



Beyond the Algebraic Product: Generalizing ANFIS with Parametric and Non-Parametric T-Norms

Bruna Novack¹, Juliano Buss¹, Gabriel Silva¹, Helida Santos^{2*}, Giancarlo Lucca², Adenauer Yamin¹ and Renata Reiser¹

¹Graduate Program in Computing (PPGC), Federal University of Pelotas (UFPel), Pelotas, RS, Brazil

²Center for Computational Sciences, Federal University of Rio Grande (FURG), Rio Grande, RS, Brazil

Citation: Bruna Novack, Juliano Buss, Gabriel Silva, Helida Santos, Giancarlo Lucca, et al. (2026) Beyond the Algebraic Product: Generalizing ANFIS with Parametric and Non-Parametric T-Norms. *J of Pion Artf Int Research* 2(5), 1-14. WMJ-JPAIR-155

Abstract

Machine Learning and Fuzzy Logic effectively complement each other in neuro-fuzzy systems, combining data-driven learning with interpretable reasoning under uncertainty. A prominent architecture is the Adaptive Neuro-Fuzzy Inference System (ANFIS), which employs the Takagi-Sugeno method to model complex nonlinear systems. However, standard ANFIS is traditionally limited to the algebraic product operation in its antecedent layer and often encounters computational bottlenecks on CPU hardware when handling high-dimensional datasets. In this work, we propose a generalization of the ANFIS antecedent layer by systematically integrating various families of parametric and non-parametric t-norms. We evaluate this architecture in two distinct environments: a traditional CPU-based baseline and a GPU-accelerated framework. Experiments conducted on various benchmark datasets demonstrate that replacing the fixed product operator with flexible t-norms improves predictive accuracy and adaptability, and statistical tests confirm the benefits of the proposed generalization.

***Corresponding author:** Helida Santos, Center for Computational Sciences, Federal University of Rio Grande (FURG), Rio Grande, RS, Brazil.

Received: 16.09.2026

Accepted: 21.09.2026

Published: 13.10.2026

Keywords: Fuzzy Logic, ANFIS generalization, Fuzzy Aggregators, Machine Learning, Neural Networks

This research was partially supported by Brazilian funding agencies: CAPES, CNPq (309160/2019-7, 3305805/2021-5, 150160/2023-2, 318141/2025-6), PqG/ FAPERGS (21/ 2551-0002057-1), FAPERGS (24/2551-0001396-2), FAPERGS/CNPq (23/2551-0000126-8), and PRONEX (16/ 2551-0000488-9).

I. Introduction

Nowadays, Machine Learning (ML) [1] has been present in a plethora of fields, such as recommendation systems [2], fuzzy systems [3]–[5], among others. There exist various ML approaches, but Neural Networks (NNs) [6] are believed to be crucial to intelligent algorithms. The diversity of current application areas includes detecting potential climate change-related disasters [7], supporting medical diagnosis [8], and even predicting stock behavior in finance [9].

Modeling uncertainty with Fuzzy Logic (FL) [10] is a viable strategy since it leverages a formal framework for the development of science and computational technologies. By handling uncertainty inherent in human reasoning, FL employs linguistic variables associated with the membership relations of Fuzzy Sets (FSs), allowing complex phenomena to be represented in a flexible and realistic manner.

Unlike systems that are strict in mapping inputs to decisions using true-or-false values, Fuzzy Inference Systems (FIS) interpret complex, imprecise inputs through qualitative representations and inference rules provided by experts, derived from numerical data or statistical methods. The process begins with the fuzzification step, converting crisp inputs (numerical data) derived from measurements or observations into fuzzy input sets. Next, the fuzzy inference mechanism, expressed as linguistic statements, activates the fuzzy rules. So, the operations are made with fuzzy sets, including rule antecedent matching, implication, and fuzzy inference. The fuzzy input sets, related to the rule antecedents, and the output set, associated with the consequent, are evaluated to identify the rules to be applied. After generating the fuzzy output set, the defuzzification process converts the qualitative information generated by the FIS into a final numerical value [11].

The Adaptive Neuro-Fuzzy Inference System (ANFIS) [12] constitutes a hybrid approach that integrates the adaptive learning capabilities of NNs with the approximate reasoning of Fuzzy Logic, employing the Takagi-Sugeno inference method [13]. ANFIS delivers an interpretable representation of the relationship between input variables and the system output, making it suitable for modeling complex and potentially nonlinear input-output relationships. Its architecture comprises five layers, beginning with the fuzzifica-

tion of input variables, followed by the computation and normalization of rule firing strengths, the selection of rule consequents, and the aggregation of the resulting outputs after the learning process. Through supervised learning, this hybrid architecture automatically adjusts the fuzzy rules and Membership Functions (MFs), enabling the model to effectively capture and represent patterns and nonlinear relationships present in the input data.

In [14], ANFIS was used as a tool to control nonlinear systems for efficiently controlling water levels in a didactic plant. In that work, ANFIS replicates the traditional controller response with a shorter actuation time. The ANFIS implementation used is publicly available [15] and was partially considered and adapted in this current study.

Traditional ANFIS implementations, such as the one in [15], are known to have practical bottlenecks, despite their sound theoretical basis and adoption in control and modeling systems [14]. Historically designed for CPU execution, these classic frameworks rely on sequential computations that struggle to scale when handling high-dimensional feature spaces, where rule generation grows exponentially. Besides, standard implementations were primarily designed for single-output regression tasks, requiring adaptations or thresholding heuristics to address classification problems.

To address these challenges, recent developments in neuro-fuzzy systems have increasingly incorporated deep learning techniques and frameworks. One example is ANFISpy [16], which implements neuro-fuzzy modeling within the PyTorch ecosystem. This integration enables GPU-based acceleration and automatic differentiation, removing the need to explicitly derive and implement gradient computations. In addition, ANFISpy supports multi-class classification by directly optimizing the cross-entropy loss, providing a practical connection between the interpretability of fuzzy inference systems and the computational scalability of modern deep learning workflows.

Building upon these complementary implementations, the theory-faithful CPU-based version and the modern GPU-accelerated framework, this work, a follow-up study of [17], investigates the generalization of the second layer (antecedent generation) in ANFIS. Rather than limiting ourselves to the stan-

standard algebraic product, we systematically incorporate various families of parametric and non-parametric t-norms and evaluate how structural and computational enhancements influence predictive accuracy across different input datasets.

Continuing previous studies on aggregators and fuzzy operators, this work aims to systematically evaluate the generalization of the second layer in the ANFIS architecture across both CPU and GPU computational frameworks, assessing how structural flexibility in antecedent aggregation affects predictive performance and runtime behavior.

