Traffic Sign Detection with VG-RAM Weightless Neural Networks

Alberto F. De Souza, Cayo Fontana, Filipe Mutz, Tiago Alves de Oliveira, Mariella Berger, Avelino Forechi, Jorcy de Oliveira Neto, Edilson de Aguiar, and Claudine Badue

***Abstract*— We present a biologically inspired approach to traffic sign detection based on Virtual Generalizing Random Access Memory Weightless Neural Networks (VG-RAM WNN). VG-RAM WNN are effective machine learning tools that offer simple implementation and fast training and test. Our VG-RAM WNN architecture models the saccadic eye movement system and the transformations suffered by the images captured by the eyes from the retina to the superior colliculus in the mammalian brain. We evaluated the performance of our VG-RAM WNN system on traffic sign detection using the German Traffic Sign Detection Benchmark (GTSDB). Using only 12 traffic sign images for training, our system was ranked between the first 16 methods for the prohibitory category in the German Traffic Sign Detection Competition, part of the IJCNN'2013. Our experimental results showed that our approach is capable of reliably and efficiently detect a large variety of traffic sign categories using a few training samples.**

I. Introduction

Safety during driving is a very important research topic for the automotive industry. One of the technologies that can make cars safer to drive is automatic detection and recognition of traffic signs. Such technology aims to warn the driver of inappropriate actions, e.g., speeding, taking a wrong turn in a one-way street, as well as to help the driver in difficult situations, e.g., bad weather, tiredness, sleeplessness etc. Although the traffic sign detection process could be simplified as the appearance of certain traffic signs are fixed, sometimes even described by law, in real-world situations, given the substantial appearance variation, for instance, due to different light conditions, weather, viewpoint changes, ageing of the traffic sign and even deformations, simple approaches are not reliable and more robust methods are necessary.

While humans are capable of detecting the large variety of existing traffic signs efficiently, automatic systems are still a challenge. Given the high industrial relevance, automatic traffic sign detection and recognition has been attracting many researchers' attentions in recent years [1]. Despite many recent advances, traffic sign detection is still a complex real-world problem, which makes it one important application for advanced and autonomous driving systems. A useful detection system needs to cope with sign rotation, different lighting conditions, perspective changes, occlusion and all kinds of weather conditions.

Given an image of a scene, the general problem of traffic sign identification is to try and identify one or more traffic signs in the image using a priori information about the shape, color or features present in the traffic signs. The current solutions in the literature commonly involves segmentation of traffic signs from the scenes (traffic sign detection), feature extraction from the traffic sign regions, and recognition. In this paper, we are only interested in the traffic sign detection part of the identification problem.

For traffic sign detection, a variety of techniques have been proposed in the literature (see overviews in [1], [2], [3]), which can be grouped in three main categories: color-based methods, shape-based methods and feature-based methods. Color-based methods [4] usually apply color segmentation combined with edge detection techniques to find specific shapes corresponding to traffic signs in images. Shape-based methods rely mostly in edge information to extract geometric constraints that correspond to traffic signs, like circles in the images [5]. In addition, radial symmetry [6] can be employed to detect regular shapes like triangles, squares, octagons, etc. Feature-based methods apply machine learning techniques to special features detected in the images. Among the most commonly used techniques are Neural Networks [7], Support Vector Machines [8] and AdaBoost methods [9].

In this paper, we present a biologically inspired approach to traffic sign detection based on Virtual Generalizing Random Access Memory Weightless Neural Networks (VG-RAM WNN [10]). VG-RAM WNN are effective machine learning tools that offer simple implementation and fast training and test. Our VG-RAM WNN systems model the biological saccadic eye movement system and the transformation suffered by the images captured by the eyes from the retina to the superior colliculus of the mammalian brain.

We evaluated the performance of our system using the German Traffic Sign Detection Benchmark (GTSDB) (http://benchmark.ini.rub.de) [11]. Our experimental results showed that our approach is capable of reliably and efficiently detect a large variety of traffic sign categories **using only 12 traffic sign images for training**.

This paper is organized as follows. After this introduction, in Section II we briefly discuss the saccadic eye movement system and the transformation suffered by the images captured by the eyes from the retina to the superior colliculus of the mammalian visual system. In Section III,

Manuscript received on March 1st, 2013. This work was supported in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico-CNPq-Brasil (grants 552630/2011-0, 308096/2010-0, and 314485/2009-0) and Fundação de Amparo à Pesquisa do Espírito Santo-FAPES-Brasil (grant 48511579/2009).

Alberto F. De Souza, Cayo Fontana, Filipe Mutz, Tiago Alves de Oliveira, Mariella Berger, Avelino Forechi, Jorcy de Oliveira Neto, Edilson de Aguiar, and Claudine Badue are with the Departamento de Informática, Universidade Federal do Espírito Santo, Vitoria, ES, 29075-910, Brazil (phone: +55-27-4009-2138; fax: +55-27-4009-5848; e-mails: alberto@lcad.inf.ufes.br, claudine@lcad.inf.ufes.br).

we present a biologically inspired VG-RAM WNN architecture for traffic sign detection. In Section IV, we describe our experimental methodology and analyze our experimental results. Our conclusions and directions for future work follow in Section V.

II. Saccadic System and Mapping from the Retina to the Superior Colliculus

The saccadic eye movement subsystem of the mammalian visual system is the main responsible for pointing the fovea towards objects of interest [12]. The fovea is the central region of the retina that has the highest density of receptors and thus affords the greatest visual acuity. The saccade system produces rapid eye movements (saccades) that shift the fovea rapidly to a visual target (saccade target) in the visual field. The purpose of the saccade is to move the eyes as quickly as possible. As there is no time for large visual feedback to significantly modify the course of a saccade, corrections to the direction of eyes movement are typically made in successive saccades.

The saccadic eye movements are controlled by the midbrain's superior colliculus (SC). The images captured by the eyes are transformed into electrical impulses by the retina and, through the optic nerve, are projected into the SC and other cerebral areas [12]. The neural projection from the retina to SC follows a retinotopic mapping, i.e., neighboring regions in the retina are projected onto neighboring regions of the SC [13]. Before a saccadic movement, cells in the SC are activated and a winner-takes-it-all behavior leads to the selection of a point in the visual field retinotopically mapped in the SC—this point is the target of the saccade [14].

