LETTER Communicated by Jochen Triesch

A Self-Organized Neural Comparator

Guillermo A. Ludueña
luduena@itp.uni-frankfurt.de
Claudius Gros
gros07@itp.uni-frankfurt.de
Institute for Theoretical Physics, Goethe University, Frankfurt am Main, Hessen 60438, Germany

Learning algorithms need generally the ability to compare several streams of information. Neural learning architectures hence need a unit, a comparator, able to compare several inputs encoding either internal or external information, for instance, predictions and sensory readings. Without the possibility of comparing the values of predictions to actual sensory inputs, reward evaluation and supervised learning would not be possible.

Comparators are usually not implemented explicitly. Necessary comparisons are commonly performed by directly comparing the respective activities one-to-one. This implies that the characteristics of the two input streams (like size and encoding) must be provided at the time of designing the system. It is, however, plausible that biological comparators emerge from self-organizing, genetically encoded principles, which allow the system to adapt to the changes in the input and the organism. We propose an unsupervised neural circuitry, where the function of input comparison emerges via self-organization only from the interaction of the system with the respective inputs, without external influence or supervision.

The proposed neural comparator adapts in an unsupervised form according to the correlations present in the input streams. The system consists of a multilayer feedforward neural network, which follows a local output minimization (anti-Hebbian) rule for adaptation of the synaptic weights. The local output minimization allows the circuit to autonomously acquire the capability of comparing the neural activities received from different neural populations, which may differ in population size and the neural encoding used. The comparator is able to compare objects never encountered before in the sensory input streams and evaluate a measure of their similarity even when differently encoded.

1 Introduction

In order to develop a complex targeted behavior, an autonomous agent must be able to relate and compare information received from the environment

with internally generated information (see Billing, 2010). For example, it is often necessary to decide whether the visual image currently being perceived is similar to an image encoded in some form in memory.

For artificial agents, such basic comparison capabilities are typically either hard-coded or initially taught, both processes involving the inclusion of predefined knowledge (Bovet and Pfeifer, 2005a, 2005b). However, living organisms acquire this capability only autonomously, by interaction with the acquired data, possibly without any explicit feedback from the environment (O'Reilly & Munakata, 2000). We can therefore hypothesize the presence of a neural circuitry in living organisms that is capable of comparing the information that different populations of neurons receive. It cannot therefore in general be assumed that these populations have a similar configuration, holding the information in the same encoding or even managing the same type of information.

A system encompassing these characteristics must be based on some form of unsupervised learning: it must self-organize in order to acquire its basic functionality autonomously. The task of an unsupervised learning system is to elucidate the structure in the input data without using external feedback. Thus, all the information should be inferred from the correlations found in the input and in its own response to the input data stream.

Unsupervised learning can be achieved using neural networks and has been implemented for a range of applications (see, e.g., Sanger, 1989; Atiya, 1990; Likhovidov, 1997; Furao, Ogura, & Hasegawa, 2007; Tong, Liu, & Tong, 2008). Higher accuracy is generally expected from supervised algorithms. However, Japkowicz, Myers, and Gluck (1995) and Japkowicz (2001) have shown that for the problem of binary classification, unsupervised learning in a neural network can perform better than standard supervised approaches in certain domains.

Neural and other less biologically inspired algorithms are often expressed in mathematical terms based on vectors and their respective elementary operations like vector subtraction, conjunction, and disjunction. Implementations in terms of artificial neural networks hence typically involve the application of these operations to the output of groups of neurons. These basic operations are, however, not directly available in biological neural circuitry, which is based exclusively on local operations. Connections among groups of neurons evolve during the growth of the biological agent and may induce the formation of topological maps (Kohonen, 1990) but generically do not result in one-to-one neural operations. For instance, these one-to-one neural interactions would involve an additional global summation of the result for the case of a scalar product.

It is unclear whether operations like vector operations are performed directly by biological systems. In any case, their implementation should be robust to the changes in the development of the system and its adaptation to different types of input. In effect, the basic building blocks of most known learning algorithms are the mathematical functions that computers

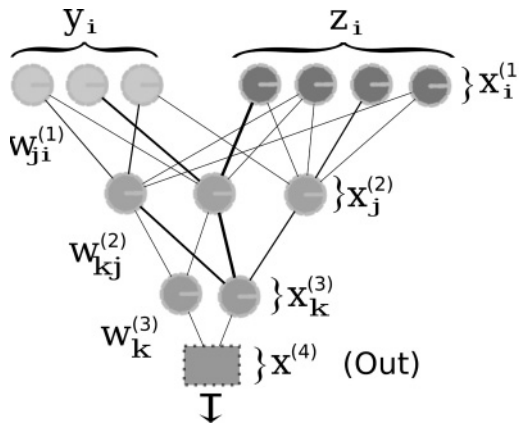

Figure 1: Architecture of the proposed neural comparator, with intermediate layers $\mathbf{x}^{(2)}$ and $\mathbf{x}^{(3)}$ and output $\mathbf{x}^{(4)}$. The comparator autonomously learns to compare the inputs $\mathbf{y}$ and $\mathbf{z}$. The output is close to zero when the two inputs are identical or correlated and close to unity when they are different or uncorrelated.

are based on. These are, however, not necessarily present, convenient, or viable in a biological system. Our aim is to elucidate how a basic mathematical function can emerge naturally in a biological system. We present, for this purpose, a model of how the basic function of comparison can emerge in an unsupervised neural network based on local rules for adaption and learning. Our adaptive "comparator" neural circuit is capable of self-organized adaption, with the correlations present in the data input stream as the only basis for inference.

The circuit autonomously acquires the capability of comparing the information received from different neural populations, which may differ in size and in the encoding used. The comparator proposed is based on a multilayer feedforward neural network, where the input layer receives two signals: $\mathbf{y}$ and $\mathbf{z}$ (see Figure 1). These two input streams can be unrelated, selected randomly, or, with a certain probability, encode the same information. The task of the neural comparator is then to determine, for any pair of input signals $\mathbf{y}$ and $\mathbf{z}$, whether they are semantically related. Generally any given pair ($\mathbf{y}$, $\mathbf{z}$) of semantically related inputs is presented to the system only once. The system has hence to master the task of discriminating generically between related and unrelated pairs of inputs, and not the task of extracting statistically repeatedly occurring patterns.

The strength of the synapses connecting neurons is readjusted using anti-Hebbian rules. Due to the readjustment of the synaptic weights, the network minimizes its output without the help of external supervision. As a consequence, the network is able to autonomously learn to discriminate

whether the two inputs encode the same information independent of whether the particular input configuration has been encountered before. The system will respond with a large output activity whenever the input pair ($\mathbf{y}$, $\mathbf{z}$) is semantically unrelated and with inactivity for related pairs.