The remainder of this paper is organized as follows. Section II reviews fundamental concepts of Fuzzy Logic, inference systems, and the five-layer ANFIS architecture. Section III introduces the generalized antecedent framework, the experimental protocol, the benchmark datasets, and the computational environments evaluated. Section IV presents the empirical results, execution time, and statistical assessments under the traditional CPU-based implementation. Section V evaluates the generalized model scaled to the GPU-accelerated ANFISpy architecture, analyzing multi-class accuracy, runtime dynamics, and non-parametric statistical tests. Section VI provides an in-depth architectural and operational comparison between the CPU and GPU implementations. Finally, Section VII concludes the paper and outlines future research directions.

II. Preliminary Concepts

This section aims to review the core concepts and ideas of the theory of fuzzy sets and adaptive neural networks that underpin our novel architectural proposal.

A. Fuzzy Logic

The theory of fuzzy sets and fuzzy logic [10] is said to generalize the classical setting as they allow elements to be in various sets having different membership degrees, modeling uncertainties and linguistic terms [18]. A Fuzzy Set (FS) A in a universe of discourse $X \neq \emptyset$ is defined by its Membership Function (MF) as follows.

Definition 2.1: The mapping $\mu_A(x): X \rightarrow [0, 1]$ is the MF denoting the level of membership of the element

x to the subset A , where the FS A is given by:

$$A = \{(x, \mu_A(x)): x \in X\}. \quad (1)$$

Definition 2.2: A mapping $M: [0, 1]^n \rightarrow [0, 1]$ is an n -ary aggregation function (aggregator) if it holds the following properties [19]:

- (A1) If $x_i \leq y_i$ for each $i = 1, \dots, n$, then $M(x_1, \dots, x_n) \leq M(y_1, \dots, y_n)$ (isotonicity);
- (A2) $M(0, \dots, 0) = 0$ and $M(1, \dots, 1) = 1$ (boundary conditions).

The binary aggregation function called fuzzy triangular norm [20], [21], t-norm for short, is defined as follows:

Definition 2.3: A binary function $T: [0, 1]^2 \rightarrow [0, 1]$, such that $\forall x, y, z \in [0, 1]$, is a t-norm if it satisfies: commutativity, associativity, monotonicity, and the boundary condition: $T(x, 1) = x$.

In this work, we consider a diverse set of (parametric and non-parametric) t-norms, as shown in Table I, to analyze the effects of different logical behaviors on the performance of the generalized architecture.

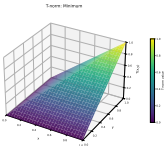
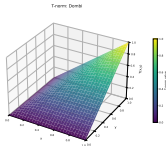
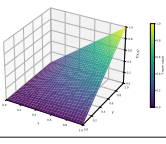
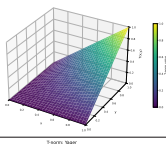
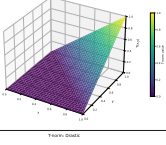
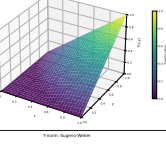
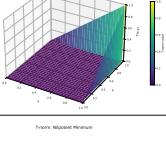
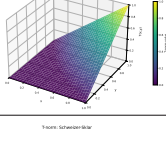
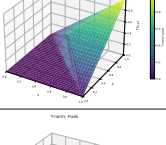
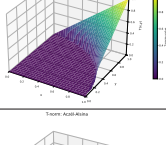
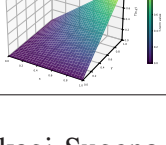
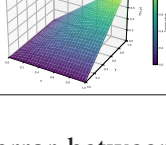
B. Fuzzy Inference System & Fuzzy Control

A Fuzzy Inference System (FIS) represents decision-making processes through linguistic *if-then* rules, enabling the development of interpretable controllers that can serve as alternatives to complex mathematical models [22]. A conventional FIS consists of four main components: a fuzzification mechanism, which transforms crisp input values into corresponding membership degrees; a knowledge base comprising the database and rule base; an inference engine responsible for evaluating and combining the activated rules; and a defuzzification unit, which converts the resulting fuzzy output into a crisp value [23], [24].

Fuzzy rules are expressed in the deductive form “If *antecedent* then *consequent*”, in which the antecedent specifies the conditions under which a rule is activated, while the consequent determines the corresponding inferred output. From a mathematical perspective, the behavior of the controller can be represented by a fuzzy relation $\mathcal{R}$ defined over the input space X and the output space Y , as given in Eq. (2) [25]:

$$\mu_{\mathcal{R}(A,B)}(x, y) = \mu_A(x) \rightarrow \mu_B(y). \quad (2)$$

Table I: Definitions of the Parametric and Non-Parametric T-Norms Used in the Study

Definition	3D graphic	Definition	3D graphic
$T_M(x, y) = \min(x, y)$		$T_\lambda^D(x, y) = \begin{cases} T_D(x, y), & \text{if } \lambda = 0, \\ T_M(x, y), & \text{if } \lambda = \infty, \\ \frac{1}{1 + \left(\left(\frac{1-x}{x} \right)^\lambda + \left(\frac{1-y}{y} \right)^\lambda \right)^{1/\lambda}}, & \text{otherwise.} \end{cases}$	
$T_P(x, y) = x \cdot y$		$T_\lambda^H(x, y) = \begin{cases} T_D(x, y), & \text{if } \lambda = \infty, \\ 0, & \text{if } \lambda = x = y = 0, \\ \frac{xy}{\lambda + (1-\lambda)(x+y-xy)}, & \text{otherwise.} \end{cases}$	
$T_L(x, y) = \max(0, x + y - 1)$		$T_\lambda^Y(x, y) = \begin{cases} T_D(x, y), & \text{if } \lambda = 0, \\ T_M(x, y), & \text{if } \lambda = \infty, \\ \max\left(1 - ((1-x)^\lambda + (1-y)^\lambda)^{1/\lambda}, 0\right), & \text{otherwise.} \end{cases}$	
$T_D(x, y) = \begin{cases} x, & \text{if } y = 1, \\ y, & \text{if } x = 1, \\ 0, & \text{otherwise.} \end{cases}$		$T_\lambda^{SW}(x, y) = \begin{cases} T_D(x, y), & \text{if } \lambda = -1, \\ T_P(x, y), & \text{if } \lambda = \infty, \\ \max\left(\left(\frac{x+y-1+\lambda xy}{1+\lambda}\right), 0\right), & \text{otherwise.} \end{cases}$	
$T_{NM}(x, y) = \begin{cases} \min(x, y), & \text{if } x + y > 1, \\ 0, & \text{otherwise.} \end{cases}$		$T_\lambda^{SS}(x, y) = \begin{cases} T_M(x, y), & \text{if } \lambda = -\infty, \\ T_P(x, y), & \text{if } \lambda = 0, \\ T_D(x, y), & \text{if } \lambda = \infty, \\ \max(x^\lambda + y^\lambda - 1, 0)^{\frac{1}{\lambda}}, & \text{otherwise.} \end{cases}$	
$T_\lambda^F(x, y) = \begin{cases} T_M(x, y), & \text{if } \lambda = 0, \\ T_P(x, y), & \text{if } \lambda = 1, \\ \log_\lambda \left(1 + \frac{(\lambda^x - 1)(\lambda^y - 1)}{\lambda - 1} \right), & \text{otherwise.} \end{cases}$		$T_\lambda^{AA}(x, y) = \begin{cases} T_D(x, y), & \text{if } \lambda = 0, \\ T_M(x, y), & \text{if } \lambda = \infty, \\ e^{-((-\log x)^\lambda + (-\log y)^\lambda)^{\frac{1}{\lambda}}}, & \text{otherwise.} \end{cases}$	

