Recurrence-Aware Long-Term Cognitive Network for Explainable Pattern Classification

Gonzalo Nápoles, Yamisleydi Salgueiro, Isel Grau, and Maikel Leon Espinosa

Abstract—Machine-learning solutions for pattern classification problems are nowadays widely deployed in society and industry. However, the lack of transparency and accountability of most accurate models often hinders their safe use. Thus, there is a clear need for developing explainable artificial intelligence mechanisms. There exist model-agnostic methods that summarize feature contributions, but their interpretability is limited to predictions made by black-box models. An open challenge is to develop models that have intrinsic interpretability and produce their own explanations, even for classes of models that are traditionally considered black boxes like (recurrent) neural networks. In this article, we propose a long-term cognitive network (LTCN) for interpretable pattern classification of structured data. Our method brings its own mechanism for providing explanations by quantifying the relevance of each feature in the decision process. For supporting the interpretability without affecting the performance, the model incorporates more flexibility through a quasi-nonlinear reasoning rule that allows controlling nonlinearity. Besides, we propose a recurrence-aware decision model that evades the issues posed by the unique fixed point while introducing a deterministic learning algorithm to compute the tunable parameters. The simulations show that our interpretable model obtains competitive results when compared to state-of-the-art white and black-box models.

Index Terms—Explainable artificial intelligence, long-term cognitive networks (LTCNs), machine-learning interpretability, recurrent neural networks.

I. INTRODUCTION

Pattern recognition techniques aim to find regularities in data stored in databases or produced by signals, processes, etc. [1]. Due to the abundance of data and the increase in computational power, machine-learning algorithms have a prominent role in pattern recognition applications. Overall, pattern classification focuses on assigning a label or category to each data point. Ground truth information is necessary for learning such a mapping from input data to labels. Several machine learning algorithms have proven successful in creating classification models with high accuracy, such as support vector machines (SVMs), random forests (RFs), ensembles, or (deep) neural networks.

However, the ubiquitousness of machine-learning algorithms deployed in today’s society has raised concerns about their accountability and transparency [2], [3]. For most high-stakes decision problems having an accurate model is not sufficient; some degree of interpretability is also needed. As stated in [4], when users perceive that an algorithm is fairer, more accountable, transparent, and explainable, they see it as a more trustworthy and useful resource. The form of this interpretability, either as a global holistic view of the model or local explanations over particular predictions, depends on the audience of the model and the domain [5].

Global interpretability can be obtained by using intrinsically interpretable machine-learning techniques, which rely on their levels of transparency [6]. Linear or logistic regression (LR) models are the simplest interpretable predictors producing explanations about the role of the features [7]. Decision trees (DTs) and decision lists are generally accurate predictors that can provide intrinsic interpretability when the structure is kept on a simulatable size [8]. On the other hand, model-agnostic post-hoc explanation methods compute local explanations from the black-box predictions to preserve their accuracy. For example, the SHAP approximation [9] of Shapley values explains the role of the features in the prediction of a given instance. Another example is the local surrogate model LIME [10], which describes the vicinity of the prediction with a linear regression, leveraging its intrinsic interpretability but limited to a particular region of the domain.

Overall, model-agnostic post-hoc methods generate explanations that are local or limited to feature attribution. Explanations provided by intrinsically interpretable models are derived from their structure and easily mappable to the problem domain. Rudin [11] accentuated the difference between explaining the predictions of a black box and the inherent explanations of the transparent models. Moreover, they argue that the community should focus on the latter to avoid unreliable explanations and potentiate explanations that are faithful to what the model actually computes. In [12], the
effects of anthropomorphic explanations are studied, and how
certain recommendations provided by systems afford human-
ness, which then influence trust and emotional assurance.
Although we consider that local explanations can be useful
for some domains, we see the development of accurate models
with inherent interpretability as an open challenge.

A type of recurrent neural network with high potential for
intrinsic interpretability is fuzzy cognitive maps (FCMs) [13].
FCMs allow modeling complex systems in terms of causal
relationships and well-defined concepts. In these networks, the
experts should provide the concepts defining the system and
the weights connecting such concepts, although the weights
could also be computed from data using learning algorithms.
In general, neural-networks operate like black boxes, where
hidden neurons and connections do not involve any clear
meaning for the problem itself. In contrast, neural con-
cepts in FCM-based models and their connections have a
precise meaning for the system under analysis and can help
explain why a solution is suitable for a given problem. FCMs
have been extensively applied to modeling complex systems
from engineering, environmental sciences, behavioral sciences,
medicine, business, and other domains [14].

However, while FCM-based models have proven effective
in scenario simulation and time-series forecasting, their
performance on pattern classification problems is arguable
(Section III will revise prominent models reported in the
literature). There are several reasons explaining the moderate
performance of FCM-based classifiers. First, the network
topology depends on the problem domain since hidden neu-
rons are not allowed. Second, if the network converges to a
unique fixed-point attractor, then the model will be able to rec-
ognize only one decision class (not necessarily the majority
one). Nápoles et al. [15] proved that, under some properties of
the weight matrix, an FCM with no input neurons converges
to a single attractor regardless the initial concept values. As an
example, Nápoles et al. [15] showed several FCMs modeling
the classification of drug resistance in HIV protein sequences
that converge to a unique fixed point, therefore only predicting
the “susceptible” decision class, which is the minority one.
Third, both the neurons’ activation values and the (causal)
weights are confined to a closed interval, thus limiting the
coverage of the activation space [16]. Finally, there is a lack
of learning algorithms with a strong mathematical foundation.

To tackle the last two issues, Nápoles et al. [17] proposed
the long-term cognitive networks (LTCNs). In this FCM-like
model, the weights are not constrained to any specific interval,
and the tunable parameters are computed using a nonsynaptic
backpropagation algorithm. However, LTCNs will not neces-
sarily produce good prediction rates in pattern classification
problems either. On the one hand, the nonsynaptic learning
method assumes that the domain expert is able to define the
weight matrix. On the other hand, the network’s convergence
to a unique fixed-point continues to be a serious problem.