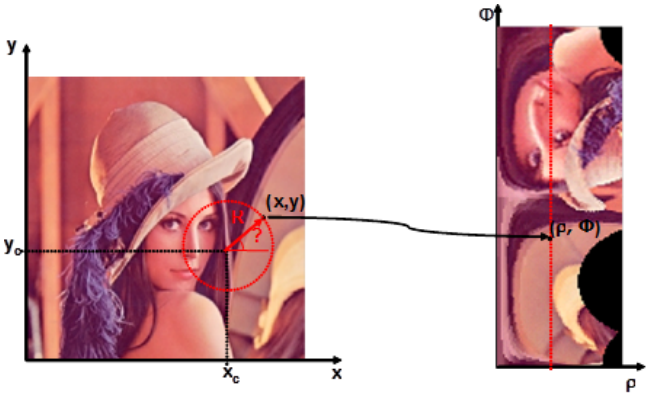


Figure 1: Log-polar transform.

The mapping from the retina to SC follows a log-polar function [13]. Figure 1 shows the log-polar transform of an image, centered on the point (x_c, y_c)—this point corresponds to the center of attention in the visual field. Note that the circle (in red) in the left image of Figure 1 becomes a straight line in the right image, and the regions around the circle's center (the fovea of the model) in the left image occupy a much larger area in the right image. The mathematical modeling of the log-polar transform commonly used in the literature is given by:

$$R = \sqrt{(x - x_c)^2 + (y - y_c)^2} \rightarrow \rho \, \alpha \log(R) \text{ and} \quad (1)$$

$$\theta = \arctan\left(\frac{(y - y_c)}{(x - x_c)}\right) \rightarrow \phi \, \alpha \, \theta \,. \quad (2)$$

In this paper, we did not employ the log-polar transform exactly as shown above, but a variant that was created to emulate more precisely the mapping from the retina to SC. Figure 2 shows this variant of the log-polar transform. As Figure 2 shows, neighboring regions in the image around the circle's center (the fovea of the model) are also neighbors in the log-polar transform (retinotopy), as occurs in the SC. This does not occur in the transform depicted in Figure 1.

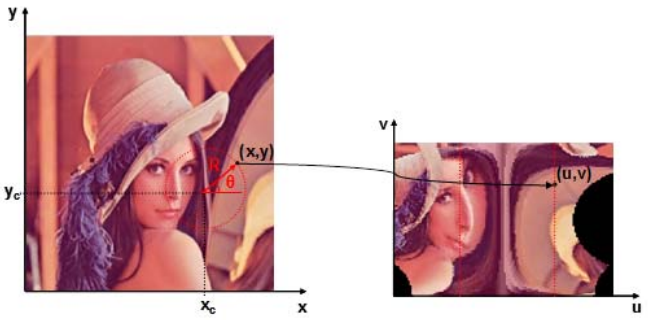


Figure 2: Our variant of the log-polar transform.

III. Traffic Sign Detection with VG-RAM WNN

A. VG-RAM WNN

RAM-based neural networks, also known as n-tuple classifiers or weightless neural networks, do not store knowledge in their connections but in Random Access Memories (RAM) inside the network's nodes, or neurons. These neurons operate with binary input values and use RAM as lookup tables: the synapses of each neuron collect a vector of bits from the network's inputs that is used as the RAM address, and the value stored at this address is the neuron's output. Training can be made in one shot and basically consists of storing the desired output in the address associated with the neuron's input vector [15].

In spite of their remarkable simplicity, RAM-based neural networks are very effective as pattern recognition tools, offering fast training and test, in addition to easy implementation [10]. However, if the network input is too large, the memory size becomes prohibitive, since it must be equal to 2^n, where n is the input size.

Virtual Generalizing RAM (VG-RAM) Weightless Neural Networks (WNN) are RAM-based neural networks that only require memory capacity to store the data related to the training set [16]. In the neurons of these networks, the memory stores the input-output pairs shown during training, instead of only the output. In the test phase, with a **distributed neuron memory** model, each neuron searches associatively its memory by comparing the input presented to the network with all inputs in the input-output pairs learned; with a **shared neuron memory** model, each neuron searches the memory of all the network's neurons. The

output of each VG-RAM WNN neuron is taken from the pair whose input is nearest to the input presented—the distance function employed by VG-RAM WNN neurons is the Hamming distance. If there is more than one pair at the same minimum distance from the input presented, the neuron's output is chosen randomly among these pairs.

Considering a distributed neuron memory model, Table 1 shows the lookup table of a VG-RAM WNN neuron with three synapses (X_1, X_2 and X_3). This lookup table contains three entries (input-output pairs), which were stored during the training phase (*entry #1*, *entry #2* and *entry #3*). During the test phase, when an input vector (*input*) is presented to the network, the VG-RAM WNN test algorithm calculates the distance between this input vector and each input of the input-output pairs stored in the neuron's lookup table. In the example of Table 1, the Hamming distance from the *input* to *entry #1* is two, because both X_2 and X_3 bits do not match the input vector. The distance to *entry #2* is one, because X_1 is the only non-matching bit. The distance to *entry #3* is three, as the reader may easily verify. Hence, for this input vector, the algorithm evaluates the neuron's output, *Y*, as *label 2*, since it is the output value stored in *entry #2*.

Table 1: VG-RAM WNN neuron lookup table.

Lookup Table	X_1	X_2	X_3	Y
entry #1	1	1	0	*label 1*
entry #2	0	0	1	*label 2*
entry #3	0	1	0	*label 3*
	↑	↑	↑	↓
input	1	0	1	*label 2*

B. VG-RAM WNN Architecture for Traffic Sign Detection

Our VG-RAM WNN architecture for traffic sign detection, named TSD, has a single bidimensional array of $m \times n$ neurons, N, where each neuron, $n_{i,j}$, has a set of synapses, $W = (w_1, w_2, ...w_{|W|})$, which are connected to the network's bidimensional input, Φ, of $u \times v$ pixels, $\varphi_{k,l}$ (Figure 3). The mapping of the elements of Φ onto the center of the *receptive field* of each neuron of N follows a log-polar function, which models the mapping from the retina to SC (Section II).

