AI

MDPI

Article

Online On-Device Adaptation of Linguistic Fuzzy Models for TinyML Systems

Javier Martín-Moreno [1,2,*], Francisco A. Márquez [1,2], Ana M. Roldán [2] and Antonio Peregrín [1,2,3,*]

[1] Centre for Advanced Studies in Physics, Mathematics and Computing (CEAFMC), University of Huelva, Huelva 21007, Spain; alfredo.marquez@dti.uhu.es
[2] Department of Information Technologies, University of Huelva, Huelva 21007, Spain; amroldan@dti.uhu.es
[3] Andalusian Institute of Data Science and Computational Intelligence (DaSCI), University of Huelva, Huelva 21007, Spain
* Correspondence: javier.martin@dti.uhu.es (J.M.-M.); peregrin@dti.uhu.es (A.P.)

Abstract

Background: Many everyday electronic devices incorporate embedded computers, allowing them to offer advanced functions such as Internet connectivity or the execution of artificial intelligence algorithms, giving rise to Tiny Machine Learning (TinyML) and Edge AI applications. In these contexts, models must be both efficient and explainable, especially when they are intended for systems that must be understood, interpreted, validated, or certified by humans in contrast to other approaches that are less interpretable. Among these algorithms, linguistic fuzzy systems have traditionally been valued for their interpretability and their ability to represent uncertainty with low computational cost, making them a relevant choice for embedded intelligence. However, in dynamic and changing environments, it is essential that these models can continuously adapt. While there are fuzzy approaches capable of adapting to changing conditions, few studies explicitly address their adaptation and optimization in resource-constrained devices. **Methods:** This paper focuses on this challenge and presents a lightweight evolutionary strategy, based on a micro genetic algorithm, adapted for constrained hardware online on-device tuning of linguistic (Mamdani-type) fuzzy models, while preserving their interpretability. **Results:** A prototype implementation on an embedded platform demonstrates the feasibility of the approach and highlights its potential to bring explainable self-adaptation to TinyML and Edge AI scenarios. **Conclusions:** The main contribution lies in showing how an appropriate integration of carefully chosen tuning mechanisms and model structure enables efficient on-device adaptation under severe resource constraints, making continuous linguistic adjustment feasible within TinyML systems.

Keywords: TinyML; edge computing; on-device learning; linguistic fuzzy systems; TinyOL

Academic Editors: Pietro Manzoni and Giovanni Delnevo

Received: 3 November 2025
Revised: 1 December 2025
Accepted: 9 December 2025
Published: 12 December 2025

Citation: Martín-Moreno, J.; Márquez, F.A.; Roldán, A.M.; Peregrín, A. Online On-Device Adaptation of Linguistic Fuzzy Models for TinyML Systems. *AI* **2025**, *6*, 325. https://doi.org/10.3390/ai6120325

1. Introduction

In recent years, connected devices have proliferated, driving the development of new solutions based on the Internet of Things (IoT). These systems, in addition to collecting information from their surroundings, enable intelligent services in a wide range of contexts, from industry and agriculture to the smart home.

In this regard, it is important to consider that IoT applications typically run their processes on devices with limited resources, while data analysis, processing and potential learning are primarily performed on cloud servers, leveraging Cloud Computing. This

setup can lead to two types of challenges: those related to high latency and those associated with the mobility of devices, which must remain continuously connected to the network [1].

To address these issues, the paradigm of Edge Computing (EC) emerged [2]. This paradigm essentially eliminates the need to send large amounts of data to the cloud. Instead, data is processed either on devices close to the data sources (known as Edge devices) or even on the devices themselves (known as Embedded devices). This approach offers several advantages:

- Data Security and Privacy: Certain applications require data privacy. In these cases, sending data to cloud systems poses a significant risk of attacks. EC addresses this by processing data locally, eliminating the need to transmit it off the device.
- Real-Time Response: Applications like traffic control and monitoring require real-time responsiveness. In cloud-based systems, this can be problematic, as an increase in the number of devices leads to higher data volumes, greater network traffic and higher bandwidth consumption. EC avoids this issue by processing data directly on the device, ensuring minimal or zero latency.
- Energy Efficiency: The rising number of IoT devices has significantly increased energy consumption in cloud servers. EC enables local or near-local data processing, eliminating the growing centralized energy demands.

IoT solutions [3], and consequently those based on EC [4], often operate in highly dynamic environments. The specific characteristics of these environments and suitable learning models for such conditions are detailed in [5]. In these scenarios, data are frequently volatile and imprecise, making response time critical. In this context, Fuzzy Rule-Based Systems (FRBSs) are a particularly appealing option due to their inference speed [6].

Another important aspect within the scope of EC is the increasing use of Microcontroller Unit (MCU)-based devices in recent years. These units typically feature limited resources, with constrained computational and storage capacities. However, they are characterized by extremely low power consumption, allowing them to operate efficiently even with small batteries [7].

Tiny Machine Learning (TinyML) [7–10] is an area where traditional Machine Learning (ML) models are adapted for deployment in MCU-based devices. Initially, these models were trained or fine-tuned externally, either through software environments or cloud services, so that the resource-constrained device was solely responsible for inference based on specific input data. The model is trained once, compressed, and subsequently deployed onto the device [8]. A further advancement occurs when the training (or part of it) is conducted directly on the device, giving rise to what is known as On-Device Learning (ODL) [11]. In this case, not only the trained model, but also the learning algorithm must be computationally efficient.

In the approaches described so far, the use of ML models can be considered static, as the model is trained once and remains unchanged over time. However, when the environment or application conditions evolve, the model may become outdated, underperform, or even become obsolete. To address this issue, the concept of training or fine-tuning the model on the device in real-time emerges. This is referred to as TinyML with Online Learning (TinyOL) [12], where the trained models are dynamic and capable of evolving over time as conditions demand. This new paradigm enables models to be trained (or partially adjusted) on the device itself, while simultaneously performing inference based on input data.

Figure 1 schematically represents the various ML paradigms discussed, as applied to resource-constrained or MCU-based devices.

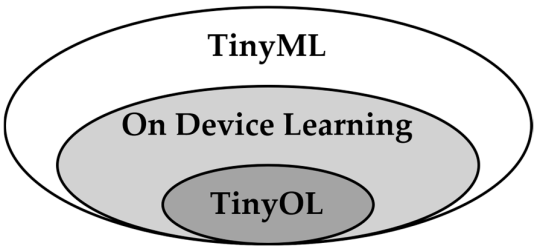


Figure 1. Diagram of TinyML, ODL and TinyOL paradigms.

FRBSs [13] have been employed in numerous devices since their inception. Historically, they have been used to address a wide range of problems, including control, regression and classification tasks. These systems share a common approach: modelling knowledge through a descriptive language based on fuzzy logic predicates, where a set of rules defines the system's behavior in response to specific inputs.

The advantages of such systems lie in several key aspects, including the following [14]:

- Fuzzy logic provides the most "natural" way to express information ambiguity. As a result, generated models are able to effectively represent environmental uncertainty.
- These systems are easily interpretable, as they represent knowledge extracted from data in a manner closely aligned with human reasoning, using rules and linguistic terms. This type of interpretability is currently referred to as by design, inherently interpretable models, model-based interpretability, transparent models and intrinsically interpretable [15–21], as opposed to interpretability achieved through external techniques, that is, post hoc interpretability or explainability.
- They are highly flexible systems, meaning they can be readily adapted. Both their knowledge base and inference engine can be parameterized, allowing them to evolve online with minimal computational effort.

These advantages make FRBSs highly appealing for use in IoT devices [22–24].

Within this context, the key scientific contributions delivered by this work are detailed as follows:

- A comprehensive discussion on the TinyML concept and the potentially feasible models.
- Insights of the feasible implementation requirements for FRBSs in such environments.
- A solution based on an online tuning mechanism for FRBSs, specifically a TinyOL Linguistic Fuzzy model for real-world scenarios. This model has been implemented in a specific case study to illustrate its functionality.

It is therefore important to emphasize that this proposal aims to enable classical linguistic FRBS, whose main characteristic is the use of human interpretable rules, to operate in EC scenarios where dynamic, on-device adaptation is required.

This paper is organized as follows: Section 2 reviews the TinyML philosophy, describing its foundations, models and emerging challenges. Section 3 outlines the requirements for designing FRBSs in TinyOL environments and presents the proposed TinyOL Linguistic Fuzzy model. Section 4 introduces the case study developed using this model and analyses the results obtained. Finally, Section 5 provides the main conclusions derived from this work.

2. Technical Background on TinyML, On-Device Learning and Tiny Fuzzy Systems

This section delves deeper into the TinyML paradigm, discussing the principles of this philosophy and the types of ML models implemented in the literature. Additionally, it describes the main challenges associated with TinyML as well as the TinyOL paradigm, which is the focus of this work. Finally, it addresses existing FRBS adaptation techniques.

These foundational elements serve as the basis for outlining the characteristics that FRBSs developed within this paradigm should possess.

2.1. Principles of TinyML

As previously introduced, TinyML is the paradigm that enables the execution of ML models on resource-constrained devices.

Considering the requirements of devices for TinyML systems [25], the constraints for developing this paradigm are as follows:

- Energy Constraint: Embedded devices in EC require a minimum battery capacity of 10–100 mAh to operate autonomously. This implies that ML algorithms must incorporate specific techniques aimed at minimizing energy consumption.
- Processor Capacity Constraint: Tiny devices typically have clock speeds ranging from 10–1000 MHz. This imposes efficiency constraints on the selected ML models, particularly on highly complex ones.
- Memory Constraint: Tiny devices have significantly less than 1 MB of SRAM, with values often below 100 kB. This implies the use of models whose characteristics align with these MCU limitations.
- Cost Constraint: Devices must be as inexpensive as possible. While individual devices may be low-cost, deploying thousands of them can lead to substantial expenses as the system scales.

Figures 2 and 3 illustrate these constraints. Figure 2 provides a comparative overview of processor and memory limitations across Embedded (TinyML), Edge, On-Premise and Cloud devices. Figure 3 shows the energy consumption of various IoT-compatible devices, highlighting the distinction between those suitable for TinyML (MCUs) and those that are not.

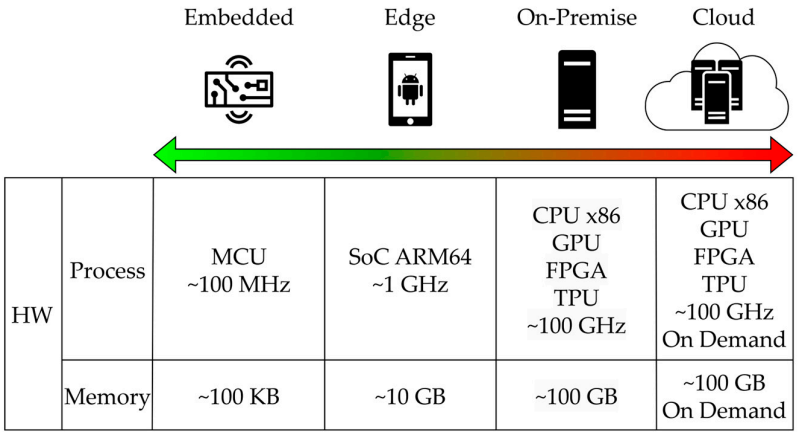


Figure 2. Embedded device hardware constraints. The gradient arrow illustrates the spectrum from resource-constrained embedded devices (green: low power consumption) to resource-rich cloud servers (red: high power consumption on-demand).

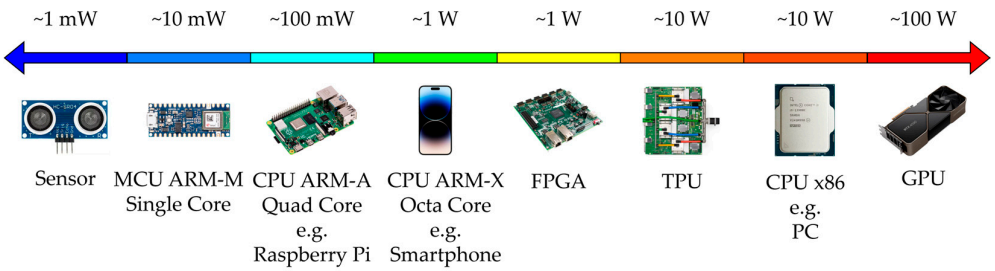


Figure 3. Embedded device energy consumption constraints.

Most small IoT hardware platforms available on the market operate below 100 MHz processor speed, with less than 1 MB of flash memory and less than 1 MB of SRAM. Regarding connectivity, Bluetooth and Wi-Fi are the most commonly used standards. Additionally, many devices include sensors such as accelerometers, gyroscopes, temperature, humidity and atmospheric pressure sensors, as well as microphones and cameras. The energy consumption of these devices typically falls within the milliwatt range. The most popular microcontroller cores used in devices for TinyML applications are from the ARM Cortex-M family, with the Cortex-M4 being particularly notable [8]. This microcontroller will be used in the application presented in this work, an nRF52840, which also forms the basis of the well-known Arduino Nano 33. Its tiny dimensions, low cost and energy efficiency, which have made it a component in tens of billions of consumer devices, should not overshadow the fact that it is a 32-bit RISC device with noteworthy computational capabilities.

2.2. TinyML Models and Use Cases

In this subsection, a selection of relevant TinyML models and use cases, drawn from the specialized literature, is presented.

There are studies aimed at describing the types of models that can be implemented in TinyML [26], but most contributions do not perform on-device model tuning [8].

The main contributions in TinyML found in the state-of-the-art literature have been organized according to the taxonomy of traditional ML algorithms in Table 1. Specifically, it highlights which works propose training or tuning the model directly on the device (ODL) and those that suggest using static models solely for inference (Infer).

Simpler classical ML models, such as Decision Trees or Linear Regression, are particularly suitable for more limited hardware. However, due to the remarkable results and advancements offered by Deep Learning models, Neural Networks have become the most widely adapted models for low-resource devices. To address memory and computational time constraints, several techniques have been developed to enable these resource-intensive models to function as inference engines directly on the devices where data is collected. These techniques include Hyperdimensional Computing [27], Swapping [28], Attention Condensers [29], Constrained Neural Architecture Search [30], Quantization [31] and the Once-For-All Network [32].

Based on this review, we can conclude that on-device inference methodologies have reached a mature stage. However, the development of on-device learning methodologies, particularly for MCU-based systems, is still in its early stages.

Table 1. TinyML state-of-the-art taxonomy. (DT: Decision Tree; KNN: k-Nearest Neighbors; ANN: Artificial Neuronal Network; RF: Random Forest; SVM: Support Vector Machine; LR: Logistic Regression; NB: Naive Bayes).

