Applied Intelligence (2025) 55:948
https://doi.org/10.1007/s10489-025-06795-6

Neural-FCM: a deep learning approach for weight matrix optimization in Fuzzy Cognitive Map classifiers

Theodoros Tziolas[1] · Konstantinos Papageorgiou[1] · Ioannis Apostolopoulos[2] · Elpiniki Papageorgiou[1]

Accepted: 12 July 2025

Abstract
The demand for interpretable and accurate machine learning models continues to grow, especially in critical domains. The data-driven Fuzzy Cognitive Map (FCM) classifier is an interpretable and transparent decision-making method. Its core element, the weight matrix, is derived using predominantly population-based supervised learning methods which often suffer from degraded performance. Recent research has adopted gradient-based learning techniques to compete with the predictive performance of black-box models. Nonetheless, such methods modify foundational principles and compromise interpretability, highlighting the necessity to improve existing approaches. In this work, we introduce a novel learning and structural modeling method, termed Neural-FCM, which leverages deep neural networks and gradient descent to enhance the accuracy and robustness of FCM learning. Neural-FCM employs a hybrid network comprising both dense and convolutional layers and is trained using a categorical cross-entropy loss function specifically aligned with FCM reasoning. This hybrid model is trained to output instance-specific weight matrices for effective and targeted FCM inference, introducing structural adaptability, a feature not supported by previous static or globally optimized approaches. Focusing on generalization across domains, the Neural-FCM approach is evaluated on different classification tasks across six widely used public datasets and one proprietary medical dataset, consistently showing improved predictive performance. Notably, the comparative analysis against standard population-based FCM learning methods reveals consistent accuracy improvements, with gains of up to 34%. While less transparent gradient-based methods also yield improved accuracy, Neural-FCM demonstrates competitive or superior performance in most cases, with accuracy improvements ranging from 1 to 6% across different domains, while preserving the underlying interpretability. The performance enhancement and the use of instance-specific matrices contribute to the broader goal of developing gradient-based models that balance computational efficiency with the intrinsic FCM interpretability.

Keywords Fuzzy Cognitive Maps · Deep learning · Classification · Pattern recognition

1 Introduction

Fuzzy Cognitive Maps (FCMs) have been extensively utilized for modeling complex systems since their inception by Kosko [1–3]. Their "white-box" structure offers an intuitive approach to knowledge representation and inference, integrating aspects from domains such as fuzzy logic, artificial neural networks (ANNs) and expert systems [4]. Due to their inherent transparency, FCMs provide a compelling framework for building trustworthy intelligent systems [5]. This attribute is extremely valuable in Artificial Intelligence (AI) applications that demand interpretability and accountability, especially in domains where trust and understanding are critical [6]. FCM inference critically relies on the accuracy of the weight matrix, making the precise determination

✉ Theodoros Tziolas
ttziolas@uth.gr

Konstantinos Papageorgiou
konpapageorgiou@uth.gr

Ioannis Apostolopoulos
japostol@iceht.forth.gr

Elpiniki Papageorgiou
elpinikipapageorgiou@uth.gr

[1] Department of Energy Systems, University of Thessaly, Gaiopolis Campus, Larisa 41500, Greece

[2] Institute of Chemical Engineering Sciences (ICE-HT), Foundation for Research and Technology Hellas (FORTH), Patras 26504, Greece

Published online: 30 August 2025

of weights essential for reliable simulations and predictions. Over recent decades, research has progressively transitioned FCMs from purely expert-driven models [7–9] to hybrid and fully data-driven approaches, leveraging machine learning (ML) methods for weight determination [10, 11].

In FCM learning, both unsupervised and supervised paradigms have been developed for defining or adjusting the weight matrix. Among unsupervised methods, Hebbian-based techniques refine an expert-defined matrix, by adapting it in response to system dynamics [12–15]. A key challenge with these methods lies in their dependence on expert-defined weight matrices, which may not always be available or sufficiently accurate. This limitation is tackled with supervised methods, also called error-based methods. The aim is to extract knowledge directly from data for determining the weight matrix, by optimizing a function that minimizes the discrepancy between the system's response and the expected output. In the literature, this is mainly achieved through evolutionary computation and swarm intelligence algorithms inspired by the principles of natural evolution [16]. These algorithms search for an optimal weight matrix by iteratively adapting a pool of potential solutions to reduce classification error, as seen in techniques like the Real-Coded Genetic Algorithm (RCGA) [17], the Structure Optimization Genetic Algorithm (SOGA) [18], the Particle Swarm Optimization (PSO) [19], and optimization methods based on artificial colonies [20, 21]. Despite their effectiveness, these methods are often computationally intensive, slow, and susceptible to convergence on suboptimal solutions [11]. Additionally, they struggle with multiclass classification tasks, especially when the model structure represents each class as an individual concept and input concepts correspond to low-level features, as described in [22]. In response, novel error-based solutions have been proposed that draw aspects from ANNs and utilize gradient-based methods for FCM learning [23–25]. While these gradient-based approaches show promise in classification and pattern recognition tasks, their emphasis on predictive performance and their adaptation to the deep model scheme often compromise FCM's foundational interpretability.

Although low-level FCM classifiers have been relatively underexplored, particularly due to the performance limitations of early learning methods [3], they could offer key advantages due to their inherent transparency [5]. Beyond achieving high accuracy, the ability to interpret the reasoning behind decisions enhances the acceptance and trustworthiness of a classification method, a key principle in the field of AI [6]. Hence, FCMs require a good trade-off between predictive accuracy and interpretability to be considered beneficial against black-box models, which typically outperform them across a range of pattern recognition tasks [3, 26]. In FCMs, interpretability refers to the ability to analyze the weight interconnections with the aid of domain knowledge in order to observe how each node influences the classification decision [5, 27]. In other words, there is a traceable cause and effect structure inherent in the model. Some early FCM-based classifiers enhanced prediction performance by incorporating black-box models within the FCM structure [28]. However, such an approach completely alters the FCM structure and thus diminishes any interpretability advantages. Even without directly integrating a black-box model, the use of bias nodes, non-standard weights, or modified inference rules, as seen in previous gradient-based methods [25, 29], can reduce interpretability at the functional level for which FCMs are renowned [5]. These challenges highlight a research gap in designing low-level FCM classifiers that effectively leverage ANN principles while preserving transparency and interpretability.

Compared to FCMs, ANNs have received far greater attention from the research community and have evolved into powerful tools capable of learning from vast datasets, making them a cornerstone of modern AI and Deep Learning (DL), despite their opaque internal mechanics. Deep models are engineered to approximate complex non-linear functions [30] and architectures with multiple processing layers and millions of interconnections have achieved remarkable success across a variety of applications [26]. Compared to the typical low-level FCMs where input nodes are fixed, have a more physical meaning, and employ static weighted interconnections [1], deep models offer enhanced modeling capacity. In fact, the rigidity of FCM's two-dimensional weight matrix has become a performance bottleneck, prompting research into more flexible representations, such as grey-valued interconnections [31]. In addition to modeling capacity, the rapid development of DL has been supported by the availability of open-source frameworks like TensorFlow [32] which simplify low-level tensor computations and accelerate prototyping. Additionally, the considerable availability of open-sourced algorithms and models is another advantage. In contrast, FCM learning methods often lack accessible software implementations, hindering progress in the field, with only a few efforts aimed at developing public Python packages [33]. Given these differences, it is reasonable to consider that FCMs could benefit from adopting certain aspects of ANN-based modeling, particularly in terms of data-driven learning, computational tools, and architectural flexibility, while still retaining their core interpretability.

In summary, although data-driven FCM classifiers can achieve competitive performance by adopting techniques from deep models, they often lose the interpretability that gives them their advantage. This trade-off has contributed to their being overlooked in favor of more powerful yet opaque methods. Nevertheless, FCMs have the potential to evolve by incorporating successful concepts, techniques and strategies from DL. Further research is required to effectively reconcile the dual goals of accuracy and interpretability, both of which are essential for building trustworthy models within the domain of AI [6]. Furthermore, the modification of their structure to tackle performance impediments, such as static weights, is another promising research area that requires methodical design to ensure the preservation of the underlying principles. Finally, adopting open-source practices with code and implementation details could significantly benefit the field, as observed in the DL community.

Considering the above, the objective of this work is to present and assess a novel weight matrix optimization method, that enhances FCM learning and structural modeling, named Neural-FCM. The proposed method integrates elements from deep models to improve accuracy while preserving interpretability. Technically, it employs a deep network consisting of dense and convolutional layers to generate a two-dimensional numeric matrix for each input instance. The network is trained using a modified Categorical Cross-Entropy (CCE) cost function, where FCM reasoning is performed prior to loss computation. As a result, Neural-FCM learns to dynamically adjust weight matrices based on instance-specific features, enhancing both predictive performance and structural flexibility. The instance-specific structure adaptation can be seen as an amplification of Fuzzy Grey Cognitive Maps (FGCM) [31]. The method is evaluated on multiple benchmark datasets to assess its generalization capabilities across domains. Results show that Neural-FCM not only improves predictive accuracy but also retains and even enhances the low-level interpretability that initially distinguished FCMs within the AI community. Therefore, it holds strong potential as a trustworthy, data-driven decision-making framework. To promote adoption and reproducibility, the method is implemented using open-source tools (TensorFlow and Python), with accompanying code provided.

To summarize, the main contribution of this work is the introduction and performance assessment of a novel learning and structure modeling method that addresses key limitations in the current state-of-the-art:

1. **Comprehensive performance and generalization assessment**: Neural-FCM demonstrates strong performance across six public datasets and one proprietary health-sector dataset. The proposed method achieves up to 34% improvement over population-based FCM learning methods with a node-per-class representation. Additionally, it delivers comparable or superior accuracy to other recent approaches.
2. **Preservation of FCM intrinsic interpretability**: Unlike existing models that introduce architectural changes, Neural-FCM achieves a fusion of DL and FCM without altering the foundational principles of low-level FCMs.
3. **Instance-specific weight matrices**: Neural-FCM generates a weight matrix for each input instance, introducing structural adaptability into FCM modeling, enhancing interpretability and promoting classification accuracy in node-per-class FCM models.

The rest of the paper is organized as follows: the next section provides an overview of the fundamentals of FCM classifiers and the Neural-FCM theory. The proposed approach is further elaborated in Sect. 3. The experimental procedure is presented in Sect. 4, whereas Sects. 5 and 6 provide and discuss the results respectively. Finally, Sect. 7 summarizes this work.

2 Background theory

This section covers the fundamentals of low-level FCM classifiers and ANNs, along with notable related works regarding FCM supervised learning methods. To aid the reader in understanding the progression of literature on these topics, the fundamental concepts are explained before presenting the related works.

2.1 Fuzzy Cognitive Map classifier

An FCM represents a system's function as a graphical model, comprising fuzzy weighted connections in the interval $[-1,1]$ between N nodes or concepts [1]. The weights are represented mathematically by a two-dimensional $N \times N$ matrix, where each entry quantifies the strength and direction of the influence between two concepts. Each node within a data-driven FCM is assigned an initial activation state based on real data. During FCM inference, these states are iteratively updated using an inference formula that employs both the weight matrix and the concept states. This iterative process continues until either convergence, where the change in states becomes negligible, or a predefined number of iterations is reached. Formally, an FCM is defined as a three-element tuple: FCM = $(\boldsymbol{c}, A, \boldsymbol{W})$, where $\boldsymbol{c}$ denotes the total concepts of the system, A the activation formula, and $\boldsymbol{W}$ the weight matrix. In what follows, each element is further elucidated.