The synaptic interconnection pattern of each neuron $n_{i,j}$ (which consubstantiates its receptive field), $\Omega_{i,j,\sigma}(W)$, follows a bidimensional Normal distribution with variance σ^2 centered at φ_{μ_k,μ_l}, where the coordinates μ_k and μ_l of Φ are given by the inverse log-polar function of the coordinates i and j of N; i.e., the distribution of coordinates k and l of the pixels of Φ to which $n_{i,j}$ connects via W follow the probability density functions:

$$\omega_{\mu_k,\sigma^2}(k) = \frac{1}{\sigma\sqrt{2\Pi}} e^{-\frac{(k-\mu_k)^2}{2\sigma^2}} \text{ and} \quad (3)$$

$$\omega_{\mu_l,\sigma^2}(l) = \frac{1}{\sigma\sqrt{2\Pi}} e^{-\frac{(l-\mu_l)^2}{2\sigma^2}}, \quad (4)$$

where σ is a parameter of the architecture, and the coordinates μ_k and μ_l of the pixel of Φ where the Normal distribution is centered at are calculated by:

$$\mu_k = \frac{u}{2} + d \cdot \cos(\theta) \text{ and} \quad (5)$$

$$\mu_l = \frac{v}{2} + d \cdot sen(\theta), \quad (6)$$

where

$$d = \frac{u}{2} \cdot \left(\frac{\alpha^{\left|\frac{i-m/2}{m/2}\right|} - 1}{\alpha - 1} \right) \text{ and} \quad (7)$$

$$\theta = \begin{cases} \pi \cdot \left(\frac{3n}{2} - \frac{j}{n} \right) + \frac{\pi}{2n}; \text{if } k < \frac{m}{2} \\ \pi \cdot \left(\frac{3n}{2} + \frac{j}{n} \right) + \frac{\pi}{2n}; \text{ if } k > \frac{m}{2} \end{cases}, \quad (8)$$

and α is the log-factor of the log-polar function and is a parameter of the architecture

The $\Omega_{i,j,\sigma}(W)$ synaptic interconnection pattern mimics that observed in many classes of biological neurons [12]. It is randomly created when the network is built and does not change afterwards; furthermore, although random, it is the same for all neurons. Moreover, the memory of all neurons is shared (Section III.A). Thus, when the network is trained, each neuron learns the association between the information collected by its synapses and a given output, but all neurons share everyone else knowledge afterwards.

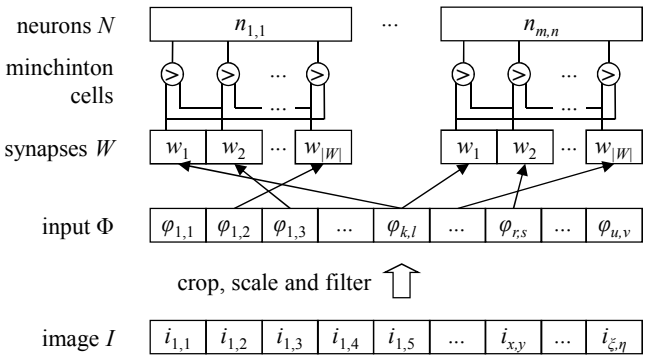


Figure 3: Schematic diagram of our VG-RAM WNN architecture for traffic sign detection.

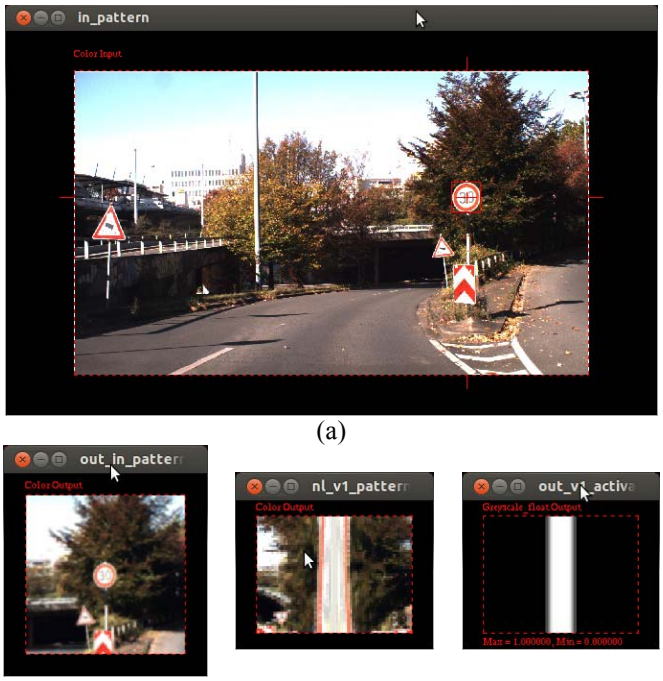
VG-RAM WNN synapses can only get a single bit from the input. Thus, in order to allow our VG-RAM WNN to deal with images, in which a pixel may assume a range of different values, we use *minchinton cells* [17]. In the proposed VG-RAM WNN architecture, each neuron's synapse, w_t, forms a minchinton cell with the next, w_{t+1} ($w_{|W|}$ forms a minchinton cell with w_1). The type of the minchinton cell we have used returns 1 (one) if the synapse w_t of the cell is connected to an input element, $\varphi_{k,l}$, whose

value is larger than the value of the element $\varphi_{r,s}$ to which the synapse w_{t+1} is connected, i.e., $\varphi_{k,l} > \varphi_{r,s}$; otherwise, it returns zero (see the synapses w_1 and w_2 of the neuron $n_{1,1}$ of Figure 3).

During training, the training image is cropped by a square centered at the traffic sign image center. The crop square size is estimated to contain the traffic sign image and part of the background. We use distinct scale factors for the crop square size to match various sizes of traffic sign images. The image patch extracted from the training image by the crop square is scaled to fit into Φ, filtered by a Gaussian filter to smooth out artifacts produced by the scaling, and its pixels are copied to Φ. The image patch's center is used as the center of the log-polar function (center of attention in the visual field) that maps Φ onto N. Each neuron is then trained to output a value different than zero if the center of its receptive field is within a circle with radius r centered at the image patch's center, and zero otherwise, where r is a parameter of the architecture. The trained value decreases as the center of the receptive field of the neurons moves away from the circle's center until it reaches zero at its border. This procedure is repeated for all images in the training set. During training, the scale factor used in the scaling mentioned above is set to a value that puts the traffic sign image precisely within the circle of radius r.

