Fast k-Fuzzy-Rough Cognitive Networks

Gonzalo Nápoles *†, Wouter Goossens*, Quinten Moesen*, Carlos Mosquera*
*Faculty of Business Economics, Hasselt University, Belgium
†Department of Cognitive Science and Artificial Intelligence, Tilburg University, The Netherlands
Email: gonzalo.napoles@uhasselt.be

***Abstract*—Fuzzy-Rough Cognitive Networks (FRCNs) are neural networks that use rough information granules with soft boundaries to perform the classification process. Unlike other neural systems, FRCNs are lazy learners in the sense that we can build the whole model when classifying a new instance. This is possible because the weight matrix connecting the neurons is prescriptively programmed. Similar to other lazy learners, the processing time of FRCNs notably increases with the number of instances in the training set, while their performance deteriorates in noisy environments. Aiming at coping with these issues, this paper presents a new FRCN-based algorithm termed Fast k-Fuzzy-Rough Cognitive Network. This variant employs a multi-thread approach for building the information granules as computed by k-fuzzy-rough sets. Numerical simulations on 35 classification datasets show a notable reduction on FRCNs' processing time, while also delivering competitive results when compared to other lazy learners in noisy environments.**

***Index Terms*—parallel granulation, noise, fuzzy-rough sets, granular cognitive mapping, lazy learners.**

I. Introduction

The continuously growing amount of raw data in our world has been challenging researchers to come up with better ways of building inference models able to extract useful insights from historical data. Pattern classification is one of the most researched topics in the field of machine learning. The classification problem consists of identifying the correct label (decision class) for an unseen object based on the available data [1]. A wide variety of models has been proposed to solve such type of problems. While often managing to deliver excellent results, many models lack the ability to explain how an object is exactly classified. That is why traditional neural networks are said to behave like *black boxes*. On the other hand, granular networks try to overcome this shortcoming by processing information granules instead of the traditional numeric representation of a classification problem [2]. The models discussed in this paper construct these information granules using either fuzzy sets [3], rough sets [4] or fuzzy-rough sets [5]. This granulation procedure allows representing the problem domain in a more concise manner. Yet granular classifiers are able to obtain competitive results when compared to traditional neural networks [6].

As an example of these granular systems, Nápoles et al. [7] introduced the *Rough Cognitive Networks* as a hybridization between rough sets [4] and fuzzy cognitive maps [8]. This lazy learner constructs a cognitive network by following a predefined set of rules that use the positive, negative and boundary rough regions for each decision class. One may notice that only the number of decision classes can enlarge the topology of the model, the number of attributes does not have an influence on the size of the network. Besides, the use of rough sets allows the model to cope better with uncertainty arising from inconsistencies. Despite all the advantages of this granular classifier, it remains to be sensitive to the value of the similarity threshold parameter. Therefore, Nápoles et al. [9] proposed an RCN-based multi-classifier system that does not require an explicit value for the similarity threshold parameter. Instead, this ensemble classifier induces different granularity degrees by using a variety of parameter values. However, the model still constructs granular regions with hard boundaries. *Fuzzy-Rough Cognitive Networks* (FRCNs) entirely suppressed the need for an explicit parameter value [10]. This lazy learner constructs information granules with soft boundaries by using fuzzy-rough sets, thus replacing the abrupt transitions between classes with more gradual ones.

In real-world problems, most datasets usually contain noisy instances which might cause FRCNs' performance to degenerate significantly. If the nearest neighbor is a mislabeled sample, the values of fuzzy lower and upper approximations may be contaminated [11]. On the other hand, the information granulation of the FRCN model may become quite extensive as the number of instances increases. This happens due to the fact that for a dataset comprised of N instances, the algorithm has to execute the distance function $N \times (N-1)/2$ times. Therefore, the processing time drastically increases when more instances are added to the dataset.

The two main contributions of this paper attempt to overcome the previously described problems. Firstly, we attempt to reduce the noise effects on the performance of the algorithm by using a k-distance function instead of using sensitive operators as the *infimum* and *supremum*. Secondly, a parallel granulation approach is introduced to reduce the processing time of deriving the fuzzy-rough regions. In that regard, our parallel solution will focus on the computation of the similarity matrix and the fuzzy-rough regions attached to each decision class. Numerical simulations show the superiority of the new variant with respect to the original FRCN algorithm and other lazy learners reported in the literature.

The structure of this paper is as follows. Section II introduces the reader to the fundamentals of fuzzy-rough set theory. The FRCN model will be outlined in Section III, whereas Section IV will elaborate on the two main contributions of this paper to the FRCN model. These contributions will be assessed by extensive experiments in Section V. The final remarks will be provided in Section VI.

II. Preliminaries on Fuzzy-Rough Sets

Let U denote the universe of discourse which incorporates all objects in the training dataset. We can define X_c as a partition of U which contains all elements associated with the D_c decision class. The membership degree of $x \in U$ to a subset X_c is given by as follows,

$$\mu_{X_c}(x) = \begin{cases} 1 & , x \in X_c \\ 0 & , x \notin X_c \end{cases}. \tag{1}$$

The FRCN classifier introduced by Nápoles et al. [10] uses the fuzzy-rough approach proposed by Inuiguchi et al. [5], which will be briefly introduced next.

Firstly, we need to determine the similarity between two instances x and y. Such a function, denoted by $\mu_P(y,x)$, can be constructed by combining the previously described membership degree $\mu_{X_c}(x)$ with a similarity degree $\varphi(x,y)$. This similarity degree denotes the complement of the normalized distance between two instances x and y.

$$\mu_P(y,x) = \mu_{X_c}(x)\varphi(x,y) = \mu_{X_c}(x)(1-\delta(x,y)). \tag{2}$$

Secondly, we use Equations (1) and (2) to analytically derive the membership function describing the lower approximations associated with the fuzzy set X_c. This is to say,

$$\mu_{P_*(X_c)}(x) = \min\left\{\mu_{X_c}(x), \inf_{y\in\mathcal{U}} \mathcal{I}(\mu_P(y,x), \mu_{X_c}(y))\right\}. \tag{3}$$

where $\mathcal{I}$ is an implication function so that $\mathcal{I}(0,0) = \mathcal{I}(0,1) = \mathcal{I}(1,1) = 0$ and $\mathcal{I}(1,0) = 1$.