References	Taxonomy			
	Infer/ODL	Algorithm	Purpose	Learning Type
[9,33]	Infer	DT	Classification	Supervised Learning
[27,34,35]	Infer	KNN		
[36,37]	ODL			
[9,28–33,35,38–43]	Infer	ANN		
[12,44–56]	ODL			
[9,33,35,57,58]	Infer	RF		
[9,33–35]	Infer	SVM		
[33,34]	Infer	LR		
[33,34]	Infer	NB		

Table 1. *Cont.*

References	Taxonomy			
	Infer/ODL	**Algorithm**	**Purpose**	**Learning Type**
[59–61]	Infer	RF	Regression	
[60]	Infer	LR		
[60,61]	Infer	SVM		
[61–63]	Infer	ANN		
[64]	ODL	K-Means	Clustering	Non-Supervised Learning
[65]	ODL	DBSCAN		
[66–68]	ODL	TEDA	Anomaly Detection	
[69]	Infer	BERT	LLM	
[70]	ODL			
[71]	Infer			Reinforcement Learning
[72]	Infer			Self-Supervised Learning

2.3. TinyML Challenges

Despite the growing accessibility of ML tools and frameworks, the development of ML applications for resource-constrained environments remains a significant challenge. Most existing models are designed without consideration for the computational and memory limitations typical of embedded IoT devices [26]. As a result, deploying these models in such environments requires additional optimization techniques such as quantization and pruning [8].

A fundamental limitation of many TinyML solutions is their reliance on static models: trained once offline, optimized, and then deployed. This static approach is ill-suited for dynamic environments, where changes over time can render pre-trained models ineffective. Online and continual learning paradigms, such as ODL, have been proposed to overcome this limitation by enabling models to adapt in real-time directly on the device.

Based on the 6 G White Paper on Edge Intelligence [73], Figure 4 illustrates the balance through the processing tasks that are performed locally on the device and those done on remote servers, as well as the availability of resources for algorithms, data privacy and response speed.

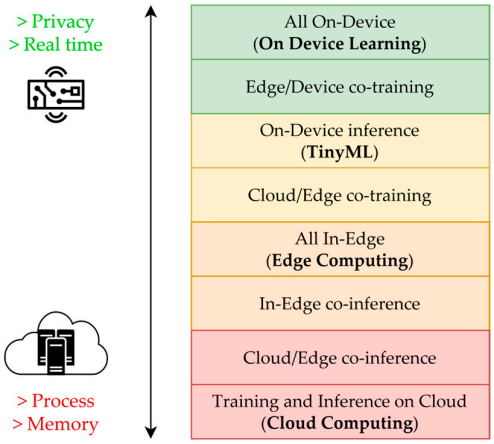


Figure 4. Level rating of Edge Intelligence. Colors indicate the trade-off between on-device learning (green), offering maximum privacy and real-time response, and cloud computing (red), providing higher processing power and larger memory capacity.

However, achieving effective ODL in TinyML contexts is non-trivial. While ODL enhances privacy and responsiveness by eliminating the need for cloud-based retraining, it introduces challenges related to energy efficiency and computational overhead. Optimizing algorithms to operate within the strict resource budgets of embedded devices is a key research area. Complementary techniques like Federated Learning offer potential solutions by enabling collaborative model training across distributed devices [74,75]. Yet, issues such as device heterogeneity and unreliable network connectivity pose additional hurdles.

TinyOL further complicates the scenario, as it emphasizes real-time, continual adaptation with minimal or no persistent data storage. This necessitates efficient incremental learning strategies that can operate on small data buffers while maintaining model accuracy and minimizing latency. Moreover, the speed of learning must be balanced with inference performance to ensure that the system remains responsive.

Overall, the development of adaptive TinyML systems calls for innovations across algorithm design, optimization strategies, and deployment frameworks, with special attention to maintaining a balance between accuracy and efficiency.

2.4. Fuzzy Systems in Tiny Computing

Fuzzy logic has played an essential role as a technological precursor to the current paradigm known as Tiny Computing, understood as the design of intelligent systems for embedded environments and devices with limited resources [76]. Since the 1970s, and on a massive scale during the 1980s and 1990s, fuzzy logic enabled the implementation of efficient control and decision-making systems in hardware with reduced computational capabilities [77–79] long before the popularization of AI based on ML.

Currently, fuzzy systems continue to evolve and coexist to enhance TinyML systems in hybrid architectures that combine fuzzy logic with lightweight ML models to perform intelligent and robust inferences in multiple applications. These include, among others, anomaly detection [80], energy optimization of smart buildings [81] and remote monitoring of healthcare environments [82]. The aim of these hybrid approaches is to improve the accuracy, energy efficiency, and adaptability of models by leveraging the interpretable reasoning of fuzzy logic together with Tiny Computing methodologies.

In summary, fuzzy logic not only anticipated the technological challenges that are now being addressed in Tiny Computing, but continues to evolve as part of hybrid solutions, serving as a key tool for the development of intelligent, low-power, high-performance applications in devices with limited resources.

2.5. Dynamic On-Device Fuzzy Systems Adaptation

Fuzzy systems offer a compelling alternative to neural networks in the context of EC, where models must operate under strict constraints in energy, memory, and interpretability. Unlike neural networks, which typically function as black-box models requiring significant computational resources for inference and online adaptation, fuzzy systems employ transparent rule-based logic that is both interpretable and computationally lightweight.

In environments with continuously evolving data, Evolving Fuzzy Systems (EFSs) [83] are valued for their ability to handle uncertainty, while continuously adapting their Rule Base (RB) and parameters without storing historical data, what makes them well-suited for real-time tasks [84,85]. To maintain their accuracy over time, EFSs apply structural adaptations such as rule adding (to handle concept drift), rule merging (to reduce redundancy), and rule pruning (to improve interpretability and prevent overfitting).

While EFSs are well-studied in terms of adaptability and learning performance, their designs generally rely on TSK-type models and numerical update mechanism, and relatively little attention has been given to their suitability for resource-constrained environ-

ments. Many existing implementations assume the availability of sufficient memory and processing power, which can limit their applicability in embedded systems, low-power devices, or edge computing scenarios.

The gap addressed by our proposal is enabling robust and adaptive linguistic fuzzy learning, preserving interpretability, while maintaining the computational efficiency required for real-time, resource-limited TinyML platforms.

3. Design of TinyOL Fuzzy Models

This section first describes, from a general perspective, the key requirements and design considerations necessary to implement FRBSs for TinyML and TinyOL environments. Then, a specific proposal is illustrated: a TinyOL Linguistic Fuzzy model based on a linguistic (Mamdani-type) fuzzy system for real-world scenarios.

3.1. FRBSs in Edge Computing, TinyML and TinyOL Environments

FRBSs are suitable techniques for EC environments due to their relatively low computational requirements in terms of CPU, memory and storage capacity. In fact, this concept has been widely applied for decades in various embedded control systems across numerous real-world devices [86,87]. Moreover, current IoT devices can also integrate FRBSs as part of their internal components [88,89]. In these cases, FRBSs are designed, tested and refined by engineers and then embedded into low cost, resource-constrained devices for deployment in real-world applications.

Additionally, FRBSs are also viable for TinyML environments. Many well-established learning and tuning mechanisms for FRBSs can be adapted, with relatively minor modifications, for use on resource-constrained devices, either during the design phase or even in deployment. In the latter case, FRBSs can operate within TinyML environments with online learning capabilities, known as TinyOL. Here, the system must support online learning or tuning of the model through an algorithm executed directly on the device autonomously, i.e., learning or tuning itself within its operational environment. This is the core novelty of the proposal presented in this work.

The following considerations must be addressed when designing or adapting an FRBS capable of on-device learning or tuning within TinyOL environments:

- Model Compactness: The FRBS designed should be as compact as possible. A reduced set of variables and rules is preferable, not so much due to the computational load during inference, which is light with a well-chosen set of inference and defuzzification operators, but primarily to minimize computational effort during the learning or tuning phase. In this phase, the volume of calculations, and consequently the time and energy required, can become excessive in less compact models. Low-granularity partitions for fuzzy variables are recommended, as these lead to fewer rules and fewer parameters to adjust subsequently.
- Learning or Tuning Mechanism Characteristics: Similarly, among the wide range of learning and tuning models proposed in the literature over the decades, those with lower computational overhead and higher efficiency should be selected for online execution on the device. In this regard, we can distinguish between learning models and tuning models [90] for elements of the Knowledge Base (KB). Learning models typically learn the RB using a fixed Data Base (DB), making them computationally heavier. Tuning models, which focus on fine-tuning the DB while assuming a fixed, unmodifiable RB, are more computationally efficient and therefore preferable. This work adopts the latter approach and describes its implementation in detail in the following Section 3.2. Additionally, adaptive inference operators [91,92] can also be adjusted or learned.

- Model Evaluation Mechanism: An autonomous evaluation mechanism may not always be feasible or straightforward in all real-world applications. To design an FRBS capable of online adaptation and improvement, it must evaluate its own accuracy to decide whether to trigger the adaptation process when necessary and to guide that process. Traditional FRBS learning or tuning processes often involve a search mechanism, frequently evolutionary [93], that generates alternative changes to the current KB (e.g., to the RB or DB) and an evaluation mechanism to assess these alternatives. An appropriate metric and evaluation strategy should be defined based on the application, including the following options:
  - Evaluation in the Real Environment: This strategy evaluates alternative KBs directly within the application. Challenges include its infeasibility in critical applications where worse-than-current alternatives cannot be tested and the amount of time required to quantify each alternative. Consequently, this strategy is generally not preferred if other evaluation strategies are feasible.
  - Evaluation via Exact Simulation: For some applications, exact simulation is possible, allowing the system to predict the output for alternative KBs based on recent real-world input data. If this option is available, it is preferred due to its precision and speed.
  - Evaluation via Surrogate Model: When evaluation in the real environment is not feasible and exact simulation is unavailable, a surrogate model can be constructed using past application data. While this approach is faster once the model is built, it depends on the accuracy of the surrogate model, which may not always reflect the evolving real application.

The choice among the three evaluation strategies depends on the constraints and requirements of the application. For safety-critical or latency-sensitive scenarios (e.g., autonomous systems), real-environment evaluation is generally avoided due to the risk of performance degradation. In such cases, exact simulation is preferred. When neither direct testing nor accurate simulation is available, such as in complex, dynamic environments, a surrogate model offers a practical trade-off. However, this requires sufficient historical data and ongoing updates to maintain fidelity. Resource availability, criticality of decisions, and system observability should guide strategy selection.
For strategies other than real-environment evaluation, the device must store some data in a buffer for evaluation purposes. This requires policies for buffer sizing and data replacement, etc., which will depend on the specific application.
Regardless of the strategy, evaluation processes should be efficient and lightweight to enable execution directly on the device while it operates.

- Synchronization: A strategy must be established to determine when the model is actively operating online and when it can engage in learning or tuning routines.

3.2. A TinyOL Linguistic Fuzzy Model

To illustrate the design concepts introduced in the previous subsection, we present a possible linguistic FRBS (Mamdani-type) model designed specifically for TinyOL, developed in accordance with the previously established guidelines and general considerations. Specifically, we opt for an adaptive system based on tuning the DB. This model will later be employed in the real-world scenario presented in Section 4.

In many real-world applications where linguistic FRBS are used for control, the operating conditions may evolve gradually over time due to smooth, non-abrupt changes in the environment. In such scenarios, and as long as these changes affect only the meaning of the linguistic terms rather than the structure of the RB, the system can be effectively updated through tuning of the linguistic terms, that is, by adjusting their associated membership

functions. This form of adaptation, also known as DB tuning of linguistic FRBS, has traditionally been employed to improve accuracy under static conditions [94], but it has never been explored as a mechanism for online adaptation in dynamic environments where the semantics of the linguistic terms themselves evolve.

Classical approaches to DB tuning can be broadly categorized into two main families:

- Free-form tuning, in which the shape of each membership function is adjusted without explicit semantic constraints.
- Structured tuning, of one of the two following:
  a. 2-tuple linguistic tuning, where only the central position of each term is modified, preserving the original shape and width alternatives [95]
  b. 3-tuple linguistic tuning, which adjusts both the position and the width of the membership functions [96]

The 2-tuple and 3-tuple approaches rely on a reduced number of parameters, thus preserving the interpretability and semantic coherence of the linguistic terms more effectively than free-form tuning. Although they offer fewer degrees of freedom, they are generally preferable when interpretability is a requirement, and particularly well suited to TinyML scenarios, where computational and memory resources are constrained.

Note that if the evolution of the operating conditions were to imply changes that affect the RB rather than only the linguistic terms (stored in the DB), a different class of adaptive strategies would be required. Addressing this scenario lies beyond the scope of the present work and is outlined in the future work of the Section 5.

The illustrative model proposed consists, in summary, of a compact highly interpretable linguistic FRBS for control or regression capable of adapting its DB online, i.e., autonomously adjusting the linguistic partitions of its variables. The model also includes a synchronization mechanism that employs a short time period for the fuzzy system inference and a longer period for DB tuning. The remainder of this section provides a detailed description of this proposal.

To achieve model compactness, the following techniques are employed:

- Symmetrical triangular linguistic labels: These labels are defined using only two parameters by employing the 3-tuple linguistic model [96], which reduces complexity while preserving sufficient degrees of freedom, unlike a single-parameter model. The 3-tuple model minimizes computational requirements and storage needs, making it suitable for IoT devices. Exceptions are the two extreme labels, which are asymmetric for expressiveness. Figure 5 illustrates how the label $L3$ shifts both its position (p) and width (w). The newly introduced label $L3'$ is denoted as the 3-tuple ($L3$, α, β), where α is the lateral displacement and β is the amplitude variation.
- Pointwise fuzzification [97] to simplify computations during the inference phase.
- Minimum T-norm for matching and inference and weighted average method (also known as the weighted mean of maxima or the height method) for defuzzification in FITA (First Infer, Then Aggregate, or Mode B) [97]. Its formula is:

$$y' = \frac{\sum_i^N h_i \times M_i}{\sum_i^N h_i} \tag{1}$$

where h_i is the matching degree between the input variables and the rule antecedent linguistic labels, and M_i represents the maximum value of the consequent linguistic label from the rule R_i. This approach is also less computationally intensive. These operators further streamline the necessary calculations, enabling real-time responses on resource-constrained devices.

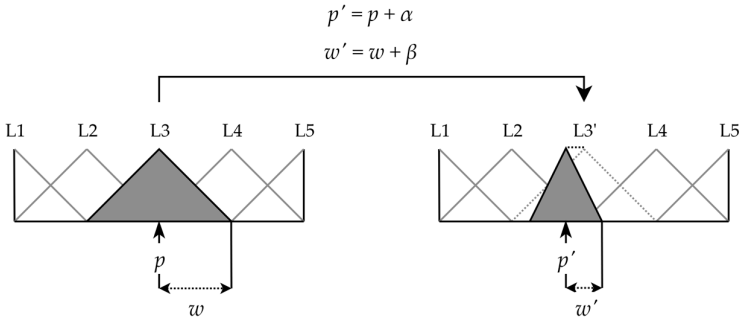


Figure 5. Symmetrical linguistic labels defined by position (p) and width (w) parameters based on the 3-tuple model. Dotted lines illustrate label variations: the gray line shows the original label, while the black line indicates the applied displacement α.