Figure 4 shows a training instance, where the TSD neural network is trained to detect a traffic sign in a training image. Figure 4(a) shows the training image with the border and center of the ground truth bounding box marked with a red square and red cross, respectively; Figure 4(b) shows the transformed (scaled and filtered) image patch extracted from the training image; Figure 4(c) shows the log-polar mapping of Φ onto N (this is only for visualization); and Figure 4(d) shows the output of N after training. As Figure 4(d) shows, neurons with the center of their receptive fields within the circle centered at the image patch's center (compare the Figure 4(c) with the Figure 4(d)) are trained to produce outputs with values higher than zero (white or gray), while those with the center of its receptive field far from the circle's center are trained to output zero (black). Note that circles become rectangles after our log-polar transform, and the regions around the center of the transform occupy a much larger area (compare the representations of the circular traffic sign with that of the triangular traffic sign in Figure 4(c)).

During testing, the test image is probed by our system at several points regularly spaced. We use 12 distinct scale factors for this probing to match various sizes of traffic sign images, mimicking the procedure of looking at the image from different distances. The probing is made using horizontal and vertical shifts of the center of the log-polar function (center of attention) in the test image proportional to the size of the crop square (window of attention) mentioned above, such that the square shifts yield partially overlapped image patches, which allows detection of traffic sign images in the boundaries of the window of attention.



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

Figure 4: Example of a training instance of our VG-RAM WNN architecture for traffic sign detection (TSD).

Each image patch extracted from the test image is scaled (to fit into Φ), filtered (by a Gaussian filter to smooth out artifacts), and its pixels are copied into Φ. The neurons then generate their output according to their receptive fields. Neurons with receptive field on regions of the image patch similar to regions of traffic sign images previously trained generate outputs with high values. After a procedure equivalent to the winner-takes-its-all behavior observed in the SC [14], the neuron with the highest output is selected and the inverse log-polar function in the coordinates of this neuron is used to compute the coordinates of a point (saccade target) in the image patch possibly belonging to a traffic sign image. The log-polar function center (center of attention) is moved (saccade) to this point in the test image and the neurons' outputs are recomputed. This procedure is repeated one or more times, or until the log-polar function center does not move anymore. A **matching score**, that quantifies the similarity between the center of the image patch and a traffic sign, is computed by comparing the image of the output of N after testing to that after training (Figure 4(d)). If the output of N after testing is similar to the output of N after training, a traffic sign image might have been detected in the image patch.

Figure 5 shows a testing instance, where neurons in the network, trained to detect traffic signs, generate their outputs according to the image region monitored by their receptive fields. Figure 5(a) shows the test image with the center of attention marked with a red cross; Figure 5(b) shows the transformed (scaled and filtered) image patch extracted from the test image; Figure 5(c) shows the log-polar mapping of Φ onto N (just for visualization); and Figure 5(d) shows the output of N before the saccade. As Figure 5(d) shows, neurons with the center of their receptive fields on a traffic

sign image generate higher outputs (compare the Figure 5(c) with the Figure 5(d)). Figure 5(e) to Figure 5(h) are equivalent to Figure 5(a) to Figure 5(d), with the difference that they illustrate the testing instance after the saccade. As Figure 5(h) shows, the output of *N* after the saccade is very similar to the output of *N* after training (compare Figure 4(d) with Figure 5(h)), which indicates that a traffic sign image might have been detected in the image patch. An animation of a single training and several subsequent saccades is available at http://youtu.be/H_LdE8fcbF4.

The TSD degree of belief that a traffic sign has really been detected in the image patch is estimated using Bayesian inference, as described below in Section III.C. The whole system final decision is regulated by a threshold: if the degree of belief is larger than this threshold, then a traffic sign has been detected. Nevertheless, the detected traffic sign image might be out of the center of the image patch extracted from the test image (as it is in Figure 5(e)). To precisely find the center of the detected traffic sign image, we employ a second VG-RAM WNN that explores the symmetry present in the traffic sign image, as described below in Section III.D.

C. *Degree of Belief in the Traffic Sign Detection*

Our system maps the matching score—that quantifies the similarity between the center of the image patch and a traffic sign (Section III.B)—into a probability measure. This probability measure is expressed as $p(D \mid M, S, X, Y)$, where D is a binary random variable, and $D = True$ if a traffic sign has been detected and $D = False$ if it has not; and $M = \{m_1, m_2, \ldots, m_{|M|}\}$, $S = \{s_1, s_2, \ldots, s_{|S|}\}$, $X = \{x_1, x_2, \ldots, x_{|X|}\}$ and $Y = \{y_1, y_2, \ldots, y_{|Y|}\}$ are discrete random variables that represent the discretization of the possible values of matching scores, scale factors, and saccade target x and y coordinates in the test image, respectively. Using the Bayes' Theorem, the probability that a traffic sign image has been detected in the image patch ($p(D = True)$), given that the network computed a matching score in the interval m_i, the image patch was extracted from the test image according to the scale factor s_i, and the saccade target is within the range of horizontal coordinates x_i and vertical coordinates y_i of the test image, can be formulated as:

$$p(D \mid M,S,X,Y) = \frac{p(D) \times p(M,S,X,Y \mid D)}{p(M,S,X,Y)}, \quad (9)$$

where

$$p(M,S,X,Y \mid D) = P(M \mid D)P(S \mid D)P(X \mid D)P(Y \mid D) \text{ and} \quad (10)$$

$$p(M,S,X,Y) = p(D) \cdot P(M \mid D)P(S \mid D)P(X \mid D)P(Y \mid D) + \\ + P(\sim D) \cdot P(M \mid \sim D)P(S \mid \sim D)P(X \mid \sim D)P(Y \mid \sim D) \quad (11)$$