1) *Takagi-Sugeno Controller*: A Takagi-Sugeno controller models the rule consequent as a linear function of the input terms, preventing a computationally exhaustive defuzzification process. In a first-order model, the output of each rule R_i is defined by $f_i(x) = \sum a_{ij}x_j + b_i$, and the overall output F_{TS} is obtained by the weighted average [13], expressed by: $F_{TS}(x) = \frac{\sum_{i=1}^n \mu_{A_i}(x) \cdot f_i(x)}{\sum_{i=1}^n \mu_{A_i}(x)}$, where the weights $\mu_{A_i}(x)$ correspond to the rule activation degree. This structure resembles a local linear regression for each activated rule [26].

Despite being robust, the manual definition of its parameters is complex in large data volumes, a gap that the ANFIS architecture addresses by integrating NN mechanisms for automatic adjustment.

C. Architecture & ANFIS Layers

Neural Networks are adaptive computational models consisting of interconnected layers of neurons, with connection weights that are iteratively updated dur-

ing the learning process to reduce the error between predicted and desired outputs [27], [28]. Within ANFIS, the NN-based architecture serves as a computational representation of the fuzzy inference mechanism, enabling the parameters associated with the antecedent Membership Functions (MFs) and the consequent functions to be learned automatically through supervised optimization [29]. This integration provides ANFIS with the adaptive capabilities of neural models while preserving the transparency and interpretability inherent to fuzzy reasoning.

ANFIS is a neuro-fuzzy architecture that embeds a Takagi-Sugeno FIS into a trainable neural-network structure, making it particularly suitable for approximating nonlinear mappings and modeling dynamic or time-dependent processes [12], [30]. Its original training procedure relies on a hybrid learning strategy. During the forward pass, the Least Squares Estimator (LSE) determines the consequent parameters of the linear functions, whereas the Gradient Descent

Method (GDM) updates the premise parameters associated with the Membership Functions. By alternating between these two optimization mechanisms, ANFIS efficiently tunes both sets of parameters during training [31].

In Fig. 1, the architecture comprises five layers (**L1-L5**), where the output of the i -th node in layer k is denoted by O_i^k :

- **L1 (Fuzzification):** extracts the membership degrees of the entries x in the fuzzy sets A_i : $O_i^1 = \mu_{A_i}(x)$.
- **L2 (Antecedent):** calculates the firing strength (ω_i) of each rule via t-norms (e.g., the product t-norm): $\omega_i = \mu_{A_i}(x) \cdot \mu_{B_i}(y)$.
- **L3 (Normalization):** normalizes the firing strength: $\bar{\omega}_i = \frac{\omega_i}{\sum \omega_j}$.
- **L4 (Consequent):** applies the linear consequent functions: $O_i^4 = \bar{\omega}_i f_i = \bar{\omega}_i(p_i x + q_i y + r_i)$.
- **L5 (Output):** computes the global output like the sum of the contributions: $f_{out} = \sum \bar{\omega}_i f_i$.

The training process seeks to reduce the total squared error $E = \sum (T_i - f_{out_i})^2$, where T_i is the target value, through forward propagation and backpropagation cycles [12].

III. Generalization and Protocol of the Novel Approach

This section presents the proposed approach and the experimental framework adopted to assess its performance. First, we introduce the generalization of the second layer, in which the conventional static product operator is replaced with a flexible selection mechanism. We then describe the experimental setup, covering the optimization of parametric t-norms and the cross-validation strategy employed for model evaluation. By systematically evaluating different t-norm classes across a range of datasets, the study investigates the extent to which alternative logical aggregation mechanisms affect the network's ability to capture nonlinear relationships and accommodate data uncertainty.

A. The proposed Generalization

The novel approach generalizes the original ANFIS architecture of [12] by turning the antecedent gener-

ation (Layer 2) more flexible. In contrast to the traditional ANFIS, which uses the algebraic product (or the product t-norm, T_P), the proposed architecture introduces an Aggregator Selection Module in the second layer of ANFIS. Consequently, the calculation of the original activation strength ω_i , expressed by $\prod_{j=1}^n \mu_{A_{ij}}(x_j)$, is replaced by a generic operator $\mathfrak{B}$ given as follows:

$$\omega_i = \mathfrak{B}(\mu_{A_{i1}}(x_1), \dots, \mu_{A_{in}}(x_n)).$$

The $\mathfrak{B}$ operator may encompass non-parametric t-norms (T_M, T_L, T_D, T_{NM}) and parameterized t-norms ($T_\lambda^F, T_\lambda^{SS}, T_\lambda^D, T_\lambda^H, T_\lambda^{SW}, T_\lambda^Y$). Moreover, $n(MF) = 2$ was fixed by attribute to avoid combinatorial explosion, which yielded $n(R) = 2^{n(A)}$ rules.

B. Configuration and Experimental Protocol

The experimental evaluation followed a structured protocol designed to assess the predictive impact of antecedent generalization across both computational frameworks. Initially, datasets were preprocessed using categorical label encoding and min-max feature normalization. To ensure statistical robustness, all models were trained using a 5-fold cross-validation scheme, with the reported performance (accuracy) corresponding to the arithmetic mean across the test folds. For the parametric t-norm families, the parameter λ was optimized via an exhaustive grid search over each family's domain, retaining the configuration that yielded the highest validation accuracy.