The concepts (or nodes) are denoted as $\boldsymbol{c} = \{c_1, c_2, \ldots, c_N\}$ and represent the N total concepts in a system, or the N total variables of a dataset, with $c_n \in [0, 1]$ denoting the initial value of concept n, for $n = 1, 2, \ldots, N$. In a low-level FCM classifier, the initial concept state values c_n are obtained by normalizing the variables to the interval $[0, 1]$ for each observation in the dataset. Variables may be either independent or dependent with input and output concepts inheriting these attributes [3]. An input concept may be either independent or dependent, whereas an output concept (class concept) is inherently dependent. Based on the interconnections among concepts, different architectures can be constructed [22]. In FCM-based classification, the activation state of the output concepts after inference, also referred to as reasoning, is used for class decision-making.

To formally denote this, let $\boldsymbol{X} \in \mathbb{R}^{M \times N}$ be a dataset with M total data instances or samples, and N variables representing $N\text{-}1$ input features of an output class k, with $k \in \{1, 2, \ldots K\}$ and $M, N, K, \in \mathbb{N}^*$. Each data instance m is then represented as a pair ($\mathbf{x}_m$, k_m), where:

- $\boldsymbol{x}_m = (\boldsymbol{x}_{m,1}, \boldsymbol{x}_{m,2}, \ldots \boldsymbol{x}_{m,N\text{-}1})$ is the vector of features for the m-th instance, with $x_{m,n} \in [0,1]$ after normalization, and $c_n = x_{m,n}$ the initial value of the n-concept.
- k_m the integer class label for the m-th instance.

To construct an FCM classifier, two common architectures are utilized, based on the representation of class concept(s) and the adaptation of FCMs to classification practices [22, 25]. These architecture representations are depicted in Fig. 1 and correspond to:

- **Class-per-output architecture**: Each class is represented with a distinct output concept in the FCM model. Class labels are encoded using one-hot encoding, transforming the integer label k_m into a K-length binary vector $\boldsymbol{k}_\mathrm{m}$, as follows (Eq. 1):

$$k_{m,i} = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases} \qquad for\ i = 1,2,\ldots\ K \tag{1}$$

As a result, $\boldsymbol{c}$ is transformed to $\boldsymbol{c}'$, with $\boldsymbol{c}' = \{c_1, c_2, \ldots, c_{N\text{-}1}, \ldots, c_{N\text{-}1+K}\}$ and the FCM architecture comprises of $N\text{-}1+K$ total concepts, where K the total output class concepts.

- **Single-output architecture**: A single concept is utilized to represent the class decision, i.e. $c_N = k_m$, constructing an FCM with N total concepts, where only one concept encodes all class information. Since FCMs operate on concepts values within the interval [0,1], class labels are normalized accordingly.

In FCM classifiers, the actual value of the output concept is initialized with a dummy value $v \in [0, 1]$. The objective of the FCM classifier is to convert the initial vector $\boldsymbol{c}$ into a state vector $\boldsymbol{a} = \{a_1, a_2, \ldots, a_N\}$ such that $a_N = k$; thus, transforming the dummy value v into the actual class label k for each instance. In the class-per-output architecture, the class label is determined by the index of the output concept with the highest activation value (Eq. 2):

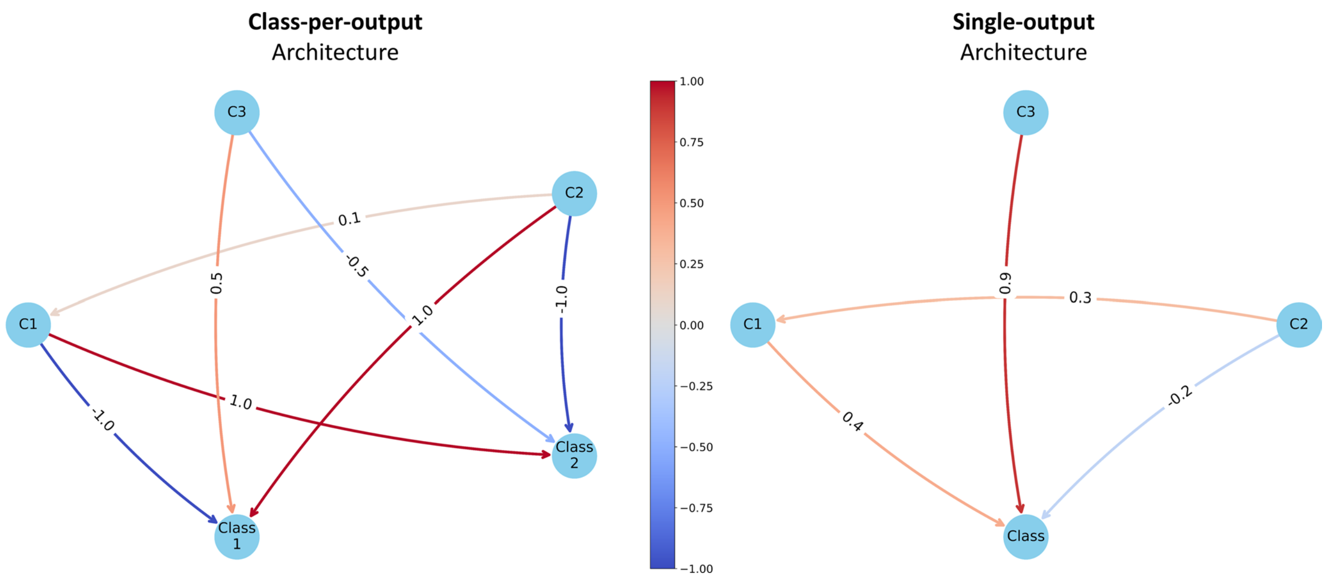

Fig. 1 Example of FCM classifier architectures as defined in [22]. Left: The class-per-output architecture, where each class is represented by a distinct output concept (e.g. Class 1 and Class 2 nodes). Right: The single-output architecture, in which all classes are encompassed within a single class concept node. As a graphical model, the class-per-output design enables a more explicit representation of how input nodes influence each class individually. In contrast, the single-output design may obscure the influence paths from input concepts to specific classes. For example, in the class-per-output architecture, concept "C1" shows a strong negative influence toward "Class 1" and strong positive influence toward "Class 2". In the single-output case, the influence of "C1" becomes less distinguishable. A divergent blue/red colormap is used to indicate the polarity and strength of the weighted interconnections among concepts

$$\widehat{k} = argmax\left(a_{N-1+k}\right), \ \ for\ k = 1,\ 2,\ \ldots,\ K \tag{2}$$

In the single-output architecture, clustering or threshold methods are commonly applied [34] to determine the class label. The algorithm essentially searches for the optimal boundaries to assign the activation state value of the output concept into a class label.

The transformation of $\boldsymbol{c}$ into $\boldsymbol{a}$ is achieved iteratively using the activation formula A (Eq. 3) and the weight matrix $\boldsymbol{W}$. Also known as the “activation rule”, this formula enables FCMs to exhibit dynamic behavior in concepts states, producing a new state vector $\boldsymbol{a}^t$ after each iteration $t \in \mathbb{N}^*$, until either convergence is achieved, or a predefined iteration limit is reached [34]. Literature, suggests that for classification, inference should generally involve a small number of iterations [25]. Although several variations of the activation formula exist [11], the most common one [35], which is also employed in this work, is presented in Eq. 3:

$$\boldsymbol{a}^{t+1} = f(\boldsymbol{a}^t\boldsymbol{W} + \boldsymbol{a}^t) \tag{3}$$

where f is the non-linear sigmoid function (Eq. 4) that keeps $\boldsymbol{a}^{t+1}$ within the defined limits (i.e. the desired [0, 1] interval for the concept states), and $\boldsymbol{a}^0 = \boldsymbol{c}$. It is worth mentioning that Eq. 3 is commonly presented as an element-wise operation [1, 36], meaning that it computes the activation value for each a_n^{t+1} concept, individually. Hence, the matrix-wise operation presented herein hides the constraint of $w_{i,j} = 0$ for $i = j$, which denotes that there is no self-activation in concepts. We employ the matrix-wise formula as it is both convenient and efficient for computing state vectors using vectorization, using computation-optimized Python libraries such as NumPy [37] and TensorFlow [38]. The loss function ensures that self-activation is avoided, as explained later in the corresponding section.

Besides ensuring the boundaries of activation states, the sigmoid function in Eq. 3 is commonly utilized with λ and b parameters in FCM implementations, to control its steepness and shift, respectively. In other words, the λ and b parameters are employed to control the magnitude of the effect at each iteration, thereby further assisting convergence [39]. Alternatively, the hyperbolic tangent function can be employed (Eq. 5) in cases where the activation state is pursued within the [−1, 1] range to consider both positive and negative effects.

$$f\left(x\right) = \frac{1}{1 + e^{-(\lambda\,x + b)}} \tag{4}$$

$$f\left(x\right) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5}$$

Finally, $\boldsymbol{W}$ in the FCM tuple represents the 2-dimensional weight matrix, with $w_{i,j} \in [-1, 1]$ denoting the influence of concept c_i on concept c_j. In data-driven FCM applications $w_{i,j}$ is calculated by a learning algorithm. Specifically, a function $g: \boldsymbol{x}_m \rightarrow y_m$ is sought, which incorporates FCMs, is parametrized by $\boldsymbol{W}$, and can be denoted as *g(FCM(x; W))*. Supervised learning paradigms are employed to find the optimal $\boldsymbol{W}$ that minimizes an error function. A comprehensive review of relevant learning algorithms can be found in [10]. It is worth mentioning, that the weight matrix is the fundamental part in defining the causality among concepts [1], as the weight values are interpretable and can be expressed in fuzzy linguistic terms. Along with the activation states of concepts produced after reasoning, the causes and effects on the systems can be assessed, enabling transparent and interpretable class decision-making.

2.1.1 Related work

Following the main objective, the establishment of a novel low-level learning method capable of generalizing across domains, this section highlights representative works that propose and evaluate FCM classification learning methods across various datasets. A summary of the relevant literature is provided in Table 1 to facilitate clarity and comparison. While the review in [3] outlines the early developments in this area, the present review extends that foundation by examining more recent approaches as well. As presented herein, several of the challenges identified in [3], regarding the construction, accuracy, interpretability and convergence of FCMs, remain unresolved. Notably, a recurring limitation in contemporary studies is the limited attention given to interpretability: most data-driven FCM classifiers do not provide any analysis or discussion of the learned weight matrices, thereby overlooking a key aspect of model transparency.

The foundational FCM classifier theory and the class-per-output architecture, known as Fuzzy Cognitive Mappers (FCMper), were initially developed in [40]. Despite the use of hybrid ANN-FCM models to improve performance, which however sacrificed the transparency of FCMs, the maximum classification rates for benchmark datasets, such as Iris [42], were substantially lower compared to more recent works. The creators of FCMper extended their previous work in [22]. Their main contribution was the comparison of various Hebbian-based learning algorithms on open datasets. They found that the class-per-output structure was less effective than the single-output architecture, which uses clustering-based class mapping. Additionally, genetic algorithms were shown to outperform Hebbian learning in terms of accuracy.