The computational complexity associated with the design of FRBSs lies primarily in the generation and tuning of the Knowledge Base. Literature offers several strategies to tackle this challenge, which can broadly be classified into:

- Gradient-based Learning Approaches (Neuro-Fuzzy Systems—NFS): These models have demonstrated significant performance in the literature [98]. By leveraging gradient-descent-based algorithms to optimize parameters in an online or offline fashion, they often provide faster convergence. Nevertheless, their reliance on local gradient information can lead to suboptimal solutions when trapped in local minimum.
- Evolutionary Approaches (Genetic Fuzzy Systems—GFS): These methods have also been extensively studied [90]. They employ bio-inspired algorithms to perform parameter search, offering strong exploratory capabilities that help avoid local optima. However, this advantage typically comes at the cost of sometimes increased computational time.

Considering these aspects, both families of methods can be employed to build FRBSs. However, the evolutionary strategy is particularly suitable in this case. This choice is supported by several facts:

- Evolutionary algorithms offer a strong advantage due to their population-based search and stochastic operators enable effective exploration of complex and poorly structured search spaces, reducing the likelihood of becoming trapped in local minimum. The literature consistently highlights this exploratory capability as a key reason for their success in optimization-driven learning tasks [90,99].
- Genetic algorithms do not depend strongly on the initial search point, unlike gradient-based methods, which are highly sensitive to initialization and can easily converge to suboptimal solutions. This robustness is well documented in the classical evolutionary computation literature [100,101].
- Linguistic (Mamdani-type) fuzzy systems are not easily differentiable, unlike TSK fuzzy models [102], where gradient-based learning is feasible because consequents are linear functions, the output is differentiable, and the error surface is smooth [103]. In contrast, linguistic fuzzy systems often involve non-differentiable inference mechanisms, such as triangular membership functions and in our case Minimum T-norm, making gradient-based tuning unsuitable [104,105].
- They allow straightforward implementation on low-resource devices, as genetic algorithms can operate efficiently under severe computational and memory constraints when the parameter space is compact, making them well suited for TinyML-oriented on-device adaptation.

Given the objective of enabling the FRBS to update its KB to maintain or improve accuracy in response to changes in the application environment, the proposed online learning or tuning mechanism focuses on DB tuning, a well-known and effective strategy [90]. This involves starting with an initial DB, which evolves as the system operates, and a predefined RB that remains fixed. This predefined RB can be learned beforehand, either from examples or via expert knowledge.

The tuning mechanism uses an evolutionary metaheuristic search [90], whose chromosome structure is illustrated in Figure 6. The parameters being adjusted correspond to the position (p_{ij}) and width (w_{ij}) of each linguistic label (L_{ij}) associated with each variable (i) and label (j), following the recommendations in Section 3.1 for using models with fewer parameters.

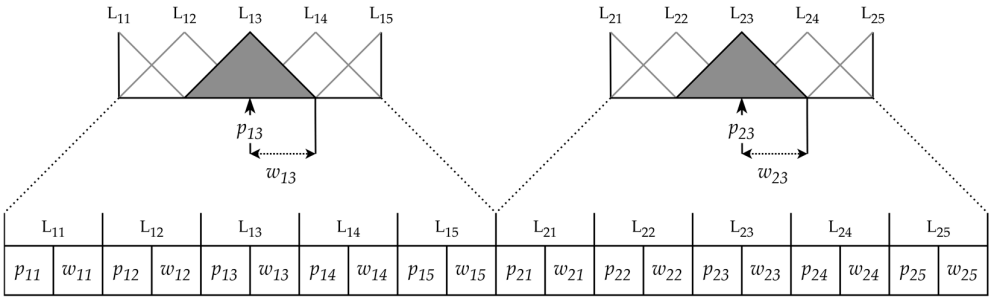


Figure 6. Chromosome encoding of a 3-tuple DB with three variables.

To make the search and evaluation mechanism efficient and suitable for resource-constrained devices, the following resources are utilized:

- Parameter discretization: The search space is reduced by discretizing parameters instead of using continuous values. For instance, linguistic labels can shift or change width in discrete steps.
- Range constraints: Parameter ranges are bounded. Specifically, labels can have limited lateral shifts and constrained width changes.
- Universe of discourse reduction: The range of each variable is restricted to relevant values, minimizing unnecessary extremes.
- Micro Genetic Algorithms (μGAs) [106] strategies: Populations composed of few individuals, periodic restarts, and high selective pressure practices to favor rapid convergence.

The evaluation and synchronization mechanisms depend more on the specific application than on the model itself. The proposed TinyOL Linguistic Fuzzy model can employ any of the three evaluation strategies (real environment, exact simulation or surrogate model) described in Section 3.1, depending on the application's requirements. Similarly, strategies for updating the DB, including periodicity, must align with the application's characteristics. In general, for evaluation using exact simulation or a surrogate model, DB tuning is performed incrementally online over time. Evaluation during the tuning phase uses a finite, recent window of data stored in a continuously updated buffer.

Figure 7 illustrates the functional components of the proposed TinyOL Linguistic Fuzzy model, the interactions among them, and their role within the online-learning pipeline. Two synchronous main loops operate in parallel. The dynamic-environment loop continuously generates input data, sends it to the device, receives the model's prediction, and evaluates the decision cost. Concurrently, each time the device acquires a new data element, the FRBS processes it to produce a response to the environment. The same data element is also stored in a buffered batch that triggers the DB tuning procedure once the buffer is full, updating the FRBS parameters with the newly optimized values.

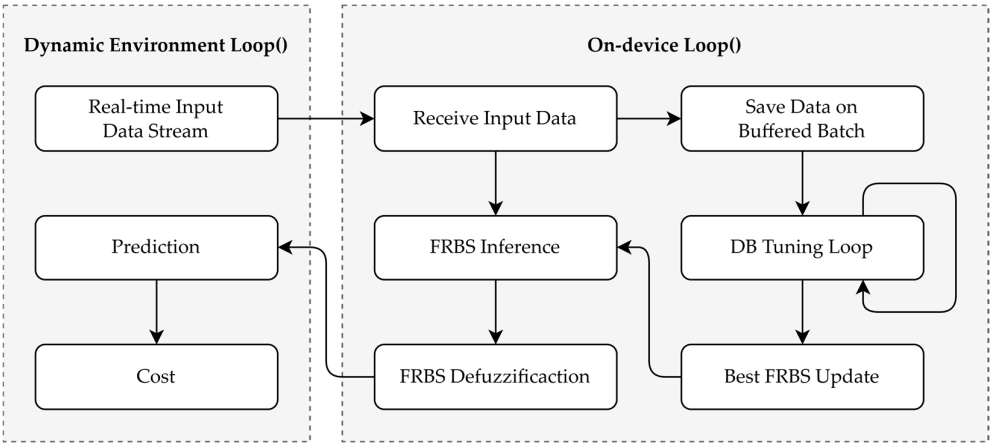


Figure 7. Flow diagram of the proposed TinyOL Linguistic Fuzzy model.

Figure 8 illustrates the workflow of the proposed TinyOL Linguistic Fuzzy model. Incoming data streams are continuously monitored, with a sliding window used to buffer recent observations. During normal operation, the system performs real-time inference on new data using the current model, producing immediate results. Simultaneously, buffered data are periodically used in the evaluation of the learning algorithm. In this phase, the system finetunes the DB parameters encoded in N chromosomes (DB_1 to DB_N) to optimize performance. During the tuning phase, the RB remains fixed. The RB consists of n fuzzy rules of the form If X_{nm} is L_{nm} (for m input variables) then Z_n, where X_{nm} denotes the m-th input variable used in rule n, L_{nm} is its associated linguistic term, and Z_n represents the linguistic output label of the rule. Keeping the RB structure static, as is standard in linguistic fuzzy system tuning, preserves interpretability and constrains the number of parameters, while allowing the DB parameters to be efficiently optimized during adaptation. Once the tuning is complete, the updated model resumes real-time inference, incorporating the learned adjustments into its predictions. This cycle enables the system to maintain accurate predictions in dynamic conditions while minimizing computational overhead.

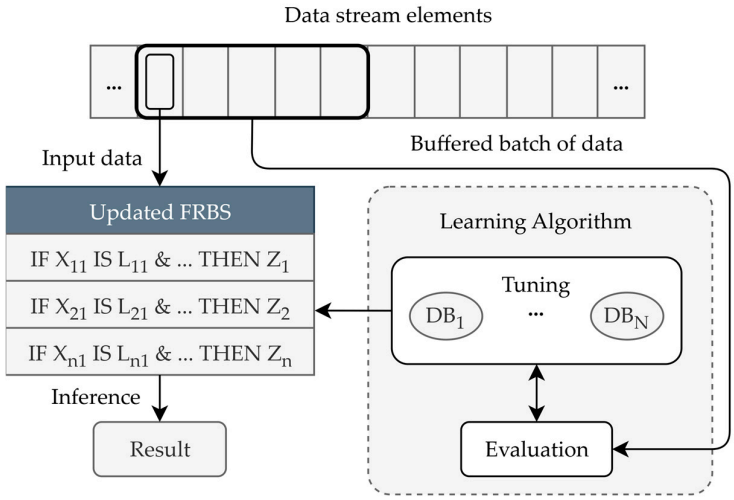


Figure 8. Diagram of the proposed TinyOL Linguistic Fuzzy model.

Finally, it should be noted that in a practical IoT deployment, several external factors may influence the behavior of the system, such as sensor noise, temporary communication interruptions, or fluctuations in available power. Linguistic FRBSs have the ability to mitigate the impact of noisy or unstable measurements due to their tolerance to imprecision and smooth membership functions. Moreover, because the TinyOL model operates entirely

on-device, it can continue to provide decisions autonomously even when communication is unavailable, with the only possible drawback being the lack of updates of remote data that it may need, depending on the application, but it will still continue to operate with the latest data received. Lastly, because both inference and online tuning have very low computational and energy requirements, the adaptive mechanism can be executed reliably even under constrained power conditions. The tuning step is lightweight enough to run without affecting the normal operation of the device or exceeding the energy budget typically available in TinyML platforms. Although power consumption during the adjustment phase can be kept very low if the system remains compact, as indicated in Section 3.1, it might be considered that it should only be executed when sufficient power resources are available, i.e., that it could be postponed.

The following section illustrates how this TinyOL Linguistic Fuzzy model can be applied in practice.

4. Case Study: TinyOL Linguistic Fuzzy Model for Energy Management

A TinyOL Linguistic Fuzzy model, suitable for EC and IoT devices, could be relevant for a wide range of domains. In this work, we present as an example an energy source management system for renewable energy installations in homes equipped with photovoltaic systems (composed of solar panels and energy storage through batteries) that are not isolated, i.e., connected to the power grid. The purpose of this example is therefore to illustrate, through a potential real-world scenario, an online adaptive TinyOL Linguistic Fuzzy model, such as the previously presented in Section 3, in comparison with the same model using TinyML (with static tuning).

The aim of the system is to determine the optimal proportion of renewable energy and conventional energy from the grid to be consumed at any given time to optimize the electricity bill cost for the household user in countries such as Spain, where electricity prices fluctuate by day and hour, and are determined the day before. Figure 9 depicts the simulated experimental setup, where the device that contains the model receives input on the available energy sources and decides which one to use in order to meet the current consumption demands.

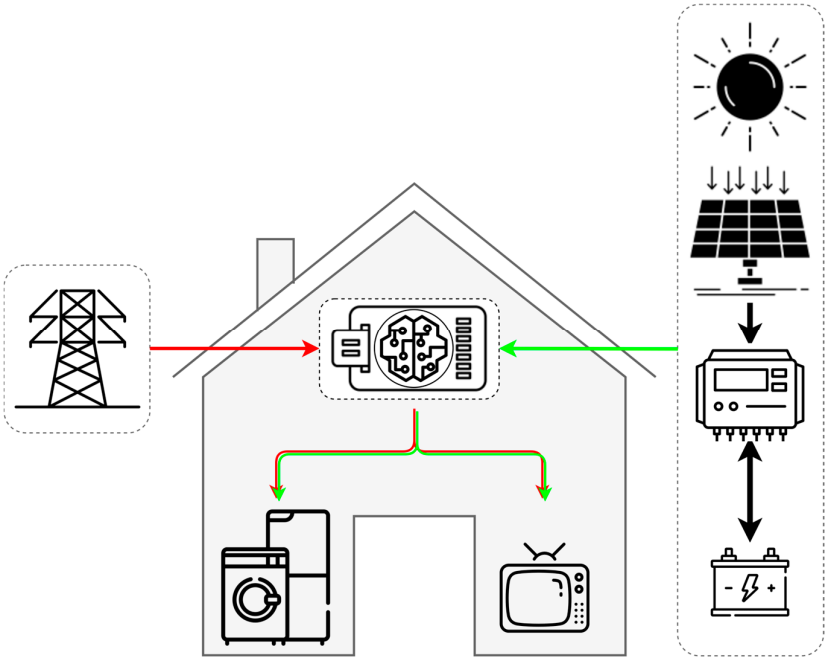


Figure 9. Simulated experimental setup. Arrows indicate energy sources for consumption: green represents renewable energy, and red represents conventional energy drawn from the grid.

This application is particularly interesting, as it intentionally focuses on sustainability by employing a small set of batteries rather than a large-capacity setup. Batteries are not only the most expensive component of the installation but also occupy considerable space, weigh heavily (requiring specific structural considerations in the home) and are subject to wear, necessitating periodic replacement. Moreover, replacing them generates chemical waste that requires costly processing. On the other hand, while the computational system in this specific application does not impose stringent energy consumption constraints, it is deliberately implemented on a resource-limited device for EC environments to demonstrate feasibility.

This type of energy management application aligns with a growing body of research on intelligent energy management in residential settings. Recent studies have explored optimization frameworks for smart homes that integrate photovoltaic storage, electric vehicle charging, and demand response to enhance efficiency and reduce costs [107,108]. Artificial intelligence and adaptive control strategies have also been applied to optimize energy consumption and system reliability in both on-grid renewable energy systems and smart buildings [109–111]. These works underline the practical relevance of lightweight, adaptive models like the TinyOL Linguistic Fuzzy approach presented here, demonstrating that resource-constrained devices can effectively contribute to sustainable, cost-efficient energy management.

4.1. Proposed Model Description

In the proposed system, decisions regarding the proportion of the sources of consumed energy (renewable through solar panels and batteries or from the grid) are made based on the following data: the expected energy consumption in the household over the next 24 h, the renewable energy available in the batteries, the expected solar radiation based on weather forecasts, and the cost of grid electricity, which can fluctuate hourly according to a day-ahead auction, as observed in countries like Spain. For example, if sufficient renewable energy is available from the solar panels at a given time, the expected household consumption is low, and the grid electricity cost is high for that time slot, it would be preferable to use renewable energy.

The decision-making mechanism is implemented via an FRBS, which makes new decisions every hour of the day. This time interval was chosen mainly because, as shown in Table 2, while shorter intervals allow finer control and responsiveness, the one-hour interval offers a practical trade-off aligned with electricity tariff stability and computational efficiency.

Table 2. Decision time and computational cost trade-off.

