JOURNAL OF ADVANCED ACADEMIC RESEARCH (JAAR)

Three-Layer Perceptron Position and Size Encoding in Shapes Classification Tasks

Carlo Ciulla[1]
[1]Faculty of Economics, Technology and Innovation, Department of Computer Science,
Western Balkans University, Tirana, Albania
Corresponding Author Carlo Ciulla, carlo.ciulla@wbu.edu.al; cxc2728@njit.edu
ORCID: https://orcid.org/0000-0002-5563-6036
Received 18 March 2026, Accepted 16 May 2026, Published 30 June 2026

ABSTRACT

This research reports on the ability of the three-layer perceptron trained with the backpropagation algorithm to solve classification tasks while encoding the spatial location, the size and the shape of two-dimensional geometrical figures in the Cartesian plane. The data set consists of images. The pixel's brightness inside the shapes is kept constant and the remnant of the image is made of zero-valued pixels' brightness. Results show that the neural network training based on backpropagation converges and the neural network classifier confirms the successfulness of the learning process with 100% correct classification rate. The hidden layer can successfully encode all the variables: position, shape, and dimension; and so, it does the output layer. Testing the three-layer perceptron with patterns not used for the learning process yields a correct classification rate between 95% and 100%. The task of encoding the shape of geometrical figures regardless of their size and their position on the Cartesian digital grid is regarded here as the ability of the three-layer perceptron to form complex decision regions. This ability is typical of this type of neural network because, with a sufficiently large hidden layer, it is a universal function approximator. The novelty of this paper is called Hidden Activation Analysis (HAA), and it consists of explicitly revealing that after the learning process the hidden layer of the three-layer perceptron can solve the pattern recognition task.

KEYWORDS

Three-layer perceptron, backpropagation, delta rule, shape, position, size.

INTRODUCTION

Overview

To clarify the essence of position (location) encoding in artificial intelligence, and more specifically in machine learning applications like neural networks, is due to refer to the conceptualization reported recently in Mai et al. (2022) that states that position encoding is the ability to encode a single point location in an embedding space. The location encoder is defined as the function that maps any coordinate in space (two or three dimensional, or more generally 'L') to a vector representation of dimension 'D', given that L << D (Mai et al., 2022). The geometrical consequence of this definition is that there exists a difference between the departing space and the embedding space. The formalism introduced by Mai et al. (2022) states that the location encoder is the neural network convolution of the position encoder (PE). The formulation of the PE is wide and follows various schemes (Li et al., 2021; Jin et al., 2022; Mai et al., 2022; Rußwurm et al., 2024).

Generalizing on this concept, we can substantiate that the convolutional power of neural networks used by deep learning structures can map the geometry of a point (its location in the original defining space) into a new geometry. This new geometry consists of hyperspace that results from the convolution of the coordinates of the point (two or more dimensions) with the weights connecting the neurons of one layer (e.g. the input layer or the hidden layer) with a neuron of the subsequent layer (e.g. the hidden or the output layer). The convolution brings the departing point (geometric location) into a new point which coordinates are the initial coordinate convolved with the weighting connections of the network's afferent layer. Using this concept, recent literature has evolved as follows regarding the ability of the neural networks to encode object's position (its location in the defining space).

Encoding three-dimensional geometric structures by means of an auto-encoder was reported as a methodology to archive 3D registration (Elbaz et al., 2017). A position encoding network consisting of a feed-forward convolutional encoder and a position encoder module, was designed to extract features and to predict the position information (Islam et al., 2020). The same research confirms considerable ability of convolutional neural networks (CNNs) to encode the position information (Islam et al., 2020). Semantic vectors of fixed length represent the spatial information of multiple objects. Such vectors are calculated by means of the convolution power of feed-forward neural networks. This representation shows that the vector representation is well suited for simple network architectures with a single hidden layer, however not trained with gradient-descend backpropagation algorithm (Mirus et al., 2019). A CNN module that models the spatial information of rectified patches was proposed to reinforce on the concept that an image (or a patch) as input to the CNN determines a tensor that can be interpreted as a detector response of the receptive field (Hinton et al., 2006; Hinton and Salakhutdinov, 2006). Hence, when the images are not aligned, the tensor is transformed into a descriptor that determines encoding of the spatial position and discards the geometrical information (Mukundan et al., 2019).

Deep neural networks have been also employed to learn the spatial characteristics (while encoding the position) and the feature selection (while encoding the type) (Wang et al., 2020). These two predictors called 'where' (position) and 'what' (type) were simultaneously encoded by a deep neural network. These encoding yields feature extraction, non-linear feature enhancement, and voxel-wise linear representation (Wang et al., 2020). Displacement between neighboring frames of pedestrian motion was predicted using the multi-layer perceptron. This prediction was able to map the pedestrian location and to predict the trajectory (Xu et al., 2018). Four types of CNNs were used to count trees from high resolution satellite images, showing that the most accurate prediction is possible using the encoder-decoder network (Yao et al., 2021). Graphical representation of spatial structures is obtainable by graph neural networks (GNNs) (Klemmer et al., 2023). These types of networks account for spatial dynamics. Moreover, GNNs were augmented by a positional encoder (PE) used to determine embedding of position coordinates, and this embedding provides the training data for the GNN. Hence, the new framework for modeling spatial data is the PN-GNN (Klemmer et al., 2023).

Motivation and Novelty

The literature on the three-layer perceptron is substantially restricted because current and past research has been giving mayor attention to position spatial encoding using convolutional neural networks (CNNs).

Because of the usage of deep learning techniques for image classification, literature relegates the three-layer perceptron to marginal importance. However, it has been shown long ago in the literature that the three-layer perceptron can form complex decision regions (Huang & Lippmann, 1987). A three-layer perceptron with a sufficiently large hidden layer is a universal function approximator (Cybenko, 1989; Hornik et al., 1989). Moreover, CNNs are not complex arbitrarily; their architecture explicitly encodes translation invariance, which three-layer perceptron lack. In this paper, to reveal the property by which, after the learning process, the hidden layer of the three-layer perceptron can solve the pattern recognition task, is 'regarded as' Hidden Activation Analysis (HAA). Results of the HAA are coherent with the ability of the three-layer perceptron to solve classification problems that require encoding of spatial location, size and shape of geometrical objects located inside an image.