Similarly, we can define the membership function for the upper approximation. To do that, we use a conjunction function $\mathcal{T}$ such that $\mathcal{T}(0,0) = \mathcal{T}(0,1) = \mathcal{T}(1,1) = 1$ and $\mathcal{T}(1,0) = 0$. Equation (4) shows this membership function for the upper approximation,

$$\mu_{P^*(X_c)}(x) = \max\left\{\mu_{X_c}(x), \sup_{y\in\mathcal{U}} \mathcal{T}_1(\mu_P(x,y), \mu_{X_c}(y))\right\}. \tag{4}$$

The lower and upper fuzzy-rough approximations are the main building-blocks of the FRCN classification model. In the next section, we will explain how to derive the network structure in a prescriptive way, without the need for an explicit learning process to compute the weight set.

III. Fuzzy-rough Cognitive Networks

When constructing an FRCN model, we first need to granulate the information space, so that a fuzzy-rough attribute space is created. This can be done by categorizing objects into granules with soft boundaries, which translates into computing the positive, negative and boundary fuzzy-rough regions by using Equations (5), (6) and (7), respectively,

$$\mu_{POS(X_c)}(x) = \mu_{P_*(X_c)}(x) \tag{5}$$

$$\mu_{NEG(X_c)}(x) = 1 - \mu_{P_*(X_c)}(x) \tag{6}$$

$$\mu_{BND(X_c)}(x) = \mu_{P^*(X_c)}(x) - \mu_{P_*(X_c)}(x). \tag{7}$$

In this model, the C_i neuron denotes either a positive (P_c), a negative (N_c) or a boundary region (B_c), or an output neuron (D_c). The weights connecting the neurons are assigned based on the following construction rules:

- R_1: If $C_i = P_c$ AND $C_j = D_c$ then $w_{ij} = 1.0$
- R_2: If $C_i = N_c$ AND $C_j = D_c$ then $w_{ij} = -1.0$
- R_3: If $C_i = P_c$ AND $C_j = D_{v\neq c}$ then $w_{ij} = -1.0$
- R_4: If $C_i = P_c$ AND $C_j = P_{v\neq c}$ then $w_{ij} = -1.0$
- R_5: IF $C_i = B_c$ AND $C_j = D_{v\neq c}$ AND $\min_{x\in\mathcal{U}} \{\mu_{BND(X_c)}(x), \mu_{BND(X_v)}(x)\} > 0$ then $w_{ij} = 0.5$

After computing all fuzzy-rough regions, we build a neural network comprised of $|\mathcal{D}|$ output neurons, between $2|\mathcal{D}|$ and $3|\mathcal{D}|$ input neurons and between $2|\mathcal{D}|(1+|\mathcal{D}|)$ and $3|\mathcal{D}|(1+|\mathcal{D}|)$ weights, depending on the number of non-empty boundary regions. Here, $\mathcal{D}$ represents the set of all decision classes. Figure 1 shows the FRCN for a classification problem with only two decision classes.

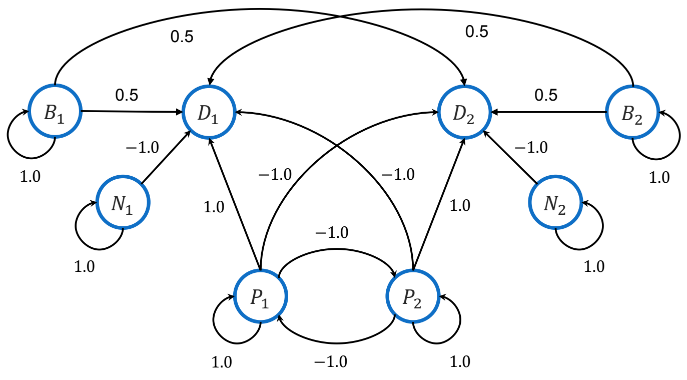

Fig. 1. Fuzzy-Rough Cognitive Network for binary classification.

The initial activation value $A_i^{(0)}$ of the neuron C_i is computed by using the similarity degree between the new object y and $x \in U$, and the membership degree of x to each granular region. When classifying a new instance, the initial activation value of decision neurons is set to zero. Equations (8), (9) and (10) formalize the initial activation rules for positive, negative and boundary neurons, respectively,

$$A_i^{(0)} = \frac{\sum_{x\in\mathcal{U}} \mathcal{T}(\varphi(x,y), \mu_{POS(X_c)}(x))}{\sum_{x\in\mathcal{U}} \mu_{POS(X_c)}(x)}, \text{if } C_i = P_c \tag{8}$$

$$A_i^{(0)} = \frac{\sum_{x\in\mathcal{U}} \mathcal{T}(\varphi(x,y), \mu_{NEG(X_c)}(x))}{\sum_{x\in\mathcal{U}} \mu_{NEG(X_c)}(x)}, \text{if } C_i = N_c \tag{9}$$

$$A_i^{(0)} = \frac{\sum_{x\in\mathcal{U}} \mathcal{T}(\varphi(x,y), \mu_{BND(X_c)}(x))}{\sum_{x\in\mathcal{U}} \mu_{BND(X_c)}(x)}, \text{if } C_i = B_c. \tag{10}$$

The activation value $A_i^{(t)}$ of neuron C_i in the t-th iteration is computed by using Equation (11) such that M is the number of neurons connected to C_i. This reasoning rule is iteratively applied to all neurons in the model [12],

$$A_i^{(t+1)} = f(\sum_{j=1}^{M} w_{ji} A_j^{(t)}) \tag{11}$$

where $f(\cdot)$ is the sigmoid transfer function used to limit neurons' values within the $[0, 1]$ interval, whereas w_{ij} represents the causal weight connecting C_i and C_j.

