A New Self-Organizing Neural Gas Model based on Bregman Divergences

Esteban J. Palomo*, Miguel A. Molina-Cabello*, Ezequiel López-Rubio* and Rafael Marcos Luque-Baena*
*Department of Computer Languages and Computer Science
University of Málaga, Bulevar Louis Pasteur, 35, 29071 Málaga, Spain
Emails: {ejpalomo,miguelangel,ezeqlr,rmluque}@lcc.uma.es

***Abstract*—In this paper, a new self-organizing neural gas model that we call Growing Hierarchical Bregman Neural Gas (GHBNG) has been proposed. Our proposal is based on the Growing Hierarchical Neural Gas (GHNG) in which Bregman divergences are incorporated in order to compute the winning neuron. This model has been applied to anomaly detection in video sequences together with a Faster R-CNN as an object detector module. Experimental results not only confirm the effectiveness of the GHBNG for the detection of anomalous object in video sequences but also its self-organization capabilities.**

1. Introduction

The Self-organizing Map (SOM) [1] has been widely used for data clustering since its publication. The SOM performs a mapping between high-dimensional data and a lower dimensional representation space preserving the topology of input data. Many SOM-like neural models have been proposed over the years, which are based on a fixed lattice topology among the neurons [2]. The Growing Neural Gas (GNG) [3] is a self-organizing neural network which learns a dynamic graph with variable numbers of neurons and connections. This graph represents input data in a more plastic and flexible way than a fixed-topology map, improving visualization capabilities and understanding of data.

These self-organizing models have their hierarchical versions, such as the Growing Hierarchical Self-Organizing Map (GHSOM) for the SOM [4] and the Growing Hierarchical Neural Gas (GHNG) for the GNG [5], in which a neuron can be expanded into a new map or graph in a subsequent layer of the hierarchy depending on the quantization error associated to that neuron or the graph it belongs to. Hierarchical models can reflect hierarchical relations present among input data in a more straightforward way.

Another possible problem present in these self-organizing models is the use of the Euclidean distance to compute the winning neuron, since this distance may not be the most suitable for all input distributions. Hence, Bregman divergences were taken into account for the GHSOM [6], since they are suited for clustering because their minimizer is the mean [7]. Moreover, the squared Euclidean distance is a particular case of the Bregman divergences. Therefore, by using Bregman divergences the most suitable divergence according to input data can be specified. In this paper, a new self-organizing neural network called the Growing Hierarchical Bregman Neural Gas (GHBNG) is proposed, which is grounded in the GHNG model and in which Bregman divergences have been considered.

On the other hand, the proliferation in recent years of a huge amount of visual information in the form of data sequences has led to a growth in the field of intelligent video surveillance. In particular, one of the most important tasks to consider is to automatically detect moving objects that are not very frequent in a scene and can be considered as anomalies. In recent years, the appearance of deep learning networks for the detection of objects in an image has meant a turning point in the detection of objects in video sequences [8]. Thus, it is possible to use pre-trained networks with thousands of data and a large number of object types to detect moving objects in a scene, providing more stable results than those obtained by classical approaches.

In order to show the possible applications of our proposal, we have also applied the GHBNG to anomalous detection in video sequences acquired by fixed IP cameras. The objects in motion on each frame are obtained by the Faster RCNN network [9]. Later, the GHBNG estimates the objects considered as anomalous after a previous training phase.

The rest of the paper will have the following structure: section 2 exhaustively describes the GHBNG model. Section 3 presents several experiments which demonstrate the self-organization capacity of the GHBNG model, in addition to its application for the detection of anomalous objects in video sequences. Finally section 4 concludes the paper.

2. The GHBNG Model

A Growing Hierarchical Bregman Neural Gas (GHBNG) network is defined as a Growing Hierarchical Neural Gas (GHNG) network [5] in which Bregman divergences are incorporated in order to compute the winning neuron. A GHBNG network can be seen as a tree of Growing Neural Gas (GNG) networks [3] where a mechanism to control the growth of each GNG graph is established. This mechanism distinguishes between a growth phase where more neurons are added until no significant improvement in the quantization error is obtained, and a convergence phase where no more units can be created. Thus, each graph contains a variable number of neurons so that its size can grow or shrink during learning. Also, each graph is the child of a unit in the upper level, except for the top level (root) graph which has no parent. An example of the structure of a GHBNG model is shown in Figure 1. Note that the structure is the same as the GHNG since the difference between these two self-organizing models resides in the way to compute the winning neuron according to the used Bregman divergence.

The definition of the GHBNG is organized in two subsections. First a review of Bregman divergences is presented. Then, the basic model for a graph and the corresponding learning algorithm are explained (Subsection 2.3). Finally we explain how new graphs are created to yield a hierarchy of graphs (Subsection 2.4).

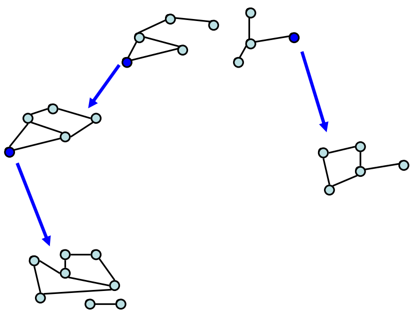
Figure 1. Structure of a GHBNG model with four graphs. The parent neurons are shown in a darker tone.