In fact, this paper reports that HAA is evidence that, using the three-layer perceptron is possible to encode position, size and geometry of shapes defined in the digital grid. Encoding is possible both in the hidden and the output layer of the network.

The geometrical shapes studied in this research are the rectangle, the square, and the circle. Their size is variable and so the position within the grid. However, despite this variability, the three-layer perceptron can learn to distinguish between shapes.

The expected evidence that the classification task is solved by the hidden layer of the network is used in this paper to bring novelty to literature. The novelty of this work is to explicitly reveal through HAA that after the learning process the hidden layer of the three-layer perceptron can solve the pattern recognition task. The implications of this work are in the domain of deep learning because this evidence revitalizes the role of the three-layer perceptron confirming that it can solve a pattern recognition problem, which is normally delegated by more complex network architectures like CNNs. Overall, this work makes a point in favor of the use of the three-layer perceptron in deep learning.

Theory

Computational systems composed of neurons distributed along cascaded layers are called artificial neural networks (ANNs). Except for the input layer, which is the gate to the computational system, neurons are computational units that receive input and calculate an output. The neurons are connected by internal variables called weighting connections. Changing the weight of the connections changes the behavior of the whole computational system. The goal is to achieve a desired input-output response by making use of an optimal set of weighting connections. The optimality depends on the accuracy of the input-to-desired-output mapping. ANNs use the learning process to achieve such optimality. The learning paradigm used to map

the network input to the desired output, is commonly referred to as supervised learning. To perform supervised learning, one of the most popular algorithms used by ANNs is the backpropagation algorithm. Backpropagation makes use of the delta rule for the calculation of the partial derivatives of the cost function. The cost function is the measure of the deviation between the net current output and the net desired output.

Each of the neurons in the hidden and output layers mathematically implements a hypersurface called hyperplane. As illustrated by Eq. (1), the hyperplane of the neuron is mathematically defined by the linear combinations of inputs and weights afferent to the neuron.

$$I^L(j) = \left(\sum_{i=0}^{n_L - 1} w^L(i,j) \cdot O^{L-1}(j) \right) - \theta^L(j);\ L = 1,2;\ j = 1 \ldots n_0 \qquad (1)$$

In Eq. (1), O^{L-1} is neuron output of the afferent layer (L - 1). θ is the firing threshold, known also as the bias term of the neuron. $w^L(i,j)$ is the weighting connection between neurons. The input pattern to the network is $O^0(j)$ with $j = 1 \ldots n_0$. Neurons are non-computational in the input layer (L = 0).

Each neuron of hidden and output layers ($L = 1, 2$) is featured by a transfer function f, as shown in Eq. (2). This function needs to be differentiable because of the necessity of the backpropagation algorithm to calculate the partial first order derivatives of the cost function with respect to the weighting connections. Indeed, differentiability is ultimately a requirement of the delta rule for the iterative weights' adjustment. The iterations are inherent to the optimization process that attempts to find the global minimum of the cost function for a specific set of values of the weights.

$$f[I^L(j)] = O^L(j) = \frac{1}{1 + e^{-I^L(j)}};\ L = 1,2;\ j = 1 \ldots n_L \qquad (2)$$

The cost function is defined by Eq. (3).

$$E = \frac{1}{2} \cdot \sum_{t=1}^{N} \sum_{j=1}^{S} \left[O_{tj}^L(j) - \Omega_{tj}^L(j) \right]^2\ L = 2; \qquad (3)$$

The learning set is inclusive of N patterns, S is the number of categories to classify, and $\Omega^L(j)$ is the desired class identification value. As articulated by Eq. (1) and Eq. (2), E is function of $w^L(i,j)$. Based on these two equations, the backpropagation algorithm uses Eq. (4) to calculate the weighting connection change at the p^{-th} iteration of the learning process. Eq. (4) is called delta rule, and it implements the steepest descend iterative optimization process.

$$\Delta w_p^L(i,j) = -\eta \cdot \left[\left(\frac{\partial E}{\partial w_p^L(i,j)} \right) \right]\ L = 1,2; \qquad (4)$$

η is the learning coefficient and it is set to be $0 < \eta < 1$. After computing the derivatives, Eq. (4) is rewritten as Eq. (5).

$$\Delta w^L(i,j) = -\eta \cdot \delta^L(j) \cdot O^L(i);\ L = 1,2; \qquad (5)$$

In Eq. (5), $\delta^L(j)$ is calculated by (6), (7). Eq. (6) is used as the correction term of the weighting connections between the neurons of the output layer and the neurons of the hidden layer. Eq. (7) is used as the correction term of the weighting connections between the neurons of the hidden layer and the neurons of the input layer.

$$\delta^L(j) = O^L(j) \cdot [1 - O^L(j)] \cdot [O^L(j) - \Omega^L(j)]; \; L = 2; \tag{6}$$

$$\delta^L(j) = O^L(j) \cdot [1 - O^L(j)] \cdot \sum_{k=1}^{n_L+1} \delta^{L+1}(k) \cdot w^{L+1}(j,k); \; L = 1 \tag{7}$$

Based on Eq. (4), the update $w_p^L(i,j)$ of the weighting connections is calculated as per Eq. (8).

$$w_p^L(i,j) = w_{p-1}^L(i,j) + \Delta w_p^L(i,j); L = 1,2; \tag{8}$$

The term $\Delta w_p^L(i,j)$ is applied to the weighting connections of each computational neurons of the neural network. Moreover, the term $\Delta w_p^L(i,j)$ is applied to the weighting connection after all N patterns of the training set are processed, hence it is inclusive of all partial corrections determined by each pattern of the training set. To fine tune the speed of the steepest descend method, the smoothing factor ε (coefficient in]0, 1[) is multiplied by the term Δw_{p-1}^L and summed to the update of the weighting connection so that Eq. (8) can be rewritten as Eq. (9).

$$w_p^L(i,j) = w_{p-1}^L(i,j) + \Delta w_p^L(i,j) + \varepsilon \cdot \Delta w_{p-1}^L(i,j); \; L = 1,2; \tag{9}$$

$w_{p-1}^L(i,j)$ and Δw_{p-1}^L are calculated at $(p\text{-}1)^{\text{-th}}$ iteration of the learning process. The mathematics reported in this section designs a three-layer artificial neural network which was implemented in ANSI C and C++, and it was used to solve two-class identification problems. Tables 1, and 2 cite the references that report the formulae used in this section of the manuscript, hence the tables create a connection between this manuscript and the literature.