The FRCNs' recurrent reasoning process continues until either a fixed point attractor, or a fixed number of iterations is reached. A fixed point attractor is reached when $A_i^{(t+1)} \approx A_i^{(t)}$ applies for each neuron i in the network. Finally, the class represented with the output neuron having the highest activation value is assigned to the new object y.

IV. Fast k-Fuzzy-Rough Cognitive Networks

Next, we introduce the key contributions of this paper which aim at improving the performance of the FRCN model with respect to both prediction power and computational efficiency. Firstly, we present the k-fuzzy-rough set model to cope better with noisy datasets. As a second contribution, we propose a parallel granulation approach when computing the distances between all instances in the training set. This method will enable the k-FRCN algorithm to build its model using multiple threads, thus speeding up its performance.

A. k-Fuzzy-Rough Sets

For a given implication function $\mathcal{I}$ and $\mathcal{T}$-norm, we notice that the membership of an object to the fuzzy lower approximation $\mu_{P_*(X_c)}$ matches with the distance from x to its nearest neighbor from different classes, while the membership to the fuzzy upper approximation $\mu_{P^*(X_c)}$ is the similarity between x and the nearest neighbor in that set X_c.

Equation (12) uses the Kleene-Dienes implicator to compute the lower membership function,

$$\begin{aligned} \mu_{P_*(X_c)}(x) &= \inf_{y \in U} \max\big(1 - \mu_P(y,x), \mu_{X_c}(y)\big) \\ &= \inf_{y \in X_c} \max\big(1 - \mu_P(y,x), 1\big) \wedge \\ &\quad \inf_{y \notin X_c} \max\big(1 - \mu_P(y,x), 0\big) \\ &= 1 \wedge \inf_{y \notin X_c} \big(1 - \mu_P(y,x)\big) \\ &= \inf_{y \notin X_c} \delta(x,y). \end{aligned} \tag{12}$$

Equation (13) adopts the standard $\mathcal{T}$-norm to compute the upper membership function,

$$\begin{aligned} \mu_{P^*(X_c)}(x) &= \sup_{y \in U} \min\big(\mu_P(x,y), \mu_{X_c}(y)\big) \\ &= \sup_{y \in X_c} \min\big(\mu_P(x,y), 1\big) \vee \\ &\quad \sup_{y \notin X_c} \min\big(\mu_P(x,y), 0\big) \\ &= \sup_{y \in X_c} \mu_P(x,y) \vee 0 \\ &= \sup_{y \in X_c} \varphi(x,y). \end{aligned} \tag{13}$$

We can try to reduce the noise effects by using a k-distance function instead of using the inf and sup operators. Given an object x and a set of objects $Y = \{y_1, y_2, \ldots, y_n\}$, we define a distance function δ_k between x and Y as the distance from x to its k-th nearest neighbor in Y. Therefore, Equations (12) and (13) can be rewritten as follows:

$$\mu_{P_*(X_c)}(x) = \inf_{y \notin X_c} \delta(x,y) = \delta_1(x, U - X_c), \tag{14}$$

$$\mu_{P^*(X_c)}(x) = \sup_{y \in X_c} \varphi(x,y) = 1 - \delta_1(x, X_c). \tag{15}$$

Notice that we get the original fuzzy-rough set formulation if parameter k is set equal to one. On the other hand, we get more flexible definitions for the upper and lower approximations when $k > 1$. We refer to this as k-fuzzy-rough sets. The approximations can then be computed using:

$$\mu_{P_*(X_c)}(x) = \min\{\mu_{X_c}(x), \delta_k(x, U - X_c)\}, \tag{16}$$

$$\mu_{P^*(X_c)}(x) = \max\{\mu_{X_c}(x), 1 - \delta_k(x, X_c)\}. \tag{17}$$

It is worth mentioning that Inuiguchi's model [5] does not assume that $\mu_P(x,x) = 1, \forall x \in U$. Instead, we compute the minimum and the maximum when computing the $\mu_{P_*(X_c)}(x)$ and $\mu_{P^*(X_c)}(x)$, respectively. This feature allows preserving the inclusiveness of $P_*(X_c)$ in the fuzzy set X_c and the inclusiveness of X_c in $P^*(X_c)$.

B. Parallel granulation process

The Fast k-FRCN algorithm is a multi-threaded variant of the FRCN classifier that improves the efficiency of the model in relatively large datasets. The calculations that are implemented in our multi-threaded approach are: i) the computation of the distance matrix, ii) the construction of the k-fuzzy-rough regions, and iii) the computation of the similarity class for a given object to be classified.

When computing the similarity distance matrix, the equal distribution of the distance computations induces balanced workloads among all threads. Only the distance values below the main diagonal in the distance matrix are computed, as the distance function used by the FRCN-based classifiers is symmetric. Figure 2 illustrates the computation of the distance matrix by using three processing threads.

In this multi-threaded model, the dataset is split into several chunks such that each thread processes a piece of data. Meanwhile, the main process waits until all tasks are completed.

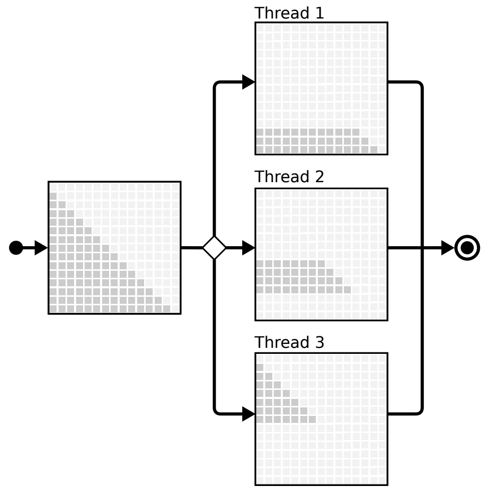

Fig. 2. Multi-threaded approach to compute the distance matrix.