2.1. Review of Bregman Divergences

Next the fundamentals of Bregman divergences and their application to clustering are reviewed. Let $\phi : \mathcal{S} \rightarrow \mathbb{R}$ be a strictly convex real valued function defined over a convex set $\mathcal{S} \subseteq \mathbb{R}^D$, where D is the dimension of the input data [10], [11], [12]. We assume that ϕ is differentiable on the relative interior $\mathrm{ri}(\mathcal{S})$ of the set $\mathcal{S}$ [7]. Then the Bregman divergence $D_\phi : \mathcal{S} \times \mathrm{ri}(\mathcal{S}) \rightarrow [0, +\infty)$ corresponding to ϕ is defined as

$$D_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - (\mathbf{x} - \mathbf{y})^T \nabla\phi(\mathbf{y}) \qquad (1)$$

where $\mathbf{x} \in \mathcal{S}$ and $\nabla\phi(\mathbf{y})$ stands for the gradient vector of ϕ evaluated at $\mathbf{y} \in \mathrm{ri}(\mathcal{S})$. Table 1 lists the Bregman divergences that we consider in this paper.

Bregman divergences are suited for clustering because their minimizer is the mean. This is the main contribution of [7], where it is proved that the class of distortion measures with respect to a set of centroids which admit an iterative minimization procedure is precisely that of Bregman divergences. Moreover, it is also proved that each Bregman divergence is uniquely associated to a regular exponential family of probability density functions, that are defined below. This way, a unique probability density function can be linked to the cluster associated to a given centroid, which enables probabilistic soft clustering. Furthermore, expectation maximization can be carried out with a reduced computational complexity for general Bregman divergences, so that specific Bregman divergences can be designed to suit the application at hand.

The property that the mean is the minimizer of a Bregman divergence is formalized next. Given an input distribution for $\mathbf{x}$ the following condition holds [12]:

$$\boldsymbol{\mu} = E[\mathbf{x}] = \arg\min_{\mathbf{y}} E[D_\phi(\mathbf{x}, \mathbf{y})] \qquad (2)$$

Let N be the number of clusters, and let $\boldsymbol{\mu}_i$ be the mean vector of the i-th cluster $\mathcal{C}_i$, $i \in \{1, ..., N\}$. Then a point $\mathbf{x}$ belongs to $\mathcal{C}_i$ if $\boldsymbol{\mu}_i$ minimizes the divergence with respect to $\mathbf{x}$:

$$\mathcal{C}_i = \left\{ \mathbf{x} \in \mathcal{S} \mid i = \arg \min_{j \in \{1,\ldots,N\}} D_\phi\left(\mathbf{x}, \boldsymbol{\mu}_j\right) \right\} \qquad (3)$$

So, we can rewrite (2) to partition $\mathcal{S}$ into N clusters $\mathcal{C}_i$:

$$\boldsymbol{\mu}_i = E[\mathbf{x} \mid \mathcal{C}_i] = \arg\min_{\mathbf{y}} E[D_\phi(\mathbf{x}, \mathbf{y}) \mid \mathcal{C}_i] \qquad (4)$$

The above equation implies that the mean of the cluster $\mathcal{C}_i$ minimizes the Bregman divergence to the samples $\mathbf{x}$ which belong to the cluster.

2.2. Basic model

For clustering and self-organizing network applications it is necessary to learn a weight vector $\mathbf{w}_i$ of each cluster i [13], so that $\mathbf{w}_i$ estimates the cluster mean vector $\boldsymbol{\mu}_i$. Stochastic gradient descent has been proposed in [12] to minimize $E[D_\phi(\mathbf{x}, \mathbf{z})]$:

$$\triangle \mathbf{w}_i = -\eta \frac{\partial D_\phi(\mathbf{x}, \mathbf{w}_i)}{\partial \mathbf{w}_i} \qquad (5)$$

where η is a suitable *step size.*

Here we propose a different approach, namely the estimation of the cluster mean vector $E[\mathbf{x} \mid \mathcal{C}_i]$ by stochastic approximation [14], [15], [16], [17]. This strategy has been successfully applied by the authors to other self-organizing models in [18], [19], [20]. The goal of stochastic approximation is to find the value of some parameter $\boldsymbol{\theta}$ which satisfies

$$\zeta(\boldsymbol{\theta}) = 0 \qquad (6)$$

where ζ is a function whose values can not be obtained directly. What we have is a random variable z which is a noisy estimate of ζ:

$$E[z(\boldsymbol{\theta}) \mid \boldsymbol{\theta}] = \zeta(\boldsymbol{\theta}) \qquad (7)$$