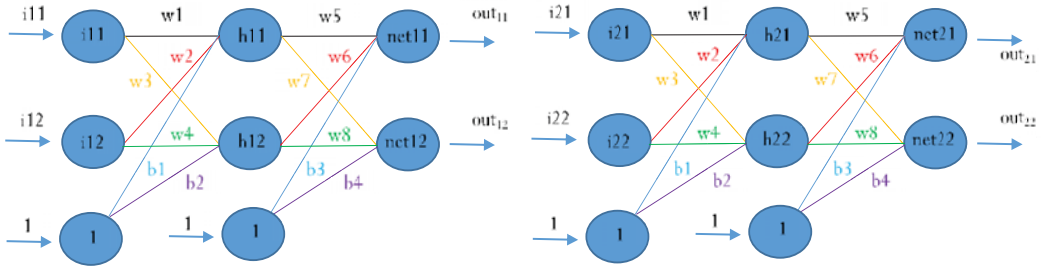

Figure 1. Three-layer artificial neural network with input patterns (i_{11} i_{12}) and (i_{21} i_{22}).

Backpropagation Example of the Three-Layer Artificial Neural Network

This section uses the same approach of the previous section, and details the mathematics involved in the training of a three-layer neural network using the backpropagation algorithm.

Figure 1 shows the illustration of the network with 2 input neurons, and 2 neurones in both hidden and output layers. The threshold logic units are termed as b1 and b2 for the hidden layer, and b3 and b4 for the output layer. The input to the threshold logic units is constant and it is 1.

Equations	Use	Literature
(1), (10), (12), (14), (16), (18), (20), (22), (24)	Input to the Neuron	Rumelhart et al., 1985; Widrow & Lehr 1990; Billings et al., 1991; Frasconi et al., 1993; Sontag, 1993; Wythoff, 1993; Yegnanarayana, 1994; Magoulas et al., 1997; Thimm et al., 1996; Marmarelis & Zhao, 1997; Tivive & Bouzerdoum, 2004; Günther & Fritsch, 2010; Abdel-Hamid et al., 2014; Park & Lek, 2016; Lv et al., 2018; Rana et al., 2018; Zajmi et al., 2018; Zhang & Qu, 2021; Dampfhoffer et al., 2023; Fekri-Ershad & Alsaffar, 2023;
(2), (11), (13), (15), (17), (19), (21), (23), (25)	Sigmoid Function	Rumelhart et al., 1985; Pineda, 1987; Widrow & Lehr, 1990; Billings et al., 1991; Sontag, 1993; De Villiers & Barnard, 1993; Wythoff, 1993; Rojas & Rojas, 1996; Thimm et al., 1996; Marmarelis & Zhao, 1997; Khotanzad & Chung, 1998; Hagan & Demuth, 1999; Park & Lek, 2016; Al-Mahasneh et al., 2017; Lv et al., 2018; Rana et al., 2018; Zajmi et al., 2018; Cong & Zhou, 2023; Zhao et al., 2024;
(3), (26)	Cost Function	Rumelhart et al., 1985; Pineda, 1987; Werbos 1990; Widrow & Lehr, 1990; Billings et al., 1991; Frasconi et al., 1993; Sontag, 1993; Wythoff, 1993; Rojas & Rojas, 1996; Magoulas et al., 1997; Chen & Zhong, 2009; Günther & Fritsch, 2010; Park & Lek, 2016; Bao et al., 2018; Min et al., 2018; Zajmi et al., 2018; Zhou, (2018); Lillicrap et al., 2020; Zhang & Qu, 2021; Dampfhoffer et al., 2023;
(4), (5), (35), (119)	Delta Rule	Rumelhart et al., 1985; Pineda, 1987; Widrow & Lehr, 1990; Billings et al., 1991; Wythoff, 1993; Yegnanarayan, 1994; Rojas & Rojas, 1996; Thimm et al., 1996; Magoulas et al., 1997; Hagan & Demuth, 1999; Bao et al., 2018; Lv et al., 2018; Zajmi et al., 2018; Zhou, 2018; Lillicrap et al., 2020; Zhang & Qu, 2021; Zhao et al., 2024;
(6)	Correction Term of the Weighting Connections Between Neurons of the Hidden Layer and Neurons of the Output Layer	Rumelhart et al., 1985; Widrow & Lehr, 1990; Billings et al., 1991; Wythoff 1993; Yegnanarayana, 1994; Thimm et al., 1996; Rojas & Rojas, 1996; Kalogirou, 2000; Chen & Zhong, 2009; Park & Lek, 2016; Al-Mahasneh et al., 2017; Zajmi et al., 2018; Zhang & Qu, 2021;
(7)	Correction Term of the Weighting Connections Between Neurons of the Input Layer and Neurons of the Output Layer	Rumelhart et al., 1985; Widrow & Lehr, 1990; Billings et al., 1991; Wythoff, 1993; Yegnanarayana, 1994; Rojas & Rojas, 1996; Thimm et al., 1996; Kalogirou, 2000; Chen & Zhong, 2009; Park & Lek, 2016; Al-Mahasneh et al., 2017; Zajmi et al., 2018;
(8)	Learning Rule	Rumelhart et al., 1985; Widrow & Lehr, 1990; Billings et al., 1991; Hecht-Nielsen, 1992; Wythoff 1993; Yegnanarayana, 1994; Rojas & Rojas, 1996; Thimm et al., 1996; Hagan & Demuth, 1999; Park & Lek, 2016; Zajmi et al., 2018;
(9)	Learning Rule (with momentum)	Rumelhart et al., 1985; Widrow & Lehr, 1990; Billings et al., 1991; Yegnanarayana, 1994; Wythoff, 1993; Park & Lek, 2016; Zajmi et al., 2018;

Table 1. Fundamental equations along with their application (left column), their use (central column), and the literature where they were found (right column).