Finally, all parts are merged. The same reasoning applies when computing the granular regions associated with each decision class and the similarity classes of each object.

V. Numerical simulations

To validate the two contributions of this paper, we perform two experiments. In the first experiment, we set up a standard classification problem context where we compare the Fast-k-FRCN to the k-FRCN implementation. For each of the 35 datasets, 4 levels of noise have been generated (0%, 5%, 10% and 20%). Both algorithms have been trained for each dataset and noise level, resulting in 140 results per algorithm. The results from this first experiment are used to check if the Fast-k-FRCN does benefit from the parallelization of the distance calculation step in terms of processing time.

In our second experiment, the k-FRCN algorithm is compared with 6 state-of-the-art granular classifiers. The first goal of this experiment is to check whether our proposal can deliver competitive results when compared with other lazy learners. Additionally, we validate if the k-fuzzy-rough sets allow the algorithm to cope better with noisy datasets.

A. Description of benchmark problems

To conduct our experiment, we selected 35 pattern classification datasets taken from the KEEL [13] and UCI ML [14] repositories. The number of instances in these datasets ranges from 1,599 to 12,960. Using fairly large datasets, we can get more insight into the reduction in time necessary to build the model. The number of attributes ranges from 2 to 240 and the number of classes from 2 to 100. A more extensive description of the datasets can be found in Table I.

All numerical attributes have been normalized and none of the included datasets have any missing values. The noise levels mentioned in this paper have artificially been added using a noise filter in Weka, which replaces the original decision class with a randomly selected class. This method is applied to the specified percentage of instances.

TABLE I
Description of the datasets

Dataset	Instances	Attributes	Classes	Imbalance
abalone	4,174	8	28	689:1
banana	5,300	2	2	1:1
bank	4,521	16	2	8:1
car	1,728	6	4	19:1
cardiotocography-10	2,126	35	10	11:1
cardiotocography-3	2,126	35	3	9:1
chess	3,196	36	2	1:1
crowdsourced-mapping	10,545	28	6	140:1
csj	2,796	34	6	2:1
frogs-mfccs	7,195	22	10	51:1
hypothyroid	3,772	29	4	1740:1
mfeat-factors	2,000	216	10	1:1
mfeat-fourier	2,000	76	10	1:1
mfeat-karhunen	2,000	64	10	1:1
mfeat-morphological	2,000	6	10	1:1
mfeat-pixel	2,000	240	10	1:1
mfeat-zernike	2,000	47	10	1:1
mushroom	8,124	22	2	1:1
musk2	6,598	167	2	5:1
nursery	12,960	8	5	2160:1
optdigits	5,620	64	10	1:1
ozone	2,536	72	2	34:1
page-blocks	5,473	10	5	175:1
pendigits	10,992	16	10	1:1
phoneme	5,404	5	2	2:1
plant-margin	1,600	64	100	1:1
plant-shape	1,600	64	100	1:1
plant-texture	1,599	64	100	1:1
segment	2,310	19	7	1:1
spambase	4,601	57	2	2:1
splice	3,190	60	3	2:1
wall-following	5,456	24	4	7:1
waveform	5,000	40	3	1:1
wine-quality-white	4,898	11	7	440:1
winequality-red	1,599	11	11	68:1

B. Lazy learners used for comparison

1) IBk [15]: IBk is an implementation of the KNN algorithm that calculates the similarity between instances using a distance function. This algorithm never generates any abstractions of the dataset, but rather compares unseen instances with already classified ones. Therefore, it is called an instance-based algorithm. The similarity between two instances x and y can be computed as follows:

$$\varphi(x,y) = -\sqrt{\sum_{i=1}^{A} f(x_i,y_i)} \qquad (18)$$

where A represents the set of attributes describing the problem instances, $f(x_i,y_i) = \{(x_i-y_i)^2$ for numerical attributes; $(x_i \neq y_i)$ for nominal attributes$\}$. The algorithm classifies an unseen instance based on the majority class of its k most similar instances. To handle noisy datasets, the algorithm keeps a record of the performance of each instance. Only instances that are known to correctly classify other instances are utilized in the classification process. Such a process can be enhanced by assigning a larger weight to neighbors which are more similar to the new instance.

2) KStar (K^*) [16]: the key difference between IBk and this method is that it uses an entropy-based distance function. This distance can be expressed as the complexity of trying to transform one instance x to another instance y. The K^* distance is defined as the sum of the probabilities of all possible transformations between two instances. One advantage of this distance measure is that it takes into account the probability of two nominal values to be similar.

3) FuzzyRoughNN (FRNN) [17]: this method is based on the fuzzy k-nearest neighbor (FNN) [18], which improves the way we determine the membership of an instance to a class in the kNN algorithm. Such a membership is computed by using the distance of the new instance from its k-nearest neighbors and the memberships of these neighbors to their classes. For an unlabeled instance x the membership to the c-th decision class is given as follows:

$$c\mu_c(x) = \sum_{y \in \mathcal{N}(x)} F_{FNN}(y,x)\ \mu_c(y), \tag{19}$$

and

$$F_{FNN}(y,x) = \frac{||x-y||^{-2/(m-1)}}{\sum\limits_{X_i \in \mathcal{N}(x)} ||x - X_i||^{2/(m-1)}}. \tag{20}$$

where $\mathcal{N}(x)$ represents the set of k-nearest neighbors of a certain instance x, while m controls the overall weighting of the similarity. As m increases, the neighbors are more evenly weighted, and their relative distances from the point being classified have less effect.