1.1 Motivation and Expected Use Case. We are motivated by a system where the information stored in two different neuronal populations is to be compared. In particular, we are interested in systems like the one presented by Bovet and Pfeifer (2005a), where two streams of information (e.g., visual input and the desired state of the visual input or the signal from the whiskers of a robot compared to the time-delayed state of these sensors) encoded in two separate neuronal populations are to be compared—in this case, in order to get a distance vector between the two. In a fixed artificial system, one could obtain this difference by simply subtracting the input from each of the streams, provided that the two neuronal populations are equal, and encode the information in the same way. This subtraction can also be implemented in such a system in a neuromorphic way simply by implementing a distance function in a neural network. However, we are interested in the case where both neuron populations have evolved mostly independently, such that they might be structurally different and might encode the information in a different way, which is expected in a biological system. Under these conditions, the neuronal circuit comparing both streams should be able to invert the encoding of both inputs in order to compare them, a task that could not be solved using a fixed distance function. In addition, we expect that such a system would be deployed in an environment where it is more probable to have different, semantically unrelated inputs than otherwise. The comparator should hence be able to solve the demanding task of autonomously extracting semantically related pairs of inputs out of a majority of unrelated and random input patterns.

2 Architecture of the Neural Comparator

The neural comparator proposed consists of a feedforward network of three layers, plus an extra layer filtering the maximum output from the third layer (see Figure 1). We will refer to the layers as $k = 1, 2, 3, 4$, where $k = 1$ corresponds to the input layer and $k = 4$ to the output layer. The output of the individual neurons is denoted by $x_i^{(k)}$, where the supraindex refers to the layer and the subindex to the index of the neuron in the layer, for instance, $x_2^{(1)}$, being the output of the second neuron in the input layer.

The individual layers are connected by synaptic weights $w_{ji}^{(k)}$. In this notation, the index i corresponds to the index of the presynaptic neuron in layer k, and j corresponds to the index of the postsynaptic neuron in layer $k + 1$. Thus $w_{3,4}^{(1)}$ is the synaptic weight connecting the fourth input neuron with the third neuron in the second layer.

Table 1: Parameters Used in the Simulations.

Parameter	$t_{\max}$	$N^{(1)}$	$N^{(2)}$	$N^{(3)}$	α ($N < 400$)
Value	10^7	$2N$	N	$\lfloor \frac{N+1}{2} \rfloor$	2.7
Parameter	$p^{(1)}_{conn}$	$p^{(2)}_{conn}$	η	p_{eq}	α ($N \geq 400$)
Value	0.8	0.3	0.003	0.2	1.0

Note: N: Input vector size; $N^{(k)}$: size of layer k; $t_{\max}$: number of steps of simulation; p_{eq}: probability of equal inputs; $p^{(k)}_{conn}$: probability of connection from the kth layer to the $(k+1)$th layer; α: sigmoid slope; η: learning rate.

The layers are generally not fully interconnected. The probability of a connection between a neuron in layer k and a neuron in layer $k+1$ is $p^{(k)}_{conn}$. The values used for the interconnection probabilities $p^{(k)}_{conn}$ are given in Table 1.

In the implementation proposed and discussed here, the output layer is special in that it consists of only selecting the maximum of all activities from the third layer. There are simple neural architectures based on local operations that could fulfill this purpose; however, for simplicity, the task of selecting the maximum activity of the third layer is done here directly by a single unit.

2.1 Input Protocol. The input vectors $\mathbf{x}^{(1)}$ consist of two parts,

$$\mathbf{x}^{(1)} = (\mathbf{y}, \mathbf{z}), \tag{2.1}$$

where $\mathbf{y}$ and $\mathbf{z}$ are the two distinct input streams to be compared. We used the following protocol for selecting pairs of $(\mathbf{y}, \mathbf{z})$:

- $\mathbf{y}$ is selected randomly at each time step, with the elements $y_i \in [0, 1]$ drawn from a uniform distribution.
- $\mathbf{z}$ is selected via

$$\mathbf{z} = \begin{cases} \text{random} & \text{with probability } 1 - p_{eq} \\ \mathbf{f}(\mathbf{y}) & \text{with probability } p_{eq} \end{cases}. \tag{2.2}$$

If the inputs $\mathbf{z}$ and $\mathbf{y}$ carry the same information, they are related by $\mathbf{z} = \mathbf{f}(\mathbf{y})$, where $\mathbf{f}$ is generically an injective transformation. This relation reduces to $\mathbf{z} = \mathbf{y}$ for the case where the encodings in the two neural populations y and z are the identity.

We consider two kinds of encoding: direct encoding with $\mathbf{f}$ being the identity and encoding through a linear transformation, which we refer to as linear encoding:

$$\mathbf{z} = \mathbf{y} \qquad \text{or} \qquad \mathbf{z} = \hat{A}\mathbf{y}, \tag{2.3}$$

where $\hat{A}$ is a random matrix. The encoding is maintained throughout individual simulations of the comparator. For the case of linear encoding, the matrix $\hat{A}$ is selected initially and not modified during a single run.

The procedure we used to generate the matrix $\hat{A}$ consists of choosing each element of the matrix as a random number taken from a continuous flat distribution of values between -1 and 1. The matrix is then normalized such that the elements of vector $\mathbf{z}$ belong to $z_i = [-1, 1]$.

2.2 Synaptic Weights Readjustment: Anti-Hebbian Rule. Each neuron integrates its inputs via

$$x_j^{(k+1)} = g\left(\sum_i w_{ji}^{(k)} x_i^{(k)}\right), \qquad g(x) = \tanh(\alpha x), \tag{2.4}$$

with $g(x)$ being the transfer function, α the gain, and $w_{ji}^{(k)}$ the afferent synaptic weights. After the information is passed forward, the synaptic weights are updated using an anti-Hebbian rule:

$$\begin{aligned} \Delta w_{ji}^{(k)}(t) &\equiv w_{ji}^{(k)}(t+1) - w_{ji}^{(k)}(t) \\ &= -\eta x_i^{(k)}(t) x_j^{(k+1)}(t). \end{aligned} \tag{2.5}$$

Neurons under an anti-Hebbian learning rule will modify their synaptic weights in order to minimize their output. Note that anti-Hebbian adaption rules generically result from information maximization principles (Bell & Sejnowski, 1995). Information maximization favors spread-out output activities for statistically independent inputs (Marković & Gros, 2010), filtering out correlated input pairs $(\mathbf{y}, \mathbf{z})$, which tend to induce a low level of output activities.

The incoming synaptic weights to neuron i of the $(k+1)$th layer are additionally normalized after each update of the synaptic weights,

$$w_{ji}^{(k)}(t+1) \;\rightarrow\; \frac{w_{ji}^{(k)}(t+1)}{\sqrt{\sum_j \left|w_{ji}^{(k)}(t+1)\right|^2}}. \tag{2.6}$$

The algorithm proposed here is based on the idea that correlated inputs will lead to a small output as a consequence of the anti-Hebbian adaption

rule. Uncorrelated pairs of input $(\mathbf{y}, \mathbf{z})$ will generally generate a substantial output because they correspond to random inputs for which the synaptic weights are not adapted to minimize the output. It is worthwhile remarking that using a Hebbian adaption rule and classifying the minimum values as uncorrelated would not achieve the same accuracy as with the proposed anti-Hebbian rule with output values between -1 and 1. The reason is that we seek a comparator capable of comparing arbitrary pairs $(\mathbf{y}, \mathbf{z})$ of input, not specific examples.