Divergence	$\mathcal{S}$	$\phi(\mathbf{x})$	$D_\phi(\mathbf{x},\mathbf{y})$
Squared Euclidean distance	$\mathbb{R}^D$	$\|\mathbf{x}\|^2$	$\|\mathbf{x}-\mathbf{y}\|^2$
Generalized I-divergence	$\mathbb{R}_+^D$	$\sum_{k=1}^D x_k \log x_k$	$\sum_{k=1}^D \left(-x_k + y_k + x_k \log \frac{x_k}{y_k}\right)$
Itakura-Saito distance	$\mathbb{R}_+^D$	$-\sum_{k=1}^D \log x_k$	$\sum_{k=1}^D \left(-1 + \frac{x_k}{y_k} - \log \frac{x_k}{y_k}\right)$
Exponential loss	$\mathbb{R}^D$	$\sum_{k=1}^D \exp x_k$	$\sum_{k=1}^D \left(\exp x_k - \exp y_k - (x_k - y_k)\exp y_k\right)$
Logistic loss	$(0,1)^D$	$\sum_{k=1}^D \left(x_k \log x_k + (1-x_k)\log(1-x_k)\right)$	$\sum_{k=1}^D \left(x_k \log \frac{x_k}{y_k} + (1-x_k)\log \frac{1-x_k}{1-y_k}\right)$

TABLE 1. BREGMAN DIVERGENCES CONSIDERED IN THIS PAPER. $\mathbb{R}_+^D$ STANDS FOR THE SET OF VECTORS OF SIZE D WITH STRICTLY POSITIVE REAL COMPONENTS.

Under these conditions, the Robbins-Monro algorithm proceeds iteratively:

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \eta(n)\, z(\boldsymbol{\theta}(n)) \tag{8}$$

where n is the time step.

In our case, the varying parameter $\boldsymbol{\theta}(n)$ is the i-th weight vector:

$$\boldsymbol{\theta}(n) = \mathbf{w}_i(n) \tag{9}$$

As said before, we aim to estimate the conditional expectation $\boldsymbol{\mu}_i = E[\mathbf{x} \mid \mathcal{C}_i]$ by stochastic approximation, so we may take

$$\zeta(\mathbf{w}_i) = \boldsymbol{\mu}_i - \mathbf{w}_i \tag{10}$$

$$z(\mathbf{w}_i) = \frac{\mathbb{I}(\mathbf{x} \in \mathcal{C}_i)}{P(\mathcal{C}_i)}(\mathbf{x} - \mathbf{w}_i) \tag{11}$$

where $\mathbb{I}$ stands for the indicator function and $P(\mathcal{C}_i)$ is the a priori probability of cluster $\mathcal{C}_i$. Please note that

$$\mathbf{x} \notin \mathcal{C}_i \Rightarrow z(\mathbf{w}_i) = \mathbf{0} \tag{12}$$

Consequently, we have that (11) satisfies the condition (7):

$$E[z(\mathbf{w}_i) \mid \mathbf{w}_i] = P(\mathcal{C}_i)\, E[z(\mathbf{w}_i) \mid \mathcal{C}_i, \mathbf{w}_i] +$$

$$P(\bar{\mathcal{C}}_i)\, E[z(\mathbf{w}_i) \mid \bar{\mathcal{C}}_i, \mathbf{w}_i] =$$

$$E[\mathbf{x} - \mathbf{w}_i \mid \mathcal{C}_i, \mathbf{w}_i] = E[\mathbf{x} \mid \mathcal{C}_i] - \mathbf{w}_i = \boldsymbol{\mu}_i - \mathbf{w}_i \tag{13}$$

where $\bar{\mathcal{C}}_i$ is the complement of cluster $\mathcal{C}_i$.

Hence equation (8) reads

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \eta(n)\, z(\mathbf{w}_i(n)) \tag{14}$$

If we take

$$\eta(n) = P(\mathcal{C}_i)\, \epsilon(n) \tag{15}$$

then (14) can be rewritten as

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) +$$

$$\epsilon(n)\, P(\mathbf{x}(n) \in \mathcal{C}_i)(\mathbf{x}(n) - \mathbf{w}_i(n)) \tag{16}$$

where $\epsilon(n)$ is the learning rate at time step n, and we no longer need the value of the a priori probability $P(\mathcal{C}_i)$. The term $\epsilon(n)P(\mathbf{x}(n) \in \mathcal{C}_i)$ is assumed to be ϵ_b for the winning neuron, ϵ_n for its direct neighbors, and zero otherwise, with $\epsilon_b > \epsilon_n > 0$. That is, $P(\mathbf{x}(n) \in \mathcal{C}_i)$ is assumed to be maximum for the winning neuron, smaller for its immediate neighbors, and zero for all other units.

2.3. Graph model

Each graph of the GHBNG is made of H neurons ($H \geq 2$) and one or more directed connections among them. Both neurons and connections can be created and destroyed during the learning process. It is not necessary that the graph is connected, as mentioned earlier. The training set for the graph will be 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 will be noted $A \subseteq \{1, ..., H\} \times \{1, ..., H\}$.