Table 1 Related work summary

Work	Learning algorithm	Main Contribution	Limitations
FCMpers [40]	Genetic	• Initial assessment of low-level FCM classifiers on benchmark datasets	• Weak predictive performance • Hybrid black-box FCMs • No weight matrix analysis
FCM classifiers [22]	Hebbian-based, Genetic	• Comprehensive comparison of learning algorithms and FCM architectures • Demonstration of the effectiveness of genetic learning and single-output architecture	• Weak predictive performance • No weight matrix analysis
Improved FCM classifiers [34]	Genetic	• A novel thresholding method for single-output architecture • Accuracy improvements compared to previous approaches and ML models	• Architecture constraints. Mimics ANNs and obscures interpretability • No weight matrix analysis
FCMs and Capsule networks [41]	PSO	• Classification accuracy improvement in Class-per-Output architecture	• Alteration of FCM reasoning • Employs fully connected ANN-like architecture
ANN-based FCM [25]	Gradient descent	• Gradient-based learning for classification • Improved classification accuracy in Class-per-Output architecture • A novel FCM-based feature transformation method	• Alteration of FCM reasoning • ANN type fully connected architecture with bias concepts • No weight matrix analysis

By proposing a new algorithm for generating thresholds in single-output architectures, the work of [34] improved the efficiency of the FCM classifier. A genetic algorithm for FCM learning and single-step reasoning were used in their experiments on open datasets, similar to [22]. Their method outperformed both traditional clustering-based class assignment methods and popular ML models on several datasets. However, limitations include the inefficiency in genetic optimization and FCM structural constraints that mimic a feedforward ANN without hidden layers. Single-output architectures may also reduce interpretability by obscuring the contribution of each input concept to the decision, particularly when numerous classes are considered, as it was illustrated in Fig. 1. Moreover, no weight matrix analysis was considered to disclose interpretability.

A class-per-output architecture was considered in [41], where the weight matrix was learned using the PSO algorithm. The novelty of this approach lies in the integration of a capsule network into the inference rules. Capsules are groups of neurons that represent specific properties of an object, and they output vectors rather than scalars, allowing them to capture more complex features. The capsule network uses top-down feedback to adjust the coupling coefficients, increasing the coefficient for the correct category and decreasing those for incorrect categories. Additionally, the scalar product between the capsule's prediction and the output of the correct category is enhanced. This enhancement improved the classification accuracy of the class-per-output architecture, preserved standard weighted interconnections, and achieved performance comparable to that of other ML methods on various datasets. However, this enhancement altered the activation formula of the FCM, and introduced additional processing steps, thereby increasing complexity.

In response to the challenges of genetic optimization, gradient-based supervised learning methods have emerged. It is worth mentioning that the current state-of-the-art in gradient-based learning methods, or deep FCMs, is mainly applied in time series prediction [29, 43]. Regarding classification and low-level modeling, the work of [25] is among those recent found in the literature. The authors developed a fully connected FCM classification architecture, adopting aspects of ANNs. This approach uses gradient descent for learning, an extended activation formula, and class-per-output architecture with bias concepts and backward links, which are particularly effective for multi-iteration reasoning. In addition, another contribution of the proposed scheme is the ability to transform the feature space by grouping observations that belong to a given class. The accuracy achieved by this method is comparable to that of other ML models employed for comparison. However, a limitation of the approach is that it modifies the low-level working principle and structure of the FCM due to the selected FCM properties and the learned weighted interconnections which exceed the standard $[-1, 1]$ range.

Lastly, for applications requiring uncertainty modeling, more complex solutions have been proposed, as the typical FCM model utilizes a static weight matrix for reasoning, an approach that entails limitations. FGCM [31, 44], in particular, employ grey numbers and weight ranges to handle uncertainty in dependencies between concepts and adapt to dynamic conditions, such as seasonality in time-series data. The learning process is predominantly carried out using population-based methods as well. Although FGCM mitigates the limitations of static matrices, it also increases computational cost and complexity, especially when interpreting white numbers in the output concepts. High-level FCM classifiers, which do not directly represent input features, are also documented in the literature. Examples include Fuzzy-Rough Cognitive Networks [45] and Deep-FCMs [46–48] which utilize information granules rather than low-level features. Such methods in which concepts do not correspond to low-level features, are beyond the scope of this work.

2.2 Artificial neural networks

ANNs, as previously discussed, are data-driven, "black-box" models that utilize ML paradigms to approximate complex, non-linear functions. According to the universal approximation theorem, a multilayer ANN can, theoretically, learn any arbitrary relationship between input and output data, given sufficient parameters and training, to an acceptable level of accuracy [49]. This concept is formalized as a function g that maps an N-dimensional input to a K-dimensional output (Eq. 6):

$$g_\theta : \mathbb{R}^N \rightarrow \mathbb{R}^K \tag{6}$$

In an ANN, θ denotes trainable weights and biases that are associated with the following elements [50]:

- **Depth**: The total number of layers in the ANN.
- **Layers**: Configuration parameters for each layer, including operation type (e.g., perceptron, convolution, residual connections [51]), number of neurons or kernels and activation functions.
- **Optimizer**: A method for adjusting weights during training. As backpropagation [52] is the dominant algorithm for training large-scale Neural Networks, an optimizer such as Adam [53] is employed to perform efficient weight updates. Backpropagation functions by calculating the gradient of the error with respect to the weights and biases, and the optimizer adjusts them in the opposite direction of the gradient.
- **Loss function**: The loss (or cost) function penalizes the network based on prediction errors [54]. For backpropagation to work, the loss function must be differentiable. The objective during training is to minimize this loss function by adjusting the weights of the network. In classification tasks, a commonly used loss function is categorical cross-entropy (CCE), which quantifies the dissimilarity between the predicted and the true distributions. For a dataset with M instances and K classes, the categorical cross-entropy is defined as (Eq. 7):

$$\mathcal{L}_{CCE} = -\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} y_{m,k}\log\left(\widehat{y}_{m,k}\right) \tag{7}$$

Where:

- $y_{m,k} \in \{0,1\}$ the binary indicator specifying if class label k is the correct classification for the m-th instance,
- $\widehat{y}_{m,k} \in [0,1]$ is the predicted probability that the m-th instance belongs to class k.

Convolutional Neural Networks (CNNs) represent a distinct type of ANNs, specifically designed to process grid-like structured data and capture spatial relationships within the input. Drawing inspiration from the way the human visual system perceives images, CNNs use the convolution operation in place of the perceptron to detect complex and non-linear patterns [50]. Compared to FCMs where the activation function f is of sigmoid type (Eqs. 4 and 5), the Rectified Linear Unit (ReLU) (Eq. 8) is commonly used in the hidden layers of ANNs due to its proved computational efficiency and faster convergence [55]:

$$ReLU\,(x) = max(0, x) \tag{8}$$

2.2.1 Related work

Within the context of deep learning, the evolution of ANNs along with variants such as CNNs, has led to the development of robust models with improved learning abilities [56]. In fact, the proper design of the model architecture is a key factor for successful learning.

One aspect of architecture that is exploited in the proposed Neural-FCM approach is the decoder architecture. A decoder is a fundamental design concept in autoencoders [57] and generative AI [58]. Essentially, a decoder, is a function g, as in Eq. 6, that maps an input $\boldsymbol{x} \in \mathbb{R}^{\mathrm{d}}$ to a higher dimension output $\boldsymbol{Y} \in \mathbb{R}^{\mathrm{D}}$, where $d < D$. In contrast to the widely used encoder architecture [51], which progressively decreases the input dimension for feature learning, the decoder operates in the exact opposite manner. It gradually increases the input vector through stacked hidden layers, enabling the model to effectively learn a representation of the data, also referred to as an embedding or latent variable, which can be used to generate new data samples.

3 Methodology

Considering the related work analysis, the core novel idea behind Neural-FCM is to investigate a decoding ANN architecture capable of generating instance-specific weight matrices for FCM-based reasoning. To illustrate this, Fig. 2 presents how Neural-FCM extends the typical black-box classification process of ANNs by integrating transparent FCM inference.

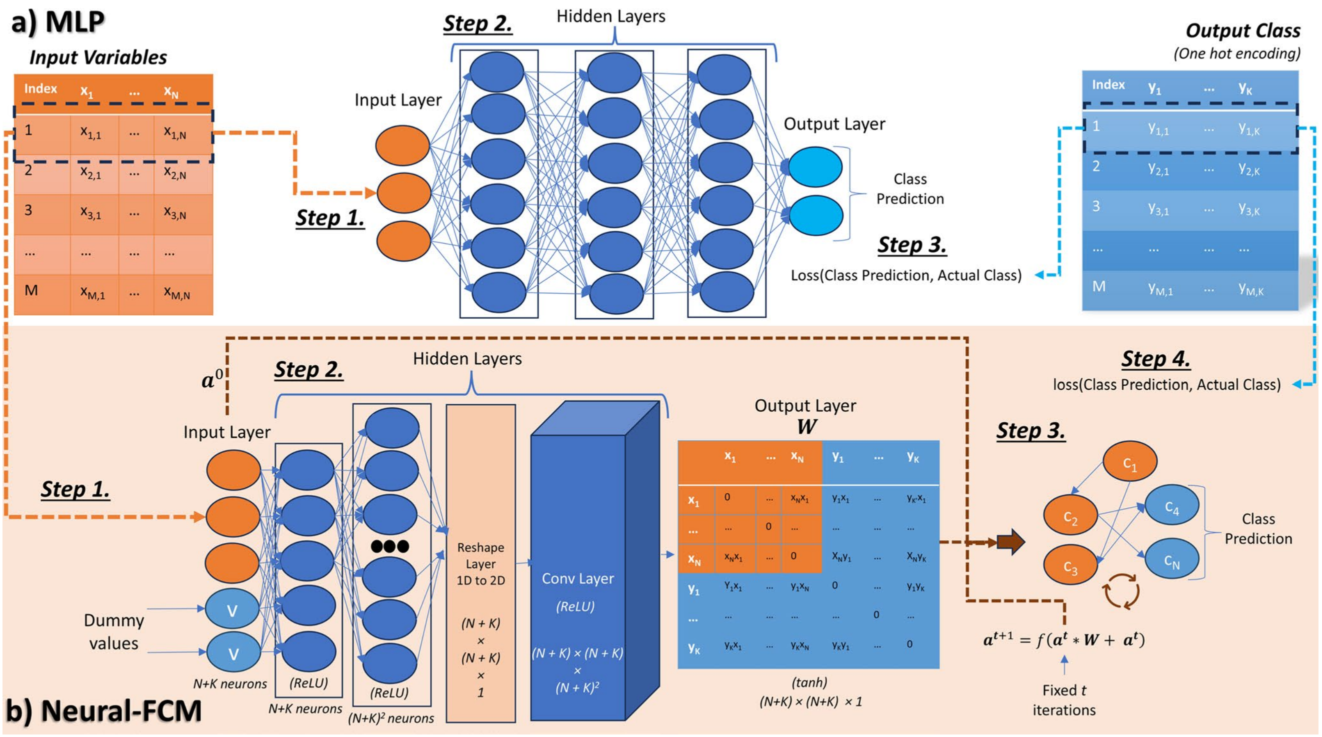

Fig. 2 Comparison between a standard MLP (top) and the proposed Neural-FCM (bottom) that elucidates the novel introduction of the additional FCM processing step and the employed ANN architecture. Neural-FCM employs a hybrid model to decode an input into an instance-specific FCM weight matrix. The weight matrix is then used for FCM inference with Eq. 3 before calculating prediction loss

Table 2 A step-by-step comparison of the processing pipeline between a typical ANN classifier and the proposed Neural-FCM model. Neural-FCM modifies the typical procedure by generating instance-specific weight matrices to be used for FCM reasoning

Processing Step	Typical ANN Classifier	Neural-FCM
Step 1	An input N-length feature vector of an m-th data instance is passed to a multilayer perceptron (MLP).	An input $(N+K)$-length vector of an m-th data instance is passed to the Neural-FCM, with N input features, and a K-length dummy vector representing K total classes.
Step 2	The hidden layers obscurely process the input vector to predict the corresponding class	The hidden layers obscurely process the input vector, decoding it into an instance-specific $(N+K)\times(N+K)$ matrix $\boldsymbol{W}$
Step 3	The output is assessed based on a differentiable cost function during training to adjust the weights. In a trained model, the predictions on new unseen data are utilized for reasoning and decision-making without any transparency.	The matrix output $\boldsymbol{W}$, the input vector (denoted as $\boldsymbol{a}^0$ in the first FCM iteration) and the inference formula A (Eq. 3) are subsequently employed for FCM construction and inference for a fixed number of iterations, as described in Sect. 2.1.
Step 4		The error for the output (class) concepts is calculated in the modified loss function after FCM inference. During training, the loss function penalizes outputs (FCM weight matrices) that produce high errors in the FCM reasoning. It is obvious that as the learning error decreases, more meaningful matrices are generated. Finally, the trained Neural-FCM can then be employed to output instance-specific weight matrices to be used for FCM class decision. Weighted interconnections and input features can be further studied to analyze causes and effects, enhancing transparency in decision-making.