When an anti-Hebbian rule is used, zero output is an optimum for any correlated input. In the case of input with equal encoding, this is reached when the synaptic weights cancel exactly ($w_{left} = -w_{right}$) in the first layer (see Figure 1). In contrast, if a Hebbian rule were used, the optimum values for correlated inputs correspond to the synaptic weights of correlated input being as large as possible. The consequence is that all synaptic weights tend to increase constantly, so that all output eventually achieves maximum values.

There remains, for anti-Hebbian adaption rules, a statistically finite probability that uncorrelated inputs will have a low output by mere chance; the terms $w_{ji}^{(k)} x_i^{(k)}$ originating from $\mathbf{y}$ and $\mathbf{z}$ may cancel out. In such cases, the comparator would be misclassifying the input. The occurrence of misclassification is reduced substantially by having multiple neurons in the third layer.

When there is an interlayer connection probability $p_{conn}^{(2)}$ well below unity, the individual neurons in the third layer will have access to different components of the information encoded in the second layer. This setup is effectively equivalent to generating different and independent parallel paths for the information transfer, adding robustness to the learning process, since only strong correlations between the input pairs $(\mathbf{y}, \mathbf{z})$, shared by the majority of paths, are then acquired by all neurons.

In addition to diminishing the possibility of random misclassification due to the multiple paths, the use of anti-Hebbian learning in the third layer minimizes the incidence of the individual parallel paths, which consistently result in $x_i^{(2)}$ outputs that are far larger than the rest (failing paths, since they are unable to learn some correlations). Thus, adding this layer results in a significant increase in accuracy with respect to an individual two-layer comparator. The accuracy is improved further by adding a filtering layer for input classification.

2.3 Input Classification. Each third-layer neuron encodes the estimated degree of correlation within the input pairs, $(\mathbf{y}, \mathbf{z})$. The fourth layer selects the most active third-layer neuron:

$$x^{(4)} = \max_j \left| x_j^{(3)} \right|. \tag{2.7}$$

By selecting the maximum of all outputs in the third layer, the circuit looks for a "consensus" among the neurons in the third layer. A given input pair (**y**, **z**) needs to be considered as correlated by all third-layer neurons in order to be classified as correlated by the fourth layer. This, together with the randomness of the interlayer connections, increases the robustness of the classification process.

There are several options for evaluating the effectiveness of the neural comparator. We discuss in section 4 an analysis in terms of fuzzy logic and consider now a measure of the accuracy of the system in terms of binary classification. The inputs **y** and **z** are classified according to the strength of the value of $x^{(4)}$. For binary classification, we use a simple threshold criterion. The inputs **y** and **z** are considered to be uncorrelated if

$$x^{(4)} > \theta,$$

and otherwise correlated. In this work, the value for the threshold θ is determined by minimizing the probability of misclassification in order to test the possible accuracy of the system. The same effect of this minimization could be achieved by keeping θ fixed and optimizing the slope α of the transfer function, equation 2.4, since θ depends on the slope α. These parameters, the slope α or the discrimination threshold θ, in principle may be optimized autonomously using information theoretical objective functions (Triesch, 2005; Marković & Gros, 2010). For simplicity we perform the optimization directly. We will show in section 3.4 that the optimal values for α and θ depend essentially on only the size N of the input. Minor adjustments of the parameters might nevertheless be desirable to maintain optimal accuracy. In any case, these readjustments can be done in a biological system by intrinsic plasticity (see Stemmler & Koch, 1999; Mozzachiodi & Byrne, 2010; Marković & Gros, 2010, 2011).

Although we did not implement the max function present in equation 2.7 in a neuromorphic form, a small neuronal circuit implementing that equation could, for instance, be realized as a winner-takes-all network (Coultrip, Granger, & Lynch, 1992; Kaski & Kohonen, 1994; Carpenter & Grossberg, 1987). Alternatively, a filtering rule different from the max function could be used for the last layer—for instance, the addition or averaging of all the inputs. We present as supporting information some results showing the behavior of the output when using averaging and sum as alternative filtering rules for the output layer. Our best results were, however, found by implementing the last layer as a max function. In this work, we discuss the behavior of the system using the max function as the last layer.

Defining a threshold θ is one way of using this system for binary classification, which we use for reporting the possible accuracy of the system. However, it is not a defining part of the model. We expect the system to be more useful for obtaining a continuous variable measuring the grade of

correlation of the inputs. As we discuss in section 4, this property can be used to apply fuzzy logic in a biological system.

3 Performance in Terms of Binary Classification

3.1 Performance Measures. In order to calculate the performance, in terms of binary classification, of the neural comparator, we need to track the number of correct and incorrect classifications. We use three measures for classification errors:

1. FP (false positives): The fraction of cases for which $x^{(4)} < \theta$ (input is classified as correlated) occurs for uncorrelated pairs of input vectors $\mathbf{y}$ and $\mathbf{z}$:

$$FP = \frac{\#\,\text{erroneously assigned as positive}}{\#\text{assigned as positive}}. \tag{3.1}$$

FN(false negatives): The fraction of cases with output activity $x^{(4)} \geq \theta$ (input classified as uncorrelated) occurring for correlated pairs of input vectors, $\mathbf{z} = \mathbf{f}(\mathbf{y})$:

$$FN = \frac{\#\,\text{erroneously assigned as negative}}{\#\text{assigned as negative}}. \tag{3.2}$$

E(overall error): The total fraction of errors E is the fraction of overall wrong classifications:

$$E = \frac{\#\,\text{erroneously assigned}}{\#\text{all assignments}}. \tag{3.3}$$

All three performance measures, E, FP, and FN, need to be kept low. This is achieved for a classification threshold θ, which minimizes $(FP + FN)$. This condition keeps all three error measures (FP, FN, and E) close to their minimum while giving FN and FP equal importance at the same time.

3.2 Mutual Information. Since the percentage of erroneous classifications, despite being an intuitive measure, is dependent on the relative number of correlated and uncorrelated inputs presented to the system, we also evaluate the mutual information (MI) (see Gros, 2008) as a measure of the information gained by the classification made by the comparator:

$$\begin{aligned}\mathrm{MI}(X,Y) &= H[X] - H[X|Y] \\ &= -\sum_{x\in X} p(x)\log\left(p(x)\right) + \sum_{y\in Y} p(y)\sum_{x\in X} p(x|y)\log\left(p(x|y)\right),\end{aligned} \tag{3.4}$$

where X in this case represents whether the inputs are equal and Y is whether the comparator classified the input as correlated; therefore, both X and Y are vectors of size two ($true/false$ corresponding to semantically related/uncorrelated). Here $\rho(x|y)$ is the conditional probability that the input had been $x = frue/false$ given that the output of the comparator is $y = true/false$ and $H(X)$ the marginal information entropy.

We will refer specifically to the mutual information, equation 3.4 , between the binary input and output of the neural comparator, also known in this context as information gain. The mutual information can also be written as $\sum_{x,y} p(x,y)\log(p(x,y)/(p(x)p(y)))$, where $p(x,y) = p(y|y)p(y)$ is the joint probability. It is symmetric in its arguments X and Y and positive definite. It vanishes for uncorrelated processes X and Y, when $\rho(x|y) = \rho(x)$, for a random output of the comparator. Finally, the mutual information is maximal when the two processes are 100% correlated, that is, when the off-diagonal probability vanishes, $\rho(x|y) = 0$ for $x \neq y$. In this case, the two marginal distributions $\rho(x) = \sum_y p(x,y)$ and $\rho(y) = \sum_x p(x,y)$ coincide, and $MI(X,Y)$ is identical to coinciding marginal entropies, $H(X) = H(Y)$.