Time Interval	Energy Price Variation	Decision Granularity	Computation Overhead	Suitability
1 h (chosen)	Variable	Low	Low	Aligned with tariff period
30 min	Stable	Moderate	Moderate	More responsive
15 min	Stable	High	High	Higher cost
1 min	Stable	Too high	Too high	Max flexibility

The aforementioned data is merged into the following two input variables for the FRBS to ensure a compact system:

- Electricity Price Trend (EPT): This variable allows the system to predict whether the price will rise, hold steady or decline over the next 24 h. It is calculated as the slope of the linear regression line that best fits the price data recorded over the preceding 24-h window.
- Energy Balance Trend (EBT): This variable merges the data corresponding to the expected energy consumption, the expected solar radiation and the amount of renewable

energy stored in the batteries. In this way, the energy balance, normalized between −1 and 1, indicates whether the current energy demand can be met using renewable energy. A value of −1 represents a scenario in which no renewable energy is available, either stored or from solar radiation, meaning that the total energy demand must be covered by the conventional power grid. A value of 0 indicates that the renewable energy available at a given moment is exactly equal to the current household consumption. Conversely, values closer to 1 indicate an excess of renewable energy after meeting consumption needs. To account for the temporal evolution of this variable, the EBT is defined as the slope of the linear regression that best fits the energy balance values recorded over the previous 24-h period. This allows the system to predict whether the EBT will rise, hold steady or decline within a normalized range from 0 to 100. In this scale, a value of 0 represents a future scenario with lower energy balance values, indicating a decrease in the proportion of renewable energy relative to consumption. A value of 50 implies that the balance between renewable energy availability and expected energy consumption will remain stable. Finally, values approaching 100 suggest an increase in the availability of renewable energy to meet consumption, thereby reducing dependence on the conventional power grid.

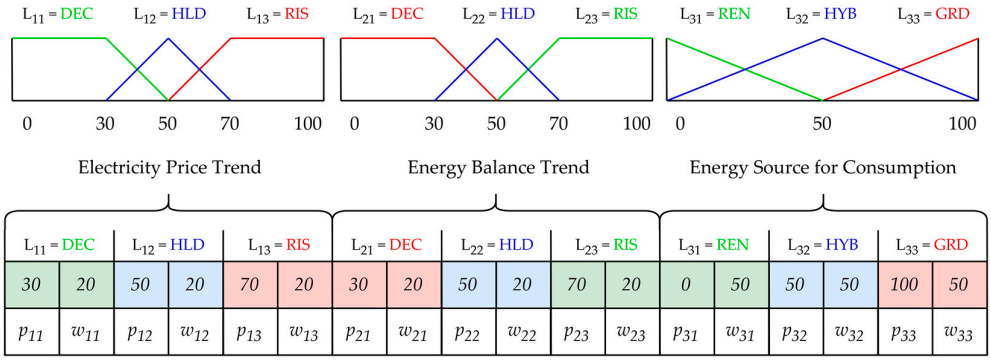
Finally, the fuzzy system's output will be denoted as the Energy Source for Consumption (ESC), which is defined as the proportion of renewable and grid energy to be used for covering the expected consumption. Its discourse universe is defined between 0 and 100, where 0 indicates covering 100% of consumption with renewable energy, 50 indicates splitting the consumption equally between renewable and grid energy, and 100 indicates covering 100% of consumption with grid energy.

Regarding the KB of the FRBS, the structure is deliberately compact: each of the two input variables and the output variable is represented using three linguistic labels, yielding a total of nine possible rules. The universes of discourse for all variables are normalized and discretized over the interval [0–100] to regulate and structure the optimization search space. Rather than permitting unconstrained continuous variation, the linguistic labels are allowed to modify their position and width only through discrete steps within this range. This approach improves interpretability and aligns with the way domain specialists conventionally describe these variables in percentage terms rather than as absolute quantities. For the input variables, the labels "DEC", "HLD", and "RIS" denote if the values tend to "decline", "hold", and "rise", respectively. For the output variable, the labels "REN," "HYB," and "GRD" correspond to "renewable", "hybrid", and "grid" energy source for consumption. Figure 10 illustrates these linguistic terms constituting the DB of the FRBS, together with the chromosome representation determined by the position and width of each label. The initial parameter values were established using expert knowledge.

The RB consist of rules defined by an expert, such as: "If the *Electricity Price Trend* for the upcoming hours is estimated to *decline* (indicating that the current price is higher than the upcoming hours price) and the *Energy Balance Trend* is estimated to *rise* (suggesting an increase in future renewable energy availability), then, the decision will favor maximizing *renewable* as *Energy Source for Consumption* (since the electricity price will decline and the renewable energy will rise)". Figure 11 illustrates the nine rules that define the RB, ensuring complete coverage of the universe of discourse.

In accordance with the scheme outlined in Section 3.2, the system alternates between inference phases using the FRBS, during which decisions are made, and real-time DB tuning phases to address changes in the environment, such as variability among the potential household consumption patterns between weekdays and weekends. BD's parameters fine-tuning is performed, for example, using an evolutionary algorithm implemented directly on the device. The aim of this process is to optimize the FRBS's responses so that, given

specific environmental conditions as input, the system produces the most appropriate decision to minimize energy costs.



L_{11} = DEC		L_{12} = HLD		L_{13} = RIS		L_{21} = DEC		L_{22} = HLD		L_{23} = RIS		L_{31} = REN		L_{32} = HYB		L_{33} = GRD	
30	*20*	*50*	*20*	*70*	*20*	*30*	*20*	*50*	*20*	*70*	*20*	*0*	*50*	*50*	*50*	*100*	*50*
p_{11}	w_{11}	p_{12}	w_{12}	p_{13}	w_{13}	p_{21}	w_{21}	p_{22}	w_{22}	p_{23}	w_{23}	p_{31}	w_{31}	p_{32}	w_{32}	p_{33}	w_{33}

Figure 10. Diagram of the DB of the FRBS, where Electricity Price Trend and Energy Balance Trend represent the fuzzy input variables, and Energy Source for Consumption represent the fuzzy output variable. Colors indicate the correspondence between each linguistic label (L_{ij}) of variable i and label j, and the position (p_{ij}) and width (w_{ij}) parameters that define it in the chromosome encoding.

	Electricity Price Trend	Energy Balance Trend	Energy Source for Consumption
R_0	Declines	Declines	Hybrid
R_1	Declines	Holds	Renewable
R_2	Declines	Rises	Renewable
R_3	Holds	Declines	Grid
R_4	Holds	Holds	Hybrid
R_5	Holds	Rises	Renewable
R_6	Rises	Declines	Grid
R_7	Rises	Holds	Grid
R_8	Rises	Rises	Hybrid

Figure 11. Table of the RB of the FRBS, constructed to provide complete coverage of the universe of discourse. Colors indicate the efficiency level of each linguistic label: green for the most favorable in minimizing consumption costs, blue for intermediate efficiency, and red for the least desirable situation for cost optimization.

The aforementioned tuning process has been designed using a bio-inspired metaheuristic that allows for real-time FRBS updates in a lightweight manner. Specifically, the CHC genetic algorithm has been selected, as it provides an effective balance between exploration and exploitation of the search space; moreover, it also provides a good basis to be complemented with μGAs strategies. To simplify the number of parameters to tune, a gene encoding based on the 3-tuple representation [96] of the fuzzy system has been adopted. This representation uses one parameter for the label position and another for its width, thereby maintaining symmetry. Consequently, the chromosome comprises only 18 genes for the three variables of the system. The search space for these 18 parameters is limited, as the discourse universe has been normalized between 0 and 100. Additionally, possible values have been discretized, significantly reducing the search space.

The evolutionary algorithm's population consists of only 10 chromosomes, which undergo crossover and evolve until just 100 trials are completed in each evolutionary run. To prevent premature convergence of the population, the model implements an incest prevention mechanism, with a crossover threshold set to a quarter of the number of genes. That is, at least 25% of the genes between two chromosomes must differ for crossover to occur. The HUX crossover operator is used, which exchanges genes between two parents

to generate two offspring. Each offspring randomly inherits a gene from each parent with equal probability.

The real-time adaptation phase of the FRBS typically requires an incremental tuning mechanism, as resource-constrained devices are not capable of storing the entire growing dataset (which increases daily as new data is generated). To address this issue, a novel variant of the CHC genetic algorithm tailored for on-device online learning environments (μCHC) is proposed. This variant introduces two main modifications:

- Iteratively Partial Executions on Buffered Data Batches: Since it is not feasible to execute the full algorithm on the entire dataset due to memory constraints, the algorithm is applied iteratively on smaller, buffered batches of data generated in real time. These batches temporarily store data generated during the previous 24 h in the device's memory. The data window shifts daily, forming a new buffered batch to update the model in the next partial execution of the evolutionary algorithm.
- Incremental Tuning through Implicit Restart: In order to make the tuning process incremental, the traditional restart mechanism of the CHC algorithm, which is triggered when chromosome similarity exceeds a predefined threshold, is replaced by an implicit restart. In this approach, each partial execution of the evolutionary algorithm begins by selecting the best-performing chromosome from the previous execution. This chromosome serves as the seed for the next run, enabling the model to adjust incrementally over time. Although the partial executions are not extensive enough to converge, they are sufficient to enhance accuracy as the model adapts to new data. To prevent the algorithm from converging after several partial executions, the rest of the initial population in each execution is initialized with random values consistent with each gene, thereby introducing diversity into the search process.

Figure 12 provides a detailed diagram of the novel incremental online tuning mechanism proposed: Data streams continuously into the device. Every 24 h, data instances form a buffered data batch, which is used to evaluate various models during each partial execution of the evolutionary tuning process. The model corresponding to the best chromosome is selected to infer decisions for the next 24-h data window while also serving as the seed for the tuning process on the new buffered batch in the subsequent execution.

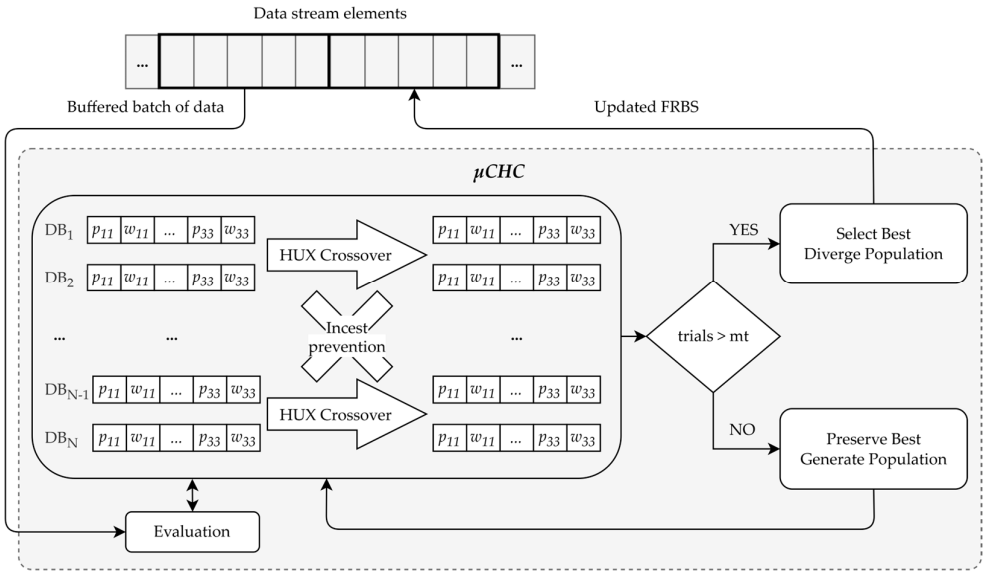


Figure 12. Diagram of the proposed μCHC.

The pseudocode of the μCHC algorithm is expressed in Algorithm 1.

Algorithm 1. μCHC

Input: L: chromosome length; p: population size; d: difference threshold; mt: max. trials

$d \leftarrow L/4$
Initialize ($P(0)$)
for each buffered batch of data **in** data stream **do**
  while *trials* < mt **do**
    Evaluate ($P(t)$)
    Preserve best chromosomes from $P(t)$
    $P'(t) \leftarrow$ Select ($P(t)$)
    $C(t) \leftarrow$ Crossover ($P'(t)$)
    Evaluate ($C(t)$)
  Increment *trials* count
  $P(t+1) \leftarrow$ Select_best ($P(t)$, $C(t)$)
  if $P(t+1) = P(t)$ **then**
    Decrement d
  if $d < 0$ **then**
    $d \leftarrow L/4$
 $P(t+1) \leftarrow$ Diverge ($P(t+1)$)
 Update FRBS parameters with best chromosome

The continuous tuning of the FRBS is made possible through the rapid evaluation of chromosomes, enabled by an exact simulation of the experimental environment (a concept introduced in Section 3.1) specifically designed for this purpose. This simulator allows the tuning algorithm to optimize the FRBS decisions in real time.

Figure 13 presents the flow diagram of the proposed incremental evolutionary DB tuning method within the overall model framework. The diagram depicts the continuous on-device operational cycle, which alternates between performing inference on each incoming instance and executing tuning procedures on buffered batches of data.

4.2. Hardware Implementation of the Proposed Model

To demonstrate the functionality of the proposed system under real-world conditions on a current resource-constrained device, a study was conducted on a subset of affordable IoT devices suitable for such applications, including some of the most well-known options. Table 3 presents the specifications of the most representative devices.

Table 3. IoT devices specifications.

Device	Basic Features		
	RAM	**Flash Memory**	**CPU Freq.**
Arduino Nano 33 BLE Sense	256 KB	1 MB	64 MHz
SparkFun Edge	384 KB	1 MB	48 MHz
STM32F746G Discovery kit	320 KB	1 MB	216 MHz
ESP32	520 KB	448 KB	240 MHz
Raspberry Pi Pico	264 KB	2 MB	133 MHz

From these, the Arduino Nano 33 BLE Sense was chosen due to its compatibility with the widely known Arduino development environment. Although it is one of the most modest devices in terms of main memory and CPU frequency, it is still capable of implementing the entire system described. This Arduino model is equipped with a

Nordic Semiconductor nRF52840 microcontroller, featuring a 32-bit ARM Cortex-M4F core. Figure 14 illustrates its main components.