For every x, the algorithm calculates the membership $\mu_c(y)$ to the decision class D_c. A crisp membership means that the membership will be one if the instance belongs to the class, otherwise the membership will be zero. The fuzzy membership proposed in [17] is computed as follows:

$$\mu_c(y) = \begin{cases} 0.51 + 0.49\frac{|\mathcal{N}_c|}{k} & if\ x\ is\ in\ class\ D_c \\ 0.49\frac{|\mathcal{N}_c|}{k} & otherwise \end{cases} \tag{21}$$

where $|\mathcal{N}_c|$ is the number of nearest neighbors that belong to decision class D_c. Afterwards, the decision class with the highest membership, calculated by Equation (19), is used to classify the unlabeled instance x.

The FRNN method [19] uses the k-nearest neighbors of x to calculate the upper and lower approximations. Thus, a $[0,1]$ valued fuzzy tolerance relation is used. This can be seen as another way to describe the similarity relation:

$$F_{FRNN}(y,x) = \min_{a \in A} F_{a_{FRNN}}(y,x) \tag{22}$$

and

$$F_{a_{FRNN}}(y,x) = 1 - \frac{|a(y) - a(x)|}{|a_{\max} - a_{\min}|}. \tag{23}$$

To calculate the membership degree of object x to a decision class D_c we first need approximations for the upper and lower boundaries $F_*C(x)$ and $F^*C(x)$. There are two variants of FRNN: FRNN-FRS and FRNN-VQRS. The second variant will be explained later in this paper. The FRNN-FRS method uses $\mathcal{T}$-norm and implicator approximations:

$$F_*c(x) = \min_{y \in \mathcal{N}(x)} \mathcal{I}(F_{FRNN}(y,x), \mu_c(y)), \tag{24}$$

$$F^*c(x) = \max_{y \in \mathcal{N}(x)} \mathcal{T}(F_{FRNN}(y,x), \mu_c(y)). \tag{25}$$

To obtain a definitive class for instance x, the algorithm combines the upper and lower approximation by calculating the mean of the two values. When this is done for every class, the class with the highest value is selected for x.

4) VQNN [19]: In fuzzy-rough set theory, *inf* and *sup* (Formula 3 and 4) operators are used to calculate the lower and upper approximations of a given set. These approximations have the tendency of being more susceptible to noise when the set is crisp. This is why vaguely quantified rough sets were brought into existence. Instead of the traditional crisp quantifiers like "all" and "at least one", VQRS uses "most" and "some" when building the lower and upper approximations, which are defined as follows:

$$\mu_{P_*Qs(y)}(x) = Qs\left(\frac{\sum\limits_{y \in \mathcal{N}(x)} \min(F_{FNN}(y,x), \mu_c(y))}{\sum\limits_{y \in \mathcal{N}(x)} F_{FNN}(y,x)}\right), \tag{26}$$

$$\mu_{P^*Qm(y)}(x) = Qm\left(\frac{\sum\limits_{y \in \mathcal{N}(x)} \max(F_{FNN}(y,x), \mu_c(y))}{\sum\limits_{y \in \mathcal{N}(x)} F_{FNN}(y,x)}\right) \tag{27}$$

such that Qs and Qm are fuzzy quantifiers that model the linguistic quantifiers "some" and "most", respectively. A general definition can be formulated, which can be used to generate different fuzzy quantifiers, as follows:

$$Q_{(\alpha,\beta)} = \begin{cases} 0 & , x \leq \alpha \\ \frac{2(x-\alpha)^2}{(\beta-\alpha)^2} & , \alpha \leq x \leq \frac{\alpha+\beta}{2} \\ 1 - \frac{2(x-\alpha)^2}{(\beta-\alpha)^2} & , \frac{\alpha+\beta}{2} \leq x \leq \beta \\ 1 & , \beta \leq x \end{cases} \tag{28}$$

where α and β are parameters that can be used to satisfy a personal definition of "some" and "most".

5) FuzzyOwnershipNN (FONN) [17]: handles the uncertainty caused by overlapping classes and inadequate knowledge. To do this, the algorithm uses the confidence for an object x being associated with the decision class D_c, which is defined as follows:

$$\tau_c(x) = \frac{\sum\limits_{y \in U} F_{FONN}(y,x)\mu_c(y)}{|U|} \tag{29}$$

where $F_{FONN}(y,x)$ is the fuzzy relation between y and x. This fuzzy relation can be written down as:

$$F_{FONN}(y,x) = \exp\big(-\sum_{a \in A} b_a(a(x) - a(y))^{2/(m-1)}\big) \tag{30}$$

where m again controls the overall weighting, while b_a defines the bandwidth of the membership,

$$b_a = \frac{|U|}{2\sum_{y\in U} ||a(x) - a(y)||^{2/(m-1)}}. \tag{31}$$

Firstly, the bandwidth is calculated for each attribute. Hereafter $\tau_c(x)$ is calculated for every class. The output of the algorithm will be the class with the highest membership value. There is no need for a parameter k in this algorithm, because distant neighbors will not influence the membership that much. Yet, every instance is considered.

6) OWANN [20], [21]: Vaguely quantified rough sets, as proposed in [22], still show some crisp behavior. Thus, ordered weighted average (OWA) fuzzy-rough sets were introduced in [23] to replace the strict minimum and maximum operators of fuzzy-rough lower and upper approximations. An OWA operator of dimension n can be defined as a mapping [24]: $R^n \rightarrow R$ with an associated weight vector $w = \langle w_1, w_2, \ldots, w_n\rangle$. This vector has to fulfill two constraints: i) $w_k \in [0,1]$ and ii) $\sum_{k=1}^{n} w_k = 1$. The purpose is to calculate an OWA aggregation of a decreasingly ordered vector of p scalar values. Hence, we can formulate the new upper and lower approximations of a fuzzy rough set as follows:

$$F_* c(x) = OWA_{w_*}\mathcal{I}\big(F_{FRNN}(y,x),\ \mu_c(y)\big), \tag{32}$$

$$F^* c(x) = OWA_{w^*}\mathcal{T}\big(F_{FRNN}(y,x),\ \mu_c(y)\big). \tag{33}$$

Similar to FRNN, the mean of the upper and lower approximation values determines the decision class that is selected for an unlabeled instance x.