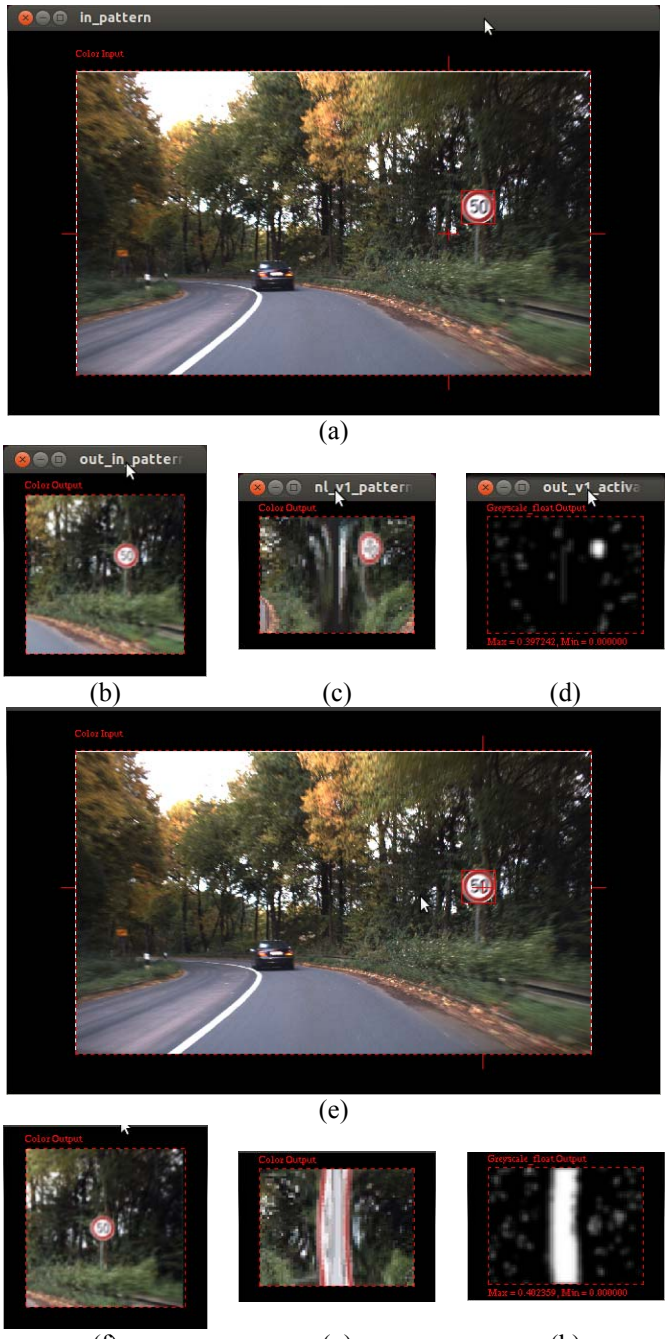


Figure 5: Example of a testing instance of our VG-RAM WNN architecture for traffic sign detection (TSD).

To estimate the values of $p(D)$, $p(\sim D)$, $p(M \mid D)$, $p(M \mid \sim D)$, $p(S \mid D)$, $p(S \mid \sim D)$, $p(X \mid D)$, $p(X \mid \sim D)$, $p(Y \mid D)$,), and $p(Y \mid \sim D)$, we used a training subset and a validation subset. We trained the network with the images of the training subset, examined the TSD output with the validation subset, and estimated the values of the terms of the Equations (10) and (11) mentioned above.

The probability that a traffic sign has (or has not) been detected, $p(D = True)$ (or $p(D = False)$), can be estimated as the percentage of saccades that hit (or do not hit) the associated ground truth bounding boxes of the validation

subset, according to the four constraints described below in this section.

The probabilities $p(M = m_i \mid D = True)$ (or $p(M = m_i \mid D = False)$) can be estimated as the percentage of matching scores in each interval m_i, given that a traffic sign has (or has not) been detected, i.e., $D = True$ (or $D = False$). The probabilities $p(S = s_i \mid D = True)$ (or $p(S = s_i \mid D = False)$) can be estimated as the percentage of image patches extracted according to a scale factor s_i, given that $D = True$ (or $D = False$). The probabilities $p(X = x_i \mid D = True)$ (or $p(X = x_i \mid D = False)$) can be estimated as the percentage of saccade targets within the range of horizontal coordinates x_i, given that $D = True$ (or $D = False$). Finally, the probabilities $p(Y = y_i \mid D = True)$ (or $p(Y = y_i \mid D = False)$) can be estimated as the percentage of saccade targets within the range of vertical coordinates y_i, given that $D = True$ (or $D = False$).

A saccade hits a ground truth bounding box if four constraints are satisfied. Let d_x and d_y be distances in the x-axis and y-axis, respectively, between the saccade target and the center of the ground truth bounding box; w_{gt} the width of the ground truth bounding box; and w_{ei} the traffic sign estimated width as a function of s_i ($w_{ei} = c/s_i$, where c is a parameter of our system). The first constraint specifies that the saccade target must be near the center of the ground truth bounding box, i.e.:

$$d_x \leq 0.8 \times \frac{w_{gt}}{2} \text{ and} \tag{12}$$

$$d_y \leq 0.8 \times \frac{w_{gt}}{2}. \tag{13}$$

The second that the image patch size must be close to the size of the ground truth bounding box, i.e.:

$$w_{gt} \times 0.7 \leq w_{ei} \leq w_{gt} \times 1.2. \tag{14}$$

The third that the matching score must be greater or equal to 0.3, and the fourth that the traffic sign image must belong to a pre-defined category of traffic signs.

Our system final decision is regulated by a threshold: if $p(D = True \mid M = m_i, S = s_i, X = x_i, Y = y_i)$ is larger than the threshold, then a traffic sign has been detected. The threshold value can either be specified by the system's user or automatically tuned using a training subset and a validation subset, by varying the threshold value until the performance of interest in terms of precision and recall is achieved (the validation subset is part of the training dataset but not of the test dataset [18]).

D. VG-RAM WNN for Traffic Sign Centralization

We employ a second VG-RAM WNN that explores the symmetry present in the detected traffic sign images to try and find their center. This second VG-RAM WNN, named TSC, has the same architecture of the first one (TSD), previously presented in Section III.B, and operates in the three steps described below.

In the first step, the center of attention of TSC (the center of its log-polar) is pointed at a traffic sign image detected by TSD (actually, TSD's saccade target). To do that, an image patch, centered at the target of the TSD's saccade, is extracted from the test image by a crop square using the same scale factor used by TSD for detection. This image patch is scaled and filtered, and its pixels are copied to the input Φ of TSC. TSC is, then, trained to learn the appearance of the image surrounding TSD's saccade target. The training procedure is identical to the one followed by TSD and described previously in Section III.B, except by the parameter r that, in the case of TSC, it is set in such a way that TSC's neurons learn to output values different than zero in a smaller region of the traffic sign image that TSD has possibly detected, instead of the whole traffic sign image.