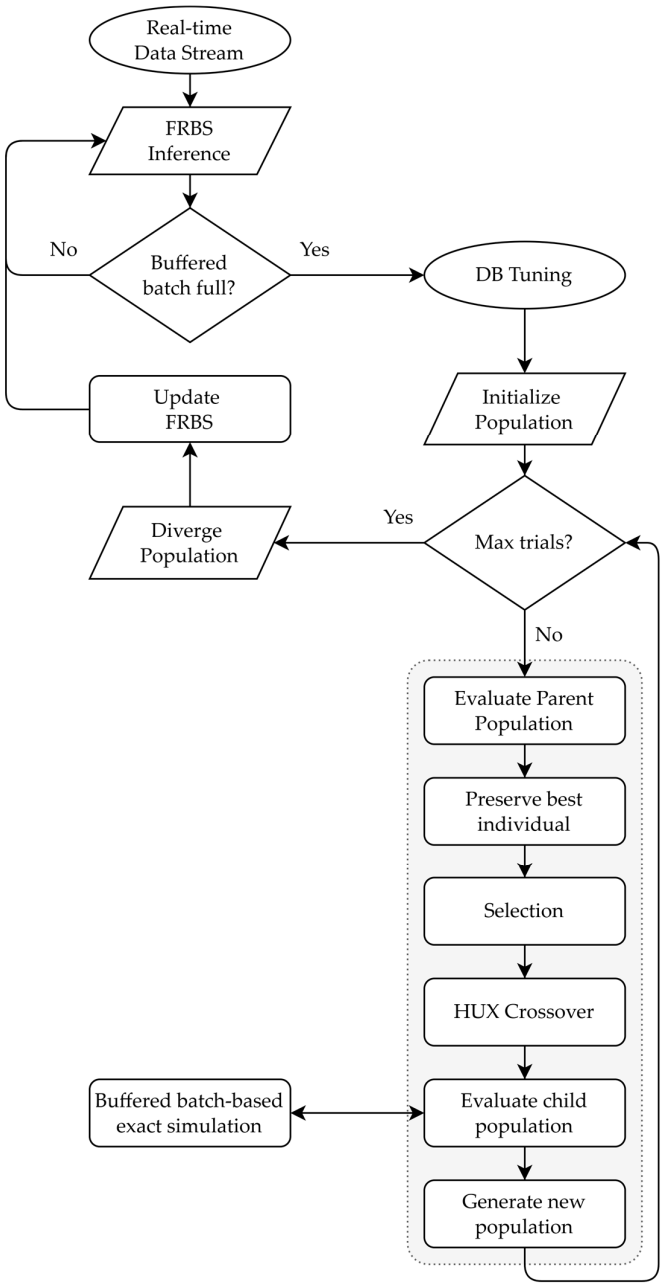


Figure 13. Flow diagram of the model including μCHC.

4.3. Experimental Platform Configuration

During the tuning phase, an exact simulation mechanism was implemented to evaluate the cost associated with each DB. This mechanism does not rely on external simulation software; instead, it is executed directly on the device to assess the decisions produced by each chromosome over the buffered batch of real data. The simulation mechanism continuously updates the levels of renewable energy stored in the battery system, considering both consumed and generated energy based on solar radiation. Each day, the cost of conventional energy, solar radiation forecast data, the amount of renewable energy available, and the expected consumption are obtained.

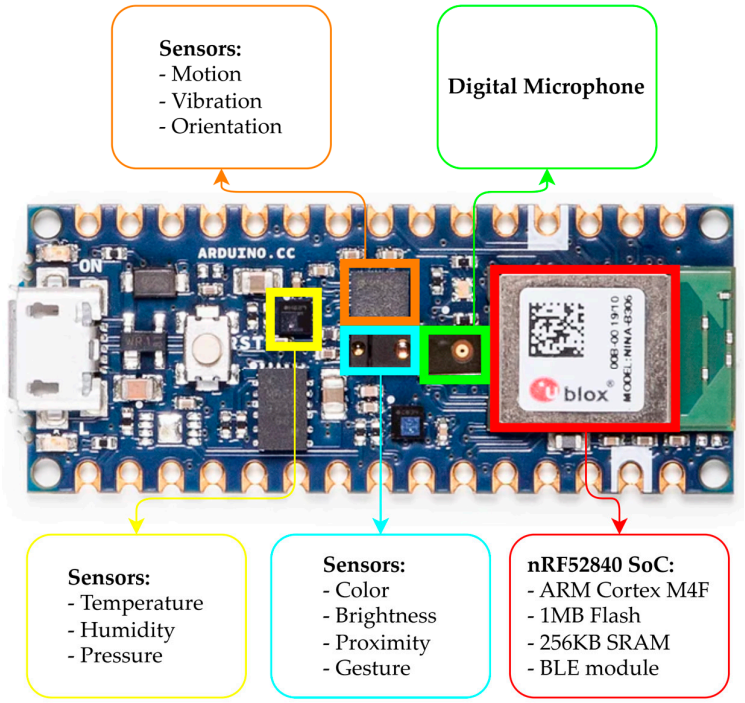


Figure 14. Arduino Nano 33 BLE Sense.

From these inputs, the model evaluates the cost of decisions based on consumption. This enables the system to infer real-time decisions while continuously adjusting to environmental changes on the device itself, optimizing the cost of its decisions.

For validation on the device, no simulation software was employed. Instead, Python 3.10 scripts running on a host computer replayed real datasets as a chronological input stream to the Arduino Nano 33 BLE Sense, allowing the device to operate under realistic conditions. The TinyOL Linguistic Fuzzy system was fully implemented in C and executed on the microcontroller, ensuring that all membership-function computations, DB evaluation and decisions occurred physically on the device.

In real time, the device receives continuous updates from the following data sources: the electricity price, expressed in euros per kWh, is retrieved through the ESIOS API (https://api.esios.ree.es/ (accessed on 8 December 2025)), while household energy consumption data, expressed in kWh, is provided by the electricity distribution company (https://cecsa.oficinavirtual.sercide.com/followup-supply (accessed on 8 December 2025)). Additionally, average annual solar radiation data, expressed in Wh/m^2 is periodically updated from the Andalusian Energy Agency service (https://www.agenciaandaluzadelaenergia.es/ (accessed on 8 December 2025)). These streams enable the system to maintain online learning on the device by incrementally updating its internal parameters as new observations arrive.

During the experimentation period, all data was collected coherently, ensuring consistency in terms of date and geolocation, which facilitates integration. The data collection period ranged from 1 November 2023 to 30 September 2024, resulting in a dataset of 8016 records defined by seven columns: hour, day, month, year, electricity cost (in €/kWh), household consumption (in kWh), and solar radiation (in Wh/m^2). These records correspond to the province of Huelva, located in southwestern Spain.

Figure 15 presents the pre-processed data in tabular format, indexed by date and time, ready for use in the simulation and training process. Figure 16 graphically represents a sample of the data used, corresponding to 24 h on 25 July 2024. The graph displays the variation in energy prices (blue), household consumption (red), and solar radiation intensity (green).

	Year	Month	Day	Hour	Cost €/kWh	Consumption kWh	Radiation Wh/m2
0	2023.0	11.0	1.0	0.0	0.05010	0.323	0.0
1	2023.0	11.0	1.0	1.0	0.04679	0.225	0.0
2	2023.0	11.0	1.0	2.0	0.03863	0.202	0.0
3	2023.0	11.0	1.0	3.0	0.03792	0.208	0.0
4	2023.0	11.0	1.0	4.0	0.03807	0.232	0.0
...	...	...	...	...	...	...	...
8011	2024.0	9.0	30.0	19.0	0.23226	0.406	0.0
8012	2024.0	9.0	30.0	20.0	0.29431	0.517	0.0
8013	2024.0	9.0	30.0	21.0	0.29181	0.549	0.0
8014	2024.0	9.0	30.0	22.0	0.18495	0.344	0.0
8015	2024.0	9.0	30.0	23.0	0.18729	0.310	0.0

8016 rows × 7 columns

Figure 15. Data used in the experimentation.

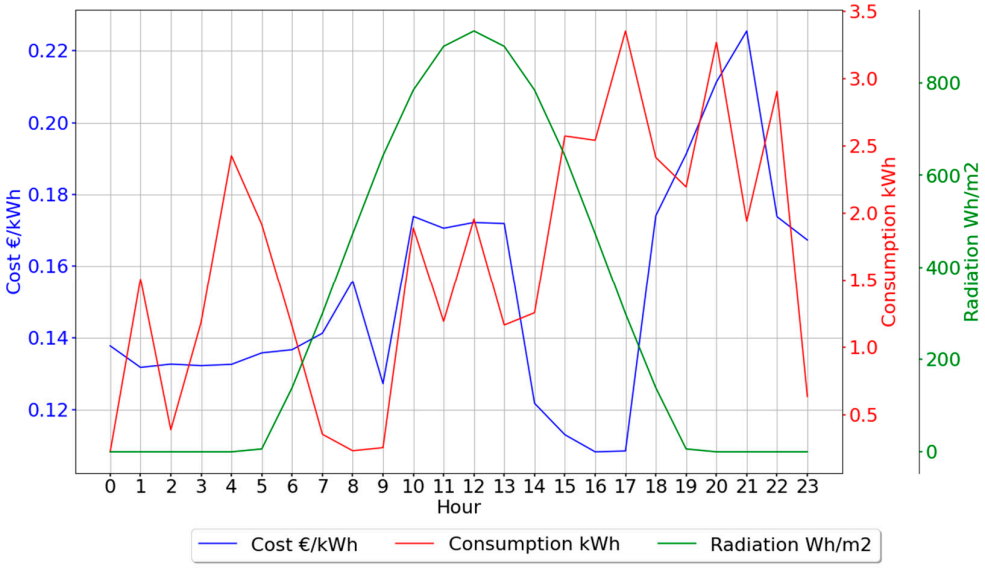


Figure 16. Visualization of a data sample for a 24-h interval on 25 July 2024.

Based on the household consumption data shown in Table 4, analyzed hourly and daily, the simulator was found to work best with the following configuration: two solar panels with a total generation capacity of 1 kWh (500 Wh each) and a battery system with a total storage capacity of 5 kWh.

Table 4. Consumption exploratory data analysis.

Metric	Household Consumption	
	Hourly (kWh)	Daily (kWh)
count	8016	334
mean	0.73	17.63
std	0.71	7.79
min	0.00	3.46
25%	0.21	11.76
50%	0.41	18.49
75%	1.05	23.80
max	3.95	38.93

4.4. Result Analysis

To evaluate the effectiveness of the proposed approach, we compared the FRBS under both deployment paradigms: a static TinyML configuration and an adaptive TinyOL configuration that enables lightweight on-device updates. This distinction allows us to assess the behavior of the model when its parameters remain fixed versus when they evolve in response to incoming data. For contextual comparison, we also include the FRBS baseline, initialized as shown in Figures 10 and 11. The performance results in terms of cost optimization and computational resource usage are presented and discussed throughout the remainder of this subsection.

First, the total cost savings percentage was compared for the same household energy consumption using different systems. The results, depicted in Figure 17, show that, initially, incorporating solar panels into the household achieves a cost saving of 25% (represented in blue) compared to using only grid energy. Secondly, when a battery storage system is added, the savings slightly increase to 26% (represented in brown). Using a decision-making process managed by the initial static FRBS, cost savings reach 25.7% (represented in violet), which surpasses the system with panels but does not outperform the system with batteries. Fourth, the red bar represents the statically adjusted fuzzy model (TinyML-based), meaning it is adjusted only once at the beginning. Finally, managing renewable energy usage through the proposed online tuning mechanism (TinyOL-based) achieves a 27% cost saving (represented in green). The TinyOL model demonstrates better results than the static TinyML model, as the former adapts to changes occurring throughout the testing period.

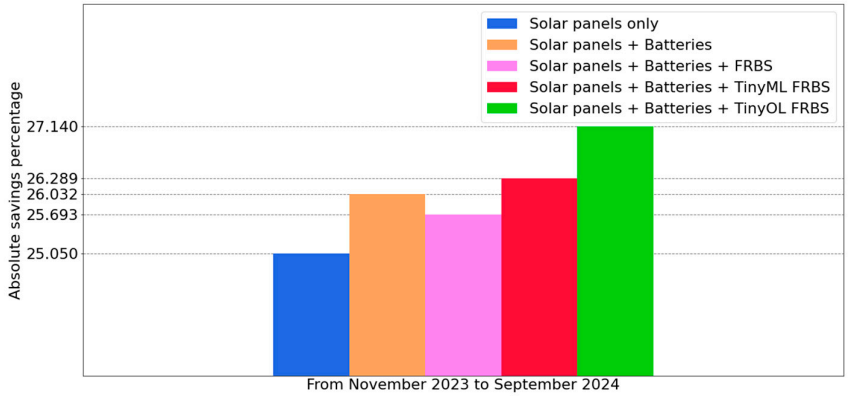


Figure 17. Total cost savings percentage.

The impact of online tuning was further analyzed by comparing the monthly cost savings percentage between the static TinyML-based FRBS (represented in red) and the online adaptive TinyOL-based FRBS (represented in green) in Figure 18, month by month during the experimentation period. Notably, the improvement observed in month 8 is significant, as it includes a partial vacation period with a markedly different consumption profile due to sudden changes on two specific occasions. The adaptive model achieves significantly better results compared to the static system.

Focusing on a specific month, July 2024, Figure 19 illustrates that the system with the online adaptive FRBS, which adjusts daily using data from the previous day, achieves notable savings peaks on certain days (shown in green) compared to the static system (shown in red). For instance, this is evident on Friday, 5 July 2024. Moreover, variations were identified between high-consumption days, such as non-working days and working days. These behavioral changes influence the savings percentage achieved, with minimal peaks observed on Saturdays.

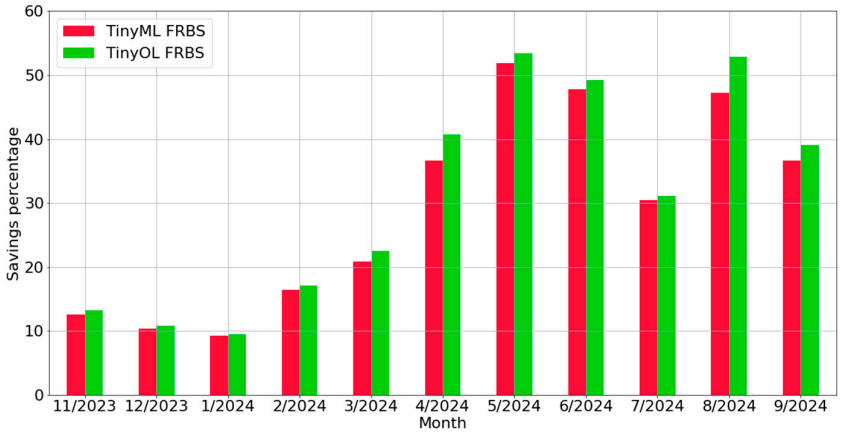


Figure 18. Monthly cost savings percentage.

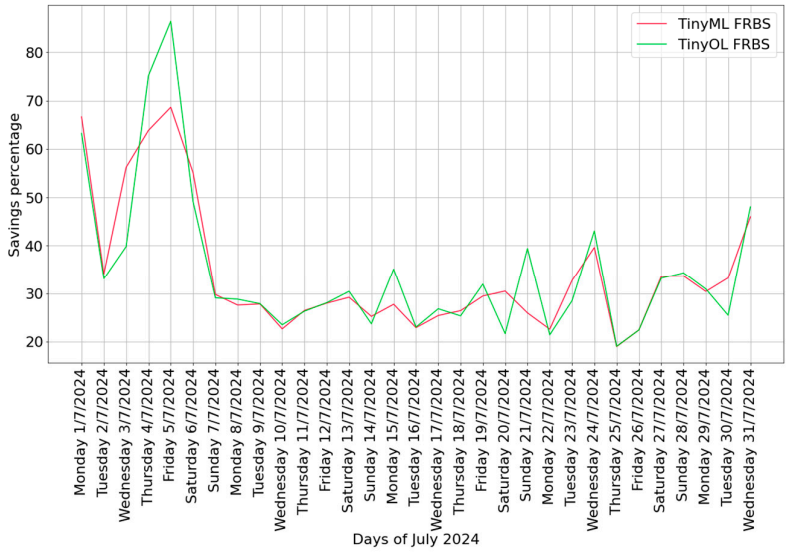


Figure 19. Daily cost savings percentage during July 2024.