In this article, we propose an LTCN-based model for inter-
pretable pattern classification in structured data, that is, tabular
datasets with well-defined features. This model solves the
remaining LTCNs’ issues while preserving the network’s inter-
pretability as much as possible. Overall, our proposal brings
four main theoretical contributions. First, we introduce a para-
metric quasi-nonlinear reasoning rule that allows controlling
nonlinearity. Second, we present a recurrence-aware decision
model for multiclass pattern classification. This decision model
is not affected by the unique fixed point when the network
converges. Third, we propose a two-step learning procedure
to adjust the weights in a deterministic way. The first step is
unsupervised and computes the weights connecting the inner
neurons (the ones mapping the problem features). The second
step is supervised and computes the weights connecting the
inner neurons with the decision ones in the recurrence-aware
model. Finally, we describe a measure to quantify the rele-
vance of each feature in the decision process as a mechanism
to provide explanations that are directly extracted from the
model.

The remainder of this article is organized as follows. Section II
presents foundations of the LTCN model, starting from
the classic FCM formalism. Section III revises a selection
of prominent pieces of research devoted to FCM-based clas-
sifiers. Section IV encloses the contributions of this article,
which include the quasi-nonlinear reasoning, the recurrence-
aware architecture, its learning algorithm, and the feature
relevance measure. Section V conducts extensive numerical
simulations, whereas Section VI presents some concluding
remarks.

II. LONG-TERM COGNITIVE NETWORKS
The LTCN model has its roots in the FCMs, which were
originally introduced in [13] as a knowledge-based method-
ology for modeling complex systems. From a connectionist
viewpoint, FCMs can be seen as recurrent neural-networks
consisting of neural concepts and signed weighted connec-
tions. Neural concepts represent variables, states, entities
related to the physical system under investigation. The signed
weight associated with each connection denotes the strength
of the causality between the corresponding neurons. Causal
relations are quantified in the [−1, 1] or [0, 1] depending on the
nonlinear transfer function attached to each neuron.

In each iteration, an FCM model produces an activation vec-
tor $A_k^{(t)} = [a_k^{(t)}], \ldots, a_k^{(t)}], a_k^{(M)}]$, where $a_k^{(t)}$ is the activation
value of the $i$-th neural entity in the $t$-th iteration, given the $k$th
initial stimulus. Equation (1) displays the recurrent reasoning
rule of this model

$$A_k^{(t)} = f(A_k^{(t-1)} W)$$

where $M$ denotes the number of neurons and $W_{M \times M}$ is
the weight matrix such that $w_{ij}$ represents the weight connect-
ing the $C_j$ and $C_i$ neurons, while $f(\cdot)$ is the transfer function
used to keep the neurons’ activation values within the allowed
activation interval.

Equation (2) presents another reasoning rule that takes into
account both the neuron’s previous activation value and the
states of connected neurons

$$A_k^{(t)} = f(A_k^{(t-1)} W + A_k^{(t-1)})$$


The neurons’ activation values are iteratively updated until 1) the network converges to a fixed-point attractor or 2) a maximal number of iterations \( T \) is reached. These states can be defined as follows.

1) **Fixed Point** (\( 3\alpha = \{1, 2, \ldots, (T-1)\} : A_k^{(t+1)} = A_k^{(t)} \forall k \geq t_a \)): The network produces the same state after \( t_a \), so \( A_k^{(t_a)} = A_k^{(t_a+1)} = A_k^{(t_a+2)} = \cdots = A_k^{(T)} \). The fixed point may be unique, which means that the network will produce the same state regardless of the neuron’s initial values.

2) **Limit Cycle** (\( 3\alpha, P \in \{1, 2, \ldots, (T-1)\} : A_k^{(t+P)} = A_k^{(t)} \forall k \geq t_a \)): The network produces the same state periodically after the period \( P \), so \( A_k^{(t_a)} = A_k^{(t_a+p)} = A_k^{(t_a+2p)} = \cdots = A_k^{(t_a+Pp)} \) where \( t_a + pP \leq T \), such that \( j \in \{1, 2, \ldots, (T-1)\} \).

3) **Chaos**: The network continues to produce different states for successive iterations.

These neural networks have proven effective for modeling complex systems, but they have been linked to serious misconceptions and theoretical issues [18]. First, the fuzzy aspect in these models is ill-defined. The reasoning process of FCMs involves no fuzzy operations whatsoever, as one would expect. Second, FCM-based models derived from historical data (using supervised learning algorithms) can hardly be considered causal. Instead, weights resulting from data-driven construction models should be interpreted as coefficients in a regression model. Last but not least, the constraint that \( w_{ji} \in [-1, 1] \) greatly hinders the predictive power of FCM-based models [16]. Perhaps that is the reason for their scarce popularity when compared to other recurrent neural networks.

The LTCN model was introduced in [17] to overcome these issues. In short, LTCNs are neither causal nor fuzzy, and their weights can take values in the real domain. The main similarities between FCMs and LTCNs are that they do not allow for hidden neurons (to retain the model’s interpretability) and share the same recurrent reasoning rule. These features make these methods somewhat similar at first sight, but the semantic differences make them quite distinct in both theory and practice.

### III. Literature Review

The literature reports several works related to the use of FCM-based models in pattern classification problems. One of the first attempts at incorporating FCMs in pattern classification applications is found in [19]. In that paper, three classification models were proposed, but not all of them achieved good results.

The mentioned study continued with [20] where the performances of the FCM classifiers were studied more in-depth. The authors investigated FCM-based classifiers’ performance when adjusting the appropriate set of parameters, such as the transfer function, the reasoning rule, and the network topology. The newly introduced classifiers presented better prediction capabilities. However, these sophisticated models are no longer interpretable since they were hybridized with black boxes.