C. Settings and Hyperparameter Optimization

Aiming at performing the comparisons in an optimal setting, we perform hyperparameter tuning on each dataset. Algorithm 1 depicted the grid search approach used in the paper, which modifies the procedure in [25] by replacing the lowest average error with the highest average Kappa value.

Algorithm 1 Hyperparameter tuning procedure

```
 1: Randomly split dataset into 5 folds
 2: for each parameter configuration cf: do
 3:     for each fold pt: do
 4:         Train model on remaining folds (not pt)
 5:         Split fold f in validation and test set
 6:         Calculate the Kappa value for the validation set
 7:         Calculate the Kappa value for the test set
 8:     end for
 9:     Calculate the average Kappa value for the validation set
10:     Calculate the average Kappa value for the test set
11: end for
12: Determine which configuration bs reported the highest
    average Kappa value on the validation set
13: return  the average Kappa value on the test set, which
    corresponds with the best configuration bs
```

Observe that we use the kappa statistic [26] as the performance measure. This statistic is more robust than the standard accuracy as it takes the chance that an instance is correctly classified by chance into account.

All algorithms that use a k parameter have been optimized with k values ranging from 1 to 10. The lower α of the *OWANN* and *VQNN* classifiers has been trained with values 0.1 and 0.2, the upper α with 0.2 and 0.3. Their β parameters also have undergone hyperparameter tuning. We have used 0.6 and 0.8 for the lower β, and 0.8 and 1 for the upper β. The values for the fuzzifier parameter of FONN ranged from 1 to 10. *KStar* was optimized by using values from 5 to 40 in steps of 5 for the global blending option.

In the case of the FRCN models, we use the Heterogeneous Manhattan-Overlap Metric (HMOM) [9] to compute the dissimilarity between two instances.

D. Exploring Fast k-FRCN's Processing Time

To assess the added value of the multi-threaded approach, we compare the time required to build the model with the number of threads ranging from 1 to 10. If the number of threads is set equal to 1, the algorithm will perform exactly as the original FRCN classifier in term of efficiency. The average time to build the model per number of threads is computed based on the results of the best performing hyperparameter configuration for each dataset. Figure 3 portrays the average time to build the model in each case.

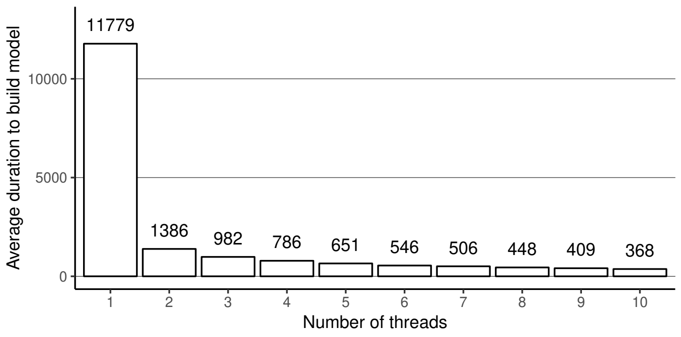

Fig. 3. Average duration in milliseconds to build model per thread setting.

Overall, the time required to build the classification model significantly decreases when the algorithm has access to at least two threads instead of one. When using two threads, the building time of the k-FRCN model is on average reduced by 88% when compared with one thread. Adding more threads only marginally reduces the processing time. Moving from two to three threads, the time gain is 30%.

E. Exploring Fast k-FRCNs' Performance

In this section, we compare the k-FRCN model against the state-of-the-art lazy classifiers revised above and the original FRCN algorithm. This performance comparison experiment has been conducted by using four different noise settings, that is, 0%, 5%, 10% and 20%.

Figures 4, 5, 6 and 7 display the average Kappa values on the 35 benchmark problems. The reader may notice that the performance of all algorithms degenerates as more noise is added to the problem. This is expected because it is not trivial for the algorithms how to distinguish incorrectly labeled instances from correctly labeled ones.

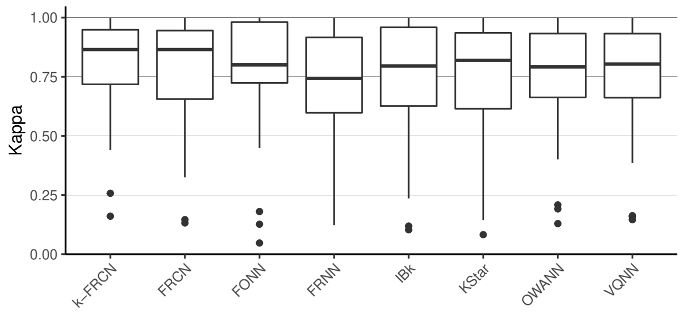

Fig. 4. Average Kappa values computed by the classification models over 35 datasets without any artificial noise.

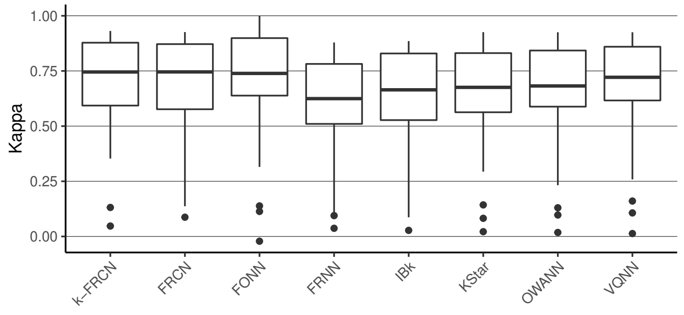

Fig. 5. Average Kappa values computed by the classification models over 35 datasets with 5% artificial noise.