The learning mechanism for a graph of the GHBNG is based on the original GNG [3], but it includes a novel procedure to control the growth of the graph. First a growth phase is performed where the graph is allowed to enlarge, until a condition is fulfilled which indicates that further growing would provide no significant improvements in the quantization error. After that, a convergence phase is executed where no unit creation is allowed in order to carry out a fine tuning of the graph. The learning algorithm is given by the following steps:

1) Start with two neurons ($H = 2$) joined two connections, one each way. 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 from $\mathcal{S}$.
3) Find the nearest unit q and the second nearest unit s in terms of Bregman divergences:

$$q = \arg \min_{i \in \{1,...,H\}} D_\phi(\mathbf{x}(n), \mathbf{w}_i(n)) \tag{17}$$

$$s = \arg \min_{i \in \{1,\ldots,H\} - \{q\}} D_\phi (\mathbf{x}(n), \mathbf{w}_i(n)) \quad (18)$$

4) Increment the age of all edges departing from q.
5) Add the squared Euclidean distance between $\mathbf{x}_n$ and the nearest unit q to the error variable e_q:

$$e_q(n+1) = e_q(n) + \|\mathbf{w}_q(n) - \mathbf{x}(n)\|^2 \quad (19)$$

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

$$\epsilon(n,i) = \begin{cases} \epsilon_b & \text{iff } n = q \\ \epsilon_n & \text{iff } (n \neq q) \wedge (n,q) \in A \\ 0 & \text{iff } (n \neq q) \wedge (n,q) \notin A \end{cases} \quad (20)$$

$$\mathbf{w}_i(n+1) = (1 - \epsilon(n,i))\,\mathbf{w}_i(n) + \epsilon(n,i)\,\mathbf{x}(n) \quad (21)$$

7) If q and s are connected by an edge, then set the age of this edge to zero. Otherwise, create it.
8) Remove edges with an age larger than a_{max}. Then remove all neurons which have no outgoing edges.
9) If the current time step n is an integer multiple of a parameter λ and the graph is in the growth phase, then make a backup copy of the full graph and insert a new unit as follows. First determine the unit r with the maximum error and the unit z with the largest error among all direct neighbors of r. 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. Finally, setup the prototype of k to be halfway between those of r and z, as follows:

$$\mathbf{w}_k(n) = \frac{1}{2}(\mathbf{w}_r(n) + \mathbf{w}_z(n)) \quad (22)$$

10) If the graph is in the growth phase and the current time step n satisfies:

$$\mod(n, 2\lambda) = \left\lfloor \frac{3}{2}\lambda \right\rfloor \quad (23)$$

where $\lfloor \cdot \rfloor$ stands for rounding towards $-\infty$, then a check is done in order to see whether the graph growth has resulted in an improvement of the quantization error. The mean quantization error per neuron of the backup and current versions of the graph are computed as the sum of their error variables divided by their number of neurons H. Let us MQE_{old} and MQE_{new} note the mean quantization errors of the backup and current versions of the graph, respectively. If the following condition holds, then the current version is destroyed, the backup copy is restored, and the graph enters the convergence phase:

$$\frac{MQE_{old} - MQE_{new}}{MQE_{old}} < \tau \quad (24)$$

where $\tau \in [0,1]$ is a parameter which controls the growth process. The higher τ, the more significant the improvement in the quantization error must be in order to continue the growth phase. Hence higher values of τ are associated with smaller graphs, and vice versa.
11) Decrease all error variables e_i by multiplying them by a constant d.
12) If the maximum number of time steps has been reached, then stop. Otherwise, go to step 2.

2.4. Hierarchical model

As mentioned before, the GHBNG is defined as a tree of graphs. The procedure to learn such hierarchy is detailed next. The process starts by training the root graph with the overall set of training samples. Each time that a graph must be trained with training set $\mathcal{S}$, this is done according to the algorithm specified in Subsection 2.3. If the resulting number of neurons is $H = 2$, then the graph is pruned because it is too small to represent any important features of the input distribution. Otherwise, a new graph is created for each unit i and the training process is invoked recursively with the receptive field of unit i as the training set:

$$\mathcal{S}_i = \left\{ \mathbf{x} \in \mathcal{S} \mid i = \arg \min_{j \in \{1,\ldots,H\}} D_\phi(\mathbf{x}, \mathbf{w}_j) \right\} \quad (25)$$

This recursive process continues until a prespecified number of levels is reached. The elimination of the graphs with less than 3 neurons and the split of the training set given by (25) work together in order to attain a parsimonious hierarchy, i.e. one with a reduced number of graphs and neurons. This is because lower graphs in the tree cannot have many neurons because their training sets are smaller. It is worth noting that many of the created graphs will eventually be pruned right after their training, so that the fact that a graph is created for each unit does not lead to uncontrolled growth.

3. Experimental Results

3.1. Experimental Setup

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. All training were carried out using 2 epochs, independently of the number of training samples M. Since the GHBNG is based on the GHNG, our proposal has the same parameters than this model. In turn, the GHNG is based on the GNG so that the parameter setup is the same as recommended in the original GNG paper [3], whose values are provided in Table 2. As mentioned before, the parameter τ (which is also present in the GHNG model) controls the growth process where the smaller τ, the bigger the architecture size. This parameter must be tuned for each experiment holding $\tau \in [0, 1]$.