Given the differing computational profiles of the two architectures, distinct training hyperparameters were established:

- **CPU-Based Implementation:** Evaluated on the first 15 datasets. To contain the exponential growth of rules under sequential execution, each attribute was partitioned into $n(MF) = 2$ membership functions, yielding $2^{n(A)}$ rules. The stopping criterion was set to 2 epochs, as preliminary empirical observations showed early convergence with negligible accuracy gains in subsequent iterations.
- **GPU-Based Implementation:** Evaluated across all 21 benchmark datasets. Leveraging batched GPU acceleration and efficient gradient backpropagation via PyTorch, the model capacity was expanded to $n(MF) = 3$ membership functions

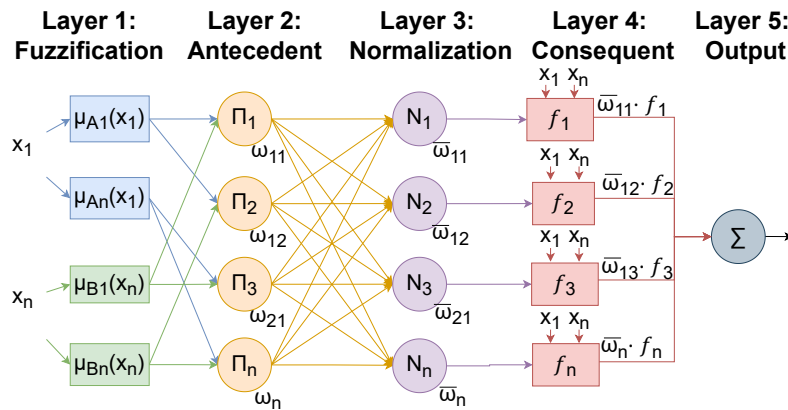


Figure 1: An adapted Sugeno structure of the ANFIS model [12].

per input feature, enabling finer fuzzy granulation of the input domain. Furthermore, models were trained for 100 epochs, allowing end-to-end optimizers to converge effectively on complex decision boundaries without encountering the runtime constraints of the CPU framework.

In both experimental regimes, the original formulation, which relies on the algebraic product t-norm (T_P), served as the baseline for comparison against the 11 proposed t-norm variants.

C. Considered Datasets and Execution Scope

The experimental evaluation was conducted on 21 benchmark datasets obtained from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository [32]. Table II summarizes their main characteristics, including their identifier (**Id**), dataset name (**Dataset name**), number of instances (**Inst.**), input attributes (**Att.**), and target classes (**Cla.**). Because the evaluated ANFIS models require exclusively continuous or numerical inputs, all categorical features were mapped prior to training.

Due to the fundamental computational differences between the evaluated implementations, their experimental execution scopes were defined as follows:

- **CPU-Based Implementation:** Evaluated on the first 15 datasets (Ids 1–15). The remaining 6 datasets (Ids 16–21: *Contraceptive*, *Magic*, *Pageblocks*, *Shuttle*, *Wine*, and *Wisconsin*) could not be executed on this architecture. Since these benchmarks feature higher input dimensionality (ranging from 9 up to 13 attributes), the rule base

grew to prohibitive sizes ($2^9 = 512$ up to $2^{13} = 8192$ fuzzy rules). Under sequential CPU processing and the standard hybrid optimization routines, this combinatorial explosion led to critical memory exhaustion and process termination (operating system *killed* signal) during training.

- **GPU-Based Implementation:** Successfully executed across the complete suite of all 21 datasets (Ids 1–21). Leveraging GPU tensor parallelism, batched inputs, and automatic differentiation via PyTorch, this framework managed the computational demands of larger rule bases, circumventing out-of-memory crashes encountered on the CPU.

D. Computational Environment

The experiments performed on the datasets, seen in Table II, were conducted on a machine equipped with an AMD Ryzen Threadripper PRO 5955WX processor (16 physical cores and 32 threads), 64 GB of RAM, x86-64 architecture supporting 32-bit and 64-bit instructions, and Ubuntu 20.04 LTS OS.

The implementation was written in Python and followed the ANFIS structure from the repository used in [14], publicly available at [15].

Once the execution was complete, the data was processed. The results were collected, organized, and analyzed to compare the aggregator classes of the new approach with the baseline performance (original ANFIS), culminating in tables for consolidating and discussing the experimental findings, which will be presented in the following section.

Table II: Features of the Datasets Used in the Experiments

Id	Dataset name	Inst.	Att.	Cla.
1	Appendicitis	106	7	2
2	Balance	625	4	3
3	Banana	5300	2	2
4	Bupa	345	6	2
5	Ecoli	336	7	8
6	Glass	214	9	6
7	Haberman	306	3	2
8	Hayes-roth	160	4	3
9	Iris	150	4	3
10	Led7digit	500	7	10
11	Newthyroid	215	5	3
12	Phoneme	5404	5	5
13	Pima	768	8	2
14	Titanic	2201	3	2
15	Yeast	1484	8	10
16	Contraceptive	1473	9	3
17	Magic	19020	10	2
18	Pageblocks	5473	10	5
19	Shuttle	58000	9	7
20	Wine	178	13	3
21	Wisconsin	683	9	2

E. Overview of Evaluated ANFIS Structures

To evaluate the impact of antecedent generalization under different computational paradigms, this study considers two distinct implementations:

1) *Traditional CPU-Based Architecture*: The classic implementation follows the formulation by Jang [12], which serves as the basis for the implementation by Meggs [15]. It employs a hybrid learning procedure: during the forward pass, consequent parameters are updated via the Least Squares Estimator (LSE) while antecedent parameters remain fixed; during the backward pass, membership function parameters are refined using the Gradient Descent Method (GDM) [12], [31]. This structure is tied to the algebraic product t-norm (T_P) in Layer 2 and executes sequentially on CPU hardware [12], [15]. Consequently, it presents critical scalability bottlenecks as rule combinations grow exponentially and offers limited support for multi-class classification, which typically requires applying regression thresholds [14].

2) *GPU-Accelerated ANFISpy Framework*: The second approach utilizes the ANFISpy framework, introduced by Monteiro and Wasques [16], which is built natively on PyTorch. This architecture employs GPU-accelerated batched tensor operations and evaluates closed-form generalized t-norm expressions for r inputs. Instead of relying on a decoupled hybrid algorithm, ANFISpy enables gradient-based end-to-end optimization using contemporary optimizers such as Adam and automatic differentiation [16]. Furthermore, the framework extends rule consequents to accommodate m target classes by defining $m \cdot r$ linear functions combined with a Softmax layer and the Cross-Entropy loss function, thereby providing native, scalable multiclass classification [16].

Readers are encouraged to consult [12], [15] and [16] for further details regarding the implementation of each paradigm.

IV. CPU-Based Architecture: Results and Assessments

This section presents the empirical results and a comparative performance analysis of the proposed generalized architecture against the standard ANFIS baseline. The evaluation focuses on classification accuracy across 15 diverse datasets, assessing how integrating 12 t-norm classes affects the model's predictive power.

Table III summarizes these results, organized by dataset (rows in alphabetical order) and fuzzy aggregators (columns), including the baseline product t-norm. The table includes the overall average accuracy per aggregator (M_{Agg}) and the average performance per dataset for the new model (M_{DS}). Values in bold indicate the best performance for each dataset.

The comparative analysis (Table III) shows that the generalized architecture surpassed the baseline in 11 of 15 datasets (73.3%). In 86.6% of cases, the results were equal to or better than the original ANFIS. Notably, in the *Led7digit* dataset (Id 10), the new approaches achieved an average accuracy of 0.1670, a significant improvement over the 0.1360 achieved by the standard model.