Furthermore, Table 2 provides a side-by-side comparison of the generic processing steps involved in a traditional ANN classifier and the proposed Neural-FCM approach, offering a high-level conceptual overview. To complement this, Algorithm 1 presents a concise pseudocode summarizing the procedural flow of Neural-FCM learning. The following subsections elaborate in more detail on the working principles and technical components of the proposed methodology.

Algorithm 1

```
Input: Input vector $\mathbf{x} \in \mathbb{R}$^(N+K), one-hot label vector $\mathbf{y} \in$ {0,1}^K
Neural_FCM_decoder: $\theta \in \mathbb{R} \mid \theta \geq 0$ // $\theta$ trainable weights
Decoder output: Predicted one-hot vector $\hat{\mathbf{y}} \in$ {0,1}^C, weight matrix W ∈ [-1,1]^((N+K)×(N+K))
FCM iterations: t ∈ {0,1,2,…, 6}

1:  function NEURAL_FCM_LEARNING(x, y, Neural_FCM_decoder):
2:      // Step 1: Initialize FCM concept state vector
3:      $\mathbf{a}^0 \leftarrow \mathbf{x}$
4:
5:      // Step 2: Decode weight matrix from input vector
6:      $\mathbf{W} \leftarrow$ Neural_FCM_Decoder($\mathbf{x}, \theta$) // Deep network with dense and convolutional layers
7:
8:      // Step 3: FCM inference using W
9:      $\mathbf{a}^{\text{final}} \leftarrow$ FCM_Inference_Formula($\mathbf{a}^0$, $\mathbf{W}$, t) // Equation 3
10:
11:     // Step 4: Map final concept states to class scores
12:     logits ← output_mapping($\mathbf{a}^{\text{final}}$) // Slice the array to obtain the K output concepts
13:     $\hat{\mathbf{y}}$ ← Softmax(logits)  // One-hot predicted probabilities
14:
15:     // Step 5: Compute loss and backpropagate
16:     loss ← CrossEntropy($\hat{\mathbf{y}}$, $\mathbf{y}$)
17:     $\theta$ ← Backpropagate(loss, Neural_FCM_Decoder)
18:
19:     return Neural_FCM_decoder
```

3.1 Neural Fuzzy Cognitive Map

To implement Neural-FCM learning, a decoding ANN architecture for transforming input vectors into weight matrices and a differentiable FCM-based loss function for penalizing inaccurate matrices are essential components.

3.1.1 Learning

For a dataset $\boldsymbol{X} \in \mathbb{R}^{M\times(N+K)}$ with $M, N, K \in \mathbb{N}^*$, and a data instance $\boldsymbol{m} = (\boldsymbol{x}_m, \boldsymbol{y}_m)$ where $\boldsymbol{x}_m = (\boldsymbol{x}_{m,1}, \boldsymbol{x}_{m,2}, \dots \boldsymbol{x}_{m,N})$ represents the N input features and $\boldsymbol{y}_m = (\boldsymbol{y}_{m,N+1}, \boldsymbol{y}_{m,N+2}, \dots \boldsymbol{y}_{m,N+K})$ represents the one-hot encoded (Eq. 1) K class labels, Neural-FCM can be defined as a decoding function approximation (Eq. 9):

$$g_\theta : [\boldsymbol{x}_m; \boldsymbol{v}] \rightarrow \boldsymbol{W}_m \tag{9}$$

Here, $\boldsymbol{W}_m \in [-1,1]^{(N+K)\times(N+K)}$ is an instance-specific matrix describing the weighted interconnections among the $N+K$ FCM concepts, $\boldsymbol{v}$ a dummy vector of length K and $[\boldsymbol{x}_m; \boldsymbol{v}]$ the concatenated input vector of length $N+K$, and θ the decoder's trainable weights. The dummy vector $\boldsymbol{v}$ ensures consistency during matrix multiplication within the FCM activation formula (Eq. 3). Since $\boldsymbol{v}$ represents the class concepts, all elements take a fixed value of **0.5**, or the median of the defined [0,1] interval. Hereafter, for clarity and unless otherwise specified, the concatenated input vector $[\boldsymbol{x}_m; \boldsymbol{v}]$ will be denoted as $\boldsymbol{x}_m$.

Using the FCM model $FCM(\boldsymbol{x}_m, A, \boldsymbol{W}_m)$ where A is the activation formula (Eq. 3) that transforms $\boldsymbol{x}_m$ to a final activation vector $\boldsymbol{a}_m$, the objective of learning is to find the optimal $\boldsymbol{W}_m$ that minimizes the classification error as it is denoted in Eq. 10:

$$\boldsymbol{W}^*_m = \underset{\boldsymbol{W}_m}{\text{argmin}} \sum{}_{m=1}^{M} CCE(a_{m,k}, y_{m,k}), \quad for\ k = N+1,\ N+2,\ \dots,\ N+K \tag{10}$$

Where:

- $\boldsymbol{W}^*_m$ is the optimal weight matrix for instance m. To comply with the semantics of FCMs [1], each element of the matrix $\boldsymbol{W}^*_m$ is constrained to the interval $[-1,1]$ where negative values represent inhibitory influence and positive values represent excitatory influence among concepts.
- CCE is the classification loss of Eq. 7,
- $a_{m,k}$ is the value of the k-th class concept in the final activation vector.
- $y_{m,k}$is the binary indicator for the one-hot encoded class label.

Unlike traditional FCM learning approaches that produce a single and static weight matrix shared across all inputs, Eqs. (9) and (10) define a dynamic formulation, where a distinct matrix $\boldsymbol{W}_m$ is generated for each input $\boldsymbol{x}_m$. This per-instance matrix generation allows the model to adapt its internal structure to the context of each input, thus introducing structural dynamicity into the reasoning process, aiming to facilitate both reasoning accuracy and data fitting, as further exemplified in the results section.

3.1.2 Decoding architecture

Considering the employed ANN architecture for decoding the input vector into a weight matrix, it comprises both dense (fully connected) and convolutional layers. Dense layers are utilized in the beginning of the network to process and expand input vectors, while convolutional layers are found towards the end to process grid structure data. Although the whole decoding process could be achieved with merely dense layers, the utilization of convolutional layers, results in higher efficiency. The network's architecture is empirically chosen and further presented herein. It is worth mentioning that as with ANNs and essentially any parametric ML algorithm, each dataset may be benefited by re-configuring the Neural-FCM architecture and by experimenting with more layers, neurons and other properties. In more detail, the ANN architecture consists of:

1. **Initial Layer**: The first hidden layer contains the same number of neurons as the length of $\boldsymbol{x_m}$ and is employed to trigger initial fully connected weighted relationships between data attributes.
2. **Decoding Dense Layer**: A subsequent dense layer with $(N+K)^2$ neurons processes and enlarges the input vector. Both the current and the previous layer employ the Rectified Linear Unit (ReLU) activation function (Eq. 8) known for its computational efficiency and reduction of the vanishing gradient problem [55].
3. **Reshaping Layer**: The third consecutive layer reshapes the $(N+K)^2$-length vector into an $(N+K)\times(N+K)\times 1$ matrix.
4. **Convolutional Layer**: This layer applies $(N+K)^2$ convolutional filters, each with the commonly employed window size of 3×3, zero-padding, and ReLU activation.
5. **Output Layer**: Finally, the output layer applies a single convolution filter of 1×1 size with zero-padding, in order to shrink the $(N+K)\times(N+K)\times(N+K)^2$ tensor and to produce the desired $(N+K)\times(N+K)\times 1$ output matrix. To ensure the weight matrix $\boldsymbol{W_i}$ conforms to the FCM constraint of $w_{i,j}\in[-1, 1]$, the hyperbolic tangent (Eq. 5) activation function is employed as the output activation.

An abstract representation of the Neural-FCM architecture, that thoroughly described herein, is depicted in Fig. 2.

3.1.3 Loss function

A critical component of Neural-FCM implementation is the custom loss function that penalizes non-conforming weight matrices. TensorFlow [32] and Keras are employed to define and prototype this custom loss function, which internally incorporates matrix multiplication operations from Eq. 3 before computing CCE (Eq. 7). TensorFlow's automatic differentiation [59] simplifies gradient computations, allowing efficient implementation of the FCM reasoning step within the loss function. To differentiate automatically, the framework tracks operations and their order during the forward pass and computes gradients during the backward pass by traversing the operation list in reverse. This section emphasizes the high-level details of the proposed modified CCE loss function and shuns the low-level concepts of backpropagation.