Finally, Figure 20 shows the impact of decisions made at each hour (the minimum unit available to evaluate the model's performance). In this case, the online adaptive FRBS achieves better results during periods of high consumption and renewable energy availability (from 7 to 10 a.m. and 12 to 2 p.m.), where the FRBS's decisions have greater weight.

Overall, accumulated gains of TinyOL FRBS compared to Tiny ML FRBS in terms of cost savings percentage is 1.17% from 1 November 2023 to 30 September 2024. Table 5 shows monthly cost savings percentage of TinyOL compared to TinyML, supplementing Figure 18. Similarly, Table 6 shows daily cost savings percentage, supplementing Figure 19, and Table 7 shows hourly cost savings percentage, supplementing Figure 20.

The proposed μCHC operates under a fixed number of trials. Let n denote the number of fitness evaluations performed during one buffered batch. Since all selection and crossover operations are constant-time for a fixed population size, the total runtime is linear in the number of trials. Therefore, the time complexity is $O(n)$. In our implementation, each buffered batch uses $n = 100$ trials.

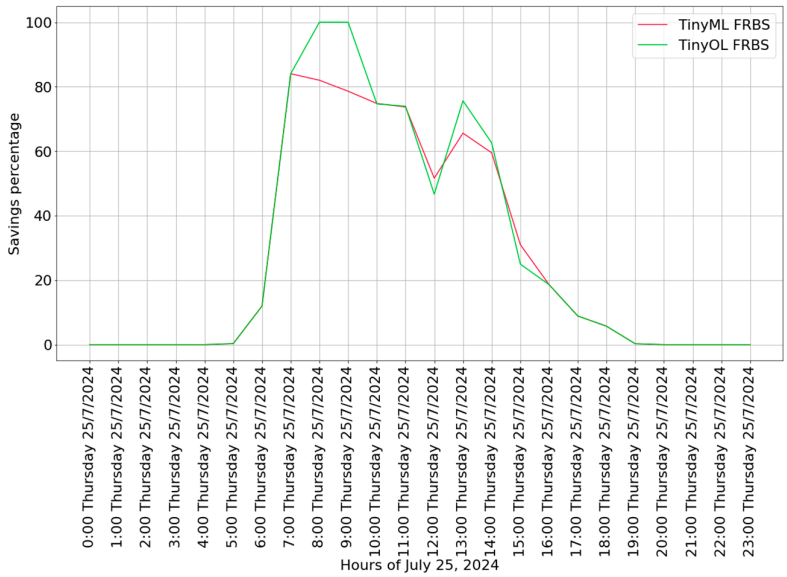


Figure 20. Hourly cost savings percentage.

Table 5. Monthly gains of TinyOL FRBS compared to Tiny ML FRBS in terms of cost savings percentage.

Month	Cost Savings (%)
November 2023	0.72
December 2023	0.58
January 2024	0.21
February 2024	0.77
March 2024	2.19
April 2024	6.88
May 2024	3.46
June 2024	2.88
July 2024	0.97
August 2024	11.73
September 2024	4.02

Table 6. Daily gains of TinyOL FRBS compared to Tiny ML FRBS in terms of cost savings percentage in July 2024.

Day	Cost Savings (%)	Day	Cost Savings (%)
Mon. 1 July 2024	−10.26	Wed. 17 July 2024	1.88
Tue. 2 July 2024	−1.57	Thu. 18 July 2024	−1.47
Wed. 3 July 2024	−37.47	Fri. 19 July 2024	3.52
Thu. 4 July 2024	31.29	Sat. 20 July 2024	−12.73
Fri. 5 July 2024	56.70	Sun. 21 July 2024	18.19
Sat. 6 July 2024	−13.37	Mon. 22 July 2024	−1.52
Sun. 7 July 2024	−0.99	Tue. 23 July 2024	−6.30
Mon. 8 July 2024	1.70	Wed. 24 July 2024	5.67
Tue. 9 July 2024	0.09	Thu. 25 July 2024	0.08
Wed. 10 July 2024	1.11	Fri. 26 July 2024	−0.01
Thu. 11 July 2024	−0.20	Sat. 27 July 2024	−0.54
Fri. 12 July 2024	0.13	Sun. 28 July 2024	1.02
Sat. 13 July 2024	1.76	Mon. 29 July 2024	0.74
Sun. 14 July 2024	−2.07	Tue. 30 July 2024	−11.64
Mon. 15 July 2024	10.20	Wed. 31 July 2024	3.82
Tue. 16 July 2024	0.14		

Table 7. Hourly gains of TinyOL FRBS compared to Tiny ML FRBS in terms of cost savings percentage on Thursday 25 July 2024.

Hour	Cost Savings (%)	Hour	Cost Savings (%)
0:00	0.00	12:00	−10.25
1:00	0.00	13:00	29.19
2:00	0.00	14:00	7.61
3:00	0.00	15:00	−8.75
4:00	0.00	16:00	0.00
5:00	0.00	17:00	0.00
6:00	0.00	18:00	0.00
7:00	0.00	19:00	0.00
8:00	100.00	20:00	0.00
9:00	100.00	21:00	0.00
10:00	−0.51	22:00	0.00
11:00	0.85	23:00	0.00

The memory footprint is constant because both the population size (10 chromosomes) and the FRBS parameter dimensionality (17 parameters per chromosome) are fixed. The algorithm therefore requires storage for 10×17 integer parameters, yielding a constant-space design. Formally, space complexity is $O(1)$.

Regarding the computational resource consumption of the model, global variables utilize 45,760 bytes out of the 262,144 bytes available in RAM, equivalent to only 17% of the total. Of this, 180 bytes are used to maintain the FRBS in memory (DB: 3 variables $\times$ 3 labels $\times$ 2 parameters $\times$ 4 bytes (int) + RB: 9 rules $\times$ 3 labels identifier $\times$ 4 bytes (int)), 288 bytes to store the daily buffered data batch (24 examples $\times$ 3 variables $\times$ 4 bytes (float)), and 680 bytes for the μCHC evolutionary algorithm's population (10 chromosomes $\times$ 17 parameters $\times$ 4 bytes (int)).

In terms of processing speed, the time required for a partial tuning using the proposed evolutionary algorithm is 217 milliseconds. Specifically, the time taken to evaluate each chromosome on the buffered data batch is only 2 milliseconds, while inferring a decision using the FRBS each hour takes 0.03 milliseconds. To quantify the energy consumption of the model, a base power consumption of 0.125 W (5 V $\times$ 0.025 A) was considered. Therefore, the estimated energy required per inference is 3.75 μJ (0.03 ms $\times$ 0.125 W), whereas each partial tuning consumes approximately 27.1 mJ (217 ms $\times$ 0.125 W). These results demonstrate that an economic processor suitable for IoT tasks can efficiently implement the proposed compact model, supporting both low-energy inference and occasional tuning operations, and could even accommodate more frequent intervention intervals without significant energy overhead.

It is important to clarify that the purpose of this case study is to validate the behavior of the proposed adaptive mechanism under evolving operating conditions rather than to characterize long-term energy consumption patterns. The available dataset includes several phases where the underlying conditions change, providing controlled forms of concept drift that are sufficient for evaluating whether the system can adapt effectively. Accordingly, the experimental analysis focuses on the stability and responsiveness of the online tuning process, rather than on modelling extended temporal dynamics of energy consumption.

The results demonstrate that the proposed model for designing fuzzy systems at the edge with online tuning not only significantly improves cost savings by adjusting in real-time to changing environmental conditions, but also operates efficiently on low-resource IoT devices. The compact implementation of the selected evolutionary model, optimized for online learning environments, strikes an appropriate balance between accuracy and computational resource consumption, enabling continuous adjustment without compromis-

ing the device's processing or memory capacity. Furthermore, the evaluation model based on a real-time exact simulator facilitates the rapid assessment of chromosomes, ensuring that the system can adapt efficiently within the required time frame.

The analysis using different measurement units (annual, monthly, daily and hourly) highlights that the online adaptive FRBS achieves greater cost savings in the selected application compared to a model that employs a static FRBS, adjusted only once during the design phase.

The resulting fuzzy DB as of 30 September 2024 is shown in Figure 21. For comparison, the initial FRBS, depicted in Figure 10, is overlaid using dotted gray lines.

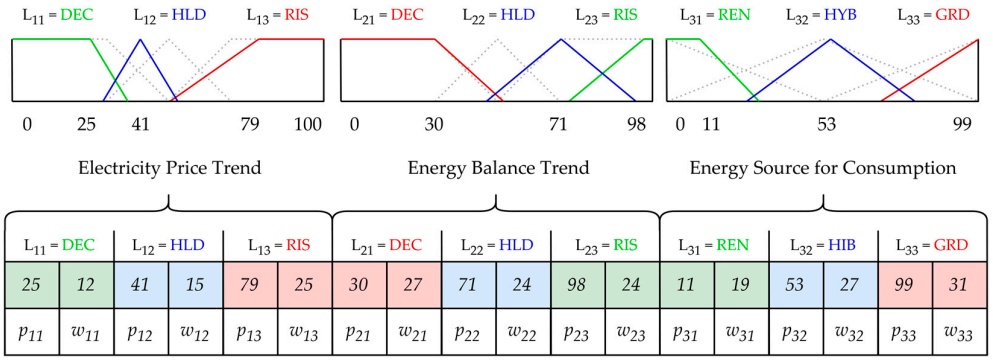


Figure 21. Diagram of the resulting DB of the FRBS on 30 September 2024. Colors indicate the correspondence between each linguistic label (L_{ij}) of variable i and label j, and the position (p_{ij}) and width (w_{ij}) parameters that define it in the chromosome encoding.

Following the notion that rule-based models constitute transparent and intrinsically interpretable model systems [15–21], we quantify the interpretability of our fuzzy model using standard structural and semantic indicators commonly employed in interpretable FRBS design. First, the Rule-Base Complexity (RBC) [112] remains extremely compact, with only 9 rules and 2 antecedents per rule, which lies at the lower end of typical FRBS structures and enhances cognitive accessibility. Second, the Linguistic Label Compactness Index (LLCI) [113], based on the 18 parameters of the symmetric triangular labels, reflects the reduced parametric burden characteristic of highly interpretable fuzzy models. Third, the Membership Function Overlap Index (MFOI) [113] remains low (0.25 initially and 0.08 after online adaptation), indicating controlled overlap and low ambiguity between terms, values consistent with the range of high interpretability reported in the literature. Finally, to address the requirement that trustworthy AI systems maintain interpretability over time and avoid semantic drift, we compute a Semantic Stability Index (SSI) [114] of 0.87, showing that the linguistic meaning of the labels remains largely preserved after online tuning. Together, these indicators demonstrate that the proposed FRBS retains both structural transparency and semantic consistency throughout the learning process.

Overall, these results support the viability of the proposed model as an effective solution for online adjustment of the FRBS at the edge, laying the groundwork for future research on integrating adaptive learning techniques into IoT devices.

Beyond the specific scenario evaluated in this section, the proposed online-adaptive linguistic fuzzy approach presented in Section 3 exhibits strong potential for scalability across a wide range of application domains. Because the mechanism relies on incremental adaptation of interpretable fuzzy DB, it can be transferred to any context where systems must operate under non-stationary conditions using limited computational resources. This includes online calibration of embedded controllers, adaptive decision-making in autonomous robots, real-time sensor fusion in IoT monitoring networks, and evolving quality-assessment or anomaly-detection modules in industrial settings. The fact that

the adaptation process only requires compact data buffers and lightweight evolutionary updates facilitates deployment in heterogeneous platforms, from microcontrollers to edge devices. These characteristics demonstrate that the proposed method is not restricted to the use case presented, but constitutes a general framework with broad universality and expansion potential.

5. Conclusions

Recent technological advances have enabled the incorporation of increasing levels of intelligence into resource-constrained devices, endowing them with the ability to learn and dynamically adapt to their surroundings. In this context and considering that fuzzy systems have historically been a key tool in such devices, particularly in applications such as control systems, due to their ability to provide a good balance between interpretability and precision, and low computational requirements, this work proposes an approach to extend fuzzy systems by adapting them to the Tiny Online Learning philosophy. Specifically, this approach allows linguistic fuzzy systems to learn online, adapting a part of their knowledge base in environments where conditions may change dynamically. To illustrate the feasibility of this idea, an online tuning mechanism that enables a linguistic fuzzy system to evolve continuously rather than remain static is proposed. This is achieved by dynamically adapting the definitions of its linguistic labels using optimized evolutionary metaheuristics, a micro genetic algorithm, to align with new environmental conditions without requiring complete retraining, and maintaining good semantic stability, as the data buffer used helps to avoid destroying previous knowledge or responding to changes that may only be temporary very quickly. Such an approach broadens the applicability of linguistic fuzzy systems to current TinyML and Edge AI scenarios, especially where high interpretability is required and only a low-power device is available.

Finally, as a practical application of a fuzzy system in a dynamic environment, a system aimed at optimizing the costs associated with energy consumption in households equipped with hybrid photovoltaic installations is proposed. The system manages the proportions of various energy sources to optimize profitability. The results demonstrate that the proposed model increases savings compared to the same model without online adaptation. Furthermore, the system has been shown to operate efficiently on low-resource hardware, using only 17% of the available RAM and achieving minimal processing times, close to real-time performance.

The limitation of this illustrative model, proposed to demonstrate that linguistic FRBSs can be used for online adaptation directly on a resource-constrained device, would arise when changing environmental conditions exceed what can be accommodated solely through the adjustment of linguistic terms. In such cases, a model capable of modifying its RB would be required. Moreover, a general limitation of this and any other online adaptive linguistic fuzzy system is that the knowledge base needed for proper operation in a given application may grow beyond what a resource-constrained device can support in terms of memory or computational performance, although such large knowledge bases would only be expected in very complex, and likely uncommon, applications.

Overall, the proposed approach contributes to the development of explainable and adaptive intelligence for resource-constrained devices, advancing the integration of fuzzy logic and evolutionary learning within the TinyML domain.

As a direction for future research, we plan to extend this study both by studying data buffer update policies to achieve a compromise between computational consumption and adaptive capacity, taking into account the past, but avoiding abrupt changes due to noise or other circumstances, and by supporting major changes in the environment that require adaptation not only of the linguistic concepts but also of the rule base itself.