The results indicate that the proposed variants exhibit superior consistency, with T_{λ}^{SW} and T_{λ}^F emerging as the most effective aggregators. Performance gains are particularly evident in datasets such as *Led7digit*, while in higher-dimensional datasets (e.g., *Ecoli* and *Glass*), flexible operators capture more complex relationships. This suggests that parametric t-norms improve adaptability to nonlinear dependencies.

A. Execution time

Table IV presents the runtime performance across the 15 evaluated datasets. The results indicate that the computational cost of the proposed ANFIS generalization is primarily driven by the dimensionality of the input space rather than the number of instances. Since the number of fuzzy rules in the ANFIS architecture grows exponentially with the number of attributes (2^n for two membership functions per input), datasets with high dimensionality, such as *Yeast* ($n = 8$) and *Glass* ($n = 9$), exhibited the highest execution times, reaching up to 73.37h and 41.35h, respectively. The impact of the rule base size becomes

evident when comparing the *Banana* (Id 3) and *Pima* (Id 13) datasets. Although *Banana* contains significantly more instances (5300 vs. 768), its low dimensionality ($n = 2$) results in only 4 rules, allowing execution to complete in 1.12h. In contrast, *Pima* ($n = 8$, 256 rules) requires over 32h. This confirms that the antecedent layer complexity is the main bottleneck for the A2-ANFIS model. Furthermore, the variability in execution time across different t-norms was found to be marginal, ranging from 1% to 16%. This suggests that while the choice of fuzzy aggregator significantly impacts predictive accuracy, the computational overhead of switching between different t-norms (e.g., Minimum T_M vs. Dombi T_{λ}^D) is negligible within the overall training process.

B. Statistical Analysis of Results

To validate the proposed extensions, a statistical analysis was performed following Demšar's [33] recommendations. The **Friedman test** yielded a p-value of 0.041, rejecting the null hypothesis and confirming that varying the t-norms statistically alters the system's performance.

The **Nemenyi post-hoc test** identified the best average ranks for T_{λ}^{SW} (4.7), T_L (5.4), and T_{λ}^{SS} (5.5), all outperforming the **standard ANFIS** (7.4). Despite the conservative nature of the Nemenyi test for $k = 12$ algorithms, a clear trend of performance gain was observed with the proposed operators.

For a sensitive pairwise comparison (control vs. treatment), the **Wilcoxon signed-rank test** confirmed that T_{λ}^{SW} ($p = 0.0251$) and T_L ($p = 0.0425$) significantly outperform the original ANFIS ($p < 0.05$). These results indicate that replacing the algebraic product with flexible t-norms yields a genuine optimization of inference capacity, with T_{λ}^{SW} standing out as the most robust choice.

V. GPU-Based Architecture: Results and Assessments

This section presents the experimental results and comparative performance analysis of the generalized antecedent layer implemented within the GPU-accelerated ANFISpy framework [16]. Unlike the CPU baseline, which was computationally bounded to 15 datasets, this environment evaluates the full bench-

Table III: Accuracy Results of the T-Norm Classes Across 15 Datasets Using the CPU-Based Implementation

Id	ANFIS (T_P)	T_λ^D	T_λ^F	T_λ^H	T_λ^{SS}	T_λ^{SW}	T_λ^Y	T_λ^{AA}	T_M	T_L	T_D	T_{NM}	M_DS
1	0.8775	0.8684	0.8775	0.8775	0.9152	0.8957	0.8957	0.8775	0.8775	0.8957	0.8684	0.8775	0.8842
2	0.6864	0.6864	0.6864	0.6864	0.6864	0.6928	0.6864	0.6864	0.6864	0.6864	0.6864	0.6864	0.6870
3	0.5674	0.5672	0.5674	0.5674	0.5674	0.5617	0.5666	0.5666	0.5666	0.5613	0.5632	0.5613	0.5652
4	0.7101	0.7101	0.7101	0.7101	0.7420	0.7101	0.7101	0.7101	0.7101	0.7101	0.7101	0.7101	0.7130
5	0.2620	0.2709	0.2709	0.2709	0.2710	0.3215	0.2710	0.2709	0.2709	0.2710	0.2650	0.2650	0.2745
6	0.4577	0.4670	0.4577	0.4577	0.4577	0.4720	0.4670	0.4577	0.4577	0.4672	0.4670	0.4577	0.4624
7	0.7451	0.7451	0.7451	0.7451	0.7614	0.7451	0.7451	0.7451	0.7451	0.7451	0.7451	0.7451	0.7466
8	0.7496	0.7496	0.7496	0.7496	0.7496	0.7496	0.7496	0.7696	0.7496	0.7496	0.7496	0.7496	0.7514
9	0.9733	0.9733	0.9733	0.9733	0.9733	0.9733	0.9733	0.9733	0.9733	0.9733	0.9733	0.9733	0.9733
10	0.1360	0.2260	0.2300	0.2260	0.1360	0.1360	0.1360	0.1960	0.2300	0.1360	0.1100	0.1040	0.1696
11	0.8140	0.8140	0.8140	0.8140	0.8140	0.8140	0.8140	0.8140	0.8140	0.8140	0.8140	0.8140	0.8140
12	0.7515	0.7519	0.7517	0.7518	0.7515	0.7519	0.7519	0.7517	0.7517	0.7519	0.7519	0.7519	0.7518
13	0.7799	0.7786	0.7812	0.7799	0.7799	0.7877	0.7786	0.7812	0.7812	0.7877	0.7786	0.7812	0.7814
14	0.7760	0.7760	0.7760	0.7760	0.7760	0.7760	0.7760	0.7760	0.7760	0.7760	0.6820	0.7760	0.7675
15	0.3235	0.3228	0.3221	0.3241	0.3235	0.3302	0.3228	0.3221	0.3221	0.3302	0.3228	0.3235	0.3242
M_{Agg}	0.6407	0.6472	0.6475	0.6473	0.6470	0.6478	0.6429	0.6465	0.6475	0.6437	0.6325	0.6384	

Table IV: Runtime Analysis by Dataset Under the CPU-Based Implementation

Id	Rules	Inst.	Avg(h)	Max(h)
1	2^7	106	3.16	4.47
2	2^4	625	0.59	0.59
3	2^2	5300	1.12	1.12
4	2^6	345	1.92	1.92
5	2^7	336	3.45	3.45
6	2^9	214	41.35	45.81
7	2^3	306	1.19	1.19
8	2^4	160	1.29	1.29
9	2^4	150	1.46	1.46
10	2^7	500	4.89	4.89
11	2^5	215	1.80	1.80
12	2^5	5404	14.58	16.89
13	2^8	768	32.83	46.51
14	2^3	2201	6.35	6.35
15	2^8	1484	73.37	74.45

mark of 21 datasets under a higher model capacity ($n(MF) = 3$) and 100 training epochs across the 12 t-norm classes.