Besides penalizing the network for classification efficiency, the proposed loss function further ensures that the derived weight matrices conform to the standard model architecture [1]. Moreover, it penalizes interconnections to acquire a matrix for a feed-forward FCM architecture with (1) recurrence only between input concepts, and (2) no self-connections, as the one presented in Fig. 1 (Left). This is achieved by performing three calculations, or in other words, by combining three loss functions in the final weighted total loss. Following the previously defined notations, these losses are defined as:

1. **Classification Loss**: For the employed class-per-output FCM classifier the CCE loss function is employed. Equation 7 is transformed into Eq. 11 which calculates the cross-entropy loss between the true labels (one-hot representation) of the m-th instance, and the $a_{m,\,N+k}$ output concepts of the final activation state vector $\boldsymbol{a}$. The activation values of output concepts are passed through a softmax activation function (Eq. 12) to yield probabilities.

$$\mathcal{L}_{classification} = -\frac{1}{M}\sum\nolimits_{m=1}^{M}\sum\nolimits_{k=1}^{K} y_{m,k}\log\left(\widehat{a}_{m,\,N+k}\right) \tag{11}$$

$$\widehat{a}_{m,\,N+k} = f\left(a_{m,\,N+k}\right) = \frac{e^{a_{m,\,N+k}}}{\sum_{k}^{N+K} e^{a_{m,\,N+k}}} \tag{12}$$

2. **Diagonal Loss**: This loss aims to vanish self-connections among concepts by minimizing the diagonal entries of the weight matrix $\boldsymbol{W_m}$ (Eq. 13), following the restrictions defined with Eq. 3. The standard FCM architecture requires the constraint of $w_{i,j}=0$ for $i=j$ which practically means that a concept value is only affected by other concept values in the model. Thus, enhancing the interpretability of identified causes and effects [1].

$$\mathcal{L}_{diagonal} = \frac{1}{M}\frac{1}{N+K}\sum{}_{m=1}^{M}\sum{}_{i=1}^{N+K}\boldsymbol{W}_{m,i,i}{}^{2} \tag{13}$$

3. **Output Loss**: This loss diminishes connections from the output concepts toward input concepts, following the FCM classifier architecture that allows interconnections only between input features [22]. This practically means that class concepts are modeled as dependent variables that do not affect the values of input concepts. Thus, the k rows, for $k = N+1, 2, \ldots N+K$, of the weight matrix $\boldsymbol{W}_m$ of each m instance, need to have zero values. This is formulated in Eq. 14 as the output loss.

$$\mathcal{L}_{output} = \frac{1}{M}\frac{1}{K}\frac{1}{N+K}\sum{}_{m=1}^{M}\sum{}_{i=1}^{K}\sum{}_{j=1}^{N+K}\boldsymbol{W}_{m,i,j}{}^{2} \tag{14}$$

The total loss (Eq. 15) is calculated as the summation of the previous losses. Emphasis is given in the classification loss by empirically utilizing a weight parameter $w_e = 2$:

$$Total\ loss = w_e Loss_{classification} + Loss_{diagonal} + Loss_{output} \tag{15}$$

The steps that are performed inside the loss function for each training instance $\boldsymbol{x}_m$ are the following:

step 1. Employ $\boldsymbol{x}_m$ and $\boldsymbol{W}_m$ to construct an FCM with $N+K$ total concepts. Moreover, this step clarifies that besides the $\boldsymbol{W}_m$ that derives from the Neural-FCM output layer, and the one-hot encoded label vector $\boldsymbol{y}_m$, the Neural-FCM loss function necessitates the input $\boldsymbol{x}_m$ as a function parameter.
step 2. Perform FCM inference with (Eq. 3) and the input $\boldsymbol{x}_m$ for t iterations. Iterations t is an additional parameter that must be predefined.
step 3. Assess the total error using Eq. 15.
step 4. Backpropagate to update $\boldsymbol{W}_m$ outputs based on the computed error.

3.1.4 Model tuning

Lastly, an important consideration in this study is the selection of FCM model parameters, which play a crucial role in the model's performance. These parameters are determined through a systematic experimentation process, aiming to balance accuracy, efficiency, and generalizability. Specifically, the most effective FCM classifier parameters, such as the number of iterations t in the FCM inference formula and the λ-slope parameters in the sigmoid function (Eq. 4) are evaluated, following recommendations from the literature [25]. Experiments are conducted using $t \in \{1, 2, 3, 4, 5, 6\}$ reflecting a range of inference depths, and $\lambda \in \{1, 2, 3, 4, 5\}$ capturing varying levels of nonlinearity in the sigmoid activation. These parameter combinations result in 30 distinct FCM models, each systematically evaluated across 7 datasets to ensure robustness and reliability of the findings. The datasets are chosen to provide diverse challenges for the models, further validating the general applicability of the selected parameters.

For the ANN training of Neural-FCM, a batch size of 126 is employed, a choice made empirically to balance computational efficiency and learning stability. The chosen batch size represents a practical trade-off between these considerations, as early experiments indicated, ensuring consistent performance across datasets and enhancing convergence while improving training speed. Additionally, the Adam optimizer [53] a well-established algorithm known for its adaptive learning rate and computational efficiency, is utilized to streamline the training process, along with the K-fold method for cross-validation.

4 Experiment setup

4.1 Datasets

Following the state-of-the-art [22, 25, 41], a variety of datasets are employed for assessing the generalization of the proposed approach. Particularly, six of these datasets are publicly available, derive from public databases such as the UCI Machine Learning Repository [60] and are popular for benchmarking ML and low-level FCM algorithms across diverse domains. The seventh dataset is proprietary, derived from the health sector, and contains sensitive medical information regarding Corona Artery Disease (CAD) [61]. Each dataset varies in size, complexity, and feature composition, offering a comprehensive evaluation platform for assessing Neural-FCM's accuracy and generalization. Moreover, the inclusion of health datasets introduces real-world complexity, particularly in healthcare-related decision-making. The characteristics of the employed datasets are summarized in Table 3. While the datasets employed are of moderate scale, they represent a standard testbed for low-level FCM classification learning, as used in prior studies. The use of these benchmarks allows a direct and fair comparison of Neural-FCM with existing approaches in the field.

Table 3 Datasets information

Dataset	Attributes	Classes	Instances
CAD [61]	22	2	571
Pima Diabetes [62]	7	2	768
Thyroid [63]	5	3	215
Breast Cancer Wisconsin [64]	30	2	569
Iris [65]	4	3	150
Wine [66]	13	3	178
German Credit Data [67]	19	2	1000

Table 4 Best metrics attained by Neural-FCM based on the λ, t, parameters

Dataset	Accuracy	F1-Score	s/Epoch	Epochs	λ	t
CAD	0.7949 ± 0.0470	0.7934 ± 0.0472	0.252 ± 0.012	45.3 ± 12.3	1	6
Diabetes	0.7539 ± 0.0617	0.6826 ± 0.1199	0.141 ± 0.009	111.1 ± 43.7	1	1
Thyroid	0.9448 ± 0.0481	0.9080 ± 0.1002	0.059 ± 0.047	411.2 ± 213.0	1	1
Breast Cancer Wisconsin	0.9789 ± 0.0262	0.9774 ± 0.0258	0.135 ± 0.005	106.1 ± 42.58	5	1
Iris	0.9667 ± 0.0333	0.9646 ± 0.0367	0.047 ± 0.001	549.4 ± 223.0	1	1
Wine	0.9833 ± 0.0255	0.9846 ± 0.0237	0.076 ± 0.003	405.9 ± 321.3	2	6
German Credit Data	0.7600 ± 0.0359	0.6769 ± 0.0387	0.189 ± 0.012	78.5 ± 34.0	3	1

Table 5 Worst metrics attained by Neural-FCM based on the λ, t, parameters

Dataset	Accuracy	F1-Score	s/Epoch	Epochs	λ	t
CAD	0.6495 ± 0.1368	0.5606 ± 0.2160	0.194 ± 0.014	30.5 ± 18.1	5	3
Diabetes	0.6496 ± 0.0542	0.4124 ± 0.0671	0.167 ± 0.013	234.0 ± 147.6	4	2
Thyroid	0.8093 ± 0.1290	0.6150 ± 0.2313	0.092 ± 0.004	168.2 ± 28.5	4	4
Breast Cancer Wisconsin	0.7485 ± 0.2158	0.6485 ± 0.2944	0.159 ± 0.008	64.8 ± 15.0	4	2
Iris	0.7667 ± 0.2534	0.7298 ± 0.2859	0.053 ± 0.001	541.3 ± 290.0	5	2
Wine	0.8886 ± 0.1609	0.8785 ± 0.1888	0.065 ± 0.006	498.5 ± 582.7	3	3
German Credit Data	0.7070 ± 0.0550	0.4718 ± 0.1024	0.367 ± 0.031	85.2 ± 72.8	5	5

4.2 Preprocessing

Standard preprocessing steps are applied to all datasets to ensure uniformity before model development. These steps include:

- Data normalization in the range of [0,1] using the min-max normalization method.
- Integer (ordinal) encoding columns that contain text.
- One-hot encoding transformation of class labels for defining class-per-output FCM architectures and for utilizing CCE as a cost function.
- Splitting of input/label vectors. In the input vector the class attributes are initialized with the dummy value of 0.5 as previously discussed (Sect. 3.1.1).

Moreover, the custom loss function for Neural-FCM requires both the model's output (weight matrices) and the original input vectors for FCM construction and inference. Thus, input vectors and one-hot encoded output vectors were concatenated and passed together as a single argument to the loss function, which splits them internally. This approach aligns with the high-level frameworks of Keras, which typically expects only two parameters in a cost function definition.

4.3 Hardware and software

All experiments are conducted in a customized workstation that is equipped with an Intel Core i9 −11900KF @ 3.5 GHz processor, 16 GB of DDR4 RAM, and NVIDIA GeForce RTX 3080 Ti with 12GB of GDDR6X memory, running windows 10 professional. The software is developed with the Python programming language (version 3.10) and with the TensorFlow (version 2.9.1) [32]. Other useful tools that are employed herein are the CUDA toolkit (version 11.8) and the cuDNN for GPU processing, and the Matplotlib [68], NumPy [37], and pandas [69] python libraries for plotting and data processing. The development code can be found in the authors' github[1].

5 Results

This section initially presents the results of parameter tuning which facilitate in discovering the proper settings for maximum predictive accuracy. Subsequently a comparative analysis with state-of-the-art methods is performed, focusing on low-level FCM classifiers that were rigorously tested across a variety of datasets. Finally, the generated instance-specific weight matrices are analyzed to acquire insights of the learned weight interconnections that facilitate interpretability.

5.1 Parameter tuning

The 10-fold cross-validation method and average results across folds are employed to ensure reliability. The reserved hold-out group is used for testing, while a validation split (20% of the training data) monitors overfitting with early stopping. The assessment considers the average classification metrics of accuracy, and F1-score achieved in the test dataset. Tables 4 and 5 summarize the performance of the

[1] https://github.com/theotziol/Neural-FCM-Classifier https://drive.google.com/file/d/1DZhYXxN8P8sB0eaoz3yTEcsSpfxUsVcO/view?usp=sharing.

best and the worst parameter configurations (λ, t), respectively. In addition, the average time per epoch (in seconds) and the average epochs that are required to train Neural-FCM with the employed hardware setup and the corresponding parameters are presented to disclose computation requirements.

The results indicate that the Neural-FCM achieves increased classification performance when both sigmoid slope and number of iterations are not excessively high. Only *German Credit Data* and *Breast Cancer Wisconsin* datasets show improvement when λ parameter is increased. Datasets such as *Wine* and *CAD* datasets require additional iterations to maximize classification performance. It is worth mentioning, that when a single FCM iteration is considered, which proved the most efficient solution based on the reported experiments and the literature findings [34], the FCM operates as a typical feed forward neural network.

5.2 Comparative analysis

To further understand Neural-FCM's strengths and weaknesses, we compared its performance with existing FCM methods and ML models from the literature. The aim of the comparison is to assess if the performance achieved with the Neural-FCM is in line with previous works in the field.

Consequently, Table 6 summarizes the accuracy comparison against the implementations and models that were presented in the works of [22, 25, 34, 41, 61, 70]. In more detail, the works of [25, 41] are selected as they tackled the poor predictive accuracy limitations of Class-per-Output (CpO) architectures, proved by their contrasting analysis of their proposed FCM learning methods. In addition, well-established ML models such as Random Forests (RF) and Support Vector Machines (SVM) were evaluated among others in the referenced works. These models have consistently ranked among the most accurate in large-scale benchmark studies on UCI datasets [71]. Therefore, we do not meticulously investigate other works with ML models for accuracy comparison in the widely used public datasets. CpO architectures with genetic algorithm learning (GA) were initially examined in [22]. The extended work of [34] provided accurate FCM classifiers by proposing Single-Output (SO) architecture with a threshold algorithm. Finally, for the proprietary CAD dataset, both the multi-modal DeepFCM of [61] and the proposed ML models of [70] are employed for comparison, providing performance insights against a hybrid high-level CNN-FCM classifier and state-of-the-art ML methods respectively. Regarding the computational requirements, the relevant FCM literature conceals the allocated resources, and the time needed for learning and understates the importance of such metrics. Hence, no performance comparison can be made for similar metrics.