3.2. Self-Organization Experiments

This set of experiments has been designed to check the self-organization capabilities of the GHBNG before different Bregman divergences. We have selected two different two-dimensional input distributions ($D = 2$), namely an input distribution with the shape of a number eight and another with the shape of an M letter. The training was carried out using $M = 10,000$ input samples and $N = 20,000$ time steps for each input distribution and Bregman divergence. Two different values of the τ parameter (0.1 and 0.2) were chosen to show the effect of this parameter in the final architecture size.

The resulting GHBNGs for each Bregman divergence and τ value are given in Figures 2 and 3 for the number eight and the M letter input distributions, respectively. In these plots neurons are represented by circles and connection among neurons are plotted as straight lines, where the color and size of both neurons and connections is different depending on the layer they belong to. Thus, upper layers are painted darker and with a bigger size than deeper layers. A maximum of three layers has been plotted in order to avoid cluttered plots. Note that GHBNGs for $\tau = 0.2$ (first rows) yield architectures with less neurons than using $\tau = 0.1$ (second rows) and therefore commit more mistakes when adapting to their corresponding shape. If we focus on $\tau = 0.1$ (second rows), each Bregman divergence correctly unfolds to the shape of the two input distributions, although for the M letter some connections are in the wrong place, especially for Itakura-Saito.

3.3. Anomalous Detection in Video Sequences

We have employed the developed model in the detection of anomalous objects in video sequences. The system is composed by a camera fixed in the scenario, recording it and producing a video.

With a Faster R-CNN object detector module [9] (a deep learning technique that uses regions with convolutional neural networks), we can recognize 20 different object classes included in the PASCAL VOC 2007 dataset [21]. The input of the Faster R-CNN detector is an image and the output is a set of probabilities and its area per each detected object, where each probability exhibits the level of belonging to each class. In this work we only use the indicated probabilities, so given the frame t, the output set of probabilites $\mathbf{q}_{i,t}$ of our object detector module is as follows:

$$\mathbf{q}_{i,t} = (q_{i,t,1}, ..., q_{i,t,K}) \in \mathbb{R}^K \tag{26}$$

where i is one the detected objects in the frame t, $q_{i,t,k} \in [0, 1]$, $C_k \in Classes$ and the number of object classes is K (in this case, $K = 20$). In this case, we have incorporated a Titan X GPU as hardware resource.

In our context, we have considered animals as anomalous objects, whereas the remaining detected classes are considered non anomalous objects. In order to train our anomalous classification models, we have carried out the system with a video that does not exhibit any anomalous object (so that, there is no animals on it) and, for each frame, we have obtained the belonging probabilities to each class of each detected object by the detection module. After that, we have trained a GHBNG model per Bregman divergence with this information. According to the definition of the object detector module, we have a twenty-dimensional input distribution ($D = 20$). The training was carried out using $M = 650$ input samples, $N = 1,300$ time steps for each Bregman divergence and the value of the τ parameter was set to 0.1. A schema of this steps in order to train the GHBNG models can be observed in Figure 4.

Then, we have executed a video which presents anomalous objects in the same scenario (the video with anomalies [1] and without them [2] can be downloaded from our website). In each frame, like the previous step, the object detector provides the belonging probabilities of each detected object and this output is supplied to the anomalous classification module with a trained GHBNG model. Later, this GHBNG model calculates how much of anomalous is each detected object. This values are obtained from the minimum distance of the object to the prototypes of the model. We have considered this distance as a negative value, so an object will be most anomalous than other if its result is lower. So that, given the set of probabilities $\mathbf{q}_{i,t}$ corresponding to the frame t, the anomalous detection module calculates the value $v_{i,t} \in \mathbb{R}$ corresponding to how anomalous is the object i in t. Finally, the anomalous detection module indicates the value v_t corresponding to the most anomalous object in t in order to establish if it exists a really anomalous object in the frame t (for example, a dog) or not (for example, people or a potterplant). The operation of this process can be observed in Figure 5.

We have carried out 10 times the selected video with anomalous objects with the different kinds of GHBNG models and the produced median v_t result by each model can be observed in Figure 6. Each image exhibits the most anomalous object in each frame with its value produced by each model, respectively. This objects are divided into two groups: the real anomalous objects (animals) and the real non anomalous ones (the remaining classes). The two presented dotted lines corresponding to the values of the higher output of all real anomalous objects and the minimum output of all real non anomalous objects. The less number

1. http://www.lcc.uma.es/~miguelangel/resources/fixed_camera/video_with_anomalies.rar
2. http://www.lcc.uma.es/~miguelangel/resources/fixed_camera/video_without_anomalies.rar

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	$d = 0.995$

TABLE 2. PARAMETER SELECTION FOR THE GHBNG MODEL.