Equations	Use	Literature
(26)	Total Error	Rumelhart et al., 1985; Werbos 1990; Widrow & Lehr, 1990; Frasconi et al., 1993; Sontag, 1993; Wythoff, 1993; Rojas & Rojas, 1996; Park & Lek, 2016; Min et al., 2018;
(27), (44), (63), (73), (82), (91), (101), (110)	Chain Rule	Rumelhart et al., 1985; Werbos, 1990; Widrow & Lehr, 1990; Billings et al., 1991; Hecht-Nielsen, 1992; Hagan & Demuth, 1999; Lv et al., 2018; Zhou, 2018; Zhang & Qu, 2021; Dampfhoffer et al., 2023;
(28), (29), (30), (31), (32), (33), (46), (49), (52), (56), (59), (62), (65), (68), (71), (75), (78), (81), (84), (87), (90), (93), (96), (99), (103), (106), (108), (112), (113), (115), (118)	First Order Partial Derivative – Relevant to the Chain Rule and the Delta Rule	Newton, 1736;
(47), (57), (50), (60), (66), (69), (76), (79), (85), (88), (94), (97), (104), (107), (116)	Derivative of Sigmoid Function	Werbos, 1990; Widrow & Lehr, 1990; Rojas & Rojas, 1996;

Table 2. Fundamental equations along with their application (left column), their use (central column), and the literature where they were found (right column).

Step 1. Forward Pass

From the input layer to the hidden layer

$$net_{h11} = w_1 \cdot i_{11} + w_2 \cdot i_{12} + b_1 \cdot 1 \quad (10)$$

$$out_{h11} = \frac{1}{(1 + e^{-net_{h11}})} \quad (11)$$

$$net_{h12} = w_3 \cdot i_{11} + w_4 \cdot i_{12} + b_2 \cdot 1 \quad (12)$$

$$out_{h12} = \frac{1}{(1 + e^{-net_{h12}})} \quad (13)$$

$$net_{h21} = w_1 \cdot i_{21} + w_2 \cdot i_{22} + b_1 \cdot 1 \quad (14)$$

$$out_{h21} = \frac{1}{(1 + e^{-net_{h21}})} \quad (15)$$

$$net_{h22} = w_3 \cdot i_{21} + w_4 \cdot i_{22} + b_2 \cdot 1 \quad (16)$$

$$out_{h22} = \frac{1}{(1 + e^{-net_{h22}})} \quad (17)$$

From the hidden layer to the output layer

$$net_{11} = w_5 \cdot out_{h11} + w_6 \cdot out_{h12} + b_3 \cdot 1 \quad (18)$$

$$out_{11} = \frac{1}{(1 + e^{-net_{11}})} \quad (19)$$

$$net_{12} = w_7 \cdot out_{h11} + w_8 \cdot out_{h12} + b_4 \cdot 1 \quad (20)$$

$$out_{12} = \frac{1}{(1 + e^{-net_{12}})} \quad (21)$$

$$net_{21} = w_5 \cdot out_{h21} + w_6 \cdot out_{h22} + b_3 \cdot 1 \quad (22)$$

$$out_{21} = \frac{1}{(1 + e^{-net_{21}})} \quad (23)$$

$$net_{22} = w_7 \cdot out_{h21} + w_8 \cdot out_{h22} + b_4 \cdot 1 \quad (24)$$

$$out_{22} = \frac{1}{(1 + e^{-net_{22}})} \quad (25)$$

Step 2. Calculation of the Error of Fit

The target patterns are $(target_{11}, target_{12}) = (1, 0)$ & $(target_{21}, target_{22}) = (0, 1)$.

$$E_{total} = = \left(\frac{1}{2}\right) \cdot [(out_{11} - 1)^2 + (out_{12} - 0)^2 + (out_{21} - 0)^2 + (out_{22} - 1)^2] \quad (26)$$

Step 3. Backpropagation from Output Layer to Hidden Layer

The weighting connections w_5, w_6, b_3, w_7, w_8, b_4 are treated the same way as they are treated for the two-layer neural network. Indeed, mapping within the network architecture included between the output layer and the hidden layer is conceptually the same as that one written for the two-layers neural network.

Step 3.1 Update of the Weighting Connection w_5

To report and example this section shows how to update the weighting connection w_5. The connection w_5 needs to be updated by the delta rule using the two states net_{11} and net_{21} that the network can have based on the two input patterns $(i_{11}\, i_{12}), (i_{21}\, i_{22})$. The two input patterns belong to two different classes.

Hence the following equations hold true for the three-layer neural network

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{11}} \cdot \frac{\partial out_{11}}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial w_5} + \frac{\partial E_{total}}{\partial out_{21}} \cdot \frac{\partial out_{21}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial w_5} \quad (27)$$

$$\frac{\partial E_{total}}{\partial out_{11}} = (out_{11} - target_{11}) \quad (28)$$

$$\frac{\partial E_{total}}{\partial out_{21}} = (out_{21} - target_{21}) \quad (29)$$

$$\frac{\partial out_{11}}{\partial net_{11}} = out_{11} \cdot (1 - out_{11}) \quad (30)$$

$$\frac{\partial out_{21}}{\partial net_{21}} = out_{21} \cdot (1 - out_{21}) \quad (31)$$

$$\frac{\partial net_{11}}{\partial w_5} = \frac{\partial(w_5 \cdot out_{h11} + w_6 \cdot out_{h12} + b_3 \cdot 1)}{\partial w_5} = \frac{\partial(w_5 \cdot out_{h11})}{\partial w_5} = out_{h11} \quad (32)$$

$$\frac{\partial net_{21}}{\partial w_5} = \frac{\partial(w_5 \cdot out_{h21} + w_6 \cdot out_{h22} + b_3 \cdot 1)}{\partial w_5} = \frac{\partial(w_5 \cdot out_{h21})}{\partial w_5} = out_{h21} \quad (33)$$

$$\frac{\partial E_{total}}{\partial w_5} = [(out_{11} - target_{11})] \cdot out_{11} \cdot (1 - out_{11}) \cdot out_{h11} + [(out_{21} - target_{21})] \cdot out_{21} \cdot (1 - out_{21}) \cdot out_{h21} \quad (34)$$