In the second step, the image patch is swapped horizontally and the TSC neurons generate their output according to their receptive fields. If TSD's saccade hit the center of a traffic sign image, its swapped image will appear very similar due to the traffic sign image symmetrical form. Therefore, in this case, the TSC neurons will show activation very similar to the one learned and a TSC saccade will not move its center of attention. In the other hand, if TSD's saccade hit a point not in the center of the traffic sign image, the image region that will appear similar will be a region symmetric to the one learned by TSC. Therefore, the TSC neurons that will show activation will be those that have receptive field in regions where the swapped image is symmetrical to the learned image. A TSC saccade will, then, move its center of attention to this region. The distances in the x-axis, d_x^h, and y-axis, d_y^h, between the current center of attention (the target of TSC saccade), (x_s, y_s), and the target of the TSD saccade, (x_c, y_c), are computed and saved ($d_x^h = x_s - x_c$ e $d_y^h = y_s - y_c$). Considering the symmetrical appearance of traffic sign images, it is expected that TSC's saccade in this step is mostly horizontal (i.e., $d_y^h \approx 0$).

In the third step, the image patch is swapped vertically and the same procedure is followed. The distances in the x-axis, d_x^v, and y-axis, d_y^v, between the current center of attention (the target of TSC saccade) and the target of the TSD saccade are computed and saved. In this case, it is expected that TSC's saccade is mostly vertical (i.e., $d_x^v \approx 0$). Actually, if $|d_y^h| \leq 0.2 \times |d_x^h|$ and $|d_x^v| \leq 0.2 \times |d_y^v|$, our system consider that the image patch contains a traffic sign image and that its center has been found at the point $(x_c - \frac{d_x^h}{2}, y_c - \frac{d_y^v}{2})$. In this case, the crop square's center is moved to this point and the traffic sign detection process is considered complete. The traffic sign bounding box can then be easily computed using this point coordinates and the current scale factor.

If the first TSC saccade is not mostly horizontal or the second is not mostly vertical, TSC neurons' memories are deleted so that this procedure can be repeated starting from a point chosen randomly near (x_c, y_c) and a higher scale factor. So, while $|d_y^h| > 0.2 \times |d_x^h|$ and $|d_x^v| > 0.2 \times |d_y^v|$ and a

maximum number of iterations has not been reached, the whole process is repeated. If the maximum number of iterations is reached, our system considers that the image patch does not contain a traffic sign image. An animation of the TSC operation is available at http://youtu.be/SZ9w1XBWJqE.

IV. Experimental Evaluation and Discussion

A. Experimental Methodology

To evaluate the performance of VG-RAM WNN on traffic sign detection, we used the German Traffic Sign Detection Benchmark (GTSDB) (http://benchmark.ini.rub.de) [11]. The GTSDB dataset contains 600 images in the training dataset and 300 images in the test dataset. All images have a resolution of 1360×800 pixels. Each image might or might not contain traffic signs, which are categorized into prohibitory, danger and mandatory categories that are, in turn, subcategorized into 12, 15 and 8 subcategories, respectively.

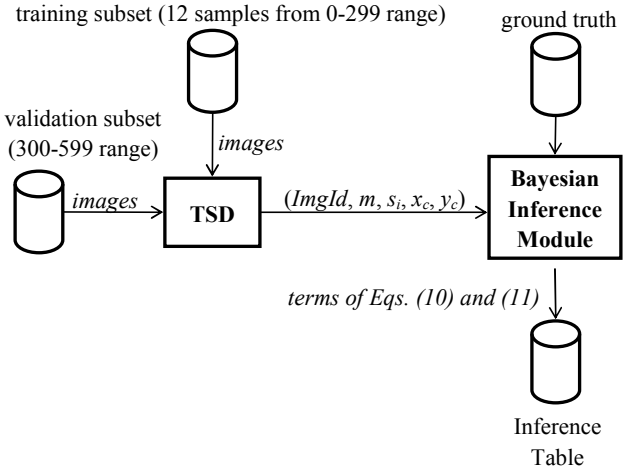


Figure 6: Flow chart of the validation phase of our system.

Figure 6 shows the flow chart of the *validation phase* of our system, which is used for estimating the values of the terms of the Equations (10) and (11). In this phase, TSD uses as input a training subset and a validation subset (see Figure 6). The training subset is composed of **only 12 traffic sign images** selected from within the first 300 images (0-299 range) of the GTSDB training dataset. These 12 traffic sign images are randomly taken from those images that contain traffic signs of the prohibitory category. The validation subset is composed of the remaining 300 images (300-599 range) of the GTSDB training dataset. We trained TSD with the 12 traffic sign images of the training subset and tested it with the whole images of the validation subset. TSD outputs a (*ImgId*, m, s, x_c, y_c) tuple for each test instance, where *ImgId* is the image identification number, m is the matching score, s is the scale factor, and x_c and y_c are the coordinates of the saccade target (we used a single saccade for each test instance). These tuples are fed to the Bayesian Inference Module, which accumulates these tuples for the whole validation subset and, after that, using the GTSDB training dataset ground truth, estimates and stores the terms of Equations (10) and (11) in the Inference Table.

Figure 7 shows the flow chart of the *test phase* of our system. In this phase, TSD uses as input a training subset and a test subset (see Figure 7). The training subset is composed of **only 12 traffic sign images**, the same used in the validation phase. The test dataset is composed of all 300 images (0-299 range) of the GTSDB test dataset, or the last 300 images (300-599 range) of the GTSDB training dataset. We trained TSD the same way as in the validation phase and tested it with the whole images of the test dataset. TSD output was fed to the Bayesian Inference Module, that, in this phase, thanks to the Inference Table built in the validation phase, is able to output a tuple that includes p, which is a short for $p(D = True \mid M = m_i, S = s_i, X = x_i, Y = y_i)$, and the same members of the tuple it received from TSD (see Figure 7). If this tuple is above a threshold, it is fed to TSC, otherwise it is discarded. TSC tries and finds the center of the traffic signs it receives or discards the tuple as well. The traffic signs properly centered by TSC are outputted as bounding boxes in the test image that can be compared with the GTSDB dataset ground truth.