Neural-FCM demonstrates predictive performance improvement across most datasets. In particular, for the CAD dataset, it outperforms the models proposed in [61, 70] which are tailored solutions with domain-specific optimization. A noteworthy finding is the substantial performance gain over traditional class-per-output (CpO) FCM-based architectures trained via standard genetic algorithms with Neural-FCM achieving up to a 34% accuracy increase in the Wine dataset. Significant improvements are also observed when compared to more sophisticated CpO learning methods such as gradient descent-based FCMs, ML models and

Table 6 Comparative accuracy performance of Neural-FCM against existing FCM learning approaches, ML models and hybrid methods reported in the literature

Dataset	DeepFCM (PSO-CNN-SO) [61]	FCM-GA (CpO) [22]	FCM (SO) [34]	FCM-PSO-Caps (CpO) [41]	FCM Gradient [25]	ML models	Neural-FCM
	Accuracy						
CAD	0.7795	-	-	-	-	0.7887 [1]RF [61, 70]	**0.7949**
Diabetes	-	-	-	-	**0.7800**	0.7700 [2]SVCrbf [25]	0.7539
Thyroid	-	0.8528	**0.9870**	0.9133	-	0.9620 RF [41]	0.9448
Breast Cancer	-	-	0.9570	0.9117	0.9600	**0.9800** [3]SCVlin [25]	0.9789
Iris	-	0.6944	**0.9850**	-	0.9700	0.9800 RF [25]	0.9667
Wine	-	0.6400	0.9250	0.9778	0.9700	0.9800 RF [25]	**0.9846**
German Credit Data	-	-	-	-	0.7000	0.7100 [4]LR [25]	**0.7600**

Abbreviations: [1]*RF*: Random Forest, [2]*SVCrbf* Support Vector Machines with radial basis function kernel, [3]*SVClin* Support Vector Machines with linear kernel, [4]*LR* Linear Regression

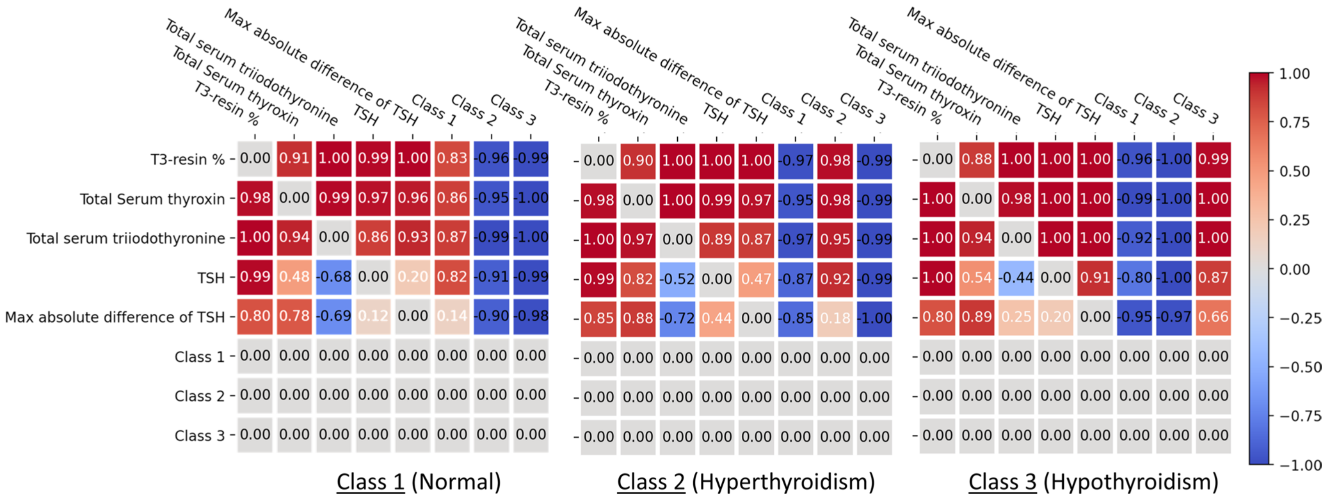

Fig. 3 Instance-specific weight matrices generated by Neural-FCM for three different cases from the Thyroid test dataset, each belonging to a distinct class. The columns {Class 1, Class 2, Class 3} correspond to the class concepts Normal, Hyperthyroidism, and Hypothyroidism, respectively, and indicate the influence of each input variable toward each classification outcome. The variation in weight values, both in strength and polarity, across instances highlights the model's dynamic reasoning mechanism. These instance-specific matrices demonstrate how Neural-FCM adapts the causal structure to suit individual inputs, preserving interpretability while improving classification accuracy

capsule network hybrids. The comparison with the SO model architecture reveals dataset-dependent trade-offs in accuracy.

5.3 Instance-specific weight matrices

The following figures (Figs. 3 and 4) present the dynamic weight matrix adaptation feature of Neural-FCM that offers structural flexibility and facilitates interpretability. In Fig. 3, the weight matrices generated are presented for three randomly selected test instances from the Thyroid dataset, each belonging to a different class. Each matrix in the figure corresponds to the Neural-FCM output and its purpose is to be exploited for transparent and interpretable reasoning. As with traditional FCMs, the weights can be interpreted as causal relationships between concepts, where positive values indicate excitatory influence and negative values indicate inhibitory influence. The term causality herein is

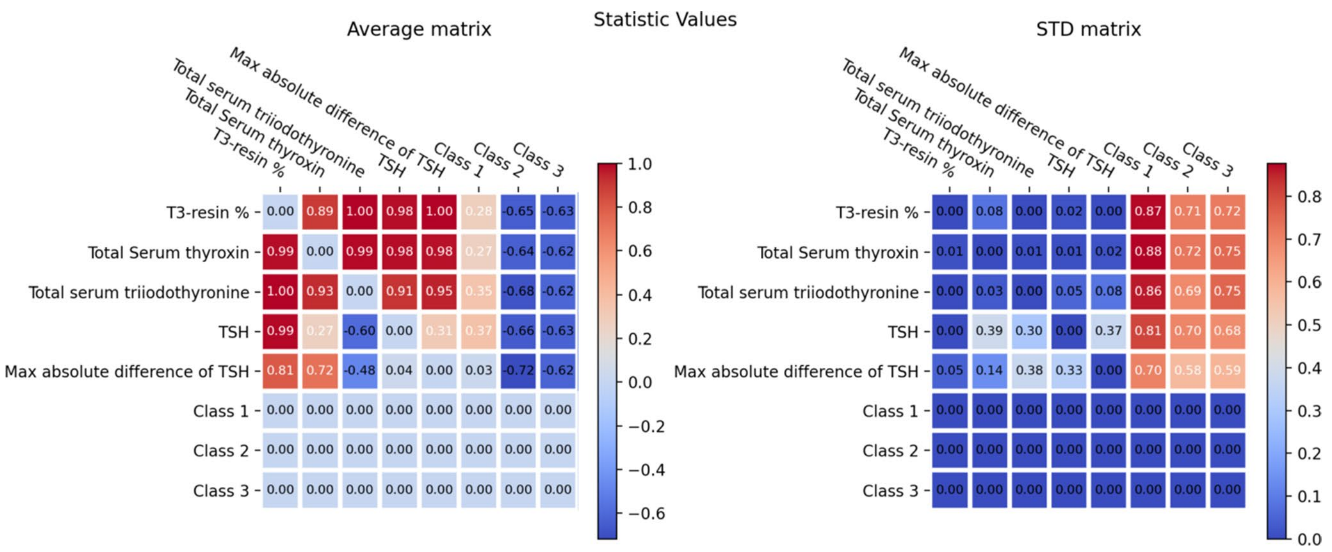

Fig. 4 Average (left) and standard deviation (right) of the Neural-FCM-generated weight matrices across all 43 test instances in the Thyroid dataset. The average matrix shows the general strength and direction of influence among concept variables across all test samples. The standard deviation matrix reveals how much these weights vary from instance to instance, highlighting the dynamic and personalized nature of Neural-FCM. Higher standard deviations in certain regions indicate that the model adapts weight strengths and dynamically reconfigures causal influences depending on input context, supporting instance-adaptive FCM reasoning depending on the input instance. In class columns particularly, the high standard deviation values indicate a key advantage of Neural-FCM over static CpO models as these weights greatly adapt to assist in decision making and accurate classification

borrowed from expert-based FCMs and is used to describe directional and strength associations. Data-driven models like Neural-FCM can capture spurious or confounded correlations that help with prediction [72] without typically providing true causality.

Neural-FCM generates a distinct matrix for each input, allowing the reasoning process to be context-sensitive and data-adaptive. This dynamic behavior is evident in the changes observed across the three instance-specific matrices. Notably, the weights in the class-related columns vary not only in magnitude but also in polarity, depending on the class and input features. This demonstrates that Neural-FCM adapts the causal structure to better fit the classification needs of each instance, a feature not achievable in static FCMs, which apply the same causal assumptions universally. While a full interpretive analysis requires domain-specific knowledge and is beyond the scope of this study, preliminary observations suggest that Neural-FCM adjusts inter-concept weights in ways that align with class-specific patterns.

To further explore this structural flexibility, Fig. 4 presents the aggregate behavior of Neural-FCM's generated weight matrices for the Thyroid test dataset. The left matrix shows average weight values, reflecting consistent patterns of influence learned by the model. In the context of traditional FCMs the average matrix can also be interpreted as a static weight matrix that aims to capture generic causal tendencies. The right matrix visualizes the standard deviation per weight, revealing which connections vary most between instances. These patterns provide an insight into how Neural-FCM adapts the causal structure dynamically per instance, a key advantage over static FCM approaches. These matrices can also be interpreted as a method to identify weight ranges, like the weight ranges that FGCM are producing.

Regarding the weight matrices in the rest of the employed datasets, similar behavior is observed. Nevertheless, for clarity purposes, only the thyroid dataset is picked for analysis and exhibition, as it consists of a few attributes and thus, the variations and the conclusions are more noticeable.

6 Discussion

6.1 Strengths and contributions

The comparative analysis validated the significant predictive performance enhancement of CpO architecture when integrated with the proposed approach. Compared to the reported most efficient architecture representation, the single-output architecture [34], Neural-FCM maintains a competitive edge. However, besides accuracy, the utilization of single-output architectures obfuscates the interpretation regarding the polarity and the degree of influence of input concepts towards the output, especially for non-binary classification tasks. Hence, it has an interpretability disadvantage over CpO architecture. The gradient-based approach [25] attempted to address the ineffectiveness of CpO architectures but at the cost of core FCM properties, introducing elements such as bias nodes, self-loops, and modified reasoning rules. Neural-FCM achieves comparable or occasionally improved accuracy without such structural compromises, thereby maintaining the foundational principles of FCMs, which are essential for interpreting causes and effects. Furthermore, the comparison with case-specific works in the CAD dataset further supported that the proposed method generalizes fairly across domains and particularly in real-world medical datasets. These comparative results validate the effectiveness of the proposed method and support its generalizability across diverse classification tasks.