The delta rule admits

$$\Delta w_5 = -\eta \cdot \frac{\partial E_{total}}{\partial w_5} \quad (35)$$

Step 4. Backpropagation from Hidden Layer to Input Layer

The example of this section shows how to calculate the update of the weighting connection w_1. For each weighting connection and each threshold logic unit of the network the following partial corrections need to be applied:

$$\frac{\partial E_{total}}{\partial out_{h11}} = \frac{\partial E_{total}}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial out_{h11}} \quad (36)$$

$$\frac{\partial E_{total}}{\partial out_{h12}} = \frac{\partial E_{total}}{\partial net_{12}} \cdot \frac{\partial net_{12}}{\partial out_{h12}} \quad (37)$$

$$\frac{\partial E_{total}}{\partial out_{h21}} = \frac{\partial E_{total}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial out_{h21}} \quad (38)$$

$$\frac{\partial E_{total}}{\partial out_{h22}} = \frac{\partial E_{total}}{\partial net_{22}} \cdot \frac{\partial net_{22}}{\partial out_{h22}} \quad (39)$$

$$\frac{\partial E_{total}}{\partial out_{h12}} = \frac{\partial E_{total}}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial out_{h12}} \quad (40)$$

$$\frac{\partial E_{total}}{\partial out_{h11}} = \frac{\partial E_{total}}{\partial net_{12}} \cdot \frac{\partial net_{12}}{\partial out_{h11}} \quad (41)$$

$$\frac{\partial E_{total}}{\partial out_{h21}} = \frac{\partial E_{total}}{\partial net_{22}} \cdot \frac{\partial net_{22}}{\partial out_{h21}} \quad (42)$$

$$\frac{\partial E_{total}}{\partial out_{h22}} = \frac{\partial E_{total}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial out_{h22}} \quad (43)$$

The sum of these partial corrections quantifies, for each weighting connection and for each threshold logic unit, the total correction to apply at each iteration. Each partial correction is part of the chain rule to determine the relations existing between E_{total} and w_1 (see Step 4.1). To help the visual connection between the formulae and the topography of the network the reader is referred to in Figure 1.

Step 4.1 Update of the Weighting Connection w_1

For equation (36) we can write

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h11}} \cdot \frac{\partial out_{h11}}{\partial net_{h11}} \cdot \frac{\partial net_{h11}}{\partial w_1}$$

$$= \frac{\partial E_{total}}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial out_{h11}} \cdot \frac{\partial out_{h11}}{\partial net_{h11}} \cdot \frac{\partial net_{h11}}{\partial w_1} \quad (44)$$

$$\frac{\partial E_{total}}{\partial net_{11}} = \frac{\partial E_{total}}{\partial out_{11}} \cdot \frac{\partial out_{11}}{\partial net_{11}} \quad (45)$$

$$\frac{\partial E_{total}}{\partial out_{11}} = [(out_{11} - target_{11})] \quad (46)$$

$$\frac{\partial\, out_{11}}{\partial\, net_{11}} = out_{11} \cdot (1 - out_{11}) \quad (47)$$

$$net_{11} = w_5 \cdot out_{h11} + w_6 \cdot out_{h12} + b_3 \cdot 1 \quad (48)$$

$$\frac{\partial\, net_{11}}{\partial\, out_{h11}} = \frac{\partial(w_5 \cdot out_{h11} + w_6 \cdot out_{h12} + b_3 \cdot 1)}{\partial\, out_{h11}} = \frac{\partial(w_5 \cdot out_{h11})}{\partial\, out_{h11}} = w_5 \quad (49)$$

$$\frac{\partial\, out_{h11}}{\partial\, net_{h11}} = out_{h11} \cdot (1 - out_{h11}) \quad (50)$$

$$net_{h11} = w_1 \cdot i_{11} + w_2 \cdot i_{12} + b_1 \cdot 1 \quad (51)$$

$$\frac{\partial\, net_{h11}}{\partial\, w_1} = i_{11} \quad (52)$$

$$\frac{\partial\, E_{total}}{\partial\, w_1} = \frac{\partial\, E_{total}}{\partial\, net_{11}} \cdot \frac{\partial\, net_{11}}{\partial\, out_{h11}} \cdot \frac{\partial\, out_{h11}}{\partial\, net_{h11}} \cdot \frac{\partial\, net_{h11}}{\partial\, w_1}$$
$$= [(out_{11} - target_{11})] \cdot out_{11} \cdot (1 - out_{11}) \cdot w_5 \cdot out_{h11} \cdot (1 - out_{h11}) \cdot i_{11} \quad (53)$$

For equation (37) we can write

$$\frac{\partial\, E_{total}}{\partial\, w_1} = \frac{\partial\, E_{total}}{\partial\, out_{h12}} \cdot \frac{\partial\, out_{h12}}{\partial\, net_{h12}} \cdot \frac{\partial\, net_{h12}}{\partial\, w_1}$$
$$= \frac{\partial\, E_{total}}{\partial\, net_{12}} \cdot \frac{\partial\, net_{12}}{\partial\, out_{h12}} \cdot \frac{\partial\, out_{h12}}{\partial\, net_{h12}} \cdot \frac{\partial\, net_{h12}}{\partial\, w_1} \quad (54)$$

$$\frac{\partial\, E_{total}}{\partial\, net_{12}} = \frac{\partial\, E_{total}}{\partial\, out_{12}} \cdot \frac{\partial\, out_{12}}{\partial\, net_{12}} \quad (55)$$

$$\frac{\partial\, E_{total}}{\partial\, out_{12}} = [(out_{12} - target_{12})] \quad (56)$$

$$\frac{\partial\, out_{12}}{\partial\, net_{12}} = out_{12} \cdot (1 - out_{12}) \quad (57)$$

$$net_{12} = w_7 \cdot out_{h11} + w_8 \cdot out_{h12} + b_4 \cdot 1 \quad (58)$$

$$\frac{\partial\, net_{12}}{\partial\, out_{h12}} = \frac{\partial(w_7 \cdot out_{h11} + w_8 \cdot out_{h12} + b_4 \cdot 1)}{\partial\, out_{h12}} = \frac{\partial(w_8 \cdot out_{h12})}{\partial\, out_{h12}} = w_8 \quad (59)$$