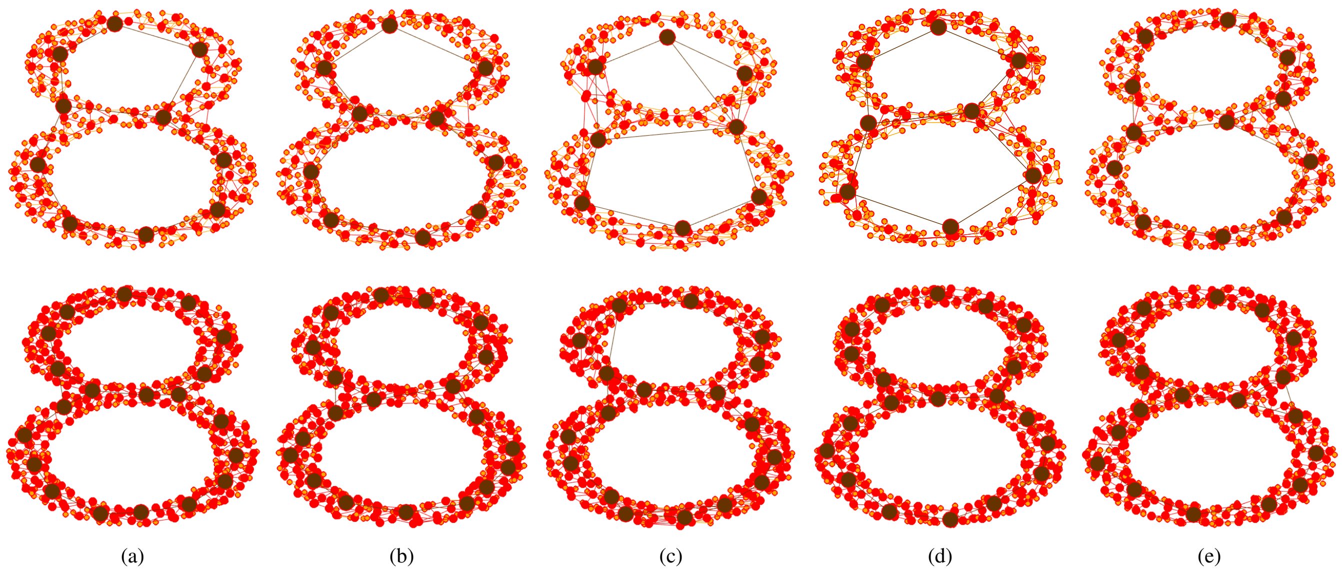

Figure 2. GHBNG results for the eight letter input distributions using $\tau = 0.2$ (first row), $\tau = 0.1$ (second row) and different Bregman divergences: (a) squared Euclidean, (b) generalized I-divergence, (c) Itakura-Saito, (d) exponential loss, and (e) logistic loss. Neurons are represented by circles and connection among neurons are plotted as straight lines, where upper layers are painted darker and with a bigger size than deeper layers.

of objects between this two lines (higher heterogeneity) the better the model, and a value between this two lines could be considered as threshold of these model in order to classify an object as anomalous. So that, an object between this two dotted lines could be considered anomalous or non anomalous, depending on the selection of the threshold.

As it can be observed in this Figure 6 the Logistic Loss and the Generalized I-Divergence models classify correctly the detected objects in the video. On the other hand, the Squared Euclidean and the Exponential Loss models yield the worst performance. In addition, we could consider a model better than other if the exhibited outputs of the anomalous and non anomalous sets are close between them, respectively, even more if a set is far away from the other. So the election of the threshold could be chosen better and the model could be suitable in a wide range of scenarios. Thus, for example, as we saw in Figure 6, the Logistic Loss model could be more appropriated than the Generalized I-Divergence model.

4. Conclusions

This paper has presented a novel growing hierarchical self-organizing model based on the neural gas approach and Bregman divergences. The main feature is its adaptability and flexibility to the data, learning a dynamic graph at each level of the hierarchy. Additionally, the use of different measures (Bregman divergences) to compute the winning neuron provides more possibilities in order to form clusters, which is suitable in real heterogeneous environments.

Thus, the self organizing capabilities of the new approach have been proved in the experimental section in a qualitative way. On the other hand, a more specific application related to detect anomalous objects in video surveillance has been considered, with interesting results which indicate its viability in this field. It is remarkable the possibilities that this model offers, obtaining the best results with the Generalized I-Divergence and Logistic Loss divergences. As future work, comparison with other methods and other performance will be addressed.

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 and TIN2016-75097-P. It is also partially supported by the Autonomous Government of Andalu-