Table V summarizes the cross-validation classification accuracies obtained across all 21 datasets, reporting the baseline ANFIS (T_P), the 11 alternative aggregators, the aggregator average (M_{Agg}), and the average performance per dataset for the proposed mod-

els (M_{DS}). Bold entries denote the highest accuracy achieved for each dataset.

The empirical data reveal that the proposed generalized architecture achieved superior or equivalent accuracy compared to the standard ANFIS baseline (T_P) across 16 out of the 21 datasets (76.19% of the test suite). Specifically:

Table V: Results of the Classes of T-Norms Applied to the Datasets in ANFISpy

Id	ANFIS (T_P)	T_λ^D	T_λ^F	T_λ^H	T_λ^{SS}	T_λ^{SW}	T_λ^Y	T_λ^{AA}	T_M	T_L	T_D	T_{NM}	M_DS
1	0.8208	0.8299	0.8303	0.8208	0.8485	0.8866	0.8580	0.8199	0.8022	0.8494	0.8017	0.7641	0.8283
2	0.1120	0.1168	0.2992	0.1168	0.0848	0.0912	0.0832	0.2880	0.1200	0.0832	0.2688	0.0784	0.1482
3	0.7989	0.8320	0.7989	0.8148	0.7678	0.7028	0.7631	0.8342	0.7963	0.6600	0.5516	0.6489	0.7428
4	0.4186	0.4186	0.4186	0.4186	0.4186	0.4186	0.4186	0.4186	0.4186	0.4186	0.4186	0.4186	0.4186
5	0.4256	0.4256	0.4256	0.4256	0.4256	0.4256	0.4256	0.4256	0.4256	0.4256	0.4256	0.4256	0.4256
6	0.3271	0.3271	0.3271	0.3271	0.3271	0.3271	0.3271	0.3271	0.3271	0.3271	0.3271	0.3271	0.3271
7	0.7344	0.7344	0.7344	0.7344	0.7344	0.7344	0.7344	0.7344	0.7344	0.7344	0.7344	0.7344	0.7344
8	0.5887	0.6053	0.5887	0.6217	0.4013	0.3770	0.3770	0.7057	0.5730	0.3770	0.4343	0.3770	0.4944
9	0.3241	0.3241	0.3241	0.3241	0.3241	0.3241	0.3241	0.3241	0.5954	0.3241	0.3375	0.3241	0.3500
10	0.5266	0.5487	0.5600	0.5488	0.5189	0.5301	0.5264	0.5413	0.5152	0.5191	0.4741	0.4926	0.5250
11	0.6977	0.6977	0.6977	0.6977	0.6977	0.6977	0.6977	0.6977	0.6977	0.6977	0.6977	0.6977	0.6977
12	0.7809	0.7685	0.7809	0.7865	0.7501	0.7396	0.7585	0.7705	0.7483	0.7203	0.7049	0.7299	0.7507
13	0.6510	0.6510	0.6510	0.6510	0.6510	0.6510	0.6510	0.6510	0.6510	0.6510	0.6510	0.6510	0.6510
14	0.4411	0.4051	0.4411	0.4415	0.4233	0.4051	0.4233	0.5146	0.3957	0.4051	0.4233	0.4051	0.4257
15	0.5189	0.4899	0.5352	0.5297	0.5284	0.3614	0.3099	0.4919	0.4919	0.3262	0.4851	0.3870	0.4488
16	0.4610	0.4774	0.4658	0.4692	0.4555	0.4227	0.4275	0.4337	0.4747	0.4220	0.4918	0.4528	0.4539
17	0.7702	0.7713	0.7697	0.7702	0.7839	0.6341	0.6488	0.6488	0.7745	0.6293	0.6477	0.7524	0.7119
18	0.9070	0.9398	0.9216	0.9124	0.9160	0.8959	0.8978	0.9033	0.9197	0.8978	0.9398	0.8960	0.9127
19	0.9655	0.9085	0.9834	0.9664	0.9747	0.8262	0.7844	0.7844	0.9683	0.8221	0.7899	0.9370	0.8859
20	0.9775	0.9495	0.9716	0.9717	0.9832	0.3314	0.3314	0.3314	0.9379	0.3314	0.3314	0.8483	0.6654
21	0.9649	0.9686	0.9575	0.9538	0.9593	0.6895	0.5619	0.5730	0.9630	0.7024	0.9261	0.8742	0.8299
M_{Agg}	0.6292	0.6281	0.6420	0.6335	0.6178	0.5463	0.5395	0.5819	0.6348	0.5392	0.5649	0.5820	0.5918

- **Peak Performance Gains:** The alternative t-norms strictly outperformed the standard model in 11 datasets (Ids 1, 2, 3, 8, 9, 10, 12, 15, 16, 17, 18, 19, 20, and 21). Notable performance improvements occurred in *Balance* (Id 2), where T_λ^F elevated accuracy from 0.1120 to 0.2992 (a 167% gain), and in *Iris* (Id 9), where the Minimum t-norm (T_M) reached 0.5954 versus 0.3241 for the baseline.
- **High-Dimensional Scalability:** In the 6 datasets that could not be executed on the CPU baseline (Ids 16–21, with up to 13 attributes), the generalized operators systematically identified better antecedent representations. For instance, T_λ^{SS} delivered the highest accuracies in *Magic* (0.7839) and *Wine* (0.9832), while T_λ^F achieved 0.9834 in *Shuttle*.
- **Aggregator Efficacy:** Across the full benchmark, the Frank family (T_λ^F) demonstrated the highest overall consistency, attaining the top average accuracy of $M_{Agg} = 0.6420$, followed closely by the Minimum operator ($T_M = 0.6348$) and Hamacher ($T_\lambda^H = 0.6335$), all outperforming the classic

product baseline (0.6292).

A. Execution Time

Table VI reports the runtime performance of ANFISpy across the 21 evaluated datasets. The empirical results demonstrate that input dimensionality (n) is the primary computational bottleneck, due to the exponential growth inherent in grid partitioning. Given $s = 3$ membership functions per attribute, the antecedent rule base scales as 3^n . Furthermore, for classification tasks with m classes, ANFISpy evaluates $m \cdot 3^n$ linear consequent functions (one dedicated equation per class for each rule), proportionally expanding the output tensor dimensions optimized during backpropagation.

This structural complexity directly dictates training runtime. The most demanding case was Wine (ID 20, $n = 13$), where $3^{13} = 1,594,323$ rules required an average runtime of 06h54m10s (peaking at 13h35m23s), followed by benchmarks with $n = 10$ and $n = 9$ (IDs 17, 18, and 19).