The study in [21] presented an approach that translates the reasoning mechanism of traditional FCMs to a set of fuzzy IF-THEN rules. Each fuzzy rule is defined as a fuzzy set concerning the summation of weighted membership grades of input linguistic terms. The impacts of fuzzy rules on output linguistic parts are transferred along with fuzzy weights and quantified by mutual subsethood. The consequent parts are then defuzzified by standard-volume-based centroid defuzzification. Finally, by describing each output as a linear combination of the defuzzified consequent parts, the model takes advantage of the mapping capability offered by the consequent parts to approximate the desired outputs. Recent studies, as found on [22], continued on the idea of creating models based on neuro-fuzzy inference systems with the advantage of obtaining the weights of connecting links to adjust the parameters of fuzzy rules. In other words, for determining the rules and obtaining the weights, in addition to the knowledge of experts, the model also exploits the existing data to adjust the inference system’s parameters.

Separately, the need to apply learning algorithms for training FCMs was discussed in [23] and [24]. Hebbian-like algorithms aimed to adjust the weights between the neurons of the FCM classifier so that it can converge to the desired state. The works in [25]–[29] reported the use of different paradigms deviating from Hebbian-based learners and employing other approaches (e.g., based on evolutionary algorithms).

In a different direction, the work in [30] extended the use of FCMs with the creation of rough cognitive networks (RCNs) as granular classifiers stemming from the hybridization of FCMs and Rough Set Theory. Such cognitive neural-networks attempted to quantify the impact of rough granular constructs over each decision class for a problem at hand. RCNs have shown substantial improvements in solving different classification problems, but the model reported some sensitivity to the similarity threshold upon which the rough information granules are built. Moreover, fuzzy-RCNs (FRCNs) [31], [32] improved on this limitation and obtained results as accurate as of the most successful black-box models with the main advantage of being able to elucidate its decision process using inclusion degrees and causal relations.

There is little doubt that FCMs have been quite useful for designing knowledge-based systems involving experts, with their intrinsic interpretability being a pivotal feature. Overall, the development of FCM-based models and classifiers is on the rise (as seen in Fig. 1). If only FCM-based models were to be as effective as black-box models, then we would have a reasoning model able to provide explanations without using any post-hoc method.

Wrapping up, most of the seen algorithms created for classification purposes dive two fold as follows: 1) low level (where neurons correspond to system variables) and 2) high level (where neurons correspond to information granules). The model to be presented in this article belongs to the first class. Many of the limitations observed correspond to open problems [33], for example, the convergence to a unique fixed point might mean the recognition of only one class. Also, a considerable number of learning algorithms for FCMs reported in the literature are meta-heuristic-based, therefore likely to suffer from both speed and convergence problems, for example, easily converge to local optima. While the literature review shows...
substantial progress on FCMs and their use for solving pattern classification problems, the above-mentioned drawbacks serve as a motivation to develop our proposal.

IV. RECURRENCE-AWARE LTCN-BASED CLASSIFIER

This section presents a recurrence-aware LTCN-based model that allows for explainable pattern classification. Section IV-A introduces a quasi-nonlinear reasoning rule that uses a parameter to control the nonlinearity degree of the recurrent reasoning process. In Section IV-B, we explain our model’s architecture and detail a two-step learning procedure to estimate the tunable parameters in a deterministic way. Finally, we introduce a measure to quantify the relevance of each problem feature in the classifier’s decision process.

A. Quasi-Nonlinear Reasoning Model

First, we propose a new reasoning rule that introduces a nonlinearity coefficient $\phi \in [0, 1]$ controlling the extent to which the model will take into account the value produced by the transfer function over the neuron’s initial activation value. Equation (3) shows this model

$$A_k^{(t)} = \phi f(A_k^{(t-1)}W + B) + (1 - \phi)A_k^{(0)}$$

where $B_{1 \times M}$ denotes the bias matrix, which can be understood as the amount of external information impacting the neuron’s state (i.e., what cannot be explained through the neuronal concepts describing the problem domain). The matrices $B$ and $W$ will be computed from historical data during the unsupervised learning step.

If $\phi = 1$, then we have a traditional long-term recurrent model such that the neurons’ activation values depend on the states of connected neurons in the previous iteration. If $\phi = 0$, then there will be no recurrence at all, so the model will narrow down to a linear regression with multiple outputs [34]. For the most part, the motivation for this model is that in traditional FCMs, the input is explicitly used to compute neurons’ activation values only in the first iteration. Moreover, some decision models might benefit from a certain degree of linearity.

Fig. 2(a)–(c) depicts the reasoning rules formalized in (1)–(3), respectively. Notice that, unlike the model in Fig. 2(b), our proposal adds the scaled initial activation value
after having transformed the incoming information flow with the transfer function. Failing to do that implies that the additional activation value (either the neuron’s initial or previous activation value) will have a limited impact on the neuron’s outcome after applying the transfer function.

Overall, one can see that the proposed quasi-nonlinear reasoning rule uses the initial activation value to compute the neurons’ states in each iteration. However, such a reasoning rule will not prevent the network from converging to undesirable states. An example of these states is the unique fixed-point attractor since it causes the network to produce the same outputs to any input. This issue will be discussed in the next sections.

B. Network Architecture and Learning

Our recurrence-aware neural classifier involves two building blocks. The first one consists of an LTCN model where each neuron maps a problem variable (feature). The role of this neural block is to capture the dynamics of the system, which can be either fixed-point attractors, cyclic, or chaotic states. Of course, the ideal situation for this model is for each decision class to be associated with a different equilibrium point. However, we will not make any assumptions on the convergence properties of this model. It is worth mentioning that this model will use the reasoning rule in (3), which involves a parameter to control nonlinearity. The second building block connects the inner neurons denoting problem features with the decision neurons. The first neural block will be trained using an unsupervised learning approach, while the second learning step will be supervised.