Besides predictive performance, the adherence to core FCM reasoning is one of Neural-FCM's central strengths. Furthermore, Neural-FCM introduces a novel feature: the ability to generate dynamic weight matrices, that is, to output a unique weight matrix for each input vector. This feature builds on ideas from FGCMs by learning to estimate weights that adapt to specific instance characteristics. In contrast to FGCMs, which require manually transitioning grey ranges into crisp values, Neural-FCM automatically performs this transition through data-driven learning. The resulting behavior is both flexible and robust, enabling the model to fine-tune its reasoning structure per instance. This instance-adaptive causal modeling provides greater precision and context awareness than static FCM approaches. As shown in Fig. 3, the generated weight matrices differ not just in values but also in polarity of key connections. For instance, in the case of a sample diagnosed with Hypothyroidism, the input concepts "*Total serum triiodothyronine*" and "*Max absolute difference of TSH*" exhibit strong positive influence on the *hypothyroidism* class concept, while exerting negative influence on the other class concepts. This suggests that higher values of these input features increase the model's confidence in assigning the *hypothyroidism* class, reflecting a case-specific learned association structure. This structural adjustment allows Neural-FCM to adapt its reasoning mechanism per input, a capability not available in static FCMs, which assign the same weight regardless of context.

Furthermore, the matrices presented in Fig. 4 offer an intuitive illustration of the key advantages of Neural-FCM over traditional static FCM models. The average matrix (left panel) represents what a static FCM would typically aim to produce, a single, fixed weight configuration meant to generalize across all inputs. Such matrices, commonly derived through population-based methods, tend to smooth out the nuances between instances, thus sacrificing context-specific reasoning. In contrast, the standard deviation matrix (right panel) captures the variability of each weight across individual instances, highlighting the adaptive behavior of Neural-FCM. Such behavior helps explain the improved classification accuracy observed in CpO architectures, particularly in tasks where subtle input shifts lead to meaningful output changes. It is important to note that the identified influence patterns in the weight matrices, while interpretable, do not imply confirmed causal relationships and cannot be interpreted as definitive proof of causality without expert-defined constraints or interventional validation.

6.2 Limitations and future directions

The experiments initially aimed to identify optimal iteration and sigmoid slope parameters, which are commonly regulated within FCM implementations. The findings align with previous research and do not provide any new outcomes, confirming superior performance for single-iteration reasoning, analogous to the operational principles of feed-forward ANNs. This implies that the decision is influenced solely by the weighted connections between input and output nodes during the first iteration, as states in input concept states are not updated through interconnections. As a result, the learning problem is simplified, requiring fewer steps to converge. Future work could address the number of FCM iterations tuning, by incorporating the selection of the number of iterations into the ANN architecture as an additional predictive output head.

Regarding computational cost, generating a distinct weight matrix for each input introduces additional memory and runtime overhead compared to traditional static FCMs. To assess this, we report the average time per epoch, and the number of training epochs required for convergence across all datasets (Tables 4 and 5), offering an empirical perspective often missing in related work. While a formal complexity analysis is beyond the scope of this paper, these metrics provide practical insight into the model's runtime behavior. It is also worth noting that the dimensions of the generated weight matrices remain modest. For instance, in the Breast Cancer dataset, the Neural-FCM model outputs a 32×32 matrix per instance, a scale significantly smaller than typical convolutional neural network (CNN) applications in computer vision [47], which often involve input tensors of size $224 \times 224 \times 3$ and process millions of samples in modest times. Given that Neural-FCM operates on low-dimensional, tabular data, the computational demands remain manageable, especially for modern GPU or CPU setups. Thus, although per-instance modeling incurs some cost, we argue that the impact is negligible and does not preclude real-time processing or practical use in small- to medium-scale applications. Nevertheless, future work should focus on a more comprehensive assessment of computation complexity and on more complex datasets that usually low-level FCMs are struggling with.

Neural-FCM, as a pure data-driven approach, inherits its known limitations in FCM learning literature [3]. Traditional FCMs derive their weight matrices from expert knowledge, while data-driven methods often produce matrices that prioritize fitting the data rather than representing genuine causal relationships. Consequently, these interconnections may serve as cost-minimizing artifacts than true causality indicators. A promising direction for improvement that is examined in our learning method is to incorporate structural constraints into the learning process. As demonstrated in Eqs. 13 and 14, the loss function can be modified to penalize undesirable connections. Future work could involve hybrid strategies where domain experts provide guidance in the form of soft constraints or allowable weight ranges, preserving causal semantics while maintaining predictive power.

Another important consideration is the model's alignment with goals in eXplainable Artificial Intelligence (XAI) [5]. While FCMs are traditionally viewed as transparent models, the integration of a deep ANN as a decoder introduces an opaque component in the reasoning chain. This added complexity raises questions about the extent to which Neural-FCM retains its interpretability. While the matrix generation process in Neural-FCM involves a deep network, it should be noted that previous population-based learning approaches are similarly non-transparent [72], relying on stochastic operations without interpretable traceability. Thus, Neural-FCM maintains the interpretability of the decision-making process by preserving the standard FCM structure, even if the learning process itself remains a black-box, as is common across FCM weight matrix optimization methods. In addition, advances in XAI, particularly in methods for explaining deep models could tackle such challenges and potentially reveal additional avenues for improving Neural-FCM.

7 Summary

This study presents Neural-FCM, a novel method for learning and structure modeling of Fuzzy Cognitive Map (FCM) classifiers that addresses key limitations in the current state-of-the-art. Existing data-driven FCM approaches often fail to achieve high predictive accuracy without sacrificing interpretability, as many recent methods modify core FCM structures to compete with black-box models. In contrast, Neural-FCM integrates principles from artificial neural networks (ANNs) to enhance FCM learning and structure modeling without altering its low-level reasoning mechanism. Neural-FCM uses a hybrid ANN decoder architecture, comprising fully connected and convolutional layers, to transform each input instance into an instance-specific weight matrix. The training process incorporates FCM inference prior to loss evaluation using categorical cross-entropy, thereby encouraging the network to generate meaningful and interpretable weight matrices. The method was evaluated on multiple public classification datasets as well as one proprietary medical dataset to establish that the proposed method generalizes across domains, a main objective of this work. The model was implemented using widely adopted prototyping tools, including Python and TensorFlow, to encourage reproducibility and adoption.

Experimental results show that Neural-FCM consistently enhances FCM classification performance. In comparative evaluations, the model achieved up to 34% improvement in accuracy over population-based FCM learning methods, prominently enhancing FCM classifier architectures that represent each class as a distinct output concept. Neural-FCM also performs competitively against recent gradient-based learning approaches and traditional machine learning classifiers. Importantly, Neural-FCM preserves the foundational working principles of FCMs, which are often compromised when FCMs are modeled using ANN frameworks. A key contribution of this work is the introduction of dynamic weight matrices, which enable structural adaptation at the instance level. This means that the polarity and magnitude of inter-concept weights can vary based on the input data, allowing more context-aware and precise reasoning. This behavior mirrors and enhances the objectives of Fuzzy Grey Cognitive Maps (FGCMs), offering the ability to identify and adjust weight ranges automatically during inference.

Several limitations also suggest directions for future work. These include the use of explainable AI (XAI) techniques to better interpret the deep model components and the incorporation of expert knowledge through constraints to reinforce causality in learned structures. Additionally, the Neural-FCM architecture presented here was empirically designed, indicating the potential for further improvement through neural network tuning and the adoption of advanced architectural features such as residual connections, regularization techniques, or attention mechanisms.

Author contributions Conceptualization: Theodoros Tziolas; Methodology: Theodoros Tziolas; Formal analysis and investigation: Theodoros Tziolas; Writing - original draft preparation: Theodoros Tziolas; Writing - review and editing: Konstantinos Papageorgiou, Ioannis Apostolopoulos, Elpiniki Papageorgiou; Resources: Elpiniki Papageorgiou; Supervision: Elpiniki Papageorgiou.

Funding Open access funding provided by HEAL-Link Greece. The authors did not support from any organization for the submitted work.

Data availability This work employed both public and private datasets. The public datasets are properly cited in text along with the UCI repository that comprises them. The private dataset can be provided by the authors on reasonable request.

Declarations

Financial interests The authors have no relevant financial or non-financial interests to disclose.

Conflicts of interest The authors declare no conflict of interest.

References

1. Kosko B (1986) Fuzzy cognitive maps. Int J Man Mach Stud 24:65–75. https://doi.org/10.1016/S0020-7373(86)80040-2
2. Papageorgiou EI, Salmeron JL (2013) A review of fuzzy cognitive maps research during the last decade. IEEE Trans Fuzzy Syst 21:66–79. https://doi.org/10.1109/TFUZZ.2012.2201727
3. Nápoles G, Leon Espinosa M, Grau I, Vanhoof K, Bello R (2018) Fuzzy cognitive maps based models for pattern classification: advances and challenges. Soft computing based optimization and decision models: to commemorate the 65th birthday of professor José. Luis Curro Verdegay 83–98. https://doi.org/10.1007/978-3-319-64286-4_5
4. Papageorgiou EI (2013) Fuzzy cognitive maps for applied sciences and engineering: from fundamentals to extensions and learning algorithms. Intelligent systems reference library (ISRL, volume 54). Springer Science & Business Media. https://doi.org/10.1007/978-3-642-39739-4
5. Apostolopoulos ID, Groumpos PP (2023) Fuzzy cognitive maps: their role in explainable artificial intelligence. Appl Sci 13. https://doi.org/10.3390/app13063412
6. Thiebes S, Lins S, Sunyaev A (2021) Trustworthy artificial intelligence. Electron Markets 31:447–464. https://doi.org/10.1007/s12525-020-00441-4
7. Amirkhani A, Papageorgiou EI, Mohseni A, Mosavi MR (2017) A review of fuzzy cognitive maps in medicine: taxonomy, methods, and applications. Comput Methods Programs Biomed 142:129–145. https://doi.org/10.1016/j.cmpb.2017.02.021
8. Papageorgiou EI, Markinos AT, Gemtos TA (2010) Soft computing technique of fuzzy cognitive maps to connect yield defining parameters with yield in cotton crop production in central Greece as a basis for a decision support system for precision agriculture application. In: Glykas M (ed) Fuzzy cognitive maps: advances in theory, methodologies, tools and applications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 325–362
9. Alipour M, Hafezi R, Papageorgiou E, Hafezi M, Alipour M (2019) Characteristics and scenarios of solar energy development

in iran: fuzzy cognitive map-based approach. Renew Sustain Energy Rev 116:109410. https://doi.org/10.1016/j.rser.2019.109410

