Learning Topologies with the Growing Neural Forest

Article *in* International Journal of Neural Systems · April 2016

DOI: 10.1142/S0129065716500192

2 authors:

Esteban J. Palomo
University of Malaga
67 PUBLICATIONS **311** CITATIONS
SEE PROFILE

Ezequiel López-Rubio
University of Malaga
115 PUBLICATIONS **668** CITATIONS
SEE PROFILE

International Journal of Neural Systems, Vol. 0, No. 0 (April, 2000) 00–00

Preprint of an article published in International Journal of Neural Systems 26(4), 1650019 (2016) DOI: 10.1142/S0129065716500192 © World Scientific Publishing Company https://www.worldscientific.com/worldscinet/ijns

LEARNING TOPOLOGIES WITH THE GROWING NEURAL FOREST

ESTEBAN JOSÉ PALOMO[a]
Department of Computer Languages and Computer Science
University of Málaga
Bulevar Louis Pasteur 35, 29071 Málaga, Spain
ejpalomo@lcc.uma.es
School of Mathematics and Computer Science
University of Yachay Tech, Ecuador
epalomo@yachaytech.edu.ec

EZEQUIEL LÓPEZ-RUBIO
Department of Computer Languages and Computer Science
University of Málaga
Bulevar Louis Pasteur 35, 29071 Málaga, Spain
ezeqlr@lcc.uma.es

In this work a novel self-organizing model called Growing Neural Forest (GNF) is presented. It is based on the Growing Neural Gas (GNG), which learns a general graph with no special provisions for data sets with separated clusters. On the contrary, the proposed GNF learns a set of trees so that each tree represents a connected cluster of data. High dimensional datasets often contain large empty regions among clusters, so this proposal is better suited to them than other self-organizing models because it represents these separated clusters as connected components made of neurons. Experimental results are reported which show the self-organization capabilities of the model. Moreover, its suitability for unsupervised clustering and foreground detection applications is demonstrated. In particular, the GNF is shown to correctly discover the connected component structure of some datasets. Moreover, it outperforms some well known foreground detectors both in quantitative and qualitative terms.

Keywords: Self organization, tree-structured model, unsupervised clustering, computer vision

1. Introduction

The proposal of the Self-Organizing Map (SOM) by Kohonen in 1982[20] was soon regarded as a key model in the artificial neural network field, which found a place in machine learning along with genetic algorithms and fuzzy systems[1,16,2]. Many self-organizing models have been developed since then[21,22]. Most of them try to address some limitations related to the original SOM, especially its fixed topology (typically an hexagonal or rectangular grid), the selection of the number of neurons prior to training, and the lack of a hierarchical representation of the input data.

One of the most successful self-organizing neural network models is the Growing Neural Gas[7] (GNG) which is based on the Neural Gas (NG) model[36] proposed in 1991. Both models store a prototype vector in each neuron, so that for each input sample the neuron with the prototype which is closest to the input sample is declared as the winning neuron. The NG overcomes the fixed-topology problem of the SOM by sorting the neurons according to their distance to the winning neuron, so that the prototypes of the neurons which are closer to the winning neuron are moved closer to the input sample. The GNG provides a neuron growth and removal mechanism by which it is no longer necessary to worry about the

[a]Corresponding author

initial number of neurons. A neuron is added whenever a neuron exhibits a large quantification error. Other dynamic topology self-organizing neural networks include those that learn the graph which best connects a fixed number of neurons [32].

A fundamental distinction must be done between two kinds of dynamic topology self-organizing models, namely hierarchical and tree-structured. Hierarchical models allow that a neuron has an associated child self-organizing network, so that the overall model is a network of networks. On the other hand, tree models do not allow such child networks and the difference with fixed topology models is that their neurons are connected so as to form trees of neurons.

Previous research on this subject include the hierarchical self-organizing approaches proposed over the last decades [42,24,30], where a hierarchy of self-organizing maps is learned. Here each neuron of a map can have a child map, in case that it represents a subset of the input which deserves a more detailed representation. The Growing Hierarchical Probabilistic Self-Orgainizing Graph (GHPSOG) builds hierarchies of graphs of neurons, which improves flexibility over fixed-topology approaches because each graph learns its own topology [31]. On the other hand, tree-structured models aim to represent the entire data set by a single tree of neurons, such as [15,41,38,5]. The TreeSOM learns a self-organizing map with a standard rectangular lattice topology, but then it builds a tree with the already trained neurons so as to discover the underlying clusters of the dataset [43].

Neither hierarchical nor tree representation of input data are addressed by the GNG or NG models. Here our goal is to propose a self-organizing neural model which learns a forest, i.e. a set of trees of neurons. This way the model is able to represent data sets which are made of several clusters with no connections among them. In other words, each tree of the forest contains several neurons that represent a connected cluster of data which has no connections with other trees. Thus flexibility is improved. Furthermore, the number of neurons is automatically determined during the unsupervised learning process since it is based on the GNG learning mechanism. Consequently, we call this model the Growing Neural Forest (GNF). It must be highlighted that our model is not related to the random ensembles of decision trees which are commonly used in machine learning, which are also called forests. Here the word 'forest' is used in the algebraic sense, i.e. a forest is a set of trees, and the trees are not generated at random but computed from a topology among the neurons.

The structure of this paper is the following. The motivation of our approach is explained in Section 2. Section 3 defines the proposed model. Experimental results are reported in Section 4. Finally, Section 5 is devoted to conclusions.

2. Motivation

Clusters can be seen as regions with a relatively large sample density, separated by gaps with a lower density [47]. These regions do not necessarily have a spherical shape in most practical applications. However, those clustering algorithms based on prototypes such that each sample is allocated to the nearest prototype according to the Euclidean distance tend to produce nearly spherical clusters. This is because the surfaces of equal Euclidean distance are spheres. Clustering methods of this kind include the k-means algorithm, the mixtures of Gaussians with homoscedastic covariance matrices, and most self-organizing neural networks including the standard SOM and the GNG. In these self-organizing neural networks, each neuron contains a prototype.

Hence it is important to realize that in this context each cluster of the dataset must be associated to a group of nearby prototypes. Irregularly shaped clusters can be adequately modeled by a graph of connections among prototypes, where each connected component of the graph is associated to a cluster, as recently proposed for mixtures of Gaussians[44]. The strategy of defining a cluster as a connected component in a graph has also been applied succesfully in recent times to nonparametric density estimation based clustering [37] and spectral clustering [45].

Interestingly enough, it has been recently found that the neurons of some cortical regions of the human brain are organized as densely connected components [13]. Other biological networks also exhibit the same kind of structure [14]. Hence, it can be said that an artificial neural network which learns connected components of neurons has a biological analogue.

The above reasons have led us to the development of a self-organizing neural model equipped with

a connection graph among neurons such that each cluster of the input dataset is represented by a connected component of the graph. The detection of the connected components in a graph is readily accomplished by searching for its spanning trees with Kruskal algorithm. This leads to an ensemble of spanning trees, i.e. a forest.

3. Definition of the self-organizing model

A Growing Neural Forest (GNF) is defined as a graph with a variable number of nodes (processing units) and edges (connections). Both nodes and edges are inserted and removed from the graph during the learning process. The current number of nodes is noted H. The training set for the graph is noted $\mathcal{S}$, with $\mathcal{S} \subset \mathbb{R}^D$, where D is the dimension of the input space. Each unit $i \in \{1, ..., H\}$ has an associated prototype $\mathbf{w}_i \in \mathbb{R}^D$ and an error variable $e_i \in \mathbb{R}$, $e_i \geq 0$. Each connection has an associated age, which is a non negative integer. The set of connections is noted $A \subseteq \{1, ..., H\} \times \{1, ..., H\}$, where no self-connections are allowed, $(i, i) \notin A$. It is also important to notice that A is an undirected graph, i.e. one cannot have $(i, j) \in A$ and $(j, i) \notin A$ at the same time.

Given the current set of connections, the Q connected components of the overall graph are called subgraphs, where $Q \leq H$. A spanning tree is associated to each subgraph, so that the set of the Q spanning trees forms a forest. The set of connections of the spanning trees is noted $\hat{A} \subseteq \{1, ..., H\} \times \{1, ..., H\}$, with $\hat{A} \subseteq A$. Please note that $\hat{A}$ is an undirected graph like A. The architecture of the proposed model is depicted in Figure 1.

The learning mechanism is inspired on the Growing Neural Gas [7], but it computes a spanning tree for each connected component (subgraph) of the overall graph, so that only those units which are connected to the winning unit in a spanning tree are updated. The learning algorithm is given by the following steps:

1. Start with two units ($H = 2$) joined by a connection, so that the graph contains only one subgraph which comprises both units. The connection set A only contains the connection between both units, and the set of the connections of the spanning trees $\hat{A}$ is initialized to A. Each prototype is initialized to a sample drawn at random from $\mathcal{S}$. The error variables are initialized to zero. The age of the connections are initialized to zero, too.

2. Draw a training sample $\mathbf{x}_n \in \mathbb{R}^D$ at random from $\mathcal{S}$. Please note that there can be a sample which is drawn two or more times. This is harmless and in fact it will happen for small datasets. The state of the network will not be the same, as the prototypes will have changed in the meantime.

3. Find the nearest unit q and the second nearest unit s in terms of Euclidean distance:

$$q = \arg \min_{i \in \{1,...,H\}} \|\mathbf{w}_i(n) - \mathbf{x}_n\| \quad (1)$$

$$s = \arg \min_{i \in \{1,...,H\} - \{q\}} \|\mathbf{w}_i(n) - \mathbf{x}_n\| \quad (2)$$

   If there are any ties in the computation of the first and second nearest units, then it is broken by choosing one of the tied units uniformly at random.

4. Increment by one the age of all edges of A which depart from q.

5. Add the squared Euclidean distance between $\mathbf{x}_n$ and the nearest unit q to the error variable e_q:

$$e_q(n+1) = e_q(n) + \|\mathbf{w}_q(n) - \mathbf{x}_n\|^2 \quad (3)$$

6. Update q and all its direct topological neighbors in its spanning tree with step size ϵ_b for unit q and ϵ_n for the neighbors, where $\epsilon_b > \epsilon_n > 0$:

$$\epsilon(n, i) = \begin{cases} \epsilon_b & \text{iff } n = q \\ \epsilon_n & \text{iff } (n \neq q) \wedge (n, q) \in \hat{A} \\ 0 & \text{iff } (n \neq q) \wedge (n, q) \notin \hat{A} \end{cases} \quad (4)$$

$$\mathbf{w}_i(n+1) = (1 - \epsilon(n, i))\,\mathbf{w}_i(n) + \epsilon(n, i)\,\mathbf{x}_n \quad (5)$$

   Please note that $(n, q) \in \hat{A} \Leftrightarrow (q, n) \in \hat{A}$ because $\hat{A}$ is an undirected graph.