Equation (4) displays the unsupervised learning rule to compute the $i$th column of the weight matrix $W_{M \times M}$ and the bias $b_i$ connected to the $i$th neuron

$$\begin{bmatrix} b_i \\ W_i \end{bmatrix} = \left( L^\top L \right)^{-1} L^\top f^{-1}(X_i)$$

where $X_i$ is the $i$th column of the training set $X_{K \times M}$ and $L$ is a $K \times (M + 1)$ matrix that results after replacing the $i$th column of $X$ with zeros and concatenating a $K \times 1$ column vector full of ones, while $K$ denotes the number of training instances. Those weights correspond to the coefficients of $M$ regression models such that $X_i$ is deemed the target variable of the $i$th model. We assume that the training set has been normalized and that they are inverse friendly (i.e., they do not cause $f$ to produce $-\infty$, $+\infty$ or any indeterminate behavior). The intuition of this procedure is that we want to approximate the $i$th problem variable given the remaining ones.

The second component of our proposal is a recurrence-aware subnetwork that connects each temporal state $A_k^{(i)}$ with the decision neurons. This subnetwork uses all states resulting from the recurrent reasoning rule for a new instance. Equation (5) shows the model used to compute the activation values of decision neurons

$$\hat{Y}_k = f(H_k^{(T)} R + Q)$$

where $\hat{Y}_k$ is the prediction for the $k$th training instance, $R_{M(T+1) \times N}$ is the outer weight matrix connecting the temporal states (including the initial state) with the $N$ decision neurons, while $Q_{1 \times N}$ is the bias weight vector attached to decision neurons. The matrices $R$ and $Q$ will be computed from historical data during the supervised learning step. In this formulation, $H_k^{(T)}$ is a $1 \times M(T + 1)$ matrix resulting from the recursive horizontal concatenation of the $T + 1$ temporal states

$$H_k^{(T)} = \left(H_k^{(T-1)}|A_k^{(T)}\right)$$

where $H_k^{(0)} = X_k$ while $\left(\cdot|\cdot\right)$ stands for the concatenation operator. Therefore, it holds that

$$H_k^{(T)} = \left(H_k^{(0)}|A_k^{(1)}|A_k^{(2)}|\ldots|A_k^{(T-2)}|A_k^{(T-1)}|A_k^{(T)}\right).$$

For the sake of clarity, we have made an explicit distinction between the inner weights connecting the features, and the outer weights connecting the temporal states with the decision neurons. The same design choice applies to the inner and outer bias weights.

Fig. 3 shows the decision model of traditional FCM-based classifiers and the one proposed in this article. In the first case, the model narrows down to a linear regression where the final state $A_k^{(T)}$ is used as independent variables. However, as stated earlier, if the network converges to the unique fixed-point attractor, $A_k^{(T)}$ will be the same for all initial activation values. If this situation comes to light, the model will produce the same decision class, as seen in [15]. In contrast, the proposed recurrence-aware model uses all temporal states as inputs of a regression model, thus preventing the classifier from producing the same decision class when converging to the unique fixed point. Therefore, our LTCN-based model will focus on the trajectory to the fixed point instead of focusing on the equilibrium point itself.

The last step concerns the supervised learning approach to adjust the tunable parameters. This means that we have to estimate the outer weights (denoted with the matrix $R$) and the outer bias weights attached to decision neurons (denoted with the matrix $Q$). Equation (8) formalizes how to compute both weight matrices in a single step using the following
pseudoinverse learning rule:
\[
\begin{bmatrix} R \\ Q \end{bmatrix} = \left( H^T | 1 \right)^{\frac{1}{2}} f^{-1}(Y)
\]
where 1 denotes a \( K \times 1 \) column vector full of ones, \((\cdot)^{\frac{1}{2}}\) represents the Moore–Penrose pseudoinverse [35], while \( Y_{K \times N} \) is a matrix containing the inverse-friendly one-hot encoding of the decision classes. The Moore–Penrose pseudoinverse is computed using the orthogonal projection method. If any matrix \( H \) has linearly independent columns (\( H^\dagger H \) is nonsingular), then \( H^\dagger = (H^\top H)^{-1}H^\top \). In contrast, if \( H \) has linearly independent rows (\( HH^\top \) is nonsingular), then \( H^\dagger = H^\top (H^\top H)^{-1} \). The former is a left inverse because \( H^\dagger H = I \) and the latter is a right inverse because \( HH^\dagger = I \). Overall, the Moore–Penrose pseudoinverse is one of the best strategies to solve the least square problem when \( H \) is not invertible.

C. Feature Relevance Measure

The advantages of our model include the ability to deal with the unique fixed point, a very low training time (to be illustrated during the experiments), and the possibility of specifying the nonlinearity degree of the reasoning model. Another reason for using FCM-based classifiers is their intrinsic interpretability. However, this does not mean we should expect to understand the model as an entire in the same way we could not easily visualize a large DT. Instead, we should focus on explaining the classifier’s decision process by using its knowledge structures (i.e., inner and outer weights) computed during the unsupervised and supervised learning steps.

In this section, we will introduce a comprehensive measure to estimate the relevance of problem features for the classifier’s decision process. It is worth recalling that our neural system does not include hidden neurons that might hinder its interpretability. Instead, it has meaningful neural concepts having temporal states. In other words, each component in the network has a well-defined meaning for the problem domain being modeled. Such a characteristics is pivotal for designing an intrinsic feature relevance score in our recurrence-aware classifier.

Before presenting our relevance score, we should clarify that neural concepts are not the same as problem features. While the latter is often static entities, the former change their states as the FCM model iterates. However, neural concepts can be used as proxies to quantify the relevance of features in the network’s decision process. Actually, the states of neural concepts produced by the subnetwork resulting from the unsupervised learning step [see (4)] can be seen as approximations of patterns encoded by the features. These hidden patterns emerge from complex correlations and associations in the training data, which can be used to classify the instances.