We will present the mutual information as the percentage of the maximally achievable mutual information,

$$\begin{aligned} \mathrm{MI\%}(X,Y) &= \frac{H[X] - H[X|Y]}{H[X]} \\ &= 1 - \frac{\sum_{y\in Y} p(y) \sum_{x\in X} p(x|y)\log\left(p(x|y)\right)}{\sum_{x\in X} p(x)\log\left(p(x)\right)}, \end{aligned} \tag{3.5}$$

which has a value between 0 and 1 and is therefore more intuitive to read as a percentage of the maximum theoretically possible. The maximum mutual information achievable by the system depends solely on the probabilities of correlation and decorrelation: p_{eq}.

The statistical correlations between the input X and the output Y can be parameterized using a correlation parameter a via

$$p(x,y) = p(x)\,p(y) + C_a(x,y), \tag{3.6}$$

where $C_a(x,y)$ are the element values of the matrix:

$$C_a = \begin{pmatrix} a & -a \\ -a & a \end{pmatrix} \qquad a \in \left[0, p_{eq}(1-p_{eq})\right]. \tag{3.7}$$

Here p_{eq} is the probability of having correlated pairs of inputs: $p(x = true) = p_{eq}$ and $p(x = false) = (1 - p_{eq})$. Using this parameterization allows us to evaluate the relative mutual information, equation 3.5, generically for a correlated joint probability $p(x, y)$, as illustrated in Figure 2. The

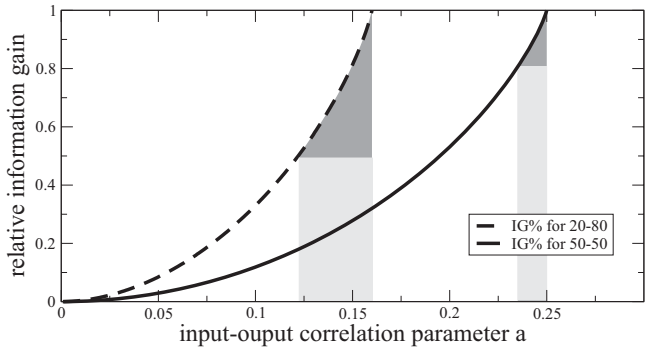

Figure 2: The relative mutual information MI%, equation 3.5, as a function of the amount of correlation a between the input and the output, defined by equation 3.6. The comparator achieves an MI% of about 50% and 80%, respectively, for fractions of $p_{eq} = 0.2$ (dashed line) and $p_{eq} = 0.8$ (solid line) of correlated input pairs, corresponding to $0.12/0.16 = 0.75$ and $0.23/0.25 = 0.92$ of all input-output correlations a. Hence, only 25% and 8%, respectively (the darker shaded areas), of all correlations are not recovered by the comparator.

parameterization, equation 3.6, hence provides an absolute yardstick for the performance of the neural comparator.

3.3 Simulation Results. We performed a series of simulations using the network parameters listed in Table 1, and for two encoding rules, equation 2.2, direct and linear encoding. The lengths N of the input vectors $\mathbf{y}$ and $\mathbf{z}$ are taken to be equal if not stated otherwise.

3.3.1 Low Probability of Equals. Since our initial motivation for the design of this system is the comparison of two input streams that are presumably most of the time different, we have studied the behavior of the system when there is a lower probability of an event where both streams are equal than otherwise. We used $p_{eq} = 0.2$ in equation 2.2. In 20% of the cases, the relation $\mathbf{z} = \mathbf{f}(\mathbf{y})$ holds, and in the remaining 80%, the two inputs $\mathbf{y}$ and $\mathbf{z}$ are completely uncorrelated (randomly drawn). Each calculation consists of $t_{\max} = 10^7$ steps, from which the last 10% of the simulation is used for evaluating performance. During the last 10% of the simulation, the system keeps learning; there is no separation between training and production stages. The purpose of taking only the last portion is to ignore the initial phase of the learning process, since at that stage, the output does not provide a good representation of the system's accuracy.

In Table 2, we present the mean values for the different measures of error, equations 3.1 to 3.3, observed for 100 independent simulations of the system. For each individual simulation, the interlayer connections are

Table 2: Errors Obtained from Averaging 100 Runs of the Comparator, Using Direct and Linear Encoding after 10^7 Steps, for Different Input Sizes N.

	Direct				Linear			
N	$\langle E \rangle$	$\langle FP \rangle$	$\langle FN \rangle$	MI%	$\langle E \rangle$	$\langle FP \rangle$	$\langle FN \rangle$	MI%
5	10.2%	5.8%	10.5%	13.2%	14.8%	8.3%	14.8%	23.8%
15	6.0%	1.2%	6.8%	44.4%	5.2%	2.8%	5.9%	41.7%
30	5.3%	1.0%	6.0%	49.5%	4.8%	1.3%	5.4%	50.8%
60	6.6%	0.6%	7.4%	45.3%	4.3%	1.0%	5.0%	54.7%
100	6.5%	0.5%	7.5%	45.5%	5.3%	0.6%	6.1%	51.5%
200	7.8%	0.9%	8.8%	37.3%	6.2%	0.9%	7.2%	44.2%
400	7.1%	0.8%	7.5%	43.5%	7.2%	0.7%	8.1%	41.5%
600	–	–	–	–	6.7%	0.5%	7.0%	50.8%

Notes: The connection probabilities used are $p_{conn}^{(1)} = 0.3$, $p_{conn}^{(2)} = 0.8$. For $N > 5$ the standard deviations amounts to 0.1% to 0.8% (decreasing with N) for the errors E, FP, and FN and 1% for MI%. For the $N = 5$ case, the standard deviation of the errors is 5% to 14% (again, decreasing with N) and for MI%, it amounts to 15%.

randomly drawn with probabilities $p_{conn}^{(k)}$, with parameters as shown in Table 1. The errors for each run are calculated using a threshold θ that minimizes the sum of errors $\langle FP \rangle + \langle FN \rangle$. Each input in the first layer has a uniform distribution of values between -1 and 1. The accuracy of the comparator is generally above 90% in terms of binary classification errors. There is, importantly, no appreciable difference in the accuracy when using direct encoding or linear encoding with random matrices.

Note that a relative mutual information of MI%$\approx$50% is substantial (Guo, Shamai, & Verdú, 2005). A relative mutual information of 50% means that the correlation between the input and the output of the neural comparator encompasses 75% of the maximally achievable correlations, as illustrated in Figure 2.