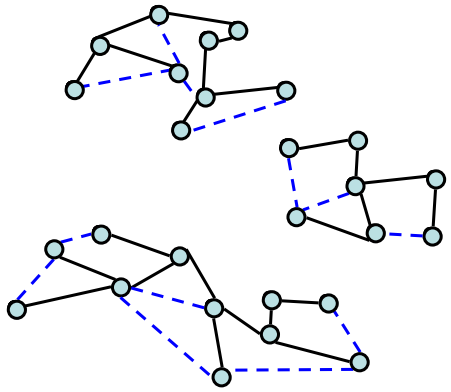


Figure 1: Structure of a GNF model with three subgraphs. The connections which do not belong the spanning trees are shown with dashed lines.

7. If q and s are connected by an edge in A, then set the age of this edge to zero. Otherwise, create an edge between q and s.

8. Remove edges in A with an age larger than a_{max}. Then remove all units which have no outgoing edges in A.

9. If the current time step n is an integer multiple of a parameter λ and the current number of units is lower than the maximum number of units H_{max}, then insert a new unit as follows. First determine the unit r with the maximum error among all units. After that, determine the unit z with the maximum error among all direct neighbors of r in A. Then create a new unit k, insert edges connecting k with r and z, and remove the original edge between r and z. After that, decrease the error variables e_r and e_z by multiplying them with a constant α, and initialize the error variable e_k to the new value of e_r. Setup the prototype of k to be halfway between those of r and z, as follows:

$$\mathbf{w}_k(n) = \frac{1}{2}\left(\mathbf{w}_r(n) + \mathbf{w}_z(n)\right) \tag{6}$$

   Finally, recompute the spanning trees by Kruskal algorithm [23] and assign $\hat{A}$ to the set of connections which belong to spanning trees.

10. Decrease all error variables e_i by multiplying them by a constant d with $0 < d < 1$.

11. If the maximum number of time steps has been reached, then halt. Otherwise, go to step 2.

It is worth noting that Kruskal algorithm is used because of its ability to compute the spanning trees of all the connected components of a graph at the same time. Also, it is important to notice that the algorithm can easily be adapted to online learning by acquiring each new incoming sample in Step 2. The learning rates ϵ_b and ϵ_n can also be chosen to decay as learning progresses. Linear or exponential decay rates could be used for this purpose.

The parameters of the above described algorithm are the following:

- The step size for the winning unit ϵ_b is a positive real number which controls how much the prototype of winning unit moves to the input sample.

- The step size for the neighbor units ϵ_n is a positive real number which controls how much the prototypes of the neighbors of the winning unit move to the input sample. We must have $\epsilon_b > \epsilon_n$ so that the neighbors move less than the winning unit.

- The a_{max} parameter is a positive integer which specifies how long an edge must have existed without having any input for which its ends are the first and second best matching units. The smaller a_{max}, the faster the old edges are pruned. Please note that the case $Q = H$ is possible, i.e. there is one connected component per neuron. However, it is undesirable because the model would not have learned any connectivity among the neurons. As the value of the a_{max} parameter increases this case becomes

unlikely because very few connections are removed. So the $Q = H$ case can be avoided by setting a_{max} to a large enough value.

- The λ parameter is a positive integer which establishes how often a new unit can be inserted into the network. The lower λ, the faster the network can grow.
- The maximum number of units H_{max} places a limit on the overall size of the network.
- The α parameter is a positive real number with $0 < \alpha < 1$. It controls the reduction in the error variables of the parent units r and z which give rise to a new unit. This is done so that r and z are not chosen again to create a new unit. The lower α, the larger the reduction of the error variables.
- The error variable decay parameter d is a positive real number with $0 < d < 1$. It controls how fast the quantization errors of the neurons are forgotten. The lower d, the faster the past errors are forgotten so that the error variables adapt faster to the new state of the network.

The computational complexity of the algorithm is derived as follows. For each input sample, the winning unit is searched (step 3), with cost $O(HD)$ where H is the current number of units. Steps 4-7 only process each neuron once, so they have complexity cost $O(HD)$. Step 8 has to process all edges, so the maximum cost is $O(H^2)$. Step 9 is executed once every λ input samples, and its cost is $O(HD + H^2 \log H)$, where the $H^2 \log H$ term comes from Kruskal algorithm. Finally, step 10 is executed in $O(H)$. For an input dataset with M samples, the overall running time is $O(H^2M + HDM + \frac{1}{\lambda}H^2 \log H)$. For large values of λ this is close to $O(H^2M + HDM)$, which is the complexity of the GNG algorithm.

The main differences of the GNF with respect to the GNG are the following. First of all, the GNF detects the connected components of the connection graph A and their associated spanning trees are computed, while the GNG does not analyze the structure of the connection graph in any way. Then the GNF uses the spanning trees to update the neuron prototypes at each learning step, so that only the neighbors of the winning neuron in its spanning tree are updated. On the other hand, the GNG updates all the neighbors of the winning neuron in the connection graph, no matter if the connections belong to the spanning tree. Since $\hat{A} \subseteq A$, the GNF restricts the number of neighbors to be updated with respect to the GNG, so that the discovery of tree structures is enhanced. To the best of our knowledge, no extension of this kind of the GNG has been attempted before.

An illustrative example of the GNF training algorithm is shown in Figure 2. In this example, a two-dimensional input distribution ($D = 2$) consisting of three balls is used as training data. The training process starts with two connected units ($H = 2$) whose prototypes are initialized to two input samples drawn uniformly at random. After $\lambda = 100$ iterations a new unit is inserted between the two initial units. Note that after λ iterations the prototypes and the error variables of these two units have been updated. At time step 200, each unit has been updated to represent a cluster and another new unit is inserted between the unit with the maximum error and its neighboring unit with maximum error. At time step 900, seven new units have been added and a connection between the biggest ball and the second biggest ball has been removed because its edge was larger than $a_{max} = 50$, so that two spanning trees are found. At time step $2,800$, more units have been inserted and updated, and the connection between the two smallest balls has also been removed. Thus three spanning trees are obtained, each representing one ball. Finally, the training algorithm halts after $20,000$ iterations and yields three spanning trees containing a grand total of $H_{max} = 50$ neurons that have been automatically adapted to the three balls present in the input distribution.

4. Experimental results

In this section, three different sets of experiments have been designed to assess the performance of the GNF model. Firstly, the experimental setup of all the experiments is presented together with a parameter sensitivity analysis (Subsection 4.1). Secondly, the self-organization capabilities of the GNF model to represent different input distributions are demonstrated in Subsection 4.2. Thirdly, in Subsection 4.3 unsupervised clustering experiments on manuscript digits and several standard becnhmark

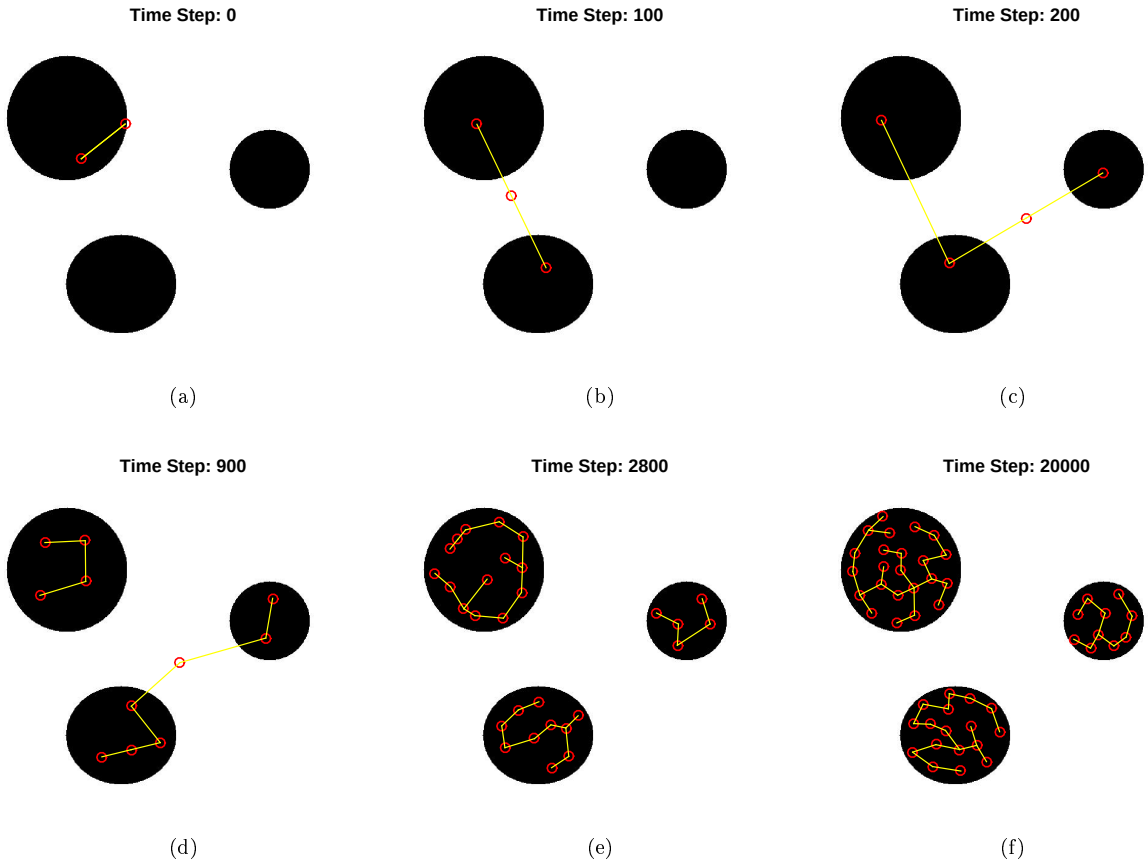


Figure 2: Example of the GNF training process for a two-dimensional input distribution at different time steps: (a) 0, (b) 100, (c) 200, (d) 900, (e) 2,800, (f) 9,000, and (g) 20,000.

data sets from the Machine Learning Repository[28] are reported. Finally, the versatility of the GNF model is demonstrated by applying this model to foreground detection in video sequences (Subsection 4.4).

4.1. *Experimental setup*

Next, the chosen setup for all of our experiments is defined. All the experiments reported on this paper have been carried out on a 64-bit Personal Computer with an Intel Core i7 2.90 GHz CPU, 8 GB RAM and standard hardware. The self-organization experiments are the only ones where synthetic data sets have been used to show the self-organization capabilities of our proposal. For this reason we have selected a fixed number of $M = 10,000$ input samples for each of these data sets and a number of 2 epochs, so that the number of time steps is $N = 20,000$. For the rest of experiments the number of input samples N depends on the data set at hand, although the number of 2 epochs is maintained.