Conversely, instance volume exerts a secondary im-

Table VI: Runtime Analysis by Dataset in ANFISpy

Id	Rules (3^n)	Inst.	Avg (hh:mm:ss)	Max (hh:mm:ss)
1	3^7	106	00:00:17	00:00:26
2	3^4	625	00:00:21	00:00:25
3	3^2	5300	00:01:24	00:01:28
4	3^6	345	00:00:24	00:00:37
5	3^7	336	00:01:05	00:01:38
6	3^9	214	00:03:36	00:05:11
7	3^3	306	00:00:09	00:00:10
8	3^4	160	00:00:04	00:00:04
9	3^4	150	00:00:04	00:00:05
10	3^7	500	00:00:38	00:00:55
11	3^5	215	00:00:17	00:00:22
12	3^5	5404	00:03:41	00:03:57
13	3^8	768	00:02:31	00:04:01
14	3^3	2201	00:00:03	00:00:03
15	3^8	1484	00:13:35	00:21:30
16	3^9	1473	00:10:39	00:17:13
17	3^{10}	19020	01:16:38	02:05:55
18	3^{10}	5473	01:29:27	02:21:28
19	3^9	58000	00:39:48	01:02:15
20	3^{13}	178	06:54:10	13:35:23
21	3^9	683	00:03:05	00:04:44

Note: 3^n denotes the number of antecedent rules for n attributes (3 MFs per variable). In classification tasks with m classes, ANFISpy generates $m \cdot 3^n$ consequent outputs.

pact on training time, thanks to batched tensor execution and native GPU acceleration in PyTorch. This behavior is highlighted by comparing Shuttle (ID 19) and Wine (ID 20): despite containing 58,000 instances versus 178, Shuttle's lower dimensionality ($n = 9$, $3^9 = 19,683$ rules) resulted in an average completion time of 39m48s. Across small-to-moderate rule bases (3^2 to 3^7 , IDs 1–15), ANFISpy scaled efficiently, finishing within a few seconds to a few minutes.

B. Statistical Analysis of Results

To rigorously evaluate the experimental findings across the 21 benchmark datasets, non-parametric statistical tests were conducted in accordance with the standard methodology proposed by Demšar [33].

The **Friedman test** was first executed to determine whether the choice of t-norm yields statistically significant discrepancies in classification performance. The test yielded a statistic of $\chi^2_F = 62.1873$ with an

associated p -value of 3.6284×10^{-9} ($p \ll 0.05$). This definitively rejects the null hypothesis of equivalent performance across all candidates, verifying that antecedent aggregator selection significantly influences ANFIS learning capability.

Regarding mean algorithmic rankings (where lower values denote superior accuracy), the Frank parametric family (T_λ^F) attained the top position with an average rank of 4.4524, followed by Hamacher (T_λ^H , 4.7143) and Dombi (T_λ^D , 5.0952), all outranking the canonical algebraic product baseline (T_P , 5.3571). Conversely, nilpotent and drastic operations displayed substantial degradation, with the Łukasiewicz t-norm (T_L) recording the lowest rank (8.6429). The post-hoc **Nemenyi test** established a Critical Distance of $CD = 3.6363$ ($\alpha = 0.05$, $k = 12$, $N = 21$). While the conservative nature of the all-pairwise Nemenyi test did not detect isolated statistical distinctions against the control model under the critical threshold, the monotonic advantage of parametric formulations re-

mains evident.

To provide a more sensitive pairwise assessment between the control baseline (T_P) and each proposed aggregator, the **Wilcoxon signed-rank test** was conducted. The analysis confirmed that the Frank aggregator (T_λ^F) achieves a statistically significant improvement over the standard product operator ($W = 8.0$, $p = 0.0469 < 0.05$). Additionally, the Hamacher operator (T_λ^H) showed a marked positive trend bordering significance ($W = 15.0$, $p = 0.0597$). In contrast, restrictive t-norms including T_L ($p = 0.0019$), T_{NM} ($p = 0.0010$), T_{SW} ($p = 0.0052$), and T_D ($p = 0.0309$) exhibited statistically significant drops in accuracy. These results confirm that smooth, parameter-tunable t-norms provide superior numerical stability and adaptive capacity in multi-class GPU-based neuro-fuzzy inference.

VI. Comparative Analysis: CPU-Based vs. GPU-Accelerated ANFIS

The empirical evaluations highlight fundamental architectural trade-offs between the CPU-based ANFIS and the GPU-accelerated ANFISpy framework. In low-dimensional settings ($n \leq 4$), the traditional CPU implementation achieves rapid analytical convergence within very few epochs by identifying linear consequents via the Least Squares Estimator (LSE). However, its sequential execution and reliance on ad hoc regression thresholds create severe memory-exhaustion bottlenecks when scaling to higher feature dimensions, rendering rule bases with more than 2^8 rules intractable.

Conversely, ANFISpy leverages PyTorch tensor parallelism to evaluate massive rule bases (up to 3^{13} rules) without memory crashes, replacing decoupled hybrid updates with smooth end-to-end backpropagation via modern gradient optimizers. Furthermore, its native multi-class formulation provides well-calibrated class probability distributions by allocating m linear consequent equations per rule. Nevertheless, this parameter expansion creates a structural trade-off: on small, moderate-dimensional datasets, allocating $m \cdot r$ consequent parameters risks over-parameterization and numerical instability in sparse input regions, whereas continuous parametric t-norm families (e.g., Frank and Hamacher) significantly improve gradient flow and overall predictive stability across large-scale

benchmarks.

VII. Conclusions

This work investigated the structural generalization of the Adaptive Neuro-Fuzzy Inference System by providing more flexibility to its second layer (antecedent rule aggregation). Moving beyond the rigid, canonical algebraic product t-norm (T_P), we systematically evaluated parametric and non-parametric t-norm families under two contrasting computational paradigms: the classical CPU-based architecture and the GPU-accelerated ANFISpy framework. In the traditional CPU baseline, 11 alternative aggregators were integrated into the standard hybrid learning scheme. In contrast, the GPU-based ANFISpy framework, which originally provided only 5 native aggregators, was significantly extended in this work to support 12 diverse t-norm classes, fully leveraging tensor parallelization, native multiclass cross-entropy loss, and automatic differentiation.

The empirical findings from both experimental stages demonstrate that the selection of antecedent operators directly affects classification accuracy and adaptive modeling capacity. In the CPU framework, the Sugeno-Weber (T_λ^{SW}) and Łukasiewicz (T_L) operators proved most effective, outperforming the baseline across 73.3% of the 15 executable datasets. In the GPU environment, evaluated across the complete suite of 21 datasets with expanded rule capacities (3^n), smooth parametric families, specifically Frank (T_λ^F) and Hamacher (T_λ^H), demonstrated superior numerical stability and gradient propagation. Non-parametric statistical evaluations (Friedman and Wilcoxon signed-rank tests) confirmed that replacing the fixed algebraic product operator with parametric aggregators such as T_λ^F yields statistically significant accuracy gains over the standard formulation ($p < 0.05$).