$$\frac{\partial\, out_{h12}}{\partial\, net_{h12}} = out_{h12} \cdot (1 - out_{h12}) \quad (60)$$

$$net_{h12} = w_3 \cdot i_{11} + w_4 \cdot i_{12} + b_2 \cdot 1 \quad (61)$$

$$\frac{\partial\, net_{h12}}{\partial\, w_1} = 0 \quad (62)$$

For equation (38) we can write

$$\frac{\partial\, E_{total}}{\partial\, w_1} = \frac{\partial\, E_{total}}{\partial\, out_{h21}} \cdot \frac{\partial\, out_{h21}}{\partial\, net_{h21}} \cdot \frac{\partial\, net_{h21}}{\partial\, w_1}$$
$$= \frac{\partial\, E_{total}}{\partial\, net_{21}} \cdot \frac{\partial\, net_{21}}{\partial\, out_{h21}} \cdot \frac{\partial\, out_{h21}}{\partial\, net_{h21}} \cdot \frac{\partial\, net_{h21}}{\partial\, w_1} \quad (63)$$

$$\frac{\partial E_{total}}{\partial net_{21}} = \frac{\partial E_{total}}{\partial out_{21}} \cdot \frac{\partial out_{21}}{\partial net_{21}} \quad (64)$$

$$\frac{\partial E_{total}}{\partial out_{21}} = [(out_{21} - target_{21})] \quad (65)$$

$$\frac{\partial out_{21}}{\partial net_{21}} = out_{21} \cdot (1 - out_{21}) \quad (66)$$

$$net_{21} = w_5 \cdot out_{h21} + w_6 \cdot out_{h22} + b_3 \cdot 1 \quad (67)$$

$$\frac{\partial net_{21}}{\partial out_{h21}} = \frac{\partial(w_5 \cdot out_{h21} + w_6 \cdot out_{h22} + b_3 \cdot 1)}{\partial out_{h21}} = \frac{\partial(w_5 \cdot out_{h21})}{\partial out_{h21}} = w_5 \quad (68)$$

$$\frac{\partial out_{h21}}{\partial net_{h21}} = out_{h21} \cdot (1 - out_{h21}) \quad (69)$$

$$net_{h21} = w_1 \cdot i_{21} + w_2 \cdot i_{22} + b_1 \cdot 1 \quad (70)$$

$$\frac{\partial net_{h21}}{\partial w_1} = i_{21} \quad (71)$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial out_{h21}} \cdot \frac{\partial out_{h21}}{\partial net_{h21}} \cdot \frac{\partial net_{h21}}{\partial w_1}$$
$$= [(out_{21} - target_{21})] \cdot out_{21} \cdot (1 - out_{21}) \cdot w_5 \cdot out_{h21} \cdot (1 - out_{h21}) \cdot i_{21} \quad (72)$$

For equation (39) we can write

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h22}} \cdot \frac{\partial out_{h22}}{\partial net_{h22}} \cdot \frac{\partial net_{h22}}{\partial w_1}$$
$$= \frac{\partial E_{total}}{\partial net_{22}} \cdot \frac{\partial net_{22}}{\partial out_{h22}} \cdot \frac{\partial out_{h22}}{\partial net_{h22}} \cdot \frac{\partial net_{h22}}{\partial w_1} \quad (73)$$

$$\frac{\partial E_{total}}{\partial net_{22}} = \frac{\partial E_{total}}{\partial out_{22}} \cdot \frac{\partial out_{22}}{\partial net_{22}} \quad (74)$$

$$\frac{\partial E_{total}}{\partial out_{22}} = [(out_{22} - target_{22})] \quad (75)$$

$$\frac{\partial out_{22}}{\partial net_{22}} = out_{22} \cdot (1 - out_{22}) \quad (76)$$

$$net_{22} = w_7 \cdot out_{h21} + w_8 \cdot out_{h22} + b_4 \cdot 1 \quad (77)$$

$$\frac{\partial net_{22}}{\partial out_{h22}} = \frac{\partial(w_7 \cdot out_{h21} + w_8 \cdot out_{h22} + b_4 \cdot 1)}{\partial out_{h22}} = \frac{\partial(w_8 \cdot out_{h22})}{\partial out_{h22}} = w_8 \quad (78)$$

$$\frac{\partial out_{h22}}{\partial net_{h22}} = out_{h22} \cdot (1 - out_{h22}) \quad (79)$$

$$net_{h22} = w_3 \cdot i_{21} + w_4 \cdot i_{22} + b_2 \cdot 1 \quad (80)$$

$$\frac{\partial net_{h22}}{\partial w_1} = 0 \quad (81)$$

For equation (40) we can write

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h12}} \cdot \frac{\partial out_{h12}}{\partial net_{h12}} \cdot \frac{\partial net_{h12}}{\partial w_1}$$
$$= \frac{\partial E_{total}}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial out_{h12}} \cdot \frac{\partial out_{h12}}{\partial net_{h12}} \cdot \frac{\partial net_{h12}}{\partial w_1} \quad (82)$$

$$\frac{\partial E_{total}}{\partial net_{11}} = \frac{\partial E_{total}}{\partial out_{11}} \cdot \frac{\partial out_{11}}{\partial net_{11}} \quad (83)$$

$$\frac{\partial E_{total}}{\partial out_{11}} = [(out_{11} - target_{11})] \quad (84)$$

$$\frac{\partial out_{11}}{\partial net_{11}} = out_{11} \cdot (1 - out_{11}) \quad (85)$$

$$net_{11} = w_5 \cdot out_{h11} + w_6 \cdot out_{h12} + b_3 \cdot 1 \quad (86)$$

$$\frac{\partial net_{11}}{\partial out_{h12}} = \frac{\partial(w_5 \cdot out_{h11} + w_6 \cdot out_{h12} + b_3 \cdot 1)}{\partial out_{h12}} = \frac{\partial(w_6 \cdot out_{h12})}{\partial out_{h12}} = w_6 \quad (87)$$

$$\frac{\partial out_{h12}}{\partial net_{h12}} = out_{h12} \cdot (1 - out_{h12}) \quad (88)$$