Since the GNF is based on the GNG, the GNF has the same parameters than the GNG (see Section 3). Therefore, the GNF parameter setup is the same as recommended in the original GNG paper[7], whose values are provided in Table 1. Of course, the parameter setup is the same for the GNG training, which has been selected together with the Self-Organizing Map (SOM) model in order to provide comparative results.

The parameters for the SOM model have been chosen as follows. The topology is rectangular; the number of rows and columns is specified for each experiment in the subsections below. The neurons are initialized to training samples drawn uniformly at random from the training set. The neighborhood

Table 1: Parameter selection for the GNF and GNG models.

Parameter description	Values
Step size for the winning unit	$\epsilon_b = 0.2$
Step size for the neighbor unit	$\epsilon_n = 0.006$
Maximum edge for an edge	$a_{max} = 50$
New units insertion	$\lambda = 100$
Maximum number of units	$H_{max} = 50$
Reduction of the error variables	$\alpha = 0.5$
Error variable decay	$d = 0.995$

function is Gaussian with initial neighborhood radius set to half the largest side of the rectangular topology, and final neighborhood radius set to 1. The learning rate is 0.5 for the ordering phase and 0.05 for the fine tuning phase. The training is done in sequential mode just like the GNG and the GNF, i.e. not batch mode.

A sensitivity analysis of GNF parameters has been conducted. To this end, the values of each parameter are varied whereas the rest of the parameters are fixed to the values shown in Table 1. The number of epochs parameter has also been taken into account. In order to analyze the parameter sensitivity a two-dimensional input distribution ($D = 2$) is proposed, which consists on three balls. This data set is the same which was used in Figure 2. The GNF vector quantization performance is evaluated by means of the mean squared error (MSE), which is a measure of the intra-cluster variance that must be minimized in a clustering task. The MSE is expressed as follows (lower is better):

$$MSE = \frac{1}{M} \sum_{i=1}^{M} \|\mathbf{w}_i - \mathbf{x}_i\|^2 \qquad (7)$$

where M is the number of samples in the distribution, $\mathbf{x}_i$ is the i-th input sample and $\mathbf{w}_i$ the prototype of the winning neuron corresponding to x_i.

The MSE results for each parameter are shown in Figure 3. In these plots, the sensitivity of each GNF parameter is evaluated for ten values of the tested parameter, where the average MSE for ten runs and the standard deviations are given. As seen, the GNF exhibits a stable behavior for all of the parameters. Please note that the selected values for our experiments, which are recommended in the original GNG paper[7], have low MSE values. These values are not the lowest but they feature low standard deviations respective to the magnitude of the MSE. It is worth noting that the MSE results for one parameter can vary if the values of the rest of parameters are changed. As seen in the experiments below, the recommended values for the GNF parameters are suitable for many unsupervised learning tasks, so that adaptation of the GNF to new applications or data sets is made easy as in the case of the GNG algorithm.

4.2. *Self-Organization experiments*

The first set of experiments is designed to show the self-organization capabilities of the GNF model by adapting its architecture to the shape of different input distributions. Hence, nine different input distributions were used for our experiments. Four of them are two-dimensional input distributions ($D = 2$), namely, the shape of an 'M' letter, the shape of a non-hollow square, two irregular separated shapes and the shape of an ellipse. Other four input distributions are three-dimensional input distributions ($D = 3$) called 'Swiss Roll', 'Swiss Hole', 'Toroidal Helix', and 'Ueda's Spiral' due to their shapes. The last input distribution represents two six-dimensional hyperspheres ($D = 6$) in order to benchmark the dimensional scaling of our proposal before high-dimensional data. Both two-dimensional and three-dimensional input distributions are shown in Figures 4 and 5, respectively. The six-dimensional input distribution is also shown in Figure 6 by plotting two dimensions at a time.

For each input distribution $M = 10,000$ randomly drawn input samples were presented to the GNF model during $N = 20,000$ time steps (2 epochs). The unfolding of the GNF models can be observed in Figures 7-9, where the supports of the input distributions are in black, the neurons are represented by red circles and the connections among neurons are plotted with yellow straight lines. Note that in these plots, the GNF architectures have a forest structure with a variable number of subgraphs

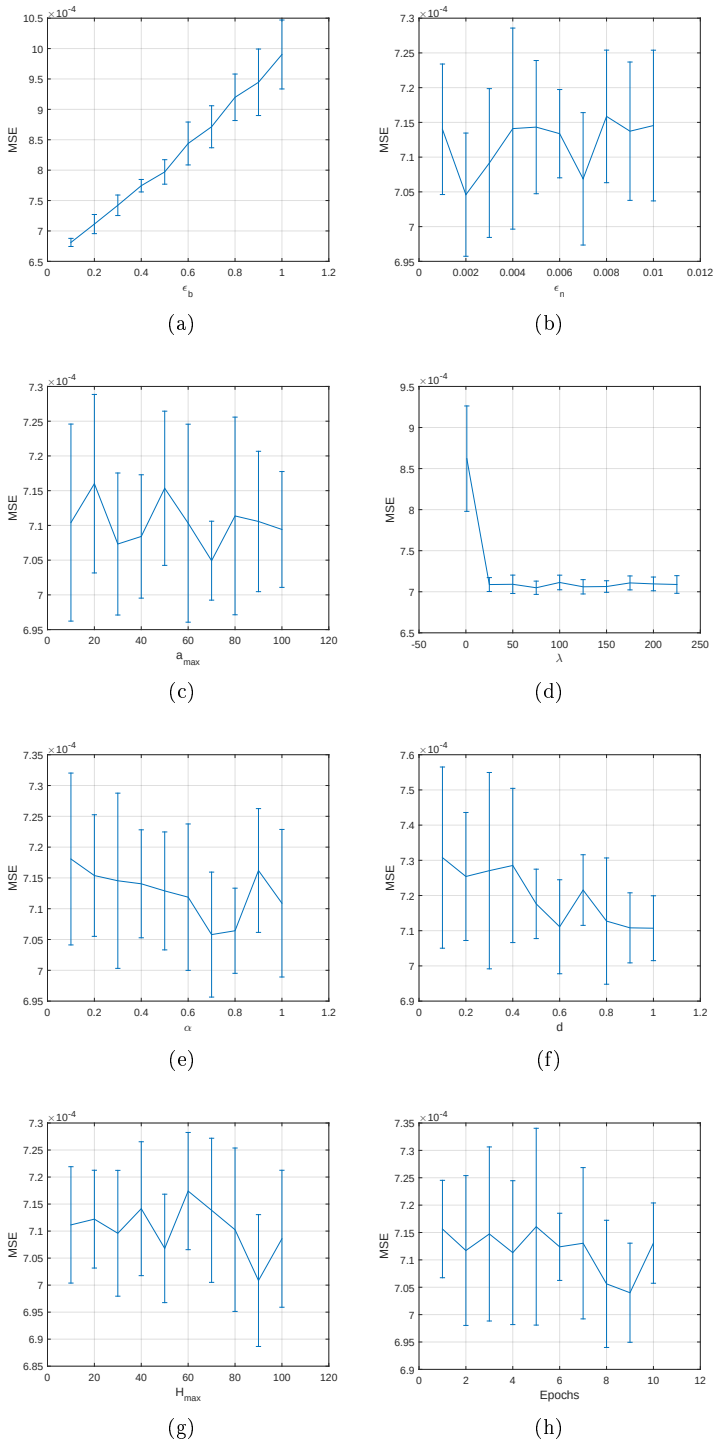
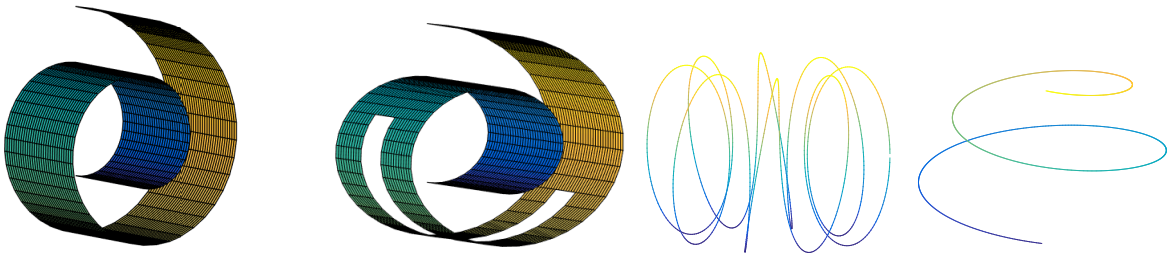


Figure 3: Average MSE and standard deviations after training the GNF with a two-dimensional input distribution using ten different values of the following parameters: (a) ϵ_b, (b) ϵ_n, (c) a_{max}, (d) λ, (e) α, (f) d, and (g) number of epochs.

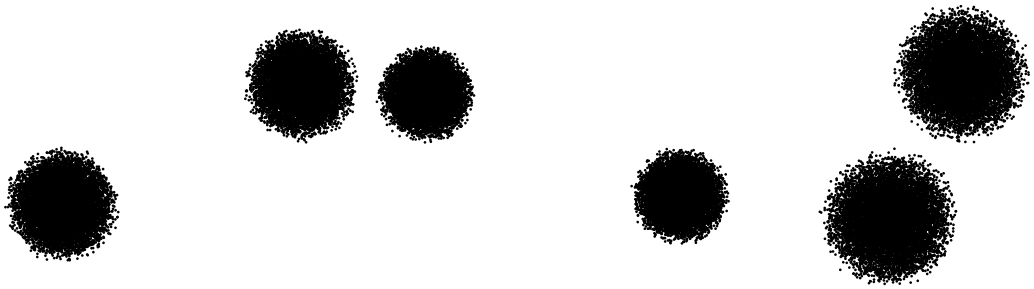
(a) (b) (c) (d)

Figure 4: Two-dimensional input distributions used for self-organization experiments: (a) 'M' letter, (b) non-hollowed square, (c) two irregular shapes, and (d) ellipse.



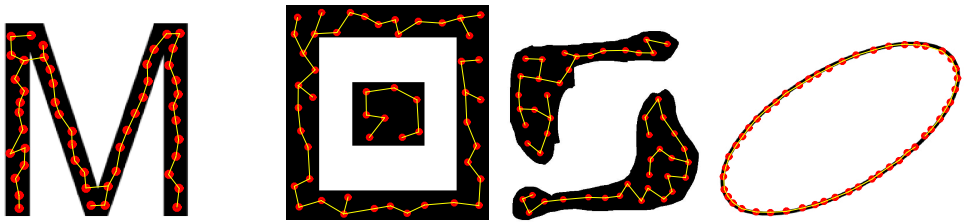
(a) (b) (c) (d)

Figure 5: Three-dimensional input distributions used for self-organization experiments: (a) 'Swiss Roll', (b) 'Swiss Hole', (c) 'Toroidal Helix', and (d) 'Ueda's Spiral'.