Equation (9) shows how to calculate the relevance score of a feature from the inner and outer weights
\[
\Omega(f_i) = \sum_{j=1}^{M} |w_{ij}| + \sum_{j=1}^{N} \sum_{t=0}^{T} |f_{ij}^{(t)}|.
\]

The intuition of the relevance score is that important features will be represented by neural concepts having outgoing weights with large absolute values. In this measure, we consider the outgoing weights obtained during both learning phases (the unsupervised one that computes the inner weights and the supervised one that computes the weights connecting the features with the decisions). Notice that we did not include the bias weight matrices \( B \) and \( Q \) in this formula since they do not map to any features and would only capture noise. Likewise, we excluded the neurons’ activation values from this calculation moved by the following assumption. If a neural concept is relevant but takes low activation values, then the pseudoinverse learning rule will compute weights with large absolute values. This is allowed in LTCN-based models since weights are neither constrained nor have any causal connotation. Actually, each weight must be analyzed by following the same statistical assumptions when interpreting coefficients in LR models.

V. NUMERICAL SIMULATIONS

The following section is devoted to the numerical simulations and the ensuing discussion. First, we present the research hypotheses and describe the datasets used for simulation purposes. Second, we study the effect of the nonlinearity parameter on the classifier’s performance. Third, we compare our recurrence-aware classifier against state-of-the-art methods used to cope with structured classification problems. Finally, we illustrate how the feature relevance measure works in a case study.

A. Methodology and Datasets

The experimentation methodology relies on four research hypotheses. First, we claim that the performance of our recurrence-aware LTCN-based classifier is not affected when the network converges to the unique fixed-point attractor. Second, we claim that some problems can benefit from adding some linearity degree to the reasoning rule. Third, we claim that the proposed neural classifier performs comparably to state-of-the-art black boxes. Finally, we claim that our LTCN-based classifier allows quantifying the relevance of each feature without the need for post-hoc procedures.

To investigate our research hypotheses, we adopt 30 pattern classification datasets taken from the study in [36] and a case study concerning cybersecurity (to be presented in the last section). For the sake of convenience, we retained the datasets with numerical features and without missing values. Besides, all features have been normalized using the min-max scaling method. Table I provides relevant information about these datasets, such as the number of instances, the number of features, the number of classes, and the imbalance ratio.

In our experiments, we use Cohen’s kappa coefficient [37] for measuring the classifiers’ performance. This measure estimates the inter-rater agreement for categorical items and ranges in \([-1, 1]\), where \(-1\) indicates no agreement between the prediction and the ground-truth values, 0 means no learning (i.e., random prediction), and 1 means total agreement or perfect performance. While accuracy is considered mainstream, the kappa coefficient is a more robust measure since
it considers the agreement occurring by chance, which is relevant for datasets with class imbalance [38], [39]. However, we will also report the accuracy values for the sake of completeness.

**B. Exploring the Quasi-Nonlinear Model**

This section studies the quasi-nonlinear reasoning model and illustrates the issues caused by the unique fixed-point attractor. For the simulations, we perform 5-fold cross-validation without any hyper-parameter tuning as we want to study the network’s performance when varying nonlinearity and the number of iterations.

Fig. 4 shows the kappa score obtained for each dataset when using the classic decision model [depicted in Fig. 3(a)] and our recurrence-aware decision model [depicted in Fig. 3(b)] for different \( \phi \) values. In this experiment, the maximal number of iterations is set to 20 while neurons use a sigmoid transfer function.

Three conclusions can be drawn from this simulation. First, the traditional decision model of FCM-based classifiers is worse than a regression model (that is to say, \( \phi = 0 \)) due to the convergence issues. It can be easily verified that the traditional FCM-based classifier converges to a unique fixed point, thus recognizing a single decision class (which causes the kappa value to be zero). Actually, the more iterations we perform, the higher the probability of observing such a behavior. Second, our recurrence-aware model is not affected by the unique fixed point regardless of the \( \phi \) value. Third, the models with larger \( \phi \) values have better discriminatory capabilities. While this is expected, we should study the cases in which \( 0 < \phi < 1 \) before jumping to definite conclusions.

Fig. 5 shows the kappa values obtained for selected datasets when varying the \( \phi \) value and the number of iterations in our model. These figures support our second hypothesis: some problems can benefit from adding some linearity degree to the reasoning rule.

In addition, we perform an ablation study to determine the contribution of both learning processes: the unsupervised learning of the problem’s dynamics and the supervised learning of the relations between the features and the decision classes. More specifically, we explored two scenarios: 1) learning the unsupervised part while replacing the supervised learned weights with a random matrix and 2) learning the supervised part while replacing the unsupervised learned weights with a random matrix.

The average performance values for the first scenario are 0.8466 in terms of accuracy and 0.7305 in terms of kappa score. For the second scenario, the accuracy and kappa scores are even lower, equal to 0.2307 and -0.015, respectively. When comparing these results with LTCN including both learning processes (see Table III), the study shows that both parts of the learning process contribute to obtain the performance of our model. Learning the supervised weights has clearly more influence in the overall performance, but this is expected since the second part of the network is trained against the ground truth. However, the unsupervised learning part is fundamental for the intrinsic interpretability of the model.

**C. Exploring the Model’s Predictive Power**

In this section, we contrast the performance of our model against state-of-the-art classifiers (both white and black boxes). The selected algorithms are SVM, LR, DT, RF, and multilayer perceptron (MLP) from the Scikit-learn library, repeated incremental pruning to produce error reduction (RIPPER) implemented in Weka, self-organized maps.
Fig. 5. Kappa values obtained for selected datasets when varying $\phi$ and the number of iterations in our model. It can be noticed that $\phi = 1$ does not necessarily yield the largest kappa score. Moreover, we can conclude that, for these problems, the performance does not increase much after performing five iterations. This happens because the network converges to a fixed-point attractor, so more iterations do not add any new information to the augmented hidden state $H^{(t)}$ depicted in (5). In practice, we just stop the recurrent reasoning process when we notice the network has converged to a fixed-point attractor. In that way, we avoid adding the same columns to the hidden state $H^{(t)}$, which might cause issues when computing the Moore–Penrose pseudoinverse. (a) D4. (b) D6. (c) D7. (d) D11. (e) D16. (f) D17. (g) D18. (h) D21. (i) D25. (j) D27. (k) D28. (l) D30.

