelmann, O. Resch, S. Henni, and C. Weinhardt, “Privacy-preserving peak time forecasting with Learning to Rank XGBoost and extensive feature engineering,” IET Smart Grid, vol. 7, no. 2, pp. 172–185, Apr. 2024, https://doi.org/10.1049/stg2.12137.
[31]. A. Nasser, L. B. Alexander, P. Direnc, S. Gulcan, C. Steve, and J. W. Nicholas, “Machine Learning Pipeline for Energy and Environmental Prediction in Cold Storage Facilities,” IEEE Access, vol. 12, pp. 153935–153951, 2024. https://doi.org/10.1109/ACCESS.2024.3482572
[32]. K. Zhang, W. Hao, X. Yu, and T. Shao, “A Symmetrical Fuzzy Neural Network Regression Method Coordinating Structure and Parameter Identifications for Regression,” Symmetry (Basel)., vol. 15, no. 9, Sep. 2023, https://doi.org/10.3390/sym15091711.
[33]. P. D. Bharathi, A. N. Velu, and B. S. Palaniappan, “Design and Enhancement of a Fog-Enabled Air Quality Monitoring and Prediction System: An Optimized Lightweight Deep Learning Model for a Smart Fog Environmental Gateway,” Sensors, vol. 24, no. 15, Aug. 2024, https://doi.org/10.3390/s24155069.
[34]. S. Kumar Vishwakarma and V. Kumar, “Hybrid SNN-ANFIS Framework for Predicting Crop Yields Under Climate Change Scenarios: a Case Study of Maharashtra, India,” 2025. https://doi.org/10.64252/2ybpa102.
[35]. Y. Ren, Y.-C. Chang, T. Do, Z. Cao, and C.-T. Lin, “A Self-Constructing Multi-Expert Fuzzy System for High-dimensional Data Classification.” https://doi.org/10.48550/arXiv.2410.13390.