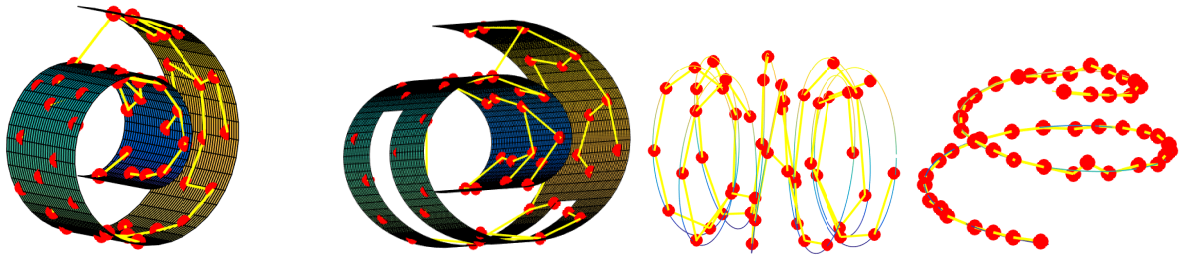
(a) (b) (c)

Figure 6: Six-dimensional input distribution consisting on two hyperspheres used for self-organization experiments. The following pairwise dimensions are plotted: (a) first vs. second, (b) third vs. fourth, and (c) fifth vs. sixth.



(a) (b) (c) (d)

Figure 7: GNF self-organization results for two-dimensional input distributions: (a)'M', (b) 'non-hollowed square', (c) 'two irregular shapes', and (d) 'Ellipse'.



(a) (b) (c) (d)

Figure 8: GNF self-organization results for three-dimensional input distributions: (a) 'Swiss Roll', (b) 'Swiss Hole', (c) 'Toroidal Helix', and (d) 'Ueda's Spiral'.

(a) (b) (c)

Figure 9: GNF self-organization results for the six-dimensional input distribution consisting on two hyperspheres. The following pairwise dimensions are plotted: (a) first vs. second, (b) third vs. fourth, and (c) fifth vs. sixth.

depending on the shape to represent. For each subgraph only its associated spanning tree is drawn.

By observing the resulting GNF forests, we can see how this model is able to appropriately adapt its architecture to the shape of each input distribution. In the case of the non-hollow square input distribution, the GNF creates two spanning trees, one of them to represent the inner square and the other one for the outer square. The same comment can be stated for the two irregular shapes input distribution, where the two shapes are separated from one another. For the 'Swiss Hole' shape, the GNF model generates only one spanning tree, which correctly fits to the shape without placing neurons on the hole. We can see how the GNF is also able to represent correctly other complex distributions such as the Toroidal Helix and Ueda's Spiral.

Moreover, quantitative results are provided for each input distribution. In particular, the peak signal-to-noise ratio (PSNR) was computed. The PSNR is defined as follows (in dB, higher is better):

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{8}$$

where MAX_I^2 is the squared Euclidean norm of the vector which joins the two furthest points in the input distribution. The mean squared error (MSE) is expressed in Eq. (7).

We have also used the Davis-Bouldin index (DB) as a metric for evaluating clustering algorithms based on a ratio of within-cluster and between-cluster distances. The Davis-Bouldin index is defined as follows (lower is better):

$$DB = \frac{1}{K} \sum_{i=1}^{K} max_{j \neq i} \{D_{ij}\} \tag{9}$$

where D_{ij} is the within-to-between cluster distance ratio for the ith and jth clusters:

$$D_{ij} = \frac{(\bar{d}_i + \bar{d}_j)}{d_{ij}} \tag{10}$$

where $\bar{d}_i$ is the average distance between each point in the ith cluster and the centroid of the ith cluster; $\bar{d}_j$ is the average distance between each point in the ith cluster and the centroid of the jth cluster; and the d_{ij} is the Euclidean distance between the centroids of the ith and jth clusters.

Finally, the silhouette (S) has been included in our evaluations, which is a measure of how appropriately the data have been clustered. The silhouette of the i-th input sample is defined as follows (higher is better):

$$s_i = \left(\frac{b_i - a_i}{max\{a_i, b_i\}} \right) \tag{11}$$

where a_i is the average dissimilarity of the i-th input sample with the rest of samples within the same cluster and b_i is the lowest dissimilarity of the i-th input sample to any other cluster of which is not a member. Note that s_i is a number in the interval $[-1, 1]$ where the closer to 1 the better the sample is clustered. We call S the average silhouette over all data. Both DB and S indexes have been widely used for the performance evaluation of clustering experiments[4,19,6,11].

Our results have been compared with two key self-organizing models in the literature in which our proposal is based, namely, the Self-Organizing Map (SOM)[20] and the Growing Neural Gas (GNG)[7]. The GNG setup is the same as the GNF (see Subsection 4.1), whereas the SOM topology was initialized to a grid of 10×5 neurons so that the three models have the same number of neurons. The MSE, PSNR, DB, S and CPU time (in seconds) for each input distribution and method are given in Table 2. By observing this table, similar results for the GNF and GNG are found, whereas poor results are yielded for the SOM in terms of MSE, PSNR, DB and S due to its fixed topology. The SOM only overcomes the GNF and GNG for the CPU time since it is a simpler model to train. Note that the GNF achieves the best results for more complex distributions as Ueda's Spiral and hyperspheres in six dimensions.

4.3. *Clustering experiments*

This set of experiments is devoted to illustrate the performance of the GNF model in a typical unsupervised clustering task. In addition to that, the capabilities of the GNF to visualize high-dimensional data are demonstrated since this model can discover the inherent structure of high-dimensional data. The chosen data sets have class label information for each sample, but these labels are not provided to the GNF, so this is not a supervised classification task. Therefore this is an unsupervised clustering

Table 2: MSE, PSNR, DB, S and CPU time (in seconds) results for self-organization experiments of the GNF, GNG and SOM models. Best results are in bold.

Data set	Method	MSE	PSNR	DB	S	CPU Time
'M' letter	GNF	1.1×10^{-3}	27.60	0.83	-0.12	1.43
	GNG	$\mathbf{1.1 \times 10^{-3}}$	**27.67**	**0.82**	**-0.11**	1.39
	SOM	4.1×10^{-3}	22.06	5.40	-0.40	**0.54**
Non-hollowed square	GNF	2.5×10^{-3}	26.01	0.81	**-0.12**	1.40
	GNG	$\mathbf{2.5 \times 10^{-3}}$	**26.10**	**0.80**	**-0.12**	1.39
	SOM	6.8×10^{-3}	21.68	4.34	-0.27	**0.58**
Two irregular shapes	GNF	$\mathbf{1.4 \times 10^{-3}}$	**28.19**	0.82	**-0.11**	1.43
	GNG	1.4×10^{-3}	28.11	**0.78**	-0.13	1.36
	SOM	3.8×10^{-3}	23.81	10.83	-0.14	**0.52**
Ellipse	GNF	$\mathbf{2.9 \times 10^{-4}}$	**34.53**	**0.51**	**-0.13**	1.38
	GNG	2.9×10^{-4}	34.5	**0.51**	**-0.13**	1.32
	SOM	1.3×10^{-3}	27.97	7.13	-0.46	**0.45**
'Swiss Roll'	GNF	6.85	21.76	0.89	**-0.09**	1.47
	GNG	**6.67**	**21.87**	**0.86**	**-0.09**	1.36
	SOM	17.72	17.63	5.17	-0.11	**0.81**
'Swiss Hole'	GNF	6.25	22.16	**0.86**	**-0.09**	1.45
	GNG	**6.20**	**22.19**	0.87	-0.10	1.40
	SOM	17.12	17.78	2.64	-0.17	**0.54**
'Toroidal Helix'	GNF	0.10	25.86	0.61	**0.62**	1.41
	GNG	**0.10**	**25.87**	**0.59**	**0.62**	1.41
	SOM	0.68	17.63	7.37	0.25	**0.46**
'Ueda's Spiral'	GNF	**1.12**	**28.45**	**0.69**	0.32	1.37
	GNG	1.15	28.25	0.71	**0.34**	1.53
	SOM	6.82	20.76	7.15	0.23	**0.53**
'Hyperspheres'	GNF	**0.32**	**24.00**	**1.58**	**-0.04**	1.68
	GNG	0.34	23.73	2.14	**-0.04**	1.69
	SOM	0.48	22.19	39.16	-0.25	**1.16**

task where the GNF is required to find the clusters without any class label information about the training samples. All the selected data sets for these experiments are widely used standard benchmark data sets in the machine learning and pattern recognition areas.

The first selected data set is the MNIST database of handwritten digits[25], which has a training data set of $M = 60,000$ input samples and a dimensionality of $D = 784$. Two versions of this data set have been taken into account by applying or not a dimensionality reduction to data. This dimensionality has been reduced to $D = 15$ by means of Principal Components Analysis (PCA) prior to training. The proportion of variance explained when 15 principal components are used is 58.63%. The training was done during $N = 120,000$ time steps (2 epochs). This time the maximum number of neurons has been reduced to $H_{max} = 20$ in order to provide a better visualization of the resulting GNF architectures, which are shown in Figure 10, for even digits, odd digits and all the digits, respectively.

Again, quantitative results are provided by means of the MSE Eq. (7), PSNR Eq. (8), DB Eq. (9), S Eq. (11) and CPU time (in seconds). These performances measures are given in Table 3 for the GNF, GNG and SOM models and each data set with PCA or without PCA. Best results are in bold. The mean and standard deviations for 10 folds were computed. By observing this table, we can see how our proposal obtain the best results according to several clustering performance measures and for both data sets with PCA and without PCA. The best CPU time is achieved by the SOM as expected due to the simplicity of this model. Nevertheless, the CPU time of our proposal is lower than the GNG CPU time.

The second experiment is carried out by using eight different standard benchmark data sets from the UCI Machine Learning Repository[28], namely, Balance Scale, Breast Cancer Wisconsin, Cloud, Contraceptive, Dermatology, Liver, Vowel and Wine. The number of input samples (M) and dimension (D) is different for each input data set, but the number of neurons (H_{max}), epochs and the rest of parameters of the GNF and GNG models are the same than used in previous experiments. We have run 10 folds for each data set and model.

The mean and standard deviations for each clustering performance measure (MSE, PSNR, DB, S and CPU time) are shown in Table 4. Note that the GNF obtains the best results, except for Dermatology and Wine data sets according only to the MSE and PSNR measures (for DB and S the GNF still achieves the best results). According to the CPU time, the GNF and GNG overcome the SOM model for the selected data sets. Therefore, these results confirm the suitability of our proposal for clustering tasks.