The architectural comparison revealed distinct operational profiles for each paradigm. While the CPU implementation offers rapid convergence on low-dimensional spaces ($n \leq 4$) via Least Squares Estimation, it hits an insurmountable scalability wall on high-dimensional datasets ($n \geq 9$) due to memory exhaustion and sequential processing constraints. Conversely, ANFISpy eliminates out-of-memory terminations and successfully executes rule bases ex-

ceeding 1.5 million rules (e.g., in *Wine*). However, its multi-class parameter formulation ($m \cdot 3^n$ linear consequents) demands careful regularization in sample-scarce settings to avoid over-parameterization, while highlighting that continuous, parameter-tunable t-norms are essential for maintaining stable backpropagation.

Future research will focus on two practical avenues directly aimed at mitigating the curse of dimensionality inherent to grid partitioning. First, we plan to integrate feature selection techniques (such as tree-based importance and mutual information filters) prior to fuzzification, effectively reducing the input dimension (n) on dense datasets like *Wine* and enabling smaller rule bases without sacrificing informative features. Second, we intend to explore data-driven clustering methods, such as Fuzzy C-Means, to define prototype-based rules instead of Cartesian grid partitions, constraining the rule set to a fixed number of clusters and eliminating the 3^n combinatorial growth during GPU backpropagation.

References

1. PN Tan, M Steinbach, V Kumar (2005) *Introduction to Data Mining*, (First Edition). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
2. P Mitra, HH Sidhwa (2025) "Collaborative filtering for personalized movie recommendations on streaming platforms," in 2025 Intl. Conf on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI) 1-6.
3. Q Khaled, U Kaymak, L Genga (2025) "Interpretable fuzzy systems for forward osmosis desalination," in 2025 IEEE Intl. Conf on Fuzzy Systems (FUZZ) 1-7.
4. TMH Gedara, V Loia, S Tomasiello (2024) "Detecting fake images using neuro-fuzzy inference systems: A brief comparative analysis," in 2024 IEEE Intl. Conf. on Fuzzy Systems (FUZZ-IEEE) 1-7.
5. A Karki, C Hardikar, MA Kumar, S Das (2023) "Improved ANFIS Model Based on Confinement Optimized PSO to Predict the Air Quality of Delhi, India," in 2023 IEEE Intl. Conf on Fuzzy Systems (FUZZ) 1-6.
6. S Russell, P Norvig, A Intelligence (1995) "A modern approach," *Artificial Intelligence* 25: 79-80.
7. AAM Lêu, SF Amorim, AFL Amorim (2024) "Aplicação de redes neurais artificiais para predição de deslizamentos de terra induzidos por chuvas," *Revista Políticas Públicas & Cidades* 13: e1344-e1344.
8. LCA Da Silva, JR Monteiro (2023) "Sistema para apoio ao diagnóstico de câncer de mama baseado em redes neurais artificiais," in *Engenharias-automatização, robótica, metrologia e energia: estudos e tendências - volume 1*, vol. 1. Editora Científica Digital 49-65.
9. GV Moura, et al. (2025) "Previsão do preço de ações da petrobras utilizando redes neurais artificiais," Master's thesis, Universidade Tecnológica Federal do Paraná.
10. LA Zadeh (1965) "Fuzzy sets," *Information and Control* 8: 338-353.
11. HH Tang, NS Ahmad (2024) "Fuzzy logic approach for controlling uncertain and nonlinear systems: a comprehensive review of applications and advances," *Systems Science & Control Engineering* 12: 2394429.
12. JS Jang (1993) "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man, and Cybernetics* 23: 665-685.
13. T Takagi, M Sugeno (1985) "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics* 15: 116-132.
14. SGLB Serri (2022) "Implementação de Controlador Neuro-Fuzzy e Sistema Supervisório para Planta Didática," Final Term Paper (B.S. in Electrical Engineering) 55.
15. T Meggs (2020) "Anfis," <https://github.com/twm/eggs/anfis>, last accessed on May 16, 2025.
16. MZ Monteiro, VF Wasques (2026) "Anfispy: a python package for neuro-fuzzy models," *Neural Computing and Applications* 38: 59.
17. B Novack, J Buss, G Silva, R Reiser, H Santos, G Lucca (2026) "Generalizing the antecedent layer of anfis: A performance analysis under classes of fuzzy aggregators," in 2026 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) 1-

- 6.
18. LA Zadeh (1975) "The concept of a linguistic variable and its application to approximate reasoning - I," Information Sciences 8: 199-249.
19. G Beliakov, A Pradera, T Calvo, et al. (2007) Aggregation functions: A guide for practitioners. Springer 221.
20. GJ Klir, B Yuan (1995) Fuzzy sets and fuzzy logic - theory and applications. Prentice Hall.
21. EP Klement, R Mesiar, E Pap (2000) Triangular Norms. Dordrecht, Netherland: Springer.
22. N Sabri, S Aljunid, M Salim, R Badlishah, R Kamaruddin, M Malek (2013) "Fuzzy inference system: Short review and design," Int. Rev. Autom. Control 6: 441-449.
23. RR Yager, LA Zadeh (2012) An introduction to fuzzy logic applications in intelligent systems. Springer Science & Business Media 165.
24. LA Zadeh (1973) "Outline of a new approach to the analysis of complex systems and decision processes," IEEE Transactions on systems, Man, and Cybernetics no. 1: 28-44.
25. TJ Ross (2010) Classical Relations and Fuzzy Relations. USA: John Wiley and Sons, Ltd.
26. G James, D Witten, T Hastie, R Tibshirani, J Taylor (2023) "Statistical learning," in An introduction to statistical learning: With applications in Python. Springer 15-67.
27. SC Wang (2003) Artificial Neural Network. Boston, MA: Springer US 81-100.
28. A Dongare, R Kharde, AD Kachare, et al. (2012) "Introduction to artificial neural network," Intl. Journal of Engineering and Innovative Technology (IJEIT) 2: 189-194.
29. K Mehrotra, CK Mohan, S Ranka (1997) Elements of artificial neural networks. MIT press.
30. P Melin, J Soto, O Castillo, J Soria (2012) "A new approach for time series prediction using ensembles of ANFIS models," Expert Systems with Applications 39: 3494-3506.
31. D Karaboga, E Kaya (2019) "Adaptive network based fuzzy inference system (ANFIS) training approaches: a comprehensive survey," Artificial Intelligence Review 52: 2263-2293.
32. J Derrac, S Garcia, L Sanchez, F Herrera (2015) "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," J. Mult. Valued Logic Soft Comput 17: 255-287.
33. J Demšar (2006) "Statistical comparisons of classifiers over multiple data sets," Journal of Machine learning research 7: 1-30.