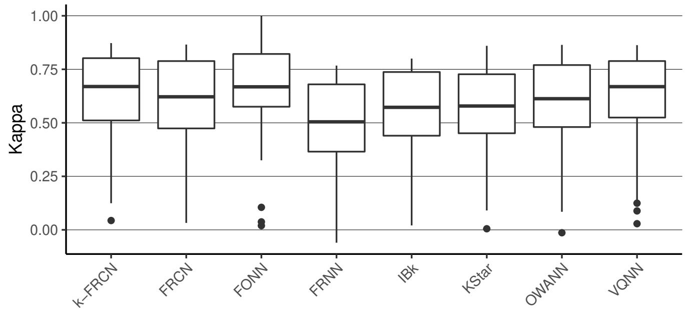

Fig. 6. Average Kappa values computed by the classification models over 35 datasets with 10% artificial noise.

To determine whether the observed performance differences are statistically significant or not, the Friedman test [27] has been conducted for each level of noise. This non-parametric test will reject the null hypothesis when at least two classifiers perform significantly different. The p-values for these tests are respectively 2.25E-5, 2.14E-18, 1.60E-21 and 2.65E-25. All p-values are smaller than the 5% significance level and therefore indicating a difference in performance between at least two algorithms in each noise environment.

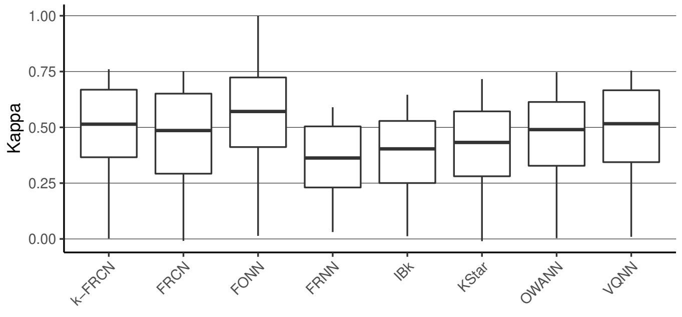

Fig. 7. Average Kappa values computed by the classification models over 35 datasets with 20% artificial noise.

Next, we further analyze the performance differences between k-FRCN and the 7 other algorithms in terms of prediction rates. This pairwise comparison is conducted using a Wilcoxon signed-rank test [28]. This test will reject the null hypothesis when there is a significant difference between the two selected algorithms. This post-hoc procedure attempts to control for type-I errors (false positive). Tables II, III, IV and V summarize the results of the pairwise comparisons for each level of noise. Besides the uncorrected p-values, we also report the negative (R^-) and the positive (R^+) ranks, and the corrected values by Holm.

TABLE II
WILCOXON SIGNED-RANK TEST (0% NOISE) BY USING THE k-FRCN ALGORITHM AS THE CONTROL METHOD.

Algorithm	R^-	R^+	$R^=$	p-value	Holm
FRCN	6	25	4	$2.57E$-4	$1.80E$-3
FONN	21	12	2	$8.58E$-1	$8.58E$-1
FRNN	9	25	1	$3.76E$-4	$2.26E$-3
IBk	14	19	2	$5.82E$-2	$1.16E$-1
KStar	12	22	1	$1.83E$-2	$7.32E$-2
OWANN	12	22	1	$1.32E$-2	$6.59E$-2
VQNN	13	21	1	$3.55E$-2	$1.06E$-1

TABLE III
WILCOXON SIGNED-RANK TEST (5% NOISE) BY USING THE k-FRCN ALGORITHM AS THE CONTROL METHOD.

Algorithm	R^-	R^+	$R^=$	p-value	Holm
FRCN	7	27	1	$1.81E$-4	$9.05E$-4
FONN	15	20	0	$8.27E$-1	$8.27E$-1
FRNN	4	31	0	$2.90E$-7	$2.03E$-6
IBk	4	31	0	$1.60E$-6	$9.62E$-6
KStar	7	28	0	$4.44E$-4	$1.77E$-3
OWANN	8	27	0	$1.84E$-3	$5.53E$-3
VQNN	10	25	0	$1.18E$-2	$2.36E$-2

Overall, the simulation results over the 35 benchmark problems show that the k-FRCN classifier significantly outperforms all algorithms except for FONN in environments with 10% and 20% noise. This clearly indicates an improvement in the model with respect to the original FRCN classifier.

TABLE IV
WILCOXON SIGNED-RANK TEST (10% NOISE) BY USING THE k-FRCN ALGORITHM AS THE CONTROL METHOD.

Algorithm	R^-	R^+	$R^=$	p-value	Holm
FRCN	4	30	1	5.42E-6	2.71E-5
FONN	18	17	0	4.41E-1	4.41E-1
FRNN	3	32	0	1.92E-9	1.34E-8
IBk	5	30	0	2.16E-8	1.30E-7
KStar	8	27	0	7.05E-6	2.82E-5
OWANN	5	30	0	1.51E-4	4.53E-4
VQNN	9	26	0	2.36E-3	4.72E-3

TABLE V
WILCOXON SIGNED-RANK TEST (20% NOISE) BY USING THE k-FRCN ALGORITHM AS THE CONTROL METHOD.