4.4. *Foreground detection experiments*

Computer vision is a common application of learning models[40,17]. The last set of experiments aims to show the versatility of the proposed model by illustrating its validity in a challenging area such as foreground detection. This task has been carried out by self-organizing neural networks in the past [29]. This can be achieved by training the GNF model with data extracted from several frames from a video sequence to learn the background of that sequence. Feature extraction consists on getting the three color components of each pixel of a frame together with its position, so that five-dimensional data sets are extracted. Then the trained model can be tested with data belonging to a new frame of the same video sequence where foreground objects are present. Finally, foreground pixels are detected by comparing the error difference commited between training and test, as proposed in[39], which is based on the principle that a neuron assigned to a pixel (input sample) during the test should be close to the neuron assigned to the same pixel during the training in case this pixel belongs to the background of the scene.

Four different video sequences from well known public repositories were selected for our experiments. These video sequences represent real situations both indoor and outdoor scenarios, and are provided with the ground truth information, which consists of a set of manually segmented images where the sets of pixels belonging to the foreground and the background are defined. The list of video sequences used in this set of experiments is given in Table 5. They come from VSSN 2006[27], the Institute for Infocomm Research[b], and the CAVIAR dataset[c].

[b] `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html`
[c] `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`

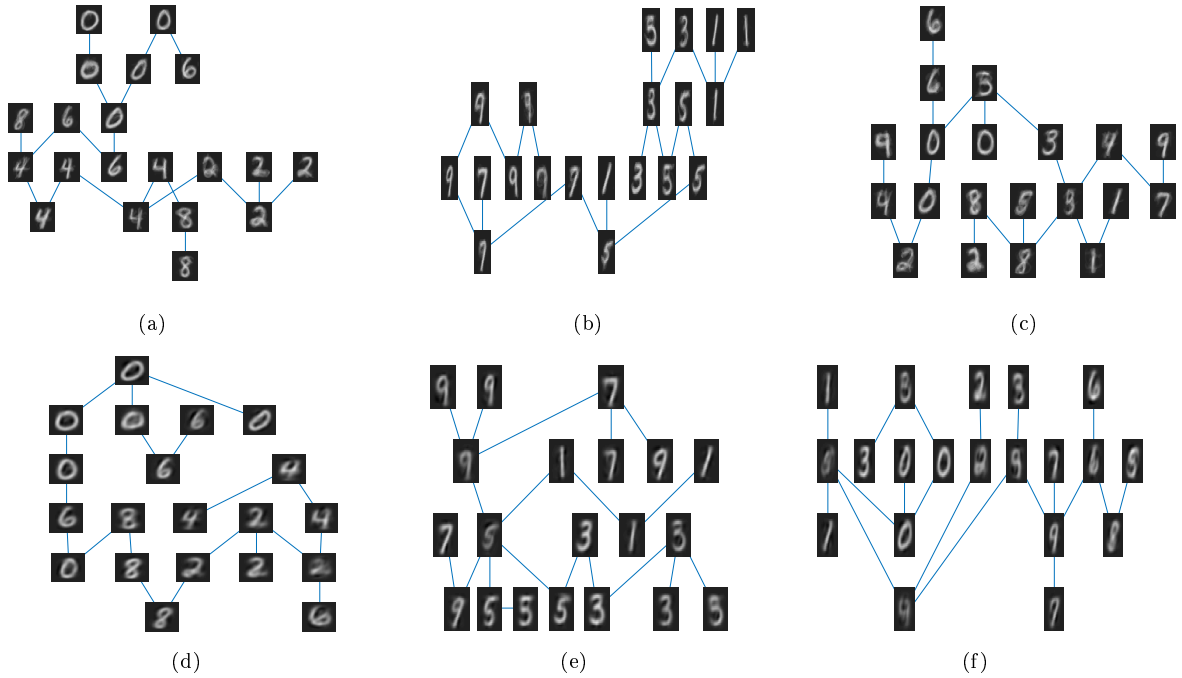


Figure 10: GNF architectures for the MNIST database: (a) even digits, (b) odd digits, (c) all digits, (d) even digits with PCA, (e) odd digits with PCA and, (f) all digits with PCA.

Table 3: MSE, PSNR, DB, S and CPU time (in seconds) results for clustering experiments of the MNIST data set for the GNF, GNG and SOM models. Best results are in bold. Standard deviations are shown in parentheses.

Data set	Method	MSE	PSNR	DB	S	CPU Time
Even digits	GNF	40.78 (0.19)	-16.10 (0.02)	**2.45 (0.05)**	**0.10 (0.00)**	18.63 (0.44)
	GNG	**40.62 (0.32)**	**-16.09 (0.03)**	2.93 (0.09)	0.10 (0.01)	26.20 (4.32)
	SOM	43.59 (0.12)	-16.39 (0.01)	11.52 (0.17)	0.06 (0.00)	**14.18 (0.53)**
Odd digits	GNF	**31.04 (0.24)**	**-14.92 (0.03)**	**2.32 (0.08)**	**0.13 (0.01)**	18.19 (0.62)
	GNG	31.06 (0.23)	-14.92 (0.03)	2.79 (0.09)	0.12 (0.00)	21.70 (1.73)
	SOM	33.69 (0.06)	-15.27 (0.01)	10.92 (0.29)	0.07 (0.01)	**14.63 (0.69)**
All digits	GNF	**39.20 (0.35)**	**-15.93 (0.04)**	**2.42 (0.07)**	**0.10 (0.01)**	**17.95 (3.01)**
	GNG	39.24 (0.22)	-15.94 (0.02)	3.10 (0.14)	0.09 (0.01)	23.25 (1.50)
	SOM	41.91 (0.05)	-16.22 (0.01)	12.12 (0.07)	0.06 (0.00)	28.86 (2.00)
Even digits (PCA)	GNF	**16.14 (0.10)**	**11.62 (0.03)**	**1.80 (0.05)**	**0.19 (0.01)**	11.86 (0.29)
	GNG	16.34 (0.20)	11.57 (0.05)	2.19 (0.12)	0.19 (0.01)	8.68 (0.33)
	SOM	20.61 (0.10)	10.56 (0.02)	8.01 (0.16)	0.12 (0.01)	**1.57 (0.18)**
Odd digits (PCA)	GNF	**12.43 (0.18)**	**11.72 (0.06)**	2.42 (0.09)	**0.25 (0.01)**	12.10 (0.46)
	GNG	12.44 (0.12)	11.72 (0.04)	**2.02 (0.04)**	0.24 (0.01)	8.61 (0.15)
	SOM	16.06 (0.06)	10.61 (0.02)	7.68 (0.20)	0.16 (0.01)	**1.72 (0.15)**
All digits (PCA)	GNF	**15.57 (0.17)**	**11.70 (0.05)**	**1.85 (0.06)**	**0.21 (0.01)**	10.42 (1.74)
	GNG	15.92 (0.19)	11.61 (0.05)	2.30 (0.09)	0.20 (0.01)	9.03 (0.62)
	SOM	19.87 (0.08)	10.64 (0.02)	8.41 (0.06)	0.13 (0.01)	**4.12 (0.31)**

Table 4: MSE, PSNR, DB, S and CPU time (in seconds) results for clustering experiments of different UCI data sets for the GNF, GNG and SOM models. Best results are in bold. Standard deviations are shown in parentheses.

Data set	Method	MSE	PSNR	DB	S	CPU Time
BalanceScale	GNF	**2.48 (0.06)**	**8.09 (0.10)**	**1.38 (0.07)**	**0.29 (0.01)**	**0.01 (0.01)**
	GNG	2.57 (0.05)	7.94 (0.09)	1.54 (0.07)	**0.29 (0.01)**	**0.01 (0.01)**
	SOM	3.34 (0.03)	6.80 (0.04)	7.43 (0.67)	-0.10 (0.04)	0.02 (0.04)
BreastCancerWisconsin	GNF	**15.10 (0.38)**	**7.30 (0.11)**	**1.90 (0.12)**	**0.32 (0.05)**	**0.01 (0.01)**
	GNG	15.12 (0.24)	7.29 (0.07)	2.04 (0.10)	0.32 (0.06)	**0.01 (0.01)**
	SOM	18.83 (0.41)	6.34 (0.09)	17.71 (7.07)	-0.08 (0.04)	0.02 (0.03)
Cloud	GNF	**2077.50 (88.92)**	**37.27 (0.19)**	**8.88 (7.01)**	**0.49 (0.01)**	**0.00 (0.01)**
	GNG	2103.04 (108.39)	37.22 (0.22)	10.17 (6.13)	0.49 (0.02)	**0.00 (0.01)**
	SOM	18925.26 (1475.88)	27.69 (0.34)	23.15 (5.47)	0.33 (0.02)	0.01 (0.03)
Contraceptive	GNF	**4.85 (0.10)**	**26.95 (0.09)**	**1.50 (0.05)**	**0.25 (0.01)**	**0.00 (0.01)**
	GNG	4.90 (0.06)	26.90 (0.05)	1.62 (0.05)	0.23 (0.01)	**0.00 (0.01)**
	SOM	6.85 (0.13)	25.45 (0.08)	4.62 (0.28)	0.03 (0.01)	0.01 (0.02)
Dermatology	GNF	30.51 (0.98)	22.66 (0.14)	**1.46 (0.12)**	0.32 (0.03)	**0.00 (0.01)**
	GNG	30.93 (0.84)	22.60 (0.12)	1.48 (0.11)	**0.33 (0.03)**	**0.00 (0.01)**
	SOM	**28.37 (1.15)**	**22.98 (0.18)**	13.61 (2.90)	0.10 (0.03)	0.01 (0.03)
Liver	GNF	**579.47 (24.25)**	**21.83 (0.18)**	**1.38 (0.19)**	**0.35 (0.05)**	**0.00 (0.01)**
	GNG	579.69 (25.16)	21.83 (0.19)	1.45 (0.10)	0.34 (0.05)	**0.00 (0.01)**
	SOM	696.33 (55.28)	21.04 (0.35)	10.07 (3.44)	-0.11 (0.03)	0.01 (0.03)
Vowel	GNF	**1.60 (0.03)**	**18.20 (0.09)**	**1.52 (0.05)**	**0.30 (0.01)**	**0.00 (0.01)**
	GNG	1.67 (0.02)	18.02 (0.04)	1.76 (0.07)	0.28 (0.01)	**0.00 (0.01)**
	SOM	2.27 (0.03)	16.69 (0.06)	8.25 (0.88)	-0.01 (0.03)	0.01 (0.03)
Wine	GNF	6372.95 (1130.51)	26.52 (0.72)	**0.63 (0.08)**	**0.67 (0.04)**	**0.00 (0.01)**
	GNG	6471.00 (346.25)	26.40 (0.23)	0.66 (0.07)	0.66 (0.03)	**0.00 (0.01)**
	SOM	**4416.74 (674.95)**	**28.10 (0.67)**	55.39 (19.79)	0.18 (0.04)	0.01 (0.03)

Table 5: List of sequences used for the experiments.