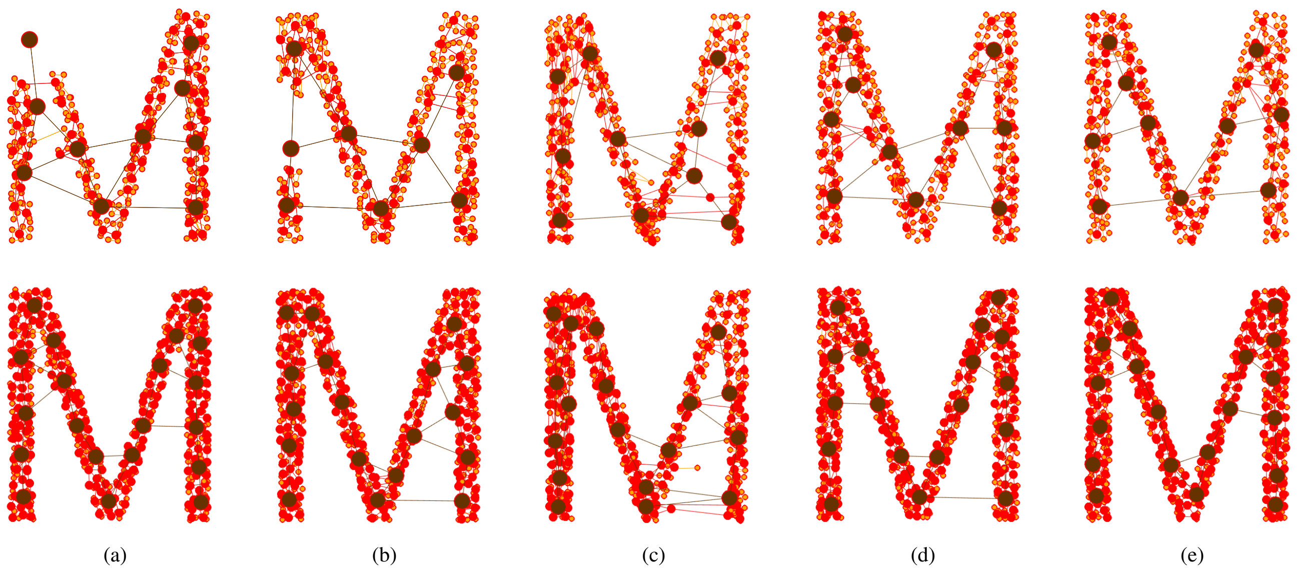

Figure 3. GHBNG results for the M letter input distributions using $\tau = 0.2$ (first row), $\tau = 0.1$ (second row) and different Bregman divergences: (a) squared Euclidean, (b) generalized I-divergence, (c) Itakura-Saito, (d) exponential loss, and (e) logistic loss. Neurons are represented by circles and connection among neurons are plotted as straight lines, where upper layers are painted darker and with a bigger size than deeper layers.

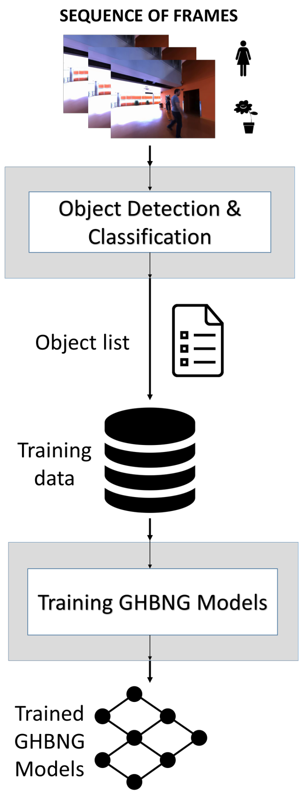

Figure 4. Schema of the training GHBNG models process.

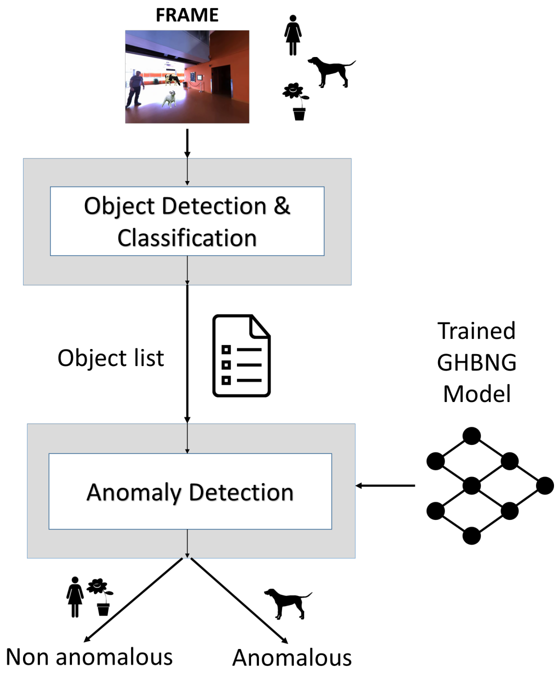

Figure 5. Schema of the operation of the anomaly detection system based on a GHBNG model.