Algorithm	R^-	R^+	$R^=$	p-value	Holm
FRCN	7	28	0	2.58E-6	1.03E-5
FONN	21	14	0	9.68E-2	9.68E-2
FRNN	2	33	0	1.46E-9	1.02E-8
IBk	2	33	0	4.07E-9	2.44E-8
KStar	4	31	0	1.01E-7	5.06E-7
OWANN	8	27	0	8.43E-5	2.53E-4
VQNN	10	25	0	6.65E-3	1.33E-2

VI. CONCLUDING REMARKS

In this paper, we have proposed a parallel and noise-tolerant variant of the Fuzzy-Rough Cognitive Networks. In summary, this research comprises two main contributions. Firstly, the introduction of k-fuzzy-rough sets allows the model to perform significantly better in noisy environments. We have compared our proposed k-FRCN classifier to 7 state-of-the-art granular classifiers, including the original FRCN model. The k-FRCN model has proven to be capable of delivering very competitive results for different noise ratios. Actually, it achieved higher prediction rates than the original FRCN classifier on more than 70% of the benchmark datasets. It should be mentioned however that FONN reported slightly better results in our study. Secondly, the parallel implementation of the information granulation procedure drastically reduced the time necessary to build the model. The processing time decreased 88% on average when operating with at least two threads, thus making our algorithm operative for large datasets. The future research will focus on further improving the FRCNs' performance in noisy environments by using soft rough sets.

REFERENCES

[1] R. O. Duda, P. E. Hart, D. G. Stork, Pattern classification, John Wiley & Sons, 2012 (2012).

[2] W. Pedrycz, G. Vukovich, Granular neural networks, Neurocomputing 36 (1) (2001) 205 – 224 (2001).

[3] L. A. Zadeh, Fuzzy sets, Information and control 8 (3) (1965) 338–353 (1965).

[4] Z. Pawlak, Rough sets, International Journal of Computer & Information Sciences 11 (5) (1982) 341–356 (Oct 1982).

[5] M. Inuiguchi, W.-Z. Wu, C. Cornelis, N. Verbiest, Fuzzy-Rough Hybridization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 425–451 (2015).

[6] G. Benjamín, G. Nápoles, R. Falcon, R. Bello, K. Vanhoof, Performance analysis of granular versus traditional neural network classifiers: Preliminary results, in: 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2018, pp. 1 – 6 (06 2018).

[7] G. Nápoles, I. Grau, E. Papageorgiou, R. Bello, K. Vanhoof, Rough cognitive networks, Knowledge-Based Systems 91 (2016) 46 – 61 (2016).

[8] B. Kosko, Fuzzy cognitive maps, International Journal of Man-Machine Studies 24 (1) (1986) 65 – 75 (1986).

[9] G. Nápoles, R. Falcon, E. Papageorgiou, R. Bello, K. Vanhoof, Rough cognitive ensembles, International Journal of Approximate Reasoning 85 (2017) 79 – 96 (2017).

[10] G. Nápoles, C. Mosquera, R. Falcon, I. Grau, R. Bello, K. Vanhoof, Fuzzy-rough cognitive networks, Neural Networks 97 (2018) 19 – 27 (2018).

[11] Q. Hu, S. An, X. Yu, D. Yu, Robust fuzzy rough classifiers, Fuzzy Sets and Systems 183 (1) (2011) 26–43 (2011).

[12] B. Kosko, Hidden patterns in combined and adaptive knowledge networks, International Journal of Approximate Reasoning 2 (4) (1988) 377 – 393 (1988).

[13] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, Journal of Multiple-Valued Logic and Soft Computing 17 (2-3) (2011) 255–287 (2011).

[14] M. Lichman, UCI machine learning repository (2013).
URL http://archive.ics.uci.edu/ml

[15] D. W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, Machine Learning 6 (1) (1991) 37–66 (January 1991).

[16] J. G. Cleary, L. E. Trigg, K*: An instance-based learner using an entropic distance measure, in: A. Prieditis, S. Russell (Eds.), Machine Learning Proceedings 1995, Morgan Kaufmann, San Francisco (CA), 1995, pp. 108 – 114 (1995).

[17] R. Jensen, C. Cornelis, Fuzzy-rough nearest neighbour classification and prediction, Theoretical Computer Science 412 (42) (2011) 5871 – 5884 (2011).

[18] J. M. Keller, M. R. Gray, J. A. Givens, A fuzzy k-nearest neighbor algorithm, IEEE Transactions on Systems, Man, and Cybernetics SMC-15 (4) (1985) 580 – 585 (July 1985).

[19] R. Jensen, C. Cornelis, Fuzzy-rough nearest neighbour classification and prediction, Theoretical Computer Science 412 (42) (2011) 5871 – 5884 (2011).

[20] E. Ramentol, S. Vluymans, N. Verbiest, Y. Caballero, R. Bello, C. Cornelis, F. Herrera, Ifrowann: Imbalanced fuzzy-rough ordered weighted average nearest neighbor classification, IEEE Transactions on Fuzzy Systems 23 (5) (2015) 1622–1637 (Oct 2015).

[21] N. Verbiest, C. Cornelis, R. Jensen, Quality, frequency and similarity based fuzzy nearest neighbor classification, in: 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2013, pp. 1–8 (July 2013).

[22] C. Cornelis, M. De Cock, A. M. Radzikowska, Vaguely quantified rough sets, in: A. An, J. Stefanowski, S. Ramanna, C. J. Butz, W. Pedrycz, G. Wang (Eds.), Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 87 – 94 (2007).

[23] C. Cornelis, N. Verbiest, R. Jensen, Ordered weighted average based fuzzy rough sets, in: J. Yu, S. Greco, P. Lingras, G. Wang, A. Skowron (Eds.), Rough Set and Knowledge Technology, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 78 – 85 (2010).

[24] R. R. Yager, On ordered weighted averaging aggregation operators in multicriteria decision making, Systems, Man and Cybernetics, IEEE Transactions on 18 (1988) 183 – 190 (02 1988).

[25] G. Nápoles, F. Vanhoenshoven, K. Vanhoof, Short-term cognitive networks, flexible reasoning and nonsynaptic learning, Neural Networks 115 (2019) 72 – 81 (2019).

[26] M. J. Warrens, Cohen's kappa is a weighted average, Statistical Methodology 8 (6) (2011) 473 – 484 (2011).

[27] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (200) (1937) 675–701 (1937).

[28] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80 – 83 (1945).