Sequence name	Source	Description
Video2	VSSN 2006	3D objects are artificially inserted to detect the foreground easily
Meeting room	Institute for Infocomm Research	Camouflage, cast shadows and illumination changes
Water surface	Institute for Infocomm Research	Camouflage, cast shadows and illumination changes
OneShopOneWait1cor	CAVIAR dataset	Busy corridor

In order to build a training dataset, we compute the median of 100 frames for each video sequence and color space. Note that by computing the median of several frames, pixel values represent the background of the scene. This way, we do not need the presence of background-only frames in the video sequences. Then, a feature extraction from each pixel of the median frame was performed. Thus, the three color components and the position (i, j) of each pixel were taken into account to build an input sample. Therefore, a five-dimensional data set ($D = 5$) was extracted from the median frame. The number of training samples M depends on the number of frame pixels of the video sequence, where each input sample represent a pixel of that median frame. Therefore, according to the resolution of the frames of each video sequence, the number of training input samples were $M = 92,160$, $M = 20,480$, $M = 20,480$, and $M = 110,592$ for the 'Video2', 'Meeting room', 'Water surface', and 'OneShopOneWait1cor' video sequences, respectively. The number of test input samples is the same than the number of training input samples for each video sequence since a frame from the same video sequence is presented to the model during test.

In addition to the RGB color space, five traditional color spaces were selected for the study: Lab, Luv, HSV, HSL and YCbCr. Both the CIELAB space, named as Lab, and the Luv were established in 1976 by the Commission Internationale de l'Eclairage (CIE). HSV (Hue, Saturation, Value) was developed in the 1970s for computer graphics applications and it is one of the most common cylindrical model. Similar color spaces, like HSB (B for Brightness), HSL (L for Lightness) or HSI (I for Intensity) were also developed to balance the advantages and disadvantages of the previous one. These cylindrical models are useful because they are more intuitive than the RGB. The last selected color space YCbCr, also written as YCBCR and sometimes abbreviated YCC, is commonly used in video and photography systems and it is composed by the luma component (Y) and the blue and red difference chroma component (Cb and Cr, respectively).

As stated above, foreground detection is carried out by comparing the error committed between training and test. Let $\mathbf{x}_k$ be a pixel in a frame and γ_k be the error difference between the training error and the test error as follows:

$$\gamma_k = ||\mathbf{w}_q^{training} - \mathbf{x}_k|| - ||\mathbf{w}_q^{test} - \mathbf{x}_k|| \tag{12}$$

The pixel $\mathbf{x}_k$ is declared to belong to the foreground of the scene if the absolute value of the error difference γ_k is higher than a threshold, as expressed in Eq. (13):

$$PredictedClass(\mathbf{x}_k) = \begin{cases} foreground & \text{iff } |\gamma_k| > T_k \\ background & otherwise \end{cases} \tag{13}$$

where $T_k = \bar{\gamma_k} + \sigma_k$, that is, the mean error difference $\bar{\gamma_k}$ plus the standard deviation of the error difference σ_k. Note that this threshold is unique for each pixel. This way the classifier is more robust against noise, as required in most classification applications of neural networks [8,10,9,3]. Foreground detection results of the GNF model using six color spaces are shown in Figures 11 and 12.

Moreover, six standard performance measures are used to provide quantitative results, as done in previous literature [33,29,34]. First the set of pixels corresponding to foreground A and the set of pixels classified as foreground by the GNF model B are defined:

$$A = \{\mathbf{x}_k \mid \chi(\mathbf{x}_k) = 1\} \tag{14}$$

$$B = \{\mathbf{x}_k \mid \widetilde{\chi}(\mathbf{x}_k) = 1\} \tag{15}$$

where $\chi(\mathbf{x}_k)$ and $\widetilde{\chi}(\mathbf{x}_k)$ are defined as:

$$\chi(\mathbf{x}_k) = \begin{cases} 1 & \text{iff } RealClass(\mathbf{x}_k) \in foreground \\ 0 & otherwise \end{cases} \tag{16}$$

$$\widetilde{\chi}(\mathbf{x}_k) = \begin{cases} 1 & \text{iff } PredictedClass(\mathbf{x}_k) \in foreground \\ 0 & otherwise \end{cases} \tag{17}$$

where $RealClass(\mathbf{x}_k)$ and $PredictedClass(\mathbf{x}_k)$ stand for the real and predicted class labels of pixel $\mathbf{x}_k$, respectively.

False negatives (FN) and false positives (FP) rates are also used in this work, which are defined as follow (lower is better):

$$FN = \frac{\text{card}\left(A \cap \overline{B}\right)}{\text{card}\left(A \cup B\right)} \tag{18}$$

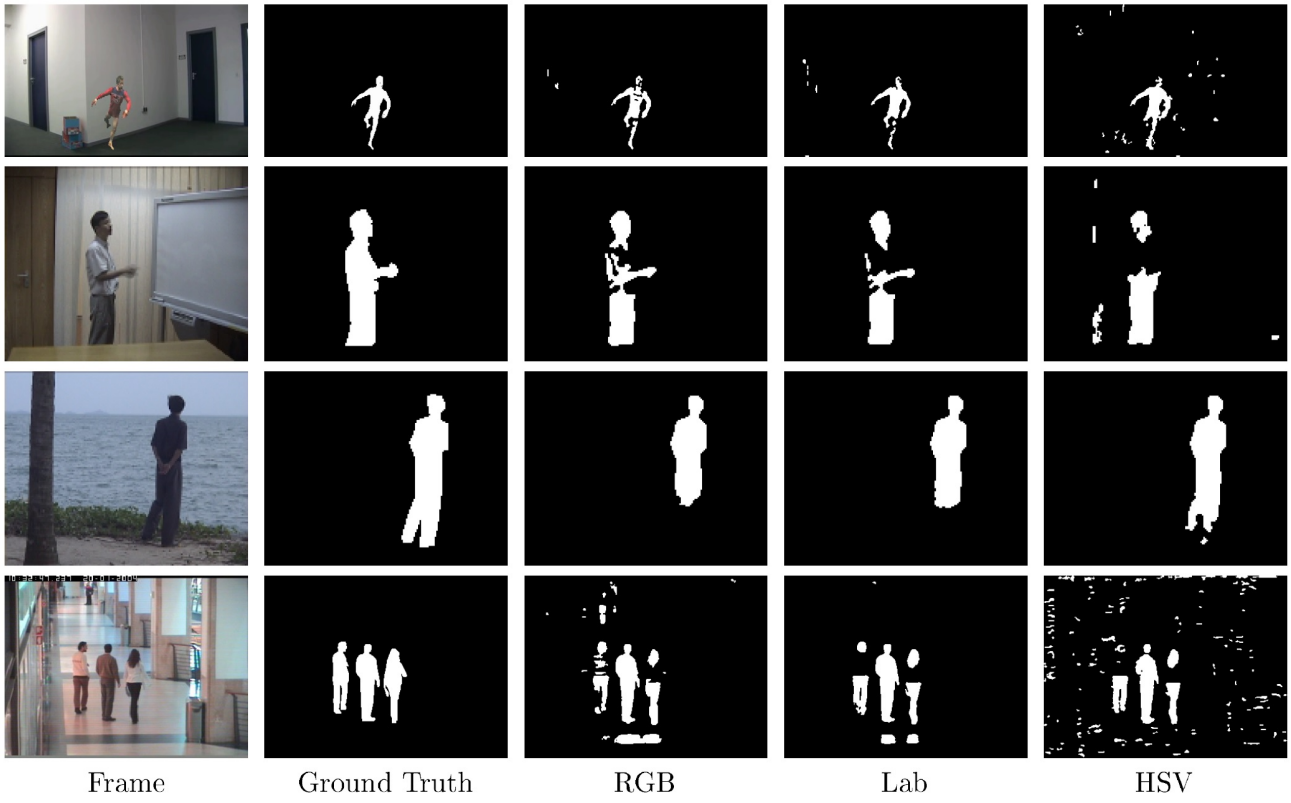


Figure 11: Foreground detection results of the GNF model for RGB, Lab and HSV color spaces. Rows from top to bottom: Video2 (frame 526), Meeting Room (frame 23242), WaterSurface (frame 1577), and OneShopOneWait1cor (frame 375). The first two columns show the original frame and Ground Truth, respectively. The last three columns show the corresponding tested color spaces.

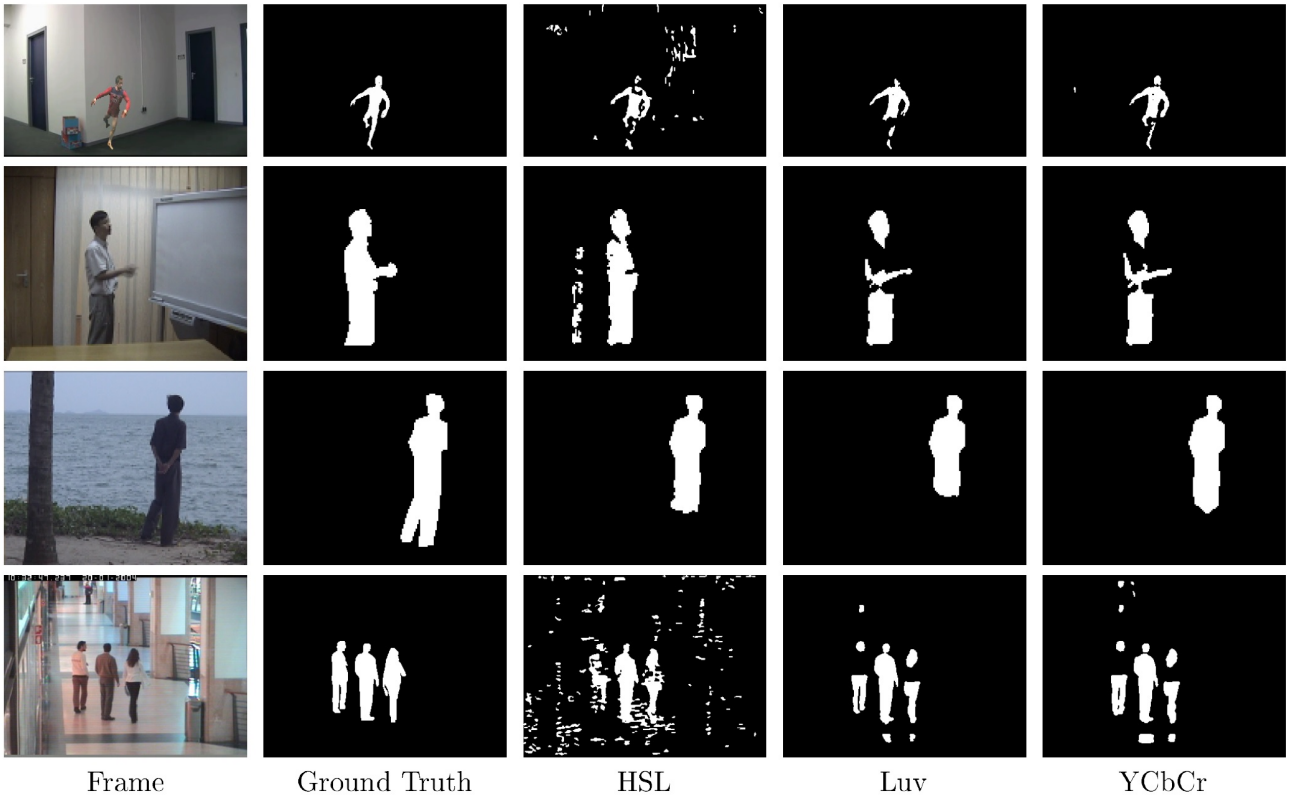


Figure 12: Foreground detection results of the GNF model for HSL, Luv and YCbCr color spaces. Rows from top to bottom: Video2 (frame 526), Meeting Room (frame 23242), WaterSurface (frame 1577), and OneShopOneWait1cor (frame 375). The first two columns show the original frame and Ground Truth, respectively. The last three columns show the corresponding tested color spaces.