$$net_{h12} = w_3 \cdot i_{11} + w_4 \cdot i_{12} + b_2 \cdot 1 \quad (89)$$

$$\frac{\partial net_{h12}}{\partial w_1} = 0 \quad (90)$$

For equation (41) we can write

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h11}} \cdot \frac{\partial out_{h11}}{\partial net_{h11}} \cdot \frac{\partial net_{h11}}{\partial w_1}$$
$$= \frac{\partial E_{total}}{\partial net_{12}} \cdot \frac{\partial net_{12}}{\partial out_{h11}} \cdot \frac{\partial out_{h11}}{\partial net_{h11}} \cdot \frac{\partial net_{h11}}{\partial w_1} \quad (91)$$

$$\frac{\partial E_{total}}{\partial net_{12}} = \frac{\partial E_{total}}{\partial out_{12}} \cdot \frac{\partial out_{12}}{\partial net_{12}} \quad (92)$$

$$\frac{\partial E_{total}}{\partial out_{12}} = [(out_{12} - target_{12})] \quad (93)$$

$$\frac{\partial out_{12}}{\partial net_{12}} = out_{12} \cdot (1 - out_{12}) \quad (94)$$

$$net_{12} = w_7 \cdot out_{h11} + w_8 \cdot out_{h12} + b_4 \cdot 1 \quad (95)$$

$$\frac{\partial net_{12}}{\partial out_{h11}} = \frac{\partial(w_7 \cdot out_{h11} + w_8 \cdot out_{h12} + b_4 \cdot 1)}{\partial out_{h11}} = \frac{\partial(w_7 \cdot out_{h11})}{\partial out_{h11}} = w_7 \quad (96)$$

$$\frac{\partial out_{h11}}{\partial net_{h11}} = out_{h11} \cdot (1 - out_{h11}) \quad (97)$$

$$net_{h11} = w_1 \cdot i_{11} + w_2 \cdot i_{12} + b_1 \cdot 1 \quad (98)$$

$$\frac{\partial\, net_{h11}}{\partial\, w_1} = i_{11} \quad (99)$$

$$\frac{\partial\, E_{total}}{\partial\, w_1} = \frac{\partial\, E_{total}}{\partial\, net_{12}} \cdot \frac{\partial\, net_{12}}{\partial\, out_{h11}} \cdot \frac{\partial\, out_{h11}}{\partial\, net_{h11}} \cdot \frac{\partial\, net_{h11}}{\partial\, w_1}$$
$$= [(out_{12} - target_{12})] \cdot out_{12} \cdot (1 - out_{12}) \cdot w_7 \cdot out_{h11} \cdot (1 - out_{h11}) \cdot i_{11} \quad (100)$$

For equation (42) we can write

$$\frac{\partial\, E_{total}}{\partial\, w_1} = \frac{\partial\, E_{total}}{\partial\, out_{h21}} \cdot \frac{\partial\, out_{h21}}{\partial\, net_{h21}} \cdot \frac{\partial\, net_{h21}}{\partial\, w_1}$$
$$= \frac{\partial\, E_{total}}{\partial\, net_{22}} \cdot \frac{\partial\, net_{22}}{\partial\, out_{h21}} \cdot \frac{\partial\, out_{h21}}{\partial\, net_{h21}} \cdot \frac{\partial\, net_{h21}}{\partial\, w_1} \quad (101)$$

$$\frac{\partial\, E_{total}}{\partial\, net_{22}} = \frac{\partial\, E_{total}}{\partial\, out_{22}} \cdot \frac{\partial\, out_{22}}{\partial\, net_{22}} \quad (102)$$

$$\frac{\partial\, E_{total}}{\partial\, out_{22}} = [(out_{22} - target_{22})] \quad (103)$$

$$\frac{\partial\, out_{22}}{\partial\, net_{22}} = out_{22} \cdot (1 - out_{22}) \quad (104)$$

$$net_{22} = w_7 \cdot out_{h21} + w_8 \cdot out_{h22} + b_4 \cdot 1 \quad (105)$$

$$\frac{\partial\, net_{22}}{\partial\, out_{h21}} = \frac{\partial(w_7 \cdot out_{h21} + w_8 \cdot out_{h22} + b_4 \cdot 1)}{\partial\, out_{h21}} = \frac{\partial(w_7 \cdot out_{h21})}{\partial\, out_{h21}} = w_7 \quad (106)$$

$$\frac{\partial\, out_{h21}}{\partial\, net_{h21}} = out_{h21} \cdot (1 - out_{h21}) \quad (107)$$

$$\frac{\partial\, net_{h21}}{\partial\, w_1} = i_{21} \quad (108)$$

$$\frac{\partial\, E_{total}}{\partial\, w_1} = \frac{\partial\, E_{total}}{\partial\, net_{22}} \cdot \frac{\partial\, net_{22}}{\partial\, out_{h21}} \cdot \frac{\partial\, out_{h21}}{\partial\, net_{h21}} \cdot \frac{\partial\, net_{h21}}{\partial\, w_1}$$
$$= [(out_{22} - target_{22})] \cdot out_{22} \cdot (1 - out_{22}) \cdot w_7 \cdot out_{h21} \cdot (1 - out_{h21}) \cdot i_{21} \quad (109)$$

For equation (43) we can write

$$\frac{\partial\, E_{total}}{\partial\, w_1} = \frac{\partial\, E_{total}}{\partial\, out_{h22}} \cdot \frac{\partial\, out_{h22}}{\partial\, net_{h22}} \cdot \frac{\partial\, net_{h22}}{\partial\, w_1}$$
$$= \frac{\partial\, E_{total}}{\partial\, net_{21}} \cdot \frac{\partial\, net_{21}}{\partial\, out_{h22}} \cdot \frac{\partial\, out_{h22}}{\partial\, net_{h22}} \cdot \frac{\partial\, net_{h22}}{\partial\, w_1} \quad (110)$$

$$\frac{\partial\, E_{total}}{\partial\, net_{21}} = \frac{\partial\, E_{total}}{\partial\, out_{21}} \cdot \frac{\partial\, out_{21}}{\partial\, net_{21}} \quad (111)$$

$$\frac{\partial\, E_{total}}{\partial\, out_{21}} = [(out_{21} - target_{21})] \quad (112)$$