(SOMs) [40], [41], FRCNs [32], and Efficient Gradient Boosting Tree (LightGBM) [42]. These algorithms are able to cope with both binary and multiclass problems. In addition, we include a generalized additive model (GAM) [43] and hybrid rule set (HyRS) [44], which can only cope with binary problems.
In our numerical experiments, we perform 5-fold nested cross-validation (i.e., with hyper-parameter tuning using the grid search method). Table II shows the hyper-parameters to be optimized and their values.

Table III shows the average accuracy and kappa values of each classifier on the 30 datasets used in our experiments. The last column portrays the average training time (in seconds) of each model using an optimistic approach based on the entire dataset since we just wanted to measure the training time. Table IV reports the same statistics for binary classifiers. The simulations were run on a laptop with the following features: Core i9 7th generation, ten physical processors, and 16-GB RAM.

The simulation results show that LTCN, SVM, RF, and LightGBM are the best-performing algorithms in terms of kappa scores, closely followed by MLP and FRCN, whereas SOM reports the worst results in our study. This comes with no surprise since these classifiers often report high prediction rates in tabular pattern classification problems. Simultaneously, it can be concluded that our algorithm’s predictive power is far superior to the white boxes included in the study except for RIPPER. This method achieves higher accuracy, but its kappa score is lower than the kappa of our proposal, which means that it is less robust against datasets with high-class imbalance.

It can be argued whether RF can still be considered a black box since we can quantify features’ relevance as a proxy for interpretability. Despite this fact, our model involves other features, such as the possibility of domain experts injecting prior knowledge into the network. In other words, the expert can modify the inner weight matrix to encode rules that have not yet been observed in the data (i.e., the expected revenue increases after adding a new product to the stock). Achieving such a degree of flexibility with RF is not trivial.

In addition, the experiments show that our method is the fastest among the most accurate algorithms. In summary, we can conclude that the proposed recurrence-aware LTCN classifier is as accurate as the black boxes included in the study while also being fairly fast.

D. Exploring the Model’s Interpretability

In this section, we explore the interpretability of our neural model using a case study. The “Phishing case” [45] is a binary classification problem described by 48 features extracted from 5000 phishing webpages and 5000 legitimate webpages and a class variable. Some of the features are: NumDots, UrlLength, NumDash, NumDashInHostname, AtSymbol, TildeSymbol, NumUnderscore, NumQueryComponents, NumAmpersand, NumHash, NumNumericChars, IpAddress, DoubleSlashInPath, PopUpWindow, ImagesOnlyInForm, and UrlLengthRT.

The LTCN classifier reported an accuracy of 97% after performing 5-fold cross-validation. In this experiment, we set the nonlinearity parameter to 0.8 and the number of iterations to 50 such that we can see what happens when we use more iterations than needed.

Fig. 6 shows the behavior of outer weights connecting the temporal states with the decision neurons. In some iterations, the supervised learning algorithm estimates the same weights repeatedly. This happens because the LTCN converges to a fixed point where the network’s state does not change as the iterations continue.

Fig. 7 displays the histogram for the normalized outer weights. It can be noted that weights follow a zero-mean normal distribution, which suggests that the learning algorithm based on the Moore–Penrose pseudoinverse learns sparse weights. This desired behavior is unexpected since we did not consider any regularization component when designing the learning algorithm.
Fig. 6. Behavior of outer weights computed during the supervised learning step. The solid line represents the mean and the shadow indicates the 95% confidence interval. It can be noticed that the weights connecting the inner neurons with the decision neurons do not change after some iterations.

Fig. 7. Distribution of outer weights. The weights follow a zero-mean Gaussian distribution, meaning that the learning method is able to produce sparse weight matrices.

Next, we compute the relevance score attached to each feature using (9). As mentioned, the intuition of this measure is that important features (represented with meaningful neural concepts in the network) will be connected through outgoing weights with large absolute values. Fig. 8 portrays the relevance scores such that $f_i$ denotes the $i$th problem feature. The largest relevance score corresponds to $f_5$ (NumDash). Other important features are: $f_6$ (NumDots), $f_3$ (PathLevel), $f_{14}$ (NumChars), $f_{25}$ (NumSensitiveWords), $f_{27}$ (PctExtHyperlinks), $f_{30}$ (InsecureForms), $f_{34}$ (PctNullSelfRedirectHyperlinks), $f_{35}$ (FrequentDomainNameMismatch), $f_{38}$ (PopUpWindow), $f_{39}$ (SubmitInfoToEmail), $f_{47}$ (ExtMetaScriptLinkRT), and $f_{48}$ (PctExtNullSelfRedirectHyperlinksRT).

Validation these scores is not an easy task: different models might focus on different features. That is why having a comprehensible measure is important to gain trustworthiness in the results [46]. Despite this fact, we insisted on comparing the relevance scores with the coefficients of a logistic model after reducing the number of features (for the sake of simplicity). Aiming at selecting the features, we used the CfsSubsetEval method [47], which evaluates the worth of a subset of features by considering the individual predictive ability of each feature along with the degree of redundancy between them.

Equation (10) depicts the LR model obtained from these features

$$g(x) = -6.83f_1 - 7.62f_3 + 26.17f_5 - 0.41f_{14} - 5.53f_{25} + 8.98f_{27} - 3.14f_{30} + 4.51f_{34} - 4.11f_{35} + 4.5f_{38} + 3.79f_{39} - 2.37f_{47} + 9.37f_{48} - 3.0917.$$

While the weights of selected features differ from those reported by our measure, the $f_5$ feature continues to have the largest weight in the LR model. We also observed an interesting behavior when using all features to build a regularized regression model: the larger the penalization, the larger the weight of $f_5$. This further confirms that this feature is quite important for the decision process of these regression-like classifiers.