We found that the performance of the comparator depends substantially on the degree of similarity of the two input signals $\mathbf{y}$ and $\mathbf{z}$ for the case when the two inputs are uncorrelated. For a quantitative evaluation of this dependency we define the Euclidean distance,

$$d = \|\mathbf{y} - \mathbf{f}^{-1}(\mathbf{z})\|, \tag{3.8}$$

where $||\cdot||$ denotes the Euclidean norm of a vector. For small input sizes N, a substantial fraction of the input vectors is relatively similar with small Euclidean distance d, resulting in a small output $x^{(4)}$. This can prevent the comparator from learning the classification effectively; thus, the best accuracy is obtained for input vectors greater than $N = 10$ (compare Table 2).

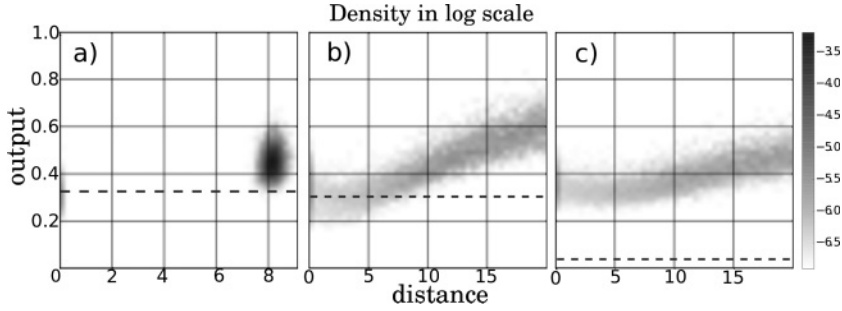

Figure 3: The probability of an output $x^{(4)}$, equation 2.7, as a function of Euclidean distance d (equation 3.8) of the input pairs $\mathbf{y}$ and $\mathbf{z}$ (see Figure 1), as a density plot, using $N = 400$, $\alpha = 1$ and direct encoding. The last 10% of simulations with 10^7 input pairs have been taken for the performance testing. (a) Unconstrained input protocol equation 3.9, for both training and for testing. (b) Unconstrained input protocol, equation 3.9, for training and constrained, equation 3.10, for testing. (c) Constrained input protocol, equation 3.10, for both training and testing.

This phenomenon can be investigated systematically by considering two distinct distributions for the Euclidean distance d. Within our input protocol, equation 2.2, the pairs $\mathbf{y}$ and $\mathbf{z}$ are statistically independent with probability $(1 - p_{eq})$. We have considered two ways of generating statistically unrelated input pairs,

- *Unconstrained:* The components y_i and z_i are selected randomly from the interval $[-1, 1]$. (3.9)
- *Constrained:* The components y_i and z_i are selected randomly such that the distance d has a flat (uniform) distribution in $[0, 1]$. (3.10)

For the case of the unconstrained input protocol, the distribution of distances d is sharply peaked for large input size N (see Figure 3). The impact of the distribution of Euclidean distances between the random input vectors $\mathbf{y}$ and $\mathbf{z}$ is presented in Figure 3, where we show the result of three separated simulations:

1. Using the unconstrained input protocol, equation 3.9, for both training and for testing. The corresponding performance errors are $FP = 1.0\%$, $FN = 10.7\%$, and $E = 9.7\%$, for a threshold $\theta = 0.34$.
2. Using the unconstrained input protocol, equation 3.9, for training and the constrained protocol, equation 3.10, for testing. The performance errors are $FP = 13.7\%$, $FN = 14.6\%$, and $E = 14.2\%$ for a threshold $\theta = 0.28$.

3. Using the constrained input protocol, equation 3.10, for both training and testing. The corresponding errors are $FP = 79.9\%$, $FN = 0.0\%$, and $E = 79.9\%$ for a threshold $\theta = 0.02$.

The accuracy of the comparator is very good for simulation 1. In this case, values close to $d \sim 0$ are almost nonexistent for random input pairs $\mathbf{y}$ and $\mathbf{z}$; random and related input pairs are clustered in distinct parts of the phase space.

The performance of the comparator drops, on the other side, with the increasing number of similar random input pairs. For case 2, the distribution of distances d is uniform, and the comparator has essentially no comparison capabilities. Since 20% of the input is correlated, the minimal error E in this case is obtained if the system assumes all input to be uncorrelated (setting an extremely small threshold). That situation results in 80% FP and 20% FN. Notice that in this case, the mutual information of the system is null. Finally in the mixed case, simulation 2, the comparator is trained with an unconstrained distribution for the distances d and tested using a constrained distribution. In this case, the comparator still acquires a reasonable accuracy of $E = 14\%$.

3.3.2 Equilibrated Input, $p_{eq} = 0.5$. In this section we expand the results for equilibrated input data sets, $p_{eq} = 0.5$ in equation 2.2. The procedure remains as described in the previous section. Again, each calculation consists of $t_{\max} = 10^7$ steps, from which the last 10% of the simulation is used for performance evaluation. This result is consistent with the intuitive notion that it is substantially harder to learn when $\mathbf{y}$ and $\mathbf{z}$ are related, when most of the input stream is just random noise and semantically correlated input pairs seldom occur. For applications, one may consider a training phase with a high-frequency p_{eq} of semantically correlated input pairs.

The use of a balanced input set does not change the general behavior but results in a substantial increase in performance (see Table 3). The accuracy of the system in terms of the percentage of correct classifications (above 95% accuracy except on very small input size) and relative mutual information MI% (∼80% of the maximum information gain) is very high. A relative mutual information of MI% $\approx$ 80% means that the system recovers over 92% of the maximally achievable correlations between the input and the output, as shown in Figure 2.

3.4 Effect of Noisy Encoding. In the previous sections, we provided results showing that the proposed comparator can achieve good accuracy despite the fact that a large part of the input is noise. In addition, the comparator is robust against a level of noise in the encoding of the inputs. Random noise in the encoding would correspond to the neural populations having rapid random reconfigurations or random changes in the individual neuron behavior above a certain level.

Table 3: Errors Obtained from Averaging 100 Runs of the Comparator, Using Linear Encoding, with $p_{eq} = 0.5$, for Different Input Sizes N.

N	$\langle E \rangle$	$\langle FP \rangle$	$\langle FN \rangle$	MI%
5	9±6%	8±7%	10±5%	58±14%
15	3.9±0.4%	0.4±0.1%	6.9±0.6%	78±1%
30	3.4±0.2%	0.3±0.1%	6.3±0.3%	81±1%
60	3.3±0.1%	0.2±0.1%	6.1±0.2%	81±1%
100	3.4±0.1%	0.2±0.1%	6.2±0.1%	82±1%
200	4.7±0.1%	0.5±0.3%	8.2±0.5%	75±1%
400	6.2±0.1%	0.4±0.1%	10.9±0.1%	70±1%
600	7.5±0.1%	1.1±0.1%	12.4±0.1%	66±1%

Note: The connection probabilities used are $p^{(1)}_{conn} = 0.3$, $p^{(2)}_{conn} = 0.8$.

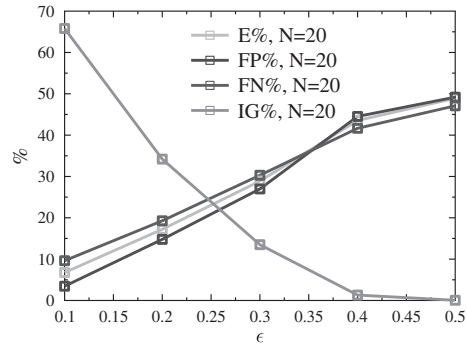

Figure 4: Errors E, FP, FN, and percentage of information gain MI% for different levels of noise ϵ in the encoding. $\epsilon = 0.5$ corresponds to a magnitude equal to the average activity of an input.