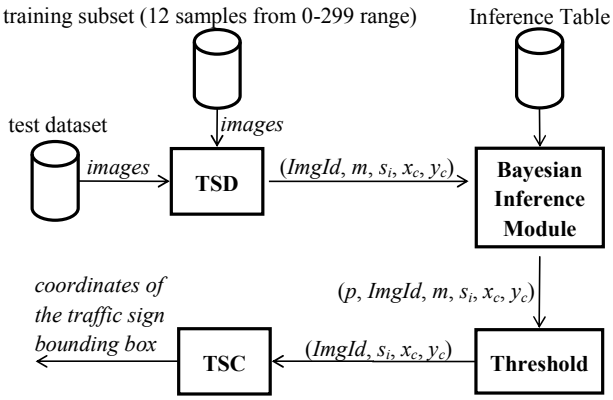


Figure 7: Flow chart of the test phase of our system.

B. Experimental Results

TSD and TSC architectures have 5 parameters: (i) the number of neurons, $m \times n$; (ii) the number of synapses per neuron, $|W|$; (iii) the size of the network input, $u \times v$; (iv) the standard deviation, σ, of the two-dimensional Normal distribution followed by the synaptic interconnection pattern of the neurons, Ω; and (v) the log-factor, α, of the log-polar function that maps Φ onto N. We have used the following parameters in our experiments, which were chosen in an ad hoc manner: (i) number of neurons equal to 65 × 49; (ii) number of synapses per neuron equal to 256; (iii) size of the network input equal to 201 × 201; (iv) σ equal to 5; and (v) α equal to 2.

We have used 12 different scale factors, which were chosen in such way that the largest traffic sign in the GTSDB database (128 × 128 pixels) could fit into the receptive field of the TSD neurons trained with a value different than zero in the case of the smaller scale factor; and the smaller traffic sign image (16 × 16 pixels) could fit in the receptive field of the TSD neurons in the case of the

highest scale factor.

To evaluate the contributions of each module of our traffic sign detection system, we examined its performance (i) considering only TSD, (ii) considering TSD and the Bayesian Inference Module, and (iii) considering the whole system. Figure 8 shows the performance of our system in a precision × recall curve for the GTSDB training dataset in the (i), (ii) and (iii) scenarios, named *Matching score*, *Probability* and *Probability+Symmetry* scenarios, respectively.

The graph of Figure 8 has three curves, one for each scenario. As the graph of Figure 8 shows, the whole system performs the best (*Probability+Symmetry* scenario), i.e., the area under the precision × recall curve is the largest, when using the whole system. However, the performance of the system using the scenarios *Matching score* and *Probability* are, at first glance, unexpected.

The graph of Figure 8 shows that the area under the precision × recall curve for the *Probability* scenario is smaller than that of the *Matching score* scenario and the reverse would at first be expected. However, this happens because the Bayesian Inference Module removes most of the tuples it receives even using a low probability as threshold. In the *Matching score* scenario, on the other hand, many tuples are accepted as valid traffic signs and some of them end up contributing to the recall of this reduced version of our system. One may reason that, in spite of that, this version should be preferred and the whole system should not include the Bayesian Inference Module. However, this would put a large burden in the TSC module, since, in the *Matching score* scenario, much more tuples would be submitted to it.

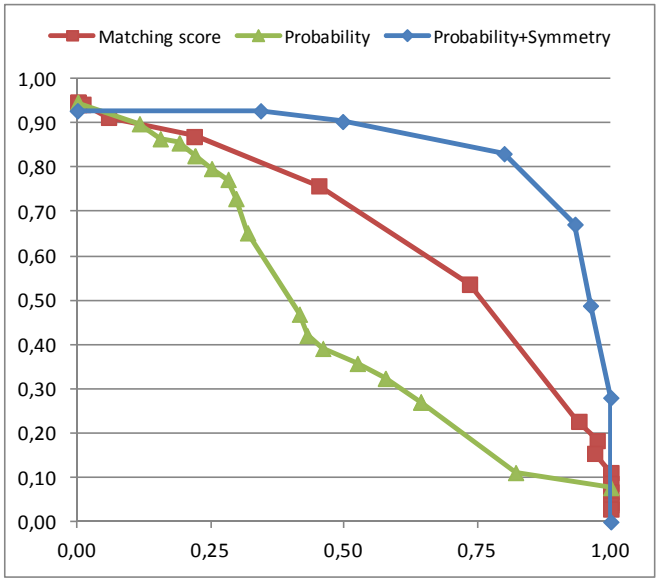


Figure 8: Performance of our system: precision × recall—GTSDB training dataset.

The difference in area between the *Probability+Symmetry* and the *Probability* curves demonstrates the benefits of using the TSC module for performing symmetry correction—most tuples that would be considered false positives are corrected by TSC and contributes for the better overall recall shown by the *Probability+Symmetry* curve.

We have submitted the results of our system for traffic sign detection to the GTSDB evaluation webpage on February 28th, 2013. Our system was ranked between the first 16 methods for the prohibitory category, showing an area of 85.12% under the precision × recall curve. The graph in Figure 9 shows the data we collected from the GTSDB webpage—red curve with squared data points—together with the performance of our system with the training dataset previously shown in Figure 8—blue curve with diamond data points. As the graph in Figure 9 shows, the performance in the training set is consistent with the performance in the testing set.

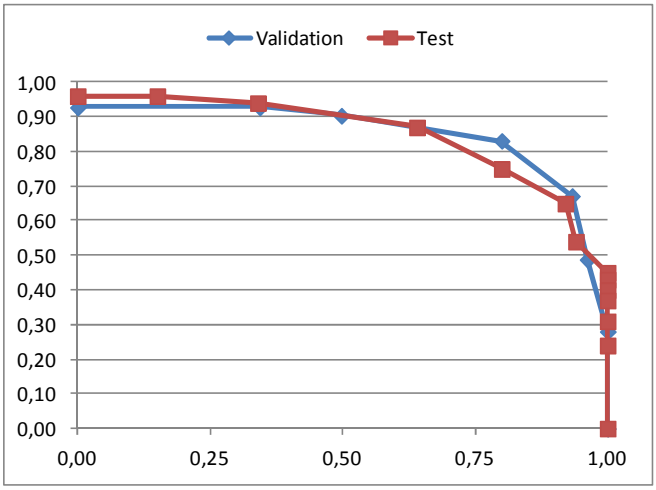


Figure 9: Performance of our system: precision × recall—GTSDB test dataset.