Author Contributions: Conceptualization, J.M.-M., F.A.M. and A.P.; methodology, J.M.-M.; software, J.M.-M.; validation, F.A.M.; formal analysis, J.M.-M.; investigation, J.M.-M.; resources, A.P.; data curation, J.M.-M.; writing—original draft preparation, J.M.-M. and F.A.M.; writing—review and editing, A.P. and A.M.R.; visualization, A.M.R.; supervision, A.P.; project administration, A.M.R.; funding acquisition, A.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Ministry of Science, Innovation and Universities of Spain, grant number PID2023-150070NB-I00.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to privacy reasons.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations and notations are used in this manuscript:

Abbreviations	
TinyML	Tiny Machine Learning
IoT	Internet of Things
EC	Edge Computing
FRBS	Fuzzy Rule-Based System
MCU	Microcontroller
ML	Machine Learning
ODL	On-Device Learning
TinyOL	TinyML with Online Learning
HW	Hardware
SoC	System-on-a-Chip
CPU	Central Processing Unit
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
FPGA	Field-Programmable Gate Array
EFS	Evolving Fuzzy System
RB	Rule Base
KB	Knowledge Base
DB	Data Base
NFS	Neuro-Fuzzy System
GFS	Genetic Fuzzy System
μGA	Micro Genetic Algorithm
EPT	Electricity Price Trend
EBT	Energy Balance Trend
ESC	Energy Source for Consumption
DEC	Linguistic input label representing a “decline” trend
HLD	Linguistic input label representing a “hold” trend
RIS	Linguistic input label representing a “rise” trend
REN	Linguistic output label representing a “renewable” energy source
HYB	Linguistic output label representing a “hybrid” energy source
GRD	Linguistic output label representing a “grid” energy source
Notations	
L_{ij}	Linguistic label j of variable i
p_{ij}	Position of linguistic label L_{ij}
w_{ij}	Width of linguistic label L_{ij}
α_{ij}	Lateral displacement of the position of linguistic label L_{ij}

β_{ij}	Amplitude variation of the width of linguistic label L_{ij}
y'	Predicted output of the model
h_i	Matching degree of the rule R_i
M_i	Maximum value of consequent linguistic label from rule R_i
DB_i	Fuzzy Data Base encoding as chromosome i
d	Difference threshold between two parent chromosomes
$trials$	Number of trials evaluated in the current partial execution of the genetic algorithm
mt	Maximum number of trials evaluated in each partial execution of the genetic algorithm
L	Chromosome length of the genetic algorithm
p	Population size of the genetic algorithm
$P(t)$	Current population of chromosomes of the genetic algorithm
$P'(t)$	Current population of parent chromosomes of the genetic algorithm
$C(t)$	Current population of child chromosomes of the genetic algorithm

References

1. Khan, T.; Tian, W.; Zhou, G.; Ilager, S.; Gong, M.; Buyya, R. Machine learning (ML)-centric resource management in cloud computing: A review and future directions. *J. Netw. Comput. Appl.* **2022**, *204*, 103405. [CrossRef]
2. Xue, T.; Zhang, Y.; Wang, Y.; Wang, W.; Li, S.; Zhang, H. Edge computing for IoT: Novel insights from a comparative analysis of access control models. *Comput. Netw.* **2025**, *270*, 111468. [CrossRef]
3. Yang, L.; Shami, A. IoT data analytics in dynamic environments: From an automated machine learning perspective. *Eng. Appl. Artif. Intell.* **2022**, *116*, 105366. [CrossRef]
4. Malazi, H.T.; Chaudhry, S.R.; Kazmi, A.; Palade, A.; Cabrera, C.; White, G.; Clarke, S. Dynamic service placement in multi-access edge computing: A systematic literature review. *IEEE Access* **2022**, *10*, 32639–32688. [CrossRef]
5. Srikanth, K.; Ungureanu, T. Organizational adaptation in dynamic environments: Disentangling the effects of how much to explore versus where to explore. *Strat. Manag. J.* **2025**, *46*, 19–48. [CrossRef]
6. Takahashi, A.; Takahashi, S. A new interval type-2 fuzzy logic system under dynamic environment: Application to financial investment. *Eng. Appl. Artif. Intell.* **2021**, *100*, 104154. [CrossRef]
7. Li, W.; Hacid, H.; Almazrouei, E.; Debbah, M. A comprehensive review and a taxonomy of edge machine learning: Requirements, paradigms, and techniques. *AI* **2023**, *4*, 729–786. [CrossRef]
8. Ray, P.P. A review on TinyML: State-of-the-art and prospects. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 1595–1623. [CrossRef]
9. Sanchez-Iborra, R.; Skarmeta, A.F. TinyML-enabled frugal smart objects: Challenges and opportunities. *IEEE Circuits Syst. Mag.* **2020**, *20*, 4–18. [CrossRef]
10. Abadade, Y.; Temouden, A.; Bamoumen, H.; Benamar, N.; Chtouki, Y.; Hafid, A.S. A comprehensive survey on TinyML. *IEEE Access* **2023**, *11*, 96892–96922. [CrossRef]
11. Dhar, S.; Guo, J.; Liu, J.; Tripathi, S.; Kurup, U.; Shah, M. A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM Trans. Internet Things* **2021**, *2*, 1–49. [CrossRef]
12. Ren, H.; Anicic, D.; Runkler, T.A. TinyOL: TinyML with online-learning on microcontrollers. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8. [CrossRef]
13. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353. [CrossRef]
14. Zhang, K.; Shao, T.; Sun, Y.; Xu, X.; Zhang, X.; Zhou, X.; Ding, K.; Huang, S. Interpretable research of fuzzy methods: A literature survey. *Inf. Fusion* **2025**, *126*, 103524. [CrossRef]
15. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef]
16. Murdoch, W.J.; Singh, C.; Kumbier, K.; Abbasi-Asl, R.; Yu, B. Interpretable machine learning: Definitions, methods, and applications. *arXiv* **2019**, arXiv:1901.04592. [CrossRef]
17. Doshi-Velez, F.; Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv* **2017**, arXiv:1702.08608. [CrossRef]
18. Magdalena, L. Fuzzy Systems Interpretability: What, Why and How. In *Fuzzy Approaches for Soft Computing and Approximate Reasoning: Theories and Applications*; Bouchon-Meunier, B., Ed.; Springer International Publishing: Cham, Switzerland, 2020; pp. 111–122. [CrossRef]
19. Arrieta, A.B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-Lopez, S.; Molina, D.; Benjamins, R.; et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI. *Inf. Fusion* **2020**, *58*, 82–115. [CrossRef]

20. Linardatos, P.; Papastefanopoulos, V.; Kotsiantis, S. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy* **2020**, *23*, 18. [CrossRef]
21. Ali, S.; Abuhmed, T.; El-Sappagh, S.; Muhammad, K.; Alonso-Moral, J.M.; Confalonieri, R.; Herrera, F. Explainable Artificial Intelligence (XAI): What We Know and What Is Left to Attain Trustworthy Artificial Intelligence. *Inf. Fusion* **2023**, *99*, 101805. [CrossRef]
22. Singh, A.; Raj, K.; Meghwar, T.; Roy, A.M. Efficient paddy grain quality assessment approach utilizing affordable sensors. *AI* **2024**, *5*, 686–703. [CrossRef]
23. Rout, R.R.; Vemireddy, S.; Raul, S.K.; Somayajulu, D.V. Fuzzy logic-based emergency vehicle routing: An IoT system development for smart city applications. *Comput. Electr. Eng.* **2020**, *88*, 106839. [CrossRef]
24. Krishnan, R.S.; Julie, E.G.; Robinson, Y.H.; Raja, S.; Kumar, R.; Thong, P.H. Fuzzy logic based smart irrigation system using internet of things. *J. Clean. Prod.* **2020**, *252*, 119902. [CrossRef]
25. Gousev, E. Recent Progress on TinyML Technologies and Opportunities. Available online: https://sites.google.com/g.harvard.edu/tinyml-fall2020/lectures#h.839rbio9569w (accessed on 8 December 2025).
26. Capogrosso, L.; Cunico, F.; Cheng, D.S.; Fummi, F.; Cristani, M. A machine learning-oriented survey on tiny machine learning. *IEEE Access* **2024**, *12*, 23406–23426. [CrossRef]
27. Zhou, A.; Muller, R.; Rabaey, J. Memory-efficient, limb position-aware hand gesture recognition using hyperdimensional computing. *arXiv* **2021**, arXiv:2103.05267.
28. Miao, H.; Lin, F.X. Enabling large neural networks on tiny microcontrollers with swapping. *arXiv* **2021**, arXiv:2101.08744. [CrossRef]
29. Huang, Q. High-Performance and Lightweight AI Model with Integrated Self-Attention Layers for Soybean Pod Number Estimation. *AI* **2025**, *6*, 135. [CrossRef]
30. Benmeziane, H.; El Maghraoui, K.; Ouarnoughi, H.; Niar, S.; Wistuba, M.; Wang, N. Hardware-Aware Neural Architecture Search: Survey and Taxonomy. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, (IJCAI), Montreal, QC, Canada, 19–27 August 2021; pp. 4322–4329. [CrossRef]
31. Ghamari, S.; Ozcan, K.; Dinh, T.; Melnikov, A.; Carvajal, J.; Ernst, J.; Chai, S. Quantization-guided training for compact TinyML models. *arXiv* **2021**, arXiv:2103.06231.
32. Lokner Lađević, A.; Kramberger, T.; Kramberger, R.; Vlahek, D. Detection of AI-generated synthetic images with a lightweight CNN. *AI* **2024**, *5*, 1575–1593. [CrossRef]
33. Delnevo, G.; Mirri, S.; Prandi, C.; Manzoni, P. An evaluation methodology to determine the actual limitations of a TinyML-based solution. *Internet Things* **2023**, *22*, 100729. [CrossRef]
34. Tsoukas, V.; Gkogkidis, A.; Boumpa, E.; Papafotikas, S.; Kakarountas, A. A gas leakage detection device based on the technology of TinyML. *Technol.* **2023**, *11*, 45. [CrossRef]
35. Berta, R.; Dabbous, A.; Lazzaroni, L.; Pau, D.; Bellotti, F. Developing a TinyML Image Classifier in a Hour. *IEEE Open J. Ind. Electron. Soc.* **2024**, *5*, 946–960. [CrossRef]
36. Disabato, S.; Roveri, M. Tiny machine learning for concept drift. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 8470–8481. [CrossRef]
37. Gupta, C.; Suggala, A.S.; Goyal, A.; Simhadri, H.V.; Paranjape, B.; Kumar, A.; Goyal, S.; Udupa, R.; Varma, M.; Jain, P. ProToNN: Compressed and accurate kNN for resource-scarce devices. In Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 July 2017; pp. 1331–1340. Available online: https://proceedings.mlr.press/v70/gupta17a.html (accessed on 8 December 2025).
38. Takele, A.K.; Villányi, B. Resource-Efficient Clustered Federated Learning Framework for Industry 4.0 Edge Devices. *AI* **2025**, *6*, 30. [CrossRef]
39. Abadade, Y.; Benamar, N.; Bagaa, M.; Chaoui, H. Empowering Healthcare: TinyML for Precise Lung Disease Classification. *Future Internet* **2024**, *16*, 391. [CrossRef]
40. Mhaouch, A.; Gtifa, W.; Machhout, M. FPGA Hardware Acceleration of AI Models for Real-Time Breast Cancer Classification. *AI* **2025**, *6*, 76. [CrossRef]
41. Edouard, P.; Campo, D. Design and validation of Withings ECG Software 2, a tiny neural network based algorithm for detection of atrial fibrillation. *Comput. Biol. Med.* **2025**, *185*, 109407. [CrossRef]
42. Khan, S.I.; Dawood, H.; Khan, M.A.; Issa, G.F.; Hussain, A.; Alnfiai, M.M.; Adnan, K.M. Transition-aware human activity recognition using an ensemble deep learning framework. *Comput. Hum. Behav.* **2025**, *162*, 108435. [CrossRef]
43. Lokhande, H.; Ganorkar, S.R. Object detection in video surveillance using MobileNetV2 on resource-constrained low-power edge devices. *Bull. Electr. Eng. Inform.* **2025**, *14*, 357–365. [CrossRef]
44. Craighero, M.; Quarantiello, D.; Rossi, B.; Carrera, D.; Fragneto, P.; Boracchi, G. On-device personalization for human activity recognition on STM32. *IEEE Embed. Syst. Lett.* **2023**, *16*, 106–109. [CrossRef]

45. Lin, J.; Zhu, L.; Chen, W.M.; Wang, W.C.; Gan, C.; Han, S. On-device training under 256kb memory. In Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022), New Orleans, LO, USA, 28 November–9 December 2022; Volume 35, pp. 22941–22954. Available online: https://proceedings.neurips.cc/paper_files/paper/2022/file/90c56c77c6df45fc8e556a096b7a2b2e-Paper-Conference.pdf (accessed on 8 December 2025).
46. Demil, S.; Bouzar-Benlabiod, L.; Paillet, G. Cost efficient mammogram segmentation and classification with NeuroMem® chip for breast cancer detection. In Proceedings of the 24th International Conference on Information Reuse and Integration for Data Science (IRI), San Diego, CA, USA, 4–6 August 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 273–278. [CrossRef]
47. Le Gallo, M.; Lammie, C.; Büchel, J.; Carta, F.; Fagbohungbe, O.; Mackin, C.; Tsai, H.; Narayanan, V.; Sebastian, A.; El Maghraoui, K.; et al. Using the IBM analog in-memory hardware acceleration kit for neural network training and inference. *APL Mach. Learn.* **2023**, *1*, 041102. [CrossRef]
48. De Vita, F.; Nawaiseh, R.M.; Bruneo, D.; Tomaselli, V.; Lattuada, M.; Falchetto, M. µ-ff: On-device forward-forward training algorithm for microcontrollers. In Proceedings of the IEEE International Conference on Smart Computing (SMARTCOMP), Nashville, TN, USA, 26–30 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 49–56. [CrossRef]
49. De Vita, F.; Nocera, G.; Bruneo, D.; Tomaselli, V.; Falchetto, M. On-device training of deep learning models on edge microcontrollers. In Proceedings of the IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), Espoo, Finland, 22–25 August 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 62–69. [CrossRef]
50. Srinivasan, R.; Mignacco, F.; Sorbaro, M.; Refinetti, M.; Cooper, A.; Kreiman, G.; Dellaferrera, G. Forward learning with top-down feedback: Empirical and analytical characterization. *arXiv* **2023**, arXiv:2302.05440.
51. Frenkel, C.; Indiveri, G. ReckOn: A 28 nm sub-mm^2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales. In Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 20–26 February 2022; IEEE: Piscataway, NJ, USA, 2022; Volume 65, pp. 1–3. [CrossRef]
52. Abernot, M.; Azemard, N.; Todri-Sanial, A. Oscillatory neural network learning for pattern recognition: An on-chip learning perspective and implementation. *Front. Neurosci.* **2023**, *17*, 1196796. [CrossRef] [PubMed]
53. Vorabbi, L.; Maltoni, D.; Borghi, G.; Santi, S. Enabling on-device continual learning with binary neural networks. *arXiv* **2024**, arXiv:2401.09916. [CrossRef]
54. Cai, H.; Gan, C.; Zhu, L.; Han, S. TinyTL: Reduce memory, not parameters for efficient on-device learning. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, BC, Canada, Virtual Conference; Volume 33, pp. 11285–11297. Available online: https://proceedings.neurips.cc/paper_files/paper/2020/file/81f7acabd411274fcf65ce2070ed568a-Paper.pdf (accessed on 8 December 2025).
55. Ren, H.; Anicic, D.; Li, X.; Runkler, T. On-device online learning and semantic management of TinyML systems. *ACM Trans. Embed. Comput. Syst.* **2024**, *23*, 55. [CrossRef]
56. Pavan, M.; Ostrovan, E.; Caltabiano, A.; Roveri, M. TyBox: An automatic design and code generation toolbox for TinyML incremental on-device learning. *ACM Trans. Embed. Comput. Syst.* **2024**, *23*, 42. [CrossRef]
57. Pereira, E.A.M.; da Silva Santos, J.F.; de Andrade Barboza, E. An energy efficient TinyML model for a water potability classification problem. *Sustain. Comput. Inform. Syst.* **2024**, *43*, 101010. [CrossRef]
58. Gomez-Bautista, A.; Mendez, D.; Alvarado-Rojas, C.; Mondragon, I.F.; Colorado, J.D. On the deployment of edge AI models for surface electromyography-based hand gesture recognition. *AI* **2025**, *6*, 107. [CrossRef]
59. Roshan, A.N.; Gokulapriyan, B.; Siddarth, C.; Kokil, P. Adaptive traffic control with TinyML. In Proceedings of the Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 25–27 March 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 451–455. [CrossRef]
60. Ahmed, K.; Hassan, M. TinyCare: A TinyML-based low-cost continuous blood pressure estimation on the extreme edge. In Proceedings of the IEEE 10th International Conference on Healthcare Informatics (ICHI), Rochester, MN, USA, 11–14 June 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 264–275. [CrossRef]
61. Kumar, S.; Rachna, P.; Hiremath, R.B.; Ramadurgam, V.S.; Shaw, D.K. Survey on implementation of TinyML for real-time sign language recognition using smart gloves. Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT), Mandya, India, 26–27 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–7. [CrossRef]
62. Liu, R.; Xie, M.; Liu, A.; Song, H. Joint optimization of risk factors and energy consumption in IoT networks with TinyML-enabled Internet of UAVs. *IEEE Internet Things J.* **2024**, *11*, 20983–20994. [CrossRef]
63. Anam, K.; Rizal, N.A.; Ilyas, Z.; Avian, C.; Muttaqin, A.Z.; Ramadhan, M.E.; Swasono, D.I. Evaluation of long short-term memory on simultaneous and proportional myoelectric control system for individual finger movements. *Res. Biomed. Eng.* **2025**, *41*, 9. [CrossRef]
64. Piątkowski, D.; Walkowiak, K. TinyML-based concept system used to analyze whether the face mask is worn properly in battery-operated conditions. *Appl. Sci.* **2022**, *12*, 484. [CrossRef]