As shown in Figure 4, the system has an accuracy decay if the encoding is affected by random noise of the same magnitude as the average input activity (0.5). For this calculation, we define the random noise in the encoding as a random number between 0 and ϵ added to each element of one of the compared inputs, $y_i \to y_i + r_i$, where $r_i \in [0, \epsilon]$. The values r_i are changed in every step of the calculation.

The addition of random noise in the encoding is seen by the system as a slightly different input. Since the system is designed to classify inputs into either different or equal, a large level of noise drives the system into classifying the input as different. However, if the input is only slightly changed, the correlation is still found by the comparator and the output remains under the threshold for classification.

3.5 Impact of the Frequency of Correlated Input and Input Size. In Figure 5, the dependence of the optimal threshold θ and the errors E, FP, FN,

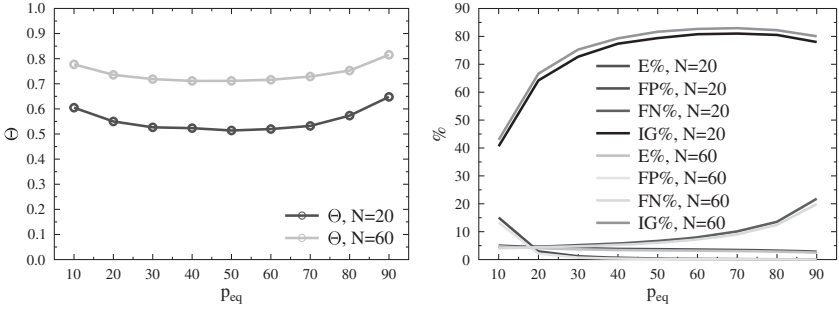

Figure 5: (Left) The errors E, FP, FN, and relative mutual information MI% for different frequencies of correlated input p_{eq} (percentages). The results are presented for input sizes $N = 20$ and 60. (Right) The optimal values of the threshold θ for different frequencies of correlated input p_{eq}. The results are again presented for input sizes $N = 20$ and 60.

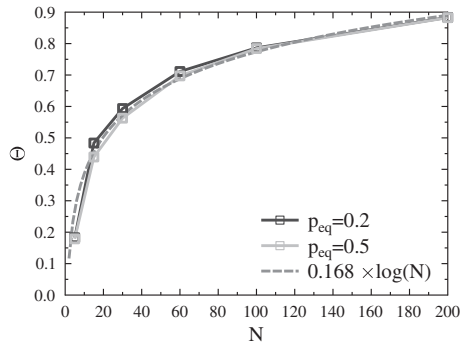

Figure 6: Optimal threshold θ versus input size N, for $\alpha = 2.7$. The behavior is markedly logarithmic.

MI% with the probability p_{eq} is shown. At constant input size, the threshold shows only a weak dependence with the probability p_{eq}. The threshold changes at its maximum for the probability of any case on the order of 10% or less. The threshold varies less than 0.1 from $p_{eq} = 0.1$ to $p_{eq} = 0.9$. This indicates that the system would still be effective if the probabilities of the events change significantly, even without readjusting the parameters α or θ or with a small readjustment if the change is extreme.

In Figure 6, the dependence of the optimal threshold θ with the input size N is presented. The threshold has a marked logarithmic dependence with respect to the system size. In effect, the threshold θ, the gain α, and the system size N are all strongly coupled, such that given an input size, the rest of the parameters are essentially fixed.

Table 4: Average Errors $\langle E \rangle$, $\langle FP \rangle$, $\langle FN \rangle$ and Relative Mutual Information MI% Obtained from 100 Runs of the Comparator for Comparing One Input of Size $N = 20$ ($N = 60$) and Another of Size $N + \Delta$, Using $p_{eq} = 50$.

	$N = 20$				$N = 60$			
Δ	$\langle E \rangle$	$\langle FP \rangle$	$\langle FN \rangle$	MI%	$\langle E \rangle$	$\langle FP \rangle$	$\langle FN \rangle$	MI%
0	3.4±0.3	0.3±0.1	6.3±0.4	80±1	3.3±0.1	0.2±0.1	6.1±0.2	81±1
5	3.1±0.3	0.3±0.1	5.7±0.5	82±1	3.4±0.3	0.3±0.1	6.3±0.2	80±1
10	2.7±0.2	0.3±0.1	4.9±0.4	84±1	2.9±0.1	0.2±0.1	5.4±0.2	83±1
20	2.2±0.2	0.3±0.1	4.0±0.4	86±1	2.6±0.1	0.2±0.1	4.8±0.1	85±1
40	1.6±0.1	0.3±0.1	2.9±0.3	89±1	2.2±0.1	0.2±0.1	4.0±0.1	87±1

3.6 Comparison of Inputs with Different Sizes. The comparator successfully compares input of different sizes. In Table 4 we show the average accuracy over 100 runs of a comparator where one of the vectors to be compared has a size N and the other has a larger size $N + \Delta$. The number of extra inputs Δ is maintained constant during the simulation. In each step, the values of the two vectors are assigned as described previously as linear encoding in section 2.1. The linear encoding is done in this case with a matrix $\hat{A}$ that has dimensions $(N + \Delta) \times N$; thus, the information gets encoded in a vector of higher dimension.

The accuracy of the comparator does not decrease; rather surprisingly, it slightly increases. There is no loss in accuracy because the uncorrelated inputs are not minimized to a value close to zero due to the anti-Hebbian adjustment of the synaptic weights, as happens only with the correlated input. We attribute the small increase in accuracy to the increase of neurons involved in the system.

3.7 Influence of Connection Density. A key ingredient in this model is the suppression of a fraction of interlayer connections with probability $1 - p_{conn}$, which is necessary to give higher-layer neurons the possibility of encoding varying features of correlated input pairs. For a systematic study, we ran simulations using a range of distinct probabilities of interconnecting the layers.

In Figure 7, we show the unconstrained performance measures for $N = 5$ when changing (left) the connection $p^{(1)}_{conn}$ from the input layer to the first layer (compare Figure 1, with constant $p^{(2)}_{conn} = 0.75$) and (right) when varying the connection $p^{(2)}_{conn}$ from the second to the third layer. In the latter case, we kept $p^{(1)}_{conn} = 0.3$ fixed.

The data presented in Figure 7 show that the neural comparator loses functionality when the network becomes fully interconnected. The optimal interconnection density varies from layer to layer and is best for 10% efferent first-layer connections and 60% links efferent from the second layer.