$$FP = \frac{\text{card}\left(\overline{A} \cap B\right)}{\text{card}\left(A \cup B\right)} \tag{19}$$

where 'card' stands for the number of elements of a set. Other measures we have used in these experiments are precision (PR), recall (RC) and the F-measure, which combines PR and RC (higher is better):

$$PR = \frac{\text{card}\left(A \cap B\right)}{\text{card}\left(B\right)} \tag{20}$$

$$RC = \frac{\text{card}\left(A \cap B\right)}{\text{card}\left(A\right)} \tag{21}$$

$$PR,\ RC \in [0,1] \tag{22}$$

$$FM = \frac{2 \cdot PR \cdot RC}{PR + RC} \tag{23}$$

The last measure taken into account is accuracy (AC) defined as follows (higher is better):

$$AC = \frac{\text{card}\left(A \cap B\right)}{\text{card}\left(A \cup B\right)} \tag{24}$$

The evaluation performance of the GNF detection for the six performance measures and the CPU time is given in Tables 6-12. The CPU time is computed as the training time plus the mean of foreground detection time. In these tables, best results for each video sequence are in bold. Note that the space color that obtains the best results depend on the video sequence at hand.

Our best results have been compared with the best results achieved by several pixel-level methods, since our proposal analyzes the scene pixel by pixel. The first method is Pfinder [48] (called here WrenGA), which features a single Gaussian. Three mixture Gaussian approaches have been also taken into account: GrimsonGMM [46], GMMV1 [18] and GMMV2 [50]. Another method is FASOM [35], which is a fuzzy approach to a self-organizing background substraction (SOBS) algorithm. The last method is MFBM [33], which is a probabilistic mixture model which handles any number of pixel features and accounts for the correlations among these features. Results from these methods have been obtained from [33], which are the best results optimizing parameters and feature vectors for each method.[d] Furthermore, the set of parameters which yields best results for each video sequence is taken, although this set of parameters is not the same for other sequences.

Accuracy and F-measure results for the GNF and the six aforementioned methods are provided in Tables 13 and 14, respectively. Our proposal yields the best results for the 'Video2' sequence, whereas it remains close to the best results for the rest of sequences. Standard methods like GMMV1 can get confused when the background is complex and dynamic, so that they learn parts of the foreground rather than the background. This is the reason of the large differences in accuracy. This can be concluded by observing the Rank sums in both tables (last rows), where the GNF obtained the second overall best Accuracy and F-measure results. Therefore, not only the GNF is able to perform foreground detection tasks in difficult scenarios, but also achieves competitive results when compared to well known foreground detection algorithms.

5. Conclusions

A novel self-organizing model, that we call Growing Neural Forest (GNF), has been presented. This model is based on the Growing Neural Gas (GNG), but it computes a spanning tree for each connected component (subgraph) of the overall graph. Therefore, the global structure of the GNF is a forest, i.e. an ensemble of trees of neurons. Hence our proposal has a greater flexibility than many self-organizing models. For example, it creates and destroys connections among pairs of neurons, unlike the SOM [21,22] and the GHSOM [21,42]. On the other hand, spanning trees are learned, unlike the GNG which features a general graph with no special structure [7].

As seen in the experiments, the GNF model has improved self-organization capabilities which allows it to represent complex input distributions, creating as many spanning trees as necessary to adapt its structure to the topology of the distribution (Subsection 4.2). Hence, the GNF succeeds in discovering the underlying structure of the input. It recognizes the connected components in the input datasets and represents them by connected components (trees) made of neurons. In addition to that, it is able to appropriately perform typical unsupervised learn-

[d] Reprinted from Computer Vision and Image Understanding, 133, Francisco Javier López-Rubio and Ezequiel López-Rubio, Features for stochastic approximation based foreground detection, 30-50, Copyright (2015), with permission from Elsevier.

Table 6: Accuracy results of the GNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

Video	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.82 (0.06)	0.80 (0.06)	0.70 (0.05)	0.58 (0.10)	0.80 (0.06)	**0.83 (0.05)**
Meeting room	0.68 (0.09)	0.67 (0.11)	0.62 (0.11)	**0.73 (0.06)**	0.63 (0.09)	0.64 (0.10)
Water surface	0.64 (0.05)	0.70 (0.04)	**0.83 (0.05)**	0.72 (0.09)	0.63 (0.06)	0.69 (0.06)
OneShopOneWait1cor	0.58 (0.01)	0.61 (0.02)	0.44 (0.03)	0.42 (0.04)	**0.64 (0.02)**	0.61 (0.02)

Table 7: F-measure results of the GNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	**0.90 (0.03)**	0.89 (0.04)	0.82 (0.04)	0.73 (0.08)	0.88 (0.04)	**0.90 (0.03)**
Meeting room	0.81 (0.07)	0.80 (0.08)	0.76 (0.08)	**0.84 (0.04)**	0.77 (0.07)	0.78 (0.08)
WaterSurface	0.78 (0.04)	0.82 (0.03)	**0.91 (0.03)**	0.83 (0.06)	0.77 (0.05)	0.81 (0.04)
OneShopOneWait1cor	0.73 (0.01)	0.76 (0.01)	0.61 (0.03)	0.59 (0.04)	**0.78 (0.01)**	0.76 (0.02)

Table 8: Precision results of the GNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.97 (0.02)	0.95 (0.04)	0.73 (0.06)	0.63 (0.10)	0.97 (0.02)	**0.98 (0.02)**
Meeting room	0.94 (0.10)	**0.97 (0.05)**	0.91 (0.05)	0.90 (0.06)	0.96 (0.09)	0.96 (0.10)
WaterSurface	0.99 (0.02)	**1.00 (0.00)**	0.99 (0.01)	0.96 (0.03)	**1.00 (0.00)**	**1.00 (0.00)**
OneShopOneWait1cor	0.77 (0.02)	0.85 (0.02)	0.57 (0.04)	0.51 (0.04)	**0.91 (0.01)**	0.84 (0.01)

Table 9: Recall results of the GNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.85 (0.06)	0.83 (0.06)	**0.93 (0.03)**	0.87 (0.06)	0.81 (0.06)	0.84 (0.05)
Meeting room	0.71 (0.06)	0.68 (0.10)	0.66 (0.10)	**0.79 (0.06)**	0.64 (0.07)	0.66 (0.08)
WaterSurface	0.65 (0.05)	0.70 (0.04)	**0.83 (0.05)**	0.74 (0.09)	0.63 (0.06)	0.69 (0.06)
OneShopOneWait1cor	**0.70 (0.02)**	0.68 (0.02)	0.65 (0.02)	0.70 (0.03)	0.68 (0.02)	0.69 (0.02)

Table 10: False positives results of the GNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.03 (0.02)	0.04 (0.03)	0.25 (0.06)	0.34 (0.09)	0.02 (0.02)	**0.02 (0.01)**
Meeting room	0.04 (0.07)	**0.02 (0.03)**	0.06 (0.03)	0.08 (0.05)	0.02 (0.05)	0.03 (0.07)
WaterSurface	0.01 (0.01)	**0.00 (0.00)**	0.01 (0.01)	0.03 (0.02)	**0.00 (0.00)**	**0.00 (0.00)**
OneShopOneWait1cor	0.17 (0.02)	0.11 (0.01)	0.32 (0.03)	0.40 (0.03)	**0.06 (0.01)**	0.12 (0.01)

Table 11: False negatives results of the GNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.15 (0.06)	0.16 (0.05)	**0.05 (0.03)**	0.08 (0.04)	0.18 (0.06)	0.15 (0.05)
Meeting room	0.28 (0.05)	0.31 (0.09)	0.32 (0.09)	**0.19 (0.06)**	0.35 (0.06)	0.33 (0.07)
WaterSurface	0.35 (0.05)	0.30 (0.04)	**0.17 (0.05)**	0.25 (0.08)	0.37 (0.06)	0.31 (0.06)
OneShopOneWait1cor	0.25 (0.02)	0.28 (0.01)	0.24 (0.01)	**0.18 (0.02)**	0.30 (0.02)	0.27 (0.02)

Table 12: CPU time results (in seconds) of the GNF model for each video sequence and color space. The CPU time is computed as the training time plus the mean of foreground detection time. Best results are in bold.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.496	0.544	0.536	**0.492**	0.573	0.502
Meeting room	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
WaterSurface	**0.006**	**0.006**	**0.006**	0.007	**0.006**	**0.006**
OneShopOneWait1cor	**0.008**	0.009	0.009	**0.008**	0.009	**0.008**

Table 13: Accuracy results for the GNF and different pixel-level methods for each video sequence. Best results are in bold. Standard deviations are in parentheses.