65. Vasconcelos, D.; Nunes, N.J.; Förster, A.; Gomes, J.P. Optimal 2D audio features estimation for a lightweight application in mosquitoes species: Ecoacoustics detection and classification purposes. *Comput. Biol. Med.* **2024**, *168*, 107787. [CrossRef] [PubMed]
66. Andrade, P.; Silva, I.; Signoretti, G.; Silva, M.; Dias, J.; Marques, L.; Costa, D.G. An unsupervised TinyML approach applied for pavement anomalies detection under the Internet of Intelligent Vehicles. In Proceedings of the IEEE International Workshop on Metrology for Industry 4.0 & IoT, Rome, Italy, 7–9 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 642–647. [CrossRef]
67. Antonini, M.; Pincheira, M.; Vecchio, M.; Antonelli, F. An adaptable and unsupervised TinyML anomaly detection system for extreme industrial environments. *Sensors* **2023**, *23*, 2344. [CrossRef] [PubMed]
68. Pereira, E.S.; Marcondes, L.S.; Silva, J.M. On-Device Learning TinyML for Anomaly Detection Based on Extreme Values Theory. *IEEE Micro* **2023**, *43*, 58–65. [CrossRef]
69. Scherer, M.; Cioflan, C.; Magno, M.; Benini, L. Work in Progress: Linear Transformers for TinyML. In Proceedings of the 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE), Valencia, Spain, 25–27 March 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 1–2. [CrossRef]
70. Vucetic, D.; Tayaranian, M.; Ziaeefard, M.; Clark, J.J.; Meyer, B.H.; Gross, W.J. Efficient Fine-Tuning of BERT Models on the Edge. In Proceedings of the 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, 27 May–1 June 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1838–1842. [CrossRef]
71. Szydlo, T.; Jayaraman, P.P.; Li, Y.; Morgan, G.; Ranjan, R. TinyRL: Towards Reinforcement Learning on Tiny Embedded Devices. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM), Atlanta, GA, USA, 17–21 October 2022; ACM: New York, NY, USA, 2022; pp. 4985–4988. [CrossRef]
72. Abououf, M.; Mizouni, R.; Singh, S.; Otrok, H.; Damiani, E. Self-Supervised Online and Lightweight Anomaly and Event Detection for IoT Devices. *IEEE Internet Things J.* **2022**, *9*, 25285–25299. [CrossRef]
73. Peltonen, E.; Bennis, M.; Capobianco, M.; Debbah, M.; Ding, A.; Gil-Castiñeira, F.; Yang, T. 6G White Paper on Edge Intelligence. *arXiv* **2020**, arXiv:2004.14850. [CrossRef]
74. Kopparapu, K.; Lin, E.; Breslin, J.G.; Sudharsan, B. TinyFedTL: Federated Transfer Learning on Ubiquitous Tiny IoT Devices. In Proceedings of the 2022 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Pisa, Italy, 21–25 March 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 79–81. [CrossRef]
75. Benhoussa, S.; Sousa, G.D.; Chanet, J.P. FedBirdAg: A Low-Energy Federated Learning Platform for Bird Detection with Wireless Smart Cameras in Agriculture 4.0. *AI* **2025**, *6*, 63. [CrossRef]
76. Ibrahim, A. *Fuzzy Logic for Embedded Systems Applications*; Elsevier: Amsterdam, The Netherlands, 2003. [CrossRef]
77. Larsen, P.M. Industrial applications of fuzzy logic control. *Int. J. Man-Mach. Stud.* **1980**, *12*, 3–10. [CrossRef]
78. Sugeno, M. *Industrial Applications of Fuzzy Control*; Elsevier Sci. Inc.: New York, NY, USA, 1985; Available online: https://dl.acm.org/doi/10.5555/537323 (accessed on 8 December 2025).
79. Lee, C.C. Fuzzy logic in control systems: Fuzzy logic controller. I. *IEEE Trans. Syst. Man Cybern.* **1990**, *20*, 404–418. [CrossRef]
80. Khatoon, A.; Wang, W.; Wang, M.; Li, L.; Ullah, A. TinyML-enabled fuzzy logic for enhanced road anomaly detection in remote sensing. *Sci. Rep.* **2025**, *15*, 20659. [CrossRef]
81. Martínez-Rojas, M.; Cano, C.; Alcalá-Fdez, J.; Soto-Hidalgo, J.M. Interpretable fuzzy control for energy management in smart buildings using JFML-IoT and IEEE Std 1855–2016. *Appl. Sci.* **2025**, *15*, 8208. [CrossRef]
82. Rama Krishna, K.; Sudha, R.; Prasad, G.N.R.; Machana, J.R. Integration of edge computing and fuzzy logic to monitor novel coronavirus. In *Advances in Fuzzy-Based Internet of Medical Things (IoMT)*; Springer: Cham, Switzerland, 2024; pp. 255–269. [CrossRef]
83. Angelov, P. Evolving Fuzzy Systems. In *Granular, Fuzzy, and Soft Computing*; Springer: New York, NY, USA, 2023; pp. 725–741. [CrossRef]
84. Yang, Z.X.; Rong, H.J. Evolving Kernel-Based Fuzzy System with Nonlinear Consequences. *Appl. Soft Comput.* **2024**, *167*, 112384. [CrossRef]
85. de Campos Souza, P.V.; Dragoni, M. EFNN-Nul0-A Trustworthy Knowledge Extraction about Stress Identification through Evolving Fuzzy Neural Networks. *Fuzzy Sets Syst.* **2024**, *487*, 109008. [CrossRef]
86. Rishabh, R.; Das, K.N. A Fusion of Decomposed Fuzzy-Based Decision-Making and Metaheuristic Optimization System for Sustainable Planning of Urban Transport. *Knowl. Based Syst.* **2025**, *324*, 113823. [CrossRef]
87. Harikumaran, M.; Vijayalakshmi, P. Optimizing Fertilizer Recommendations in Precision Agriculture: A Novel De-Fuzzification Approach with Adaptive Intelligent Optimization. *Knowl. Based Syst.* **2025**, *321*, 113550. [CrossRef]
88. Benyezza, H.; Bouhedda, M.; Rebouh, S. Zoning Irrigation Smart System Based on Fuzzy Control Technology and IoT for Water and Energy Saving. *J. Clean. Prod.* **2021**, *302*, 127001. [CrossRef]
89. Chiesa, G.; Di Vita, D.; Ghadirzadeh, A.; Herrera, A.H.M.; Rodriguez, J.C.L. A Fuzzy-Logic IoT Lighting and Shading Control System for Smart Buildings. *Autom. Constr.* **2020**, *120*, 103397. [CrossRef]

90. Fernandez, A.; Herrera, F.; Cordon, O.; del Jesus, M.J.; Marcelloni, F. Evolutionary Fuzzy Systems for Explainable Artificial Intelligence: Why, When, What For, and Where To? *IEEE Comput. Intell. Mag.* **2019**, *14*, 69–81. [CrossRef]
91. Márquez, A.A.; Márquez, F.A.; Roldán, A.M.; Peregrín, A. An Efficient Adaptive Fuzzy Inference System for Complex and High Dimensional Regression Problems in Linguistic Fuzzy Modelling. *Knowl. Based Syst.* **2013**, *54*, 42–52. [CrossRef]
92. Alcalá-Fdez, J.; Herrera, F.; Márquez, F.; Peregrín, A. Increasing Fuzzy Rules Cooperation Based on Evolutionary Adaptive Inference Systems. *Int. J. Intell. Syst.* **2007**, *22*, 1035–1064. [CrossRef]
93. Rodríguez, M.; Escalante, D.M.; Peregrín, A. Efficient Distributed Genetic Algorithm for Rule Extraction. *Appl. Soft Comput.* **2011**, *11*, 733–743. [CrossRef]
94. Cordón, O.; Herrera, F.; Hoffmann, F.; Magdalena, L. *Genetic Fuzzy Systems*; World Scientific: Singapore, 2001. [CrossRef]
95. Herrera, F.; Martínez, L. A 2-Tuple Fuzzy Linguistic Representation Model for Computing with Words. *IEEE Trans. Fuzzy Syst.* **2000**, *8*, 746–752. [CrossRef]
96. Alcalá, R.; Alcalá-Fdez, J.; Gacto, M.J.; Herrera, F. Rule Base Reduction and Genetic Tuning of Fuzzy Systems Based on the Linguistic 3-Tuples Representation. *Soft Comput.* **2007**, *11*, 401–419. [CrossRef]
97. Czogała, E.; Pedrycz, W. *Elements and Methods of Fuzzy Set Theory*; PWN: Warszawa, Poland, 1985. Available online: https://zbmath.org/0661.94029 (accessed on 8 December 2025).
98. Nauck, D.; Kruse, R. Neuro-Fuzzy Systems. In *Handbook of Fuzzy Computation*; CRC Press: Boca Raton, FL, USA, 2020; p. 319-D2. [CrossRef]
99. Gu, X.; Han, J.; Shen, Q.; Angelov, P.P. Autonomous learning for fuzzy systems: A review. *Artif. Intell. Rev.* **2023**, *56*, 7549–7595. [CrossRef]
100. Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley: Reading, MA, USA, 1989; Available online: https://dl.acm.org/doi/10.5555/534133 (accessed on 8 December 2025).
101. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1992. [CrossRef]
102. Takagi, T.; Sugeno, M. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Syst. Man Cybern.* **1985**, *15*, 116–132. [CrossRef]
103. Jang, J.S.R.; Sun, C.T.; Mizutani, E. Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [Book Review]. *IEEE Trans. Autom. Control* **1997**, *42*, 1482–1484. [CrossRef]
104. Mendel, J.M. Fuzzy logic systems for engineering: A tutorial. *Proc. IEEE* **1995**, *83*, 345–377. [CrossRef]
105. Klir, G.J.; Yuan, B. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*; Prentice Hall: Upper Saddle River, NJ, USA, 1994. Available online: https://dl.acm.org/doi/10.5555/202684 (accessed on 8 December 2025).
106. Santiago, A.; Dorronsoro, B.; Fraire, H.J.; Ruiz, P. Micro-Genetic Algorithm with Fuzzy Selection of Operators for Multi-Objective Optimization: µFAME. *Swarm Evol. Comput.* **2021**, *61*, 100818. [CrossRef]
107. Bilal, M.; Oladigbolu, J.O.; Mujeeb, A.; Al-Turki, Y.A. Cost-Effective Optimization of On-Grid Electric Vehicle Charging Systems with Integrated Renewable Energy and Energy Storage: An Economic and Reliability Analysis. *J. Energy Storage* **2024**, *100*, 113170. [CrossRef]
108. Karimianfard, H. A Robust Optimization Framework for Smart Home Energy Management: Integrating Photovoltaic Storage, Electric Vehicle Charging, and Demand Response. *J. Energy Storage* **2025**, *110*, 115259. [CrossRef]
109. Ukoba, K.; Olatunji, K.O.; Adeoye, E.; Jen, T.C.; Madyira, D.M. Optimizing Renewable Energy Systems through Artificial Intelligence: Review and Future Prospects. *Energy Environ.* **2024**, *35*, 3833–3879. [CrossRef]
110. Youssef, H.; Kamel, S.; Hassan, M.H.; Yu, J.; Safaraliev, M. A Smart Home Energy Management Approach Incorporating an Enhanced Northern Goshawk Optimizer to Enhance User Comfort, Minimize Costs, and Promote Efficient Energy Consumption. *Int. J. Hydrogen Energy* **2024**, *49*, 644–658. [CrossRef]
111. Luan, B.; Feng, X. Artificial Intelligence in Smart Buildings: A Bibliometrics-Based Visualization Analysis. *J. Asian Archit. Build. Eng.* **2025**, 1–25. [CrossRef]
112. Alonso, J.M.; Magdalena, L. HILK++: An Interpretability-Guided Fuzzy Modeling Methodology for Learning Readable and Comprehensible Fuzzy Rule-Based Classifiers. *Soft Comput.* **2011**, *15*, 1959–1980. [CrossRef]
113. Gacto, M.J.; Alcalá, R.; Herrera, F. Interpretability of Linguistic Fuzzy Rule-Based Systems: An Overview of Interpretability Measures. *Inf. Sci.* **2011**, *181*, 4340–4360. [CrossRef]
114. Gacto, M.J.; Alcalá, R.; Herrera, F. Integration of an Index to Preserve the Semantic Interpretability in the Multiobjective Evolutionary Rule Selection and Tuning of Linguistic Fuzzy Systems. *IEEE Trans. Fuzzy Syst.* **2010**, *18*, 515–531. [CrossRef]