sia (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. They also gratefully

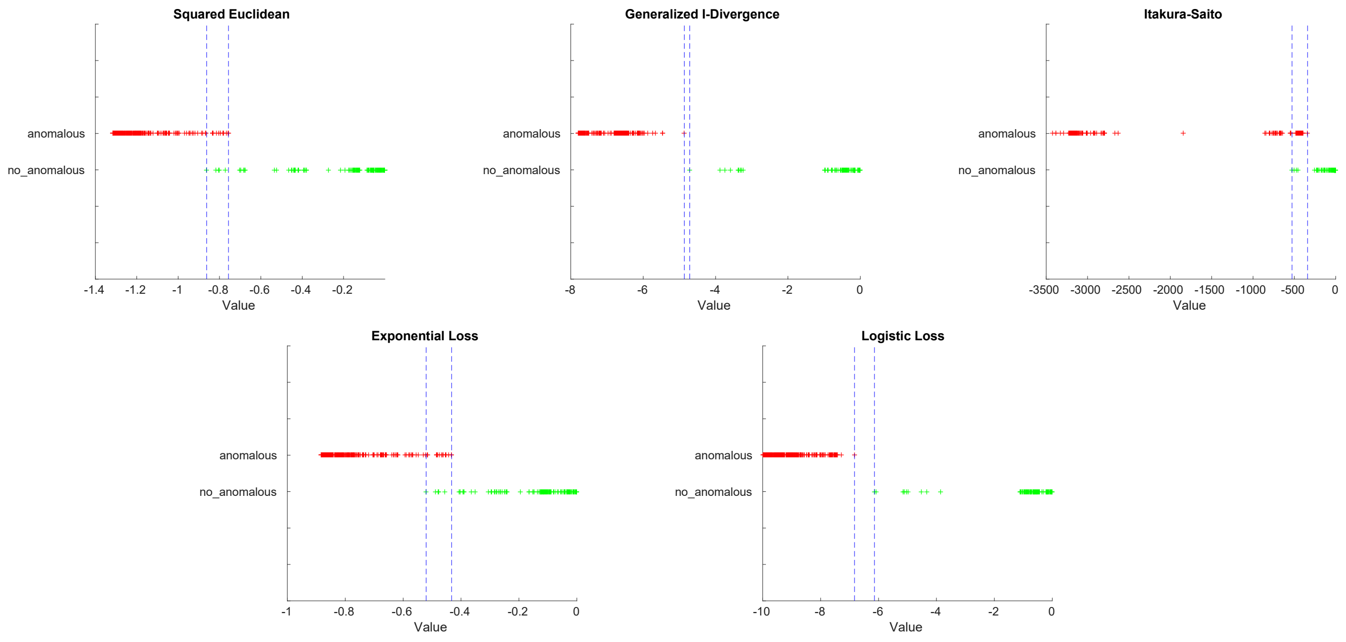

Figure 6. Most anomalous output for each anomaly classification model in each frame of the tested video sequence.

acknowledge the support of NVIDIA Corporation with the donation of two Titan X GPUs used for this research.

References

[1] T. Kohonen, “Self-organized formation of topologically correct feature maps,” *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[2] T.Kohonen, “Essentials of the self-organizing map,” *Neural Networks*, vol. 37, pp. 52–65, 2013.

[3] B. Fritzke, “A growing neural gas network learns topologies,” *Advances in Neural Information Processing Systems*, vol. 7, pp. 625–632, 1995.

[4] A. Rauber, D. Merkl, and M. Dittenbach, “The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data,” *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1331–1341, 2002.

[5] E. J. Palomo and E. López-Rubio, “The growing hierarchical neural gas self-organizing neural network,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 9, pp. 2000–2009, 2017.

[6] E. López-Rubio, E. J. Palomo, and E. Dominguez, “Bregman divergences for growing hierarchical self-organizing networks,” *International Journal of Neural Systems*, vol. 24, no. 04, p. 1450016, 2014.

[7] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, “Clustering with Bregman divergences,” *Journal of Machine Learning Research*, vol. 6, pp. 1705–1749, 2005.

[8] X. Wang, “Deep learning in object recognition, detection, and segmentation,” *Foundations and Trends in Signal Processing*, vol. 8, no. 4, pp. 217–382, 2016.

[9] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[10] L. Bregman, “The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming,” *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 3, pp. 200–217, 1967.

[11] Y. Censor and S. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1998.

[12] T. Villmann and S. Haase, “Divergence-based vector quantization,” *Neural Computation*, vol. 23, pp. 1343–1392, 2011.

[13] E. Mwebaze, P. Schneider, F. M. Schleif, J. R. Aduwo, J. A. Quinn, S. Haase, T. Villmann, and M. Biehl, “Divergence-based classification in learning vector quantization,” *Neurocomputing*, vol. 74, no. 9, pp. 1429–1435, 2011.

[14] H. J. Kushner and G. G. Yin, *Stochastic approximation and Recursive Algorithms and Applications*. New York, NY, USA: Springer-Verlag, 2003.

[15] T. Lai, “Stochastic approximation,” *Annals of Statistics*, vol. 31, no. 2, pp. 391–406, 2003.

[16] B. Delyon, M. Lavielle, and E. Moulines, “Convergence of a stochastic approximation version of the EM algorithm,” *Annals of Statistics*, vol. 27, no. 1, pp. 94–128, 1999.

[17] M. Sato and S. Ishii, “On-line EM algorithm for the normalized Gaussian network,” *Neural Computation*, vol. 12, no. 2, pp. 407–432, 2000.

[18] E. López-Rubio, J. M. Ortiz-de-Lazcano-Lobato, and D. López-Rodríguez, “Probabilistic PCA self-organizing maps,” *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1474–1489, 2009.

[19] E. López-Rubio and E. Palomo, “Growing hierarchical probabilistic self-organizing graphs,” *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 997–1008, 2011.

[20] 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*, vol. 74, no. 16, pp. 2633–2648, 2011.

[21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results,” http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.