VI. CONCLUSION

In this article, we presented a recurrence-aware LTCN-based model for interpretable pattern classification. While many FCM-based classifiers have the limitation of converging to a unique fixed point, therefore being able to recognize only one decision class, our proposal evades this by establishing temporal connections between all earlier states and the decision neurons. In that way, our model focuses on the trajectory to the equilibrium point rather than the...
fixed point itself. In addition, we proposed a quasi-nonlinear reasoning rule to control nonlinearity. Another drawback found in the literature is that many learning algorithms for FCM-like models are meta-heuristic-based and suffer from being slow. This proposal dodges these issues by employing instead a two-step deterministic learning method. Last but not least, we propose a feature relevance score as a proxy for interpretability.

The numerical simulations using 30 structured pattern classification problems supported our research hypotheses. First, we confirmed that our recurrence-aware LTCN model is not affected by the unique fixed point attractor problem, in contrast to the FCM-based classifier, which performed worse than an LR model. Second, we illustrated how some problems benefit from models having a certain linearity degree (i.e., using the neurons’ initial states when computing the temporal states). Third, we showed that our model’s discriminative capability is comparable to state-of-the-art algorithms. Moreover, we noted that the learning method produced sparse weight representations, even when we did not consider any regularization strategy. Finally, it was found that our feature relevance measure aligns well with the interpretability derived from an LR strategy. Finally, it was found that our feature relevance measure aligns well with the interpretability derived from an LR strategy. Furthermore, it was found that our feature relevance measure aligns well with the interpretability derived from an LR strategy. Therefore, it was found that our feature relevance measure aligns well with the interpretability derived from an LR strategy. The authors thank the anonymous reviewers for their comments, valuable suggestions, and recommendations.

Acknowledgment

The authors thank the anonymous reviewers for their constructive criticism, valuable comments, and suggestions.