10. Papageorgiou EI (2012) Learning algorithms for fuzzy cognitive maps—A review study. IEEE transactions on systems, man, and cybernetics, part C (Applications and Reviews). 42:150–163. https://doi.org/10.1109/TSMCC.2011.2138694
11. Felix G, Nápoles G, Falcon R, Froelich W, Vanhoof K, Bello R (2019) A review on methods and software for fuzzy cognitive maps. Artif Intell Rev 52:1707–1737. https://doi.org/10.1007/s10462-017-9575-1
12. Dickerson JA, Kosko B (1993) Virtual worlds as fuzzy cognitive maps. In: Proceedings of IEEE Virtual Reality Annual International Symposium. pp. 471–477
13. Papageorgiou EI, Stylios CD, Groumpos PP (2004) Active Hebbian learning algorithm to train fuzzy cognitive maps. Int J Approximate Reasoning 37:219–249. https://doi.org/10.1016/j.ijar.2004.01.001
14. Li S-J, Shen R-M (2004) Fuzzy cognitive map learning based on improved nonlinear Hebbian rule. In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), pp 2301–2306 vol.4
15. Stach W, Kurgan L, Pedrycz W (2008) Data-driven nonlinear Hebbian learning method for fuzzy cognitive maps. In: 2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence), pp 1975–1981
16. Slowik A, Kwasnicka H (2020) Evolutionary algorithms and their applications to engineering problems. Neural Comput Appl 32:12363–12379. https://doi.org/10.1007/s00521-020-04832-8
17. Stach W, Kurgan L, Pedrycz W, Reformat M (2005) Genetic learning of fuzzy cognitive maps. Fuzzy Sets Syst 153:371–401. https://doi.org/10.1016/j.fss.2005.01.009
18. Papageorgiou EI, Poczęta K, Laspidou C (2015) Application of fuzzy cognitive maps to water demand prediction. In: 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp 1–8
19. Liang W, Zhang Y, Liu X, Yin H, Wang J, Yang Y (2022) Towards improved multifactorial particle swarm optimization learning of fuzzy cognitive maps: A case study on air quality prediction. Appl Soft Comput 130:109708. https://doi.org/10.1016/j.asoc.2022.109708
20. Yesil E, Ozturk C, Dodurka MF, Sakalli A (2013) Fuzzy cognitive maps learning using artificial bee colony optimization. In: 2013 IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 1–8. IEEE
21. Mendonça M, Palácios RH, Papageorgiou EI, de Souza LB (2020) Multi-robot exploration using dynamic fuzzy cognitive maps and ant colony optimization. In: 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–8. IEEE
22. Papakostas GA, Koulouriotis DE, Polydoros AS, Tourassis VD (2012) Towards hebbian learning of fuzzy cognitive maps in pattern classification problems. Expert Syst Appl 39:10620–10629. https://doi.org/10.1016/j.eswa.2012.02.148
23. Madeiro SS, Zuben FJV (2012) Gradient-based algorithms for the automatic construction of fuzzy cognitive maps. In: 2012 11th International Conference on Machine Learning and Applications. pp 344–349
24. Gregor M, Groumpos PP (2013) Training fuzzy cognitive maps using gradient-based supervised learning. In: Papadopoulos H, Andreou AS, Iliadis L, Maglogiannis I (eds) Artificial intelligence applications and innovations. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 547–556
25. Szwed P (2021) Classification and feature transformation with fuzzy cognitive maps. Appl Soft Comput 105:107271. https://doi.org/10.1016/j.asoc.2021.107271
26. Dong S, Wang P, Abbas K (2021) A survey on deep learning and its applications. Comput Sci Rev 40:100379. https://doi.org/10.1016/j.cosrev.2021.100379
27. Nápoles G, Ranković N, Salgueiro Y (2023) On the interpretability of fuzzy cognitive maps. Knowl Based Syst 281:111078. https://doi.org/10.1016/j.knosys.2023.111078
28. Papakostas GA, Koulouriotis DE (2010) Classifying patterns using fuzzy cognitive maps. In: Glykas M (ed) Fuzzy cognitive maps: advances in theory, methodologies, tools and applications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 291–306
29. Sabahi S, Stanfield PM (2022) Neural network based fuzzy cognitive map. Expert Syst Appl 204:117567. https://doi.org/10.1016/j.eswa.2022.117567
30. LeCun Y, Bengio Y, Hinton G (2015) Deep learn. Nat 521:436–444
31. Salmeron JL (2010) Modelling grey uncertainty with fuzzy grey cognitive maps. Expert Syst Appl 37:7581–7588. https://doi.org/10.1016/j.eswa.2010.04.085
32. Developers T (2022) TensorFlow. Zenodo
33. Mkhitaryan S, Giabbanelli P, Wozniak MK, Nápoles G, De Vries N, Crutzen R (2022) FCMpy: a python module for constructing and analyzing fuzzy cognitive maps. PeerJ Comput Sci 8:e1078. https://doi.org/10.7717/peerj-cs.1078/supp-1
34. Froelich W (2017) Towards improving the efficiency of the fuzzy cognitive map classifier. Neurocomputing 232:83–93. https://doi.org/10.1016/j.neucom.2016.11.059
35. Stylios CD, Groumpos PP (2004) Modeling complex systems using fuzzy cognitive maps. IEEE Trans Syst Man Cybernetics - Part A: Syst Hum 34:155–162. https://doi.org/10.1109/TSMCA.2003.818878
36. Tziolas T, Papageorgiou K, Theodosiou T, Rapti A, Mastos T, Papadopoulos A, Papageorgiou E (2023) Root cause analysis with fuzzy cognitive maps and correlation coefficient. In: Massanet S, Montes S, Ruiz-Aguilera D, González-Hidalgo M (eds) Fuzzy logic and technology, and aggregation operators. Springer Nature Switzerland, Cham, pp 174–184
37. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, Río JF, del, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020) Array programming with numpy. Nature 585:357–362. https://doi.org/10.1038/s41586-020-2649-2
38. Martín Abadi A, Agarwal P, Barham E, Brevdo Z, Chen C, Citro GS, Corrado A, Davis J, Dean M, Devin S, Ghemawat I, Goodfellow A, Harp G, Irving M, Isard, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/
39. Kottas TL, Boutalis YS, Christodoulou MA (2010) Fuzzy cognitive networks: adaptive network Estimation and control paradigms. In: Glykas M (ed) Fuzzy cognitive maps: advances in theory, methodologies, tools and applications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 89–134
40. Papakostas GA, Boutalis YS, Koulouriotis DE, Mertzios BG (2008) Fuzzy cognitive maps for pattern recognition applications. Int J Pattern Recognit Artif Intell 22:1461–1486. https://doi.org/10.1142/S0218001408006910
41. Yu T, Gan Q, Feng G, Han G (2022) A new fuzzy cognitive maps classifier based on capsule network. Knowl Based Syst 250:108950. https://doi.org/10.1016/j.knosys.2022.108950
42. Fisher RA (1936) Iris [Dataset]. UCI Machine learning repository. https://doi.org/10.24432/C56C76

43. Wang J, Peng Z, Wang X, Li C, Wu J (2021) Deep fuzzy cognitive maps for interpretable multivariate time series prediction. IEEE Trans Fuzzy Syst 29:2647–2660. https://doi.org/10.1109/TFUZZ.2020.3005293
44. Froelich W, Salmeron JL (2014) Evolutionary learning of fuzzy grey cognitive maps for the forecasting of multivariate, interval-valued time series. Int J Approximate Reasoning 55:1319–1335. https://doi.org/10.1016/j.ijar.2014.02.006
45. Nápoles G, Mosquera C, Falcon R, Grau I, Bello R, Vanhoof K (2018) Fuzzy-rough cognitive networks. Neural Netw 97:19–27. https://doi.org/10.1016/j.neunet.2017.08.007
46. Sovatzidi G, Vasilakakis MD, Iakovidis DK (2022) Automatic fuzzy graph construction for interpretable image classification. In: 2022 IEEE International Conference on Image Processing (ICIP), pp 3743–3747
47. Tziolas T, Papageorgiou K, Feleki A, Theodosiou T, Rapti K, Papageorgiou E, Pantoja S, Cuinas A (2024) Deep fuzzy cognitive maps for defect inspection in antenna assembly. Procedia Comput Sci 232:97–106. https://doi.org/10.1016/j.procs.2024.01.010
48. Feleki A, Apostolopoulos ID, Moustakidis S, Papageorgiou EI, Papathanasiou N, Apostolopoulos D, Papandrianos N (2023) Explainable deep fuzzy cognitive map diagnosis of coronary artery disease: integrating myocardial perfusion imaging, clinical data, and natural language insights. Appl Sci 13. https://doi.org/10.3390/app132111953
49. Basheer IA, Hajmeer M (2000) Artificial neural networks: fundamentals, computing, design, and application. J Microbiol Methods 43:3–31. https://doi.org/10.1016/S0167-7012(00)00201-3
50. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
51. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 770–778
52. Chauvin Y, Rumelhart DE (eds) (1995) Backpropagation: theory, architectures, and applications. L. Erlbaum Associates Inc., Mahwah
53. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
54. Wang Q, Ma Y, Zhao K, Tian Y (2022) A comprehensive survey of loss functions in machine learning. Annals Data Sci 9:187–212. https://doi.org/10.1007/s40745-020-00253-5
55. Hara K, Saito D, Shouno H (2015) Analysis of function of rectified linear unit used in deep learning. In: 2015 International Joint Conference on Neural Networks (IJCNN). pp 1–8
56. Taye MM (2023) Understanding of machine learning with deep learning: architectures, workflow, applications and future directions. Computers 12. https://doi.org/10.3390/computers12050091
57. Bank D, Koenigstein N, Giryes R (2023) Autoencoders. In: Rokach L, Maimon O, Shmueli E (eds) Machine learning for data science handbook: data mining and knowledge discovery handbook. Springer International Publishing, Cham, pp 353–374
58. Feuerriegel S, Hartmann J, Janiesch C, Zschech P, Generative AI, Business (2024) Inform Syst Eng 66:111–126. https://doi.org/10.1007/s12599-023-00834-7
59. Harrison D (2021) A brief introduction to automatic differentiation for machine learning. arXiv preprint arXiv:2110.06209
60. Dheeru D, Karra Taniskidou E (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml
61. Feleki A, Apostolopoulos ID, Papageorgiou K, Papageorgiou EI, Apostolopoulos DJ, Papandrianos NI (2023) A fuzzy cognitive map learning approach for coronary artery disease diagnosis in nuclear medicine. In: Massanet S, Montes S, Ruiz-Aguilera D, González-Hidalgo M (eds) Fuzzy logic and technology, and aggregation operators. Springer Nature Switzerland, Cham, pp 14–25
62. Kayaer K, Yildirim T et al (2003) Medical diagnosis on Pima Indian diabetes using general regression neural networks. In: Proceedings of the international conference on artificial neural networks and neural information processing (ICANN/ICONIP). p 184
63. Coomans D, Broeckaert I, Jonckheer M, Massart D (1983) Comparison of multivariate discrimination techniques for clinical data—application to the thyroid functional state. Methods Inf Med 22:93–101
64. Street WN, Wolberg WH, Mangasarian OL (1993) Nuclear feature extraction for breast tumor diagnosis. In: Biomedical image processing and biomedical visualization. pp 861–870. SPIE
65. Fisher RA (1936) The use of multiple measurements in taxonomic problems. Annals Eugenics 7:179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x
66. Forina M, Leardi R, Armanino C, Lanteri S (1991) An extendible package for data exploration, classification and correlation. Institute of pharmaceutical and food analisys and technologies. Via Brigata Salerno 16147. https://doi.org/10.1002/cem.1180040210
67. Hofmann H (1994) Statlog (German Credit Data)
68. Hunter JD (2007) Matplotlib: a 2D graphics environment. Comput Sci Eng 9:90–95. https://doi.org/10.1109/MCSE.2007.55
69. team T (2020) Pandas development: pandas-dev/pandas: pandas. https://doi.org/10.5281/zenodo.3509134
70. Samaras A-D, Moustakidis S, Apostolopoulos ID, Papageorgiou E, Papandrianos N (2023) Uncovering the black box of coronary artery disease diagnosis: the significance of explainability in predictive models. Appl Sci 13. https://doi.org/10.3390/app13148120
71. Fernández-Delgado M, Cernadas E, Barro S, Amorim D (2014) Do we need hundreds of classifiers to solve real world classification problems? J Mach Learn Res 15:3133–3181
72. Tyrovolas M, Liang XS, Stylios C (2023) Information flow-based fuzzy cognitive maps with enhanced interpretability. Granul Comput 8:2021–2038. https://doi.org/10.1007/s41066-023-00417-7