We have also measured the performance of our systems in terms of time. Running in a Dell Alienware Aurora R1 machine, with an Intel Core i7-930 quad-core processor (8M Cache, 2.8 GHz) and 12 GB of RAM, TSD can be trained in 3.85 seconds (12 traffic signs) and performs a saccade in 0.42 seconds on average, while TSC performs the whole process required for symmetry detection and correction in 0.18 seconds on average. Therefore, if the whole system detects a traffic sign in a single TSD saccade, the total detection time of a traffic sign is 0.60 s = 0.42 s + 0.18 s (the consumed by the Bayesian Inference Module is negligible). However, the system may not detect a traffic sign with a single TSD saccade. Actually, to cover a single image, in our experiments we performed 780 saccades because we use several scale factors—for the highest scale factor 162 saccades are necessary. Nevertheless, the system can be used in fewer and coarser scale factors, say the smallest scale factor, which requires a total of 6 TSD saccades. In this case, using current desktop computers, our whole system can operate at a rate of about one image each 3.6 s. It is important to mention that our implementation can be optimized for taking advantage of hardware accelerators, such as GPU, FPGA or digital signal processors, improving the time performance figures just mentioned, and, in such case, we believe it can be used on-line.

C. Discussion

The traffic sign detection performance of our system is the result of two factors. First, each synapse of TSD and TSC collects the result of a comparison between two pixels, which is executed by its corresponding minchinton cell. Our configurations for both systems have hundreds of synapses per neuron and thousands of neurons. Therefore, during testing, hundreds of thousands of such comparisons are performed on each input image and the results are checked against equivalent results learned from training images. This amount of pixel comparisons allows not only for high discrimination capability but also generalization. Second, thanks to the synapse interconnection patterns and log-polar architecture, each neuron of TSD or TSC monitors a specific region of the traffic sign, which reduces the overall impact of occlusion, and varying color and illumination conditions on the performance of these systems.

We believe that our system could show much better traffic sign detection performance if proper parameter tuning were employed—we adjusted all system parameters in a ad hoc manner—and if a large number of training images were used—we used only 12 traffic sign images in all experiments. We were unable to perform parameter tuning tests with more training samples, and tests with other categories of traffic signs due to time constraints.

V. Conclusions and Future Work

In this paper, we present a new approach for traffic sign detection based on Virtual Generalizing Random Access Memory Weightless Neural Networks (VG-RAM WNN). Our experiments showed that VG-RAM WNN can be employed for traffic sign detection with good accuracy, despite a very small number of training samples. The main advantage of VG-RAM WNN against other neural network approaches employed for traffic sign recognition is its simple implementation and fast training and test. For future work, we plan to tune the parameters of our VG-RAM WNN architecture and increase the number of training examples, which can improve its performance even further. We also would like to evaluate the proposed system on traffic signs of Brazil's road environment.

References

[1] S. Escalera, X. Baró, O. Pujol, J. Vitrià, and P. Radeva, *Traffic Sign Recognition Systems*, Springer Series: SpringerBriefs in Computer Science, 2011.

[2] D. Gavrila, "Traffic sign recognition revisited", in *Proceedings of the Mustererkennung 1999, 21. DAGM*-Symposium, pp. 86-93,1999.

[3] Y. Li, "Real-time traffic sign detection: an evaluation study", in *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)*, pp. 3033-3036, 2010.

[4] Y.-Y. Nguwi and A. Z. Kouzani, "Detection and classification of road signs in natural environments", *Neural Computing and Applications,* vol. 17, no. 3, pp. 265-289. 2008.

[5] T. M. Nguyen, S. S. Ahuja, and Q. M. J. Wu, "A real-time ellipse detection based on edge grouping," *IEEE International Conference on Systems, Man and Cybernetics*, pp.3280-3286, 2009.

[6] N. Barnes, A. Zelinsky, and L. S. Fletcher, "Real-time speed sign detection using the radial symmetry detector," *IEEE Transactions on Intelligent Transportation Systems*, vol.9, no.2, pp.322-332, 2008.

[7] Y. Aoyagi and T. Asakura, "A study on traffic sign recognition in scene image using genetic algorithms and neural networks", in *Proceedings of the 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, pp. 1838-1843, 1996.

[8] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. Lopez-Ferreras, "Road-sign detection and recognition based on support vector machines," *IEEE Transactions on Intelligent Transportation Systems*, vol.8, no.2, pp.264-278, 2007.

[9] X. Baro, S. Escalera, J. Vitria, O. Pujol, and P. Radeva, "Traffic sign recognition using evolutionary adaboost detection and forest-ECOC classification", *IEEE Transactions on Intelligent Transportation Systems*, vol.10, no.1, pp.113-126, 2009.

[10] I. Aleksander, "From WISARD to MAGNUS: A family of weightless virtual neural machines", in *RAM-Based Neural Networks*, J. Austin, Ed. Singapore: World Scientific, pp. 18-30, 1998.

[11] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: the German Traffic Sign Detection Benchmark", in *Proceedings of the International Joint Conference on Neural Networks*, 2013 (Submitted).

[12] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, *Principles of Neural Science*, 4th ed. McGraw-Hill, 2000.

[13] N. Tabareau, D. Bennequin, A. Berthoz, J. Slotine, and B. Girard, "Geometry of the superior colliculus mapping and efficient oculomotor computation", *Biological Cybernetics*, vol. 97, pp. 279-292, 2007.

[14] R. A. Marino, T. P. Trappenberg, M. Dorris, D. P. Munoz, "Spatial interactions in the superior colliculus predict saccade behavior in a neural field model", *Journal of Cognitive Neuroscience*, vol. 24, no. 2, pp. 315-336, 2012.

[15] I. Aleksander, "Self-adaptive universal logic circuits", *IEEE Electronic Letters*, vol. 2, no. 8, pp. 231-232, 1966.

[16] T. B. Ludermir, A. C. P. L. F. Carvalho, A. P. Braga, and M. D. Souto, "Weightless neural models: a review of current and past works, *Neural Computing Surveys*, vol. 2, pp. 41-61, 1999.

[17] R. J. Mitchell, J. M. Bishop, S. K. Box, and J. F. Hawker, "Comparison of some methods for processing grey level data in weightless networks", in *RAM-based Neural Networks*, J. Austin, Ed. Singapore: World Scientific, pp. 61-70, 1998.

[18] F. Sebastiani, "Machine learning in automated text categorization", *ACM Computing Surveys*, vol. 34, no. 1, pp. 1-47, 2002.