References

[1] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification, 2nd ed. New York, NY, USA: Wiley, 2012.
[2] D. Gunning and D. W. Aha, “DARPA’s explainable artificial intelligence program,” AI Mag., vol. 40, no. 2, pp. 44–58, 2019.
[3] B. Goodman and S. Flaxman, “European union regulations on algorithmic decision-making and a ‘right to explanation’” AI Mag., vol. 38, no. 3, pp. 50–57, 2017.
[4] D. Shin, “User perceptions of algorithmic decisions in the personalized AI system: Perceptual evaluation of fairness, accountability, transparency, and explainability,” J. Broadcast. Electron. Media, vol. 64, no. 4, pp. 541–565, 2020.
[5] A. B. Arrieta et al., “Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,” Inf. Fusion, vol. 58, pp. 82–115, Jun. 2020.
[6] Z. C. Lipton, “The Myths of model interpretability,” in Proc. ICML Workshop Human Interpretability (WHI), vol. 61, 2016, pp. 96–100.
[7] C. Molnar, Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, 2nd ed., Leanpub, 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book
[8] I. Grau, D. Sengupta, M. G. G. Lorenzo, and A. Nowe, “An interpretable semi-supervised classifier using rough sets for amended self-labeling,” in Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE), 2020, pp. 1–8.
[9] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in Advances in Neural Information Processing Systems 30, I. Guyon et al., Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2017, pp. 4765–4774.
[10] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why should I trust you?” Explaining the predictions of any classifier,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2016, pp. 1135–1144.
[11] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” Nat. Mach. Intell., vol. 1, no. 5, pp. 206–215, 2019.
[12] D. Shin, “The perception of humanness in conversational journalism: An algorithmic information-processing perspective,” New Media Soc., to be published.
[13] B. Kosko, “Fuzzy cognitive maps,” Int. J. Man-Mach. Stud., vol. 24, no. 1, pp. 65–75, 1986.
[14] E. I. Papageorgiou and J. L. Salmeron, “A review of fuzzy cognitive maps research during the last decade,” IEEE Trans. Fuzzy Syst., vol. 21, no. 1, pp. 66–79, Feb. 2013.
[15] G. Nápoles, R. Bello, and K. Vanhoof, “How to improve the convergence on sigmoid fuzzy cognitive maps?” Intell. Data Anal., vol. 18, no. 6S, pp. S77–S88, 2014.
[16] L. Concepción, G. Nápoles, R. Falcon, K. Vanhoof, and R. Bello, “Unveiling the dynamic behavior of fuzzy cognitive maps,” IEEE Trans. Fuzzy Syst., vol. 29, no. 5, pp. 1252–1261, May 2021.
[17] G. Nápoles, F. Vanhoenshoven, R. Falcon, and K. Vanhoof, “Nonsynaptic error backpropagation in long-term cognitive networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 3, pp. 865–875, Mar. 2020.
[18] G. Nápoles et al., “Fuzzy cognitive modeling: Theoretical and practical considerations,” in Proc. 11th KES Int. Conf. Intell. Technol. (KES-IDT), vol. 1, Jun. 2019, pp. 77–87.
[19] G. A. Papakostas, Y. S. Boutalis, E. E. Kouliouriotis, and B. G. Mertzios, “Fuzzy cognitive maps for pattern recognition applications,” Int. J. Pattern Recognit. Artif. Intell., vol. 22, no. 8, pp. 1461–1486, 2008.
[20] G. A. Papakostas and D. E. Kouliouriotis, “Classifying patterns using fuzzy cognitive maps,” in Fuzzy Cognitive Maps, Berlin, Germany: Springer, 2010, pp. 291–306, doi: 10.1007/978-3-642-03220-2_12.
[21] H. J. Song, C. Y. Miao, R. Wuys, Z. Q. Shen, M. D’Hondt, and F. Catthoor, “An extension to fuzzy cognitive maps for classification and prediction,” IEEE Trans. Fuzzy Syst., vol. 19, no. 1, pp. 116–135, Feb. 2011.
[22] A. Anagnostis et al., “A deep learning approach for anthracnose infected trees classification in walnut orchards,” Comput. Electron. Agr., vol. 182, Mar. 2021, Art. no. 105998.
[23] E. I. Papageorgiou, C. D. Stylios, and P. P. Groumpos, “Active Hebbian learning algorithm to train fuzzy cognitive maps,” Int. J. Approx. Reason., vol. 37, no. 3, pp. 219–249, 2004.
[24] G. A. Papakostas, D. E. Kouliouriotis, A. S. Polydoros, and V. D. Taurasis, “Towards Hebbian learning of fuzzy cognitive maps in pattern classification problems,” Expert Syst. Appl., vol. 39, no. 12, pp. 10620–10629, 2012.
[25] E. I. Papageorgiou and P. P. Groumpos, “A new hybrid method using evolutionary algorithms to train fuzzy cognitive maps,” Appl. Soft Comput., vol. 5, no. 4, pp. 409–431, 2005.
[26] W. Stach, L. A. Kurgan, and W. Pedrycz, “A divide and conquer method for learning large fuzzy cognitive maps,” Fuzzy Sets Syst., vol. 161, no. 19, pp. 2515–2532, 2010.
[27] W. Stach, W. Pedrycz, and L. A. Kurgan, “Learning of fuzzy cognitive maps using density estimate,” IEEE Trans. Syst. Man, Cybern. B, Cybern., vol. 42, no. 3, pp. 900–912, Jun. 2012.
[28] M. León, L. Mkrtchyan, B. Depaire, D. Ruan, and K. Vanhoof, “Learning and clustering of fuzzy cognitive maps for travel behaviour analysis,” Knowl. Inf. Syst., vol. 39, no. 2, pp. 435–462, 2013.
[29] J. L. Salmeron, A. Ruiz-Celma, and A. Mena, “Learning FCMs with multi-local and balanced memetic algorithms for forecasting industrial drying processes,” Neurocomputing, vol. 232, pp. 52–57, Apr. 2017.
[30] G. Nápoles, I. Grau, E. Papageorgiou, R. Bello, and K. Vanhoof, “Rough cognitive networks,” Knowl. Based Syst., vol. 91, pp. 46–61, Jan. 2016.
[31] G. Nápoles, C. Mosquera, R. Falcon, I. Grau, R. Bello, and K. Vanhoof, “Rough cognitive networks,” Neural Netw., vol. 97, pp. 19–27, Jan. 2018.
[32] L. Concepción, G. Nápoles, I. Grau, and W. Pedrycz, “Fuzzy-rough cognitive networks: Theoretical analysis and simpler models,” IEEE Trans. Cybern., early access, Oct. 7, 2020, doi: 10.1109/TCYB.2020.3022527.
[33] G. Nápoles, M. L. Espinosa, I. Grau, K. Vanhoof, and R. Bello, “Fuzzy cognitive maps based models for pattern classification: Advantages and challenges,” in Soft Computing Based Optimization and Decision Models. Cham, Switzerland: Springer Int., Aug. 2017, pp. 83–98, doi: 10.1007/978-3-319-64286-4_5.
[34] Z. Yu et al., “Multiobjective semisupervised classifier ensemble,” IEEE Trans. Cybern., vol. 49, no. 6, pp. 2280–2293, Jun. 2019.
[35] R. Penrose, “A generalized inverse for matrices,” Math. Proc. Cambridge Philos. Soc., vol. 51, no. 3, pp. 406–413, 1955.
[36] G. Nápoles, W. Goossens, Q. Moesen, and C. Mosquera, “Fast k-fuzzy-rough cognitive networks,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), 2020, pp. 1–8.
[37] J. Cohen, “A coefficient of agreement for nominal scales,” Educ. Psychol. Meas., vol. 20, no. 1, pp. 37–46, 1960.
[38] N. Japkowicz and M. Shah, Evaluating Learning Algorithms: A Classification Perspective. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[39] A. Ben-David, “Comparison of classification accuracy using Cohen’s weighted kappa,” Expert Syst. Appl., vol. 34, no. 2, pp. 825–832, 2008.
[40] A. A. Akinduko, E. M. Mirkes, and A. N. Gorban, “SOM: Stochastic initialization versus principal components,” Inf. Sci., vols. 364–365, pp. 213–221, Oct. 2016.
[41] T. Kohonen, “Self-organized maps of sensory events,” Philos. Trans. Roy. Soc. London A, Math. Phys. Eng. Sci., vol. 361, no. 1807, pp. 1177–1186, 2003.
[42] G. Ke et al., “LightGBM: A highly efficient gradient boosting decision tree,” in Advances in Neural Information Processing Systems, vol. 30, I. Guyon et al., Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2017.
[43] D. Servén and C. Brummitt, “pyGAM: Generalized Additive Models in Python.” Mar. 2018. [Online]. Available: doi.org/10.5281/zenodo.1208723
[44] T. Wang and Q. Lin, “Hybrid predictive models: When an interpretable model collaborates with a black-box model,” J. Mach. Learn. Res., vol. 22, no. 137, pp. 1–38, 2021.
[45] K. L. Chiew, C. L. Tan, K. Wong, K. S. C. Yong, and W. K. Tiong, “A new hybrid ensemble feature selection framework for machine learning-based phishing detection system,” Inf. Sci., vol. 484, pp. 153–166, May 2019.
[46] C.-F. Juang, T.-L. Jeng, and Y.-C. Chang, “An interpretable fuzzy system learned through online rule generation and multiobjective ACO with a mobile robot control application,” IEEE Trans. Cybern., vol. 46, no. 12, pp. 2706–2718, Dec. 2016.
[47] M. A. Hall, “Correlation-based feature selection for machine learning,” Ph.D. dissertation, Dept. Doctor Philos., Univ. Waikato, Hamilton, New Zealand, 1998.