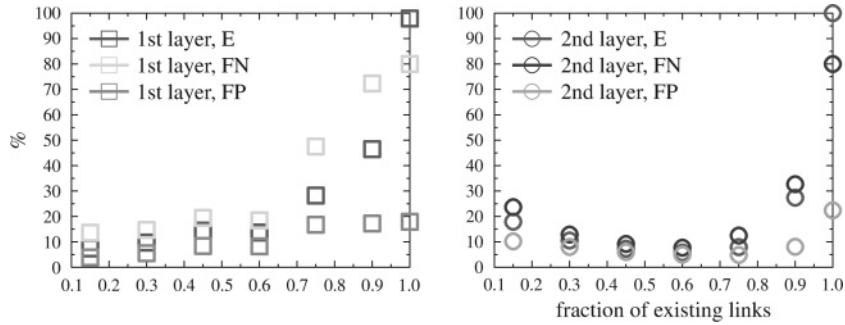

Figure 7: Effect of the probability of connection in the overall errors. (Left) The errors of the comparator with varying probability $p_{conn}^{(1)}$ of connection efferent from the first layer, with $p_{conn}^{(2)} = 0.75$. (Right) The probability $p_{conn}^{(2)}$ of efferent connection from the second layer is varied, keeping $p_{conn}^{(1)} = 0.3$ constant.

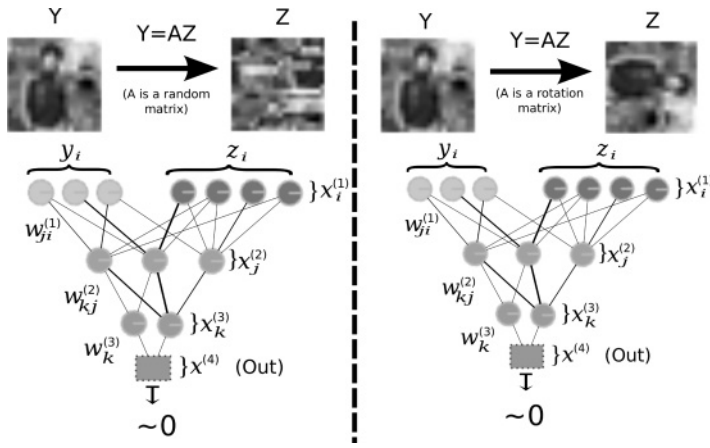

Figure 8: Two possible encryptions made by the comparator with probability p_{eq}. In both cases, after being exposed to many images, the comparator would ideally result in an output $x^4 \sim 0$. Notice that the matrix A on each side of the figure is constant during the whole simulation.

3.8 Images Comparison. We tested the comparator efficiency in comparing a set of black and white pictures of small size (20×20 pixels, i.e., $N = 400$) using linear encoding via a random matrix as in previous sections (see Figure 8). The set of pictures is very small (200 pictures) in comparison to the input data used to train the comparator ($t = 10^7$ inputs). The results are in Table 5. The limited input set has the negative effect that the comparator is not able to learn comparison only from this set. This suggests that

Table 5: Average Errors $\langle E \rangle$, $\langle FP \rangle$, $\langle FN \rangle$ and Relative Mutual Information MI% Obtained from 100 Runs of the Comparator for Comparing Black and White Pictures of Size 20×20 Pixels.

	$\langle E \rangle$	$\langle FP \rangle$	$\langle FN \rangle$	MI%
Only images $p_{eq} = 0.2$	20.6±0.3	51.9±0.6	14.4±0.2	8±1
Trained with random input $p_{eq} = 0.2$	14.5±0.2	39.3±0.3	6.2±0.1	32±1
Trained with random input $p_{eq} = 0.5$	10.8±0.1	10.7±0.2	10.9±0.1	51±1

in order for the comparator to develop its functionality, it must sample a sizable part of the possible input patterns.

As we have noted, the correlated inputs are minimized by the anti-Hebbian rule, while the uncorrelated input cannot be minimized to the same level, since those cases result in the terms $x_i^{(k)} x_j^{(k+1)}$ in equation 2.5 being essentially random. This assumption is, however, not fulfilled if the values of these terms are not well distributed (unless their values are by chance always small), which is the case if the sampling is not large enough.

As a second test, we initially trained the comparator using random data (still using $p_{eq} = 0.5$) in order to start with a functional distribution of the synaptic weights and then switched to the picture set for the last 10% of the calculation, with the comparator still learning during this stage. In this case, the comparator achieved its function (see Table 5). However, the accuracy did not fully reach that of the system when comparing randomly generated data.

We expect the accuracy of the random comparator to be at the level of the one generated by random input if the input stream explores a sizable part of the possible input. For instance, ideally the image input would be a video of the visual input in a mobile agent while exploring the environment, such that a large number of patterns are processed by the comparator. This is, however, out of the scope of this work, although follow-up work is expected.

4 Interpretation Within the Scope of Fuzzy Logic

The dependency of the output of the comparator seen in Figures 3b, 3c, and 9 can be interpreted in terms of fuzzy logic (Keller, Yager, & Tahani, 1992), offering alternative application scenarios for the neural comparator.

The error measures evaluated in Table 2, like the incidence of false positives (FP), are based on Boolean logic, and the classification is either correct or incorrect (i.e., binary). For real-world applications, the input pairs $(\mathbf{y}, \mathbf{z})$ may be similar but not equal, and the dependence of the output as a function of input similarity is an important relation characterizing the functionality of neural comparators.

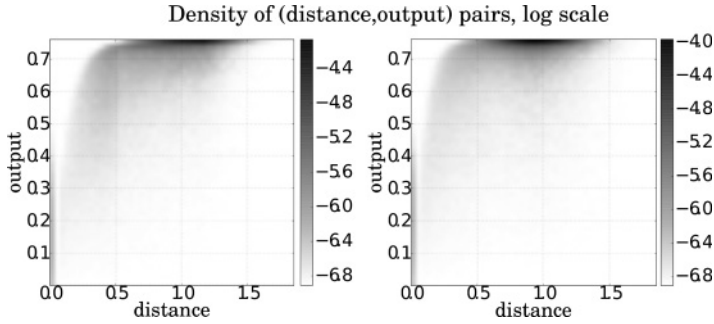

Figure 9: Density of output-distance pairs from two individual runs of the comparator, comparing inputs of dimension $N = 5$ under the direct encoding (left) and linear encoding (right), using $\alpha = 2.7$.

The comparator essentially provides a continuous variable classifying how much the input case corresponds to the case of equal input (i.e., a truth degree). Thus, the comparator can be interpreted as a fuzzy logic gate for the operator "equals" (=), since it provides a truth degree for the outcome of the discrete version of the same operator.

In Figure 9, we present, on a logarithmic scale, the density of results for the observed output $x^{(4)}$, as a function of the distance d between the respective inputs, for a single run of the comparator. Eighty percent of the input vectors were randomly drawn and later readjusted in order to fill the range of distances $d = 0.1$ to 1.5 uniformly, according to the constrained protocol, equation 3.10. In addition, 20% of the input has a distance of $d = 0$ with $\mathbf{z} = \mathbf{y}$, resulting in the high density of simulations at $d = 0$.

The uncertainty of the classification of inputs presented in Figure 9 is reflected in a probability distribution for the comparator output, shown in Figure 10 for the case of direct encoding. The output distribution is narrower for cases where the distance d corresponds to clearly correlated or uncorrelated inputs.