Video	WrenGA	GrimsonGMM	GMMV1	GMMV2	FASOM	MFBM	GNF
Video2	0.55 (0.09)	0.62 (0.08)	0.43 (0.07)	0.65 (0.08)	0.72 (0.02)	0.73 (0.04)	**0.83 (0.05)**
Meeting room	0.54 (0.09)	0.44 (0.09)	0.46 (0.15)	0.54 (0.16)	**0.78 (0.03)**	0.77 (0.03)	0.73 (0.06)
Water surface	0.77 (0.04)	0.55 (0.20)	0.68 (0.13)	0.53 (0.16)	0.83 (0.01)	**0.91 (0.01)**	0.83 (0.05)
OneShopOneWait1cor	**0.67 (0.04)**	0.38 (0.11)	0.53 (0.16)	0.37 (0.11)	0.61 (0.03)	0.66 (0.05)	0.64 (0.02)
Rank sums	15.50	24.00	23.00	22.50	10.50	**7.00**	9.50

ing tasks such as the clustering of manuscript digits and the clustering of standard benchmark data sets from the UCI Machine Learning Repository (Subsection 4.3). Both self-organization and clustering experiments have been compared to the GNG and the SOM models by using several performance measures, namely, the mean squared error (MSE), peak signal-to-noise ratio (PSNR), Davis-Boulding index (DB), silhouette coeficient (S), and CPU time. According to these quantitative results, the GNF overcomes the GNG and SOM models for these tasks. Moreover, our proposal can be used for foreground detection in video sequences by training the GNF with the median of several frames, achieving competitive results when compared to several well known algorithms specifically designed for foreground detection (Subsection 4.4).

Directions for future research include the application of the GNF model to other unsupervised learning tasks. These include vector quantization, lossy image compression, and visualization and clustering of other kinds of data. It could also be used to model visual labelled data in order to carry out classification tasks[49,26].

The GNF model has a considerable potential for extensions. An automated mechanism to determine the optimal size of the network could be added, rather than the H_{max} maximum size parameter. Time varying learning rates could be introduced, as usual in other neural models[12]. Furthermore, distance measures other than the Euclidean one could be considered. This would enable the GNF to process other kinds of data such that the Euclidean distance is inadequate. For example, Hamming distance could be used for binary data.

Acknowledgments

This work is partially supported by the Ministry of Economy and Competitiveness of Spain under grant TIN2014-53465-R, project name Video surveillance by active search of anomalous events. It is also partially supported by the Autonomous Government of Andalusia (Spain) under projects TIC-6213, project name Development of Self-Organizing Neural Networks for Information Technologies; and TIC-657, project name Self-organizing systems and robust estimators for video surveillance. All of them include funds from the European Regional Development Fund (ERDF). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Málaga.

References

1. H. Adeli and S.-L. Hung. *Machine Learning: Neural Networks, Genetic Algorithms, and Fuzzy Systems.* John Wiley & Sons, Inc., New York, NY, USA, 1994.
2. H. Adeli and H. S. Park. *Neurocomputing for Design Automation.* CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1998.

Table 14: F-measure results for the GNF and different pixel-level methods for each video sequence. Best results are in bold. Standard deviations are in parentheses.

Video	WrenGA	GrimsonGMM	GMMV1	GMMV2	FASOM	MFBM	GNF
Video2	0.53 (0.04)	0.55 (0.04)	0.45 (0.05)	0.57 (0.03)	0.60 (0.01)	0.84 (0.03)	**0.90 (0.03)**
Meeting room	0.68 (0.09)	0.57 (0.09)	0.58 (0.16)	0.68 (0.15)	**0.87 (0.02)**	**0.87 (0.02)**	0.84 (0.04)
Water surface	0.87 (0.03)	0.67 (0.18)	0.80 (0.12)	0.67 (0.15)	0.91 (0.00)	**0.95 (0.00)**	0.91 (0.03)
OneShopOneWait1cor	**0.80 (0.03)**	0.54 (0.12)	0.67 (0.16)	0.53 (0.12)	0.76 (0.02)	0.79 (0.03)	0.78 (0.01)
Rank sums	15.50	24.50	23.00	22.00	11.00	**6.50**	9.50

3. M. Ahmadlou and H. Adeli. Enhanced probabilistic neural network with local decision circles: A robust classifier. *Integrated Computer-Aided Engineering*, 17(3):197–210, Aug. 2010.
4. P. Ashok, G. Kadhar, E. Elayaraja, and V. Vadivel. Fuzzy based clustering method on yeast dataset with different fuzzification methods. In *Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on*, pages 1–6, July 2013.
5. C. A. Astudillo and B. J. Oommen. Self-organizing maps whose topologies can be learned with adaptive binary search trees using conditional rotations. *Pattern Recognition*, 47(1):96–113, 2014.
6. R. C. de Amorim and C. Hennig. Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Science*, 324(C):126–145, Dec. 2015.
7. B. Fritzke. A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, 7:625–632, 1995.
8. S. Ghosh-Dastidar and H. Adeli. Improved spiking neural networks for EEG classification and epilepsy and seizure detection. *Integrated Computer-Aided Engineering*, 14(3):187–212, Aug. 2007.
9. S. Ghosh-Dastidar and H. Adeli. A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks*, 22(10):1419–1431, 2009.
10. S. Ghosh-Dastidar and H. Adeli. Spiking neural networks. *International Journal of Neural Systems*, 19(04):295–308, 2009.
11. A. Granados, K. Koroutchev, and F. de Borja Rodriguez. Discovering data set nature through algorithmic clustering based on string compression. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):699–711, March 2015.
12. K. J. Gurubel, A. Y. Alanis, E. N. Sanchez, and S. Carlos-hernandez. A neural observer with time-varying learning rate: analysis and applications. *International Journal of Neural Systems*, 24(1):1450011, 2014.
13. M. Hinne, M. Ekman, R. Janssen, T. Heskes, and M. Van Gerven. Probabilistic clustering of the human connectome identifies communities and hubs. *PLoS ONE*, 10(1):e0117179, 2015.
14. F. Huffner, C. Komusiewicz, A. Liebtrau, and R. Niedermeier. Partitioning biological networks into highly connected clusters with maximum edge coverage. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(3):455–467, 2014.
15. H.-L. Hung and W.-C. Lin. Dynamic hierarchical self-organizing neural networks. In *IEEE International Conference on Neural Networks. IEEE World Congress on Computational Intelligence*, volume 2, pages 627–632, Jun 1994.
16. S. Hung and H. Adeli. A parallel genetic/neural network learning algorithm for MIMD shared memory machines. *IEEE Transactions on Neural Networks*, 5(6):900–909, Nov 1994.
17. J. Huo, Y. Gao, W. Yang, and H. Yin. Multi-instance dictionary learning for detecting abnormal event detection in surveillance videos. *International Journal of Neural Systems*, 24(3):1430010, 2014.
18. P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In P. Remagnino, G. Jones, N. Paragios, and C. Regazzoni, editors, *Video-Based Surveillance Systems*, pages 135–144. Springer US, 2002.
19. J. Kim and L. Billard. A polythetic clustering process and cluster validity indexes for histogram-valued objects. *Computational Statistics and Data Analysis*, 55(7):2250 – 2262, 2011.
20. T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.
21. T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.
22. T. Kohonen. Essentials of the self-organizing map. *Neural Networks*, 37:52–65, 2013.
23. J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.
24. J. Lampinen and E. Oja. Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, 2:261–272, 1992.
25. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.
26. G. Lee, M. Kwon, S. Kavuri, and M. Lee. Action-perception cycle learning for incremental emotion

recognition in a movie clip using 3D fuzzy GIST based on visual and EEG signals. *Integrated Computer-Aided Engineering*, 21(3):295–310, 2014.
27. H. Lee, J. K. Aghajan, Hamid Aggarwal, R. Cucchiara, and A. Prati. VSSN '06: Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks. 2006.
28. M. Lichman. UCI machine learning repository, 2013. http://archive.ics.uci.edu/ml.
29. E. López-Rubio, R. M. Luque-Baena, and E. Domínguez. Foreground detection in video sequences with probabilistic self-organizing maps. *International Journal of Neural Systems*, 21(3):225–246, 2011.
30. E. López-Rubio, E. J. Palomo, and E. Domínguez. Bregman divergences for growing hierarchical self-organizing networks. *International Journal of Neural Systems*, 24(4):1450016, 2014.
31. E. López-Rubio and E. Palomo-Ferrer. Growing hierarchical probabilistic self-organizing graphs. *IEEE Transactions on Neural Networks*, 22(7):997–1008, 2011.
32. E. López-Rubio, E. J. Palomo-Ferrer, J. M. Ortiz-de Lazcano-Lobato, and M. C. Vargas-González. Dynamic topology learning with the probabilistic self-organizing graph. *Neurocomputing*, 74(16):2633 – 2648, 2011.
33. F. J. López-Rubio and E. López-Rubio. Features for stochastic approximation based foreground detection. *Computer Vision and Image Understanding*, 133:30 – 50, 2015.
34. F. J. López-Rubio and E. López-Rubio. Local color transformation analysis for sudden illumination change detection. *Image and Vision Computing*, 37:31 – 47, 2015.
35. L. Maddalena and A. Petrosino. A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection. *Neural Computing and Applications*, 19(2):179–186, 2010.
36. T. Martinetz and K. Schulten. A "Neural-Gas" Network Learns Topologies. *Artificial Neural Networks*, I:397–402, 1991.
37. G. Menardi and A. Azzalini. An advancement in clustering via nonparametric density estimation. *Statistics and Computing*, 24(5):753–767, 2014.
38. J. Pakkanen, J. Iivarinen, and E. Oja. The evolving tree - a novel self-organizing network for data analysis. *Neural Processing Letters*, 20(3):199–211, 2004.
39. E. J. Palomo, E. Domínguez, R. M. Luque-Baena, and J. Muñoz. Image compression and video segmentation using hierarchical self-organization. *Neural Processing Letters*, 37(1):69–87, 2013.
40. L. Quesada and A. Leon. Unsupervised markerless 3DOF motion tracking in real-time using a single low-budget camera. *International Journal of Neural Systems*, 22(5):1250019, 2012.
41. J. Rahmel. SplitNet: learning of tree structured Kohonen chains. In *IEEE International Conference on Neural Networks*, volume 2, pages 1221–1226, Jun 1996.
42. A. Rauber, D. Merkl, and M. Dittenbach. The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, 2002.
43. E. V. Samsonova, J. N. Kok, and A. P. Ijzerman. TreeSOM: Cluster analysis in the self-organizing map. *Neural Networks*, 19(6-7):935 – 949, 2006.
44. L. Scrucca. Identifying connected components in gaussian finite mixture models for clustering. *Computational Statistics and Data Analysis*, 93:5–17, 2016.
45. T. Semertzidis, D. Rafailidis, M. Strintzis, and P. Daras. Large-scale spectral clustering based on pairwise constraints. *Information Processing and Management*, 51(5):616–624, 2015.
46. C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on*, volume 2, pages 246–252, 1999.
47. W. Stuetzle. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *Journal of Classification*, 20(1):25–47, 2003.
48. C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.
49. Y.-B. Yang, Y.-N. Li, Y. Gao, H. Yin, and Y. Tang. Structurally enhanced incremental neural learning for image classification with subgraph extraction. *International Journal of Neural Systems*, 24(7):1450024, 2014.
50. Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *Proceedings - International Conference on Pattern Recognition*, volume 2, pages 28–31, 2004.