The distributions presented in Figure 9 can be interpreted as fuzzy probability distributions for any given distance d (vertical slices), as shown in Figure 10. The probability for the input pairs $\mathbf{y}$ and $\mathbf{z}$ to be classified as different decreases with decreasing distance d between them. This shows that inputs with smaller distances in general have increasingly weaker outputs. Thus, assuming that the Euclidean distance d is a good estimator of how similar the input is, the output of the comparator provides an arguably reliable continuous variable estimating a similarity degree for the inputs; that is, the truth degree of the operator "equals" applied to the inputs.

5 Discussion

The results presented here demonstrate that the proposed neural comparator has the capability of discerning similar input vectors from dissimilar

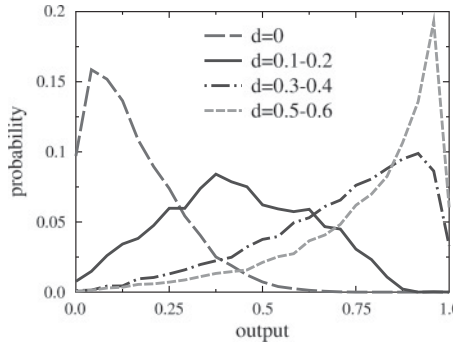

Figure 10: Output distribution for different Euclidean distances d (equation 3.8) between the inputs, as a function of output, corresponding to a vertical slice of the $N = 5$ direct encoding data presented in Figure 9.

ones even under noisy conditions. Using 80% noise, with four out of five inputs being randomly drawn, the unsupervised comparator architecture achieves a Boolean discrimination accuracy above 90%. The comparator circuit can also achieve the same accuracy when the inputs to be compared are encoded differently. If the encodings of both inputs are related by a linear relation, the accuracy of the comparison does not worsen with respect to the direct encoding case.

A key factor for the accuracy of the method is the inclusion of a slightly different path for the layer-to-layer information, provided by random suppressions of interlayer connections. However, the suppression has the potential side effect of rendering some of the correlations difficult to be learned. For this reason, a compromise needs to be found between the number of connections that must be kept so that the network will be functional and the number of connections that need to be removed to generate sufficiently different outputs in the third layer.

We find it remarkable that from a very simple model of interacting neurons under the rule of minimization of its output, the fairly complex task of identifying the similarity between unrelated inputs can emerge through self-organization without the need for any predefined or externally given information. Complexity arising from simple interactions is a characteristic of natural systems, and we believe the capacity of many living beings to perform comparison operations could potentially be based on some of the aspects included in our model.

6 Conclusion

We have presented a neuronal circuit based on a feedforward artificial neural network, which is able to discriminate whether two inputs are equal or different with high accuracy even under noisy input conditions.

Our model is an example of how algorithmic functionalities can emerge from the interaction of individual neurons under strictly local rules—in our case, the minimization of the output—without hard-wired encoding of the algorithm, external supervision, and any a priori information about the objects to be compared. Since our model is capable of comparing information in different encodings, it would be a suitable model of how seemingly unrelated information coming from different areas of a brain can be integrated and compared.

We view the architecture proposed here as a first step toward an in-depth study of the important question: Which are possible neural circuits for the unsupervised comparison of unknown objects? Our results show that anti-Hebbian adaption rules, which are optimal for synaptic information transmission (Bell & Sejnowski, 1995), allow comparing two novel objects (objects never encountered before during training) with respect to their similarity. The model is capable not only of providing binary answers—whether the two objects in the sensory stream are (are not) identical—but also giving a quantitative estimate of the degree of similarity, which may be interpreted in the context of fuzzy logic. We believe this quantitative estimate of similarity is a central aspect of any neural comparator because it may be used as a learning or reenforcement signal.

Acknowledgments

We acknowledge the support of the German Science Foundation.

References

Atiya, A. F. (1990). An unsupervised learning technique for artificial neural networks. *Neural Networks*, *3*, 707–711.

Bell, A. J., & Sejnowski, T. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computing*, *7*, 1129–1159.

Billing, E. A. (2010). Cognitive perspectives on robot behavior. In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence: Special Session on Computing Languages with Multi-Agent Systems and Bio-Inspired Devices* (pp. 373–382). Setubal, Portugal: INSTICC.

Bovet, S., & Pfeifer, R. (2005a). Emergence of coherent behaviors from homogeneous sensorimotor coupling. In *Proceedings of the 12th International Conference on Advanced Robotics (ICAR)*. Piscataway, NJ: IEEE.

Bovet, S., & Pfeifer, R. (2005b). Emergence of delayed reward learning from sensorimotor coordination. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Piscataway, NJ: IEEE.

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, *37*, 54–115.

Coultrip, R., Granger, R., & Lynch, G. (1992). A cortical model of winner-take-all competition via lateral inhibition. *Neural Networks*, *5*, 47–54.

Furao, S., Ogura, T., & Hasegawa, O. (2007). An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks*, *20*, 893–903.

Gros, C. (2008). *Complex and adaptive dynamical systems: A primer*. New York: Springer.

Guo, D., Shamai, S., & Verdú, S. (2005). Mutual information and minimum mean-square error in gaussian channels. *IEEE Transactions on Information Theory*, *51*, 1261–1282.

Japkowicz, N. (2001). Supervised versus unsupervised binary-learning by feedforward neural networks. *Machine Learning*, *42*, 97–122.

Japkowicz, N., Myers, C., & Gluck, M. (1995). A novelty detection approach to classification. In *Proceedings of the Fourteenth Joint Conference on Artificial Intelligence* (pp. 518–523). Cambridge, MA: AAAI Press.

Kaski, S., & Kohonen, T. (1994). Winner-take-all networks for physiological models of competitive learning. *Neural Networks*, *7*, 973–984.

Keller, J. M., Yager, R. R., & Tahani, W. (1992). Neural network implementation of fuzzy logic. *Fuzzy Sets and Systems*, *45*, 1–12.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*, 1464–1480.

Likhovidov, V. (1997). Variational approach to unsupervised learning algorithms of neural networks. *Neural Networks*, *10*, 273–289.

Marković, D., & Gros, C. (2010). Self-organized chaos through polyhomeostatic optimization. *Physical Review Letters*, *105*, 068702.

Marković, D., & Gros, C. (2011). Intrinsic adaptation in autonomous recurrent neural networks. *Neural Computation*, *24*, 523–540.

Mozzachiodi, R., & Byrne, J. H. (2010). More than synaptic plasticity: Role of nonsynaptic plasticity in learning and memory. *Trends in Neurosciences*, *33*, 17–26.

O'Reilly, R. C., & Munakata, Y. (2000). *Computational explorations in cognitive neuroscience*. Cambridge, MA: MIT Press.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, *2*, 459–473.

Stemmler, M., & Koch, C. (1999). How voltage-dependent conductances can adapt to maximize the information encoded by neuronal firing rate. *Nature Neuroscience*, *2*, 521–527.

Triesch, J. (2005). A gradient rule for the plasticity of a neuron's intrinsic excitability. In *Proceedings of the International Conference on Artificial Neural Networks* (pp. 65–70). New York: Springer.

Tong, H., Liu, T., & Tong, Q. (2008). Unsupervised learning neural network with convex constraint: Structure and algorithm. *Neurocomputing*, *71*, 620–625.

Received May 25, 2012; accepted October 26, 2012.