$$\frac{\partial\, out_{21}}{\partial\, net_{21}} = out_{21} \cdot (1 - out_{21}) \quad (113)$$

$$net_{21} = w_5 \cdot out_{h21} + w_6 \cdot out_{h22} + b_3 \cdot 1 \quad (114)$$

$$\frac{\partial\, net_{21}}{\partial\, out_{h22}} = \frac{\partial(w_5 \cdot out_{h21} + w_6 \cdot out_{h22} + b_3 \cdot 1)}{\partial\, out_{h22}} = \frac{\partial(w_6 \cdot out_{h22})}{\partial\, out_{h22}} = w_6 \quad (115)$$

$$\frac{\partial\, out_{h22}}{\partial\, net_{h22}} = out_{h22} \cdot (1 - out_{h22}) \quad (116)$$

$$net_{h22} = w_3 \cdot i_{21} + w_4 \cdot i_{22} + b_2 \cdot 1 \quad (117)$$

$$\frac{\partial\, net_{h22}}{\partial\, w_1} = 0 \quad (118)$$

Hence, the cumulative correction term to use for w_1 is the sum of equations (53), (72), (100), and (109) and the four zero terms as shown by the math processes of this section.

$$\frac{\partial\, E_{cumulative}}{\partial\, w_1} = [(out_{11} - target_{11})] \cdot out_{11} \cdot (1 - out_{11}) \cdot w_5 \cdot out_{h11} \cdot (1 - out_{h11}) \cdot i_{11} + 0 + [(out_{21} - target_{21})] \cdot out_{21} \cdot (1 - out_{21}) \cdot w_5 \cdot out_{h21} \cdot (1 - out_{h21}) \cdot i_{21} + 0 + 0 + [(out_{12} - target_{12})] \cdot out_{12} \cdot (1 - out_{12}) \cdot w_7 \cdot out_{h11} \cdot (1 - out_{h11}) \cdot i_{11} + [(out_{22} - target_{22})] \cdot out_{22} \cdot (1 - out_{22}) \cdot w_7 \cdot out_{h21} \cdot (1 - out_{h21}) \cdot i_{21} + 0 \quad (119)$$

RESULTS

Data Sets and Problems

The experimental sessions were conducted in the following manner. Each image contained one shape only. Each shape was reproduced in more images by changing the position or the size across the grid of digital samples. Each pattern was an image of 147 x 168 pixels. Hence the three-layer perceptron input comprised of 24,696 values. The pixel brightness was either 1000.0 inside the shape or 0.0 outside the shape. Each of the images of the training and test sets was calculated separately with no need for interpolation or extrapolation. The use of 1000.0 or 0.0 for pixel brightness does not cause numerical instability in the sigmoid because the pixel brightness is scaled into [0, 1] before the image is processed by the network. The hidden layer was inclusive of 10 neurons for all the experiments. Since the network's performance was sensitive to the hidden layer size, 10 is the number of neurons which was empirically determined as optimal provider of successful learning process. The success of the learning process was measured by the ability of the network to reach the global minimum through the steepest descend optimization. The learning rate was set to 0.001 or to 0.01. The smoothing factor (momentum) was set to 0.005 or to 0.05.

Three two-class problems were generated: 1. Rectangular and Square shapes (Table 3), 2. Circular and Square shapes (Table 4), 3. Circular and Rectangular shapes (Table 5). The output layer has 2 neurons (one for each class). Each learning process encompassed one of the three

two-class problems. Each data set was either composed of 1. Shapes at different position across the digital grid, or 2. Shapes at different positions and with different size. Each learning process is detailed in Tables 3, 4, and 5. In these tables are presented, from left to right, the number of patterns (images) of each learning process, the size of the shapes in pixels, the classification rate, and the first separation layer (and neuron). The tables also show, from left to right, the variables of the problem (Shape, Size, Position), the number of patterns (images) used for validation (after the learning process), the size of the shapes, and the correct classification score of the validation test. Figure 2 shows shapes sizes with varying positions.

After the Learning Processes

After the learning processes the error of fit was satisfactorily low (below 0.31 units), which means it was such to assure 100% correct classification of the patterns used for the learning process. A neural network builder loaded the values of the weighting connections and the values of the bias of the threshold logic units; and each image was feed-forwarded through the net. The recognition rate was such to assert that the three-layer perceptron encoded successfully the three variables shape, position and size.

The expected finding revealed by the Hidden Activation Analysis is the main result. The network's hidden layer was able in some cases to encode the three variables hence assuring separation of the two classes. *This analysis is the novelty of this research. This aspect is reported in Tables 3, 4, and 5 (see column labelled 'First Separation - Hidden Layer, Neuron; Output Layer'), which shows the neuron able to separate the two classes*. In the hidden layer, HID 1 is the first neuron, HID 2 is the second neuron, and so on. The Hidden Activation Analysis, which looks at the numerical values of the sigmoidal output at the hidden layer, is reported in Figure 3 for a case that involves all the 10 neurons. In this case the network successfully separates the two classes by means of each the 10 neurons of the hidden layer, hence it would not require additional processing of the output layer.

The main result, however, is revealed by HAA and it is that the hidden layer can encode the three variables at the same time. The results of HAA are reported in Tables 3, 4, and 5 in the column labelled "First Separation revealed by HAA or Output Layer Separation. Hidden Layer Neuron (HID) or Output Layer (OUT)". This column reports the number (identifier, e.g. HID 1) of the hidden layer neuron where the pattern recognition problem is solved by the network consequently to the learning process; or OUT which means that the network uses the output layer to solve the pattern recognition task. That means that after the learning process, the net can solve the pattern recognition task at a specific hidden neuron (e.g. HID 1). If the solution is not found at any of the hidden neurons, then it is found at the output layer ("Output Layer Separation").

After the learning process, the network was able to generalize to shapes at unobserved positions and sizes. Each row in Tables 3, 4, and 5 is corroborated by the test session described by the four right-most columns which show the number of images used by the tests, the size of the shapes, and the correct classification rate. The correct classification rate of the patterns used for the learning process was 100% in all data sets (see tables). The correct classification rate of the

images with shapes at unobserved positions and sizes (patterns not used for the learning process) was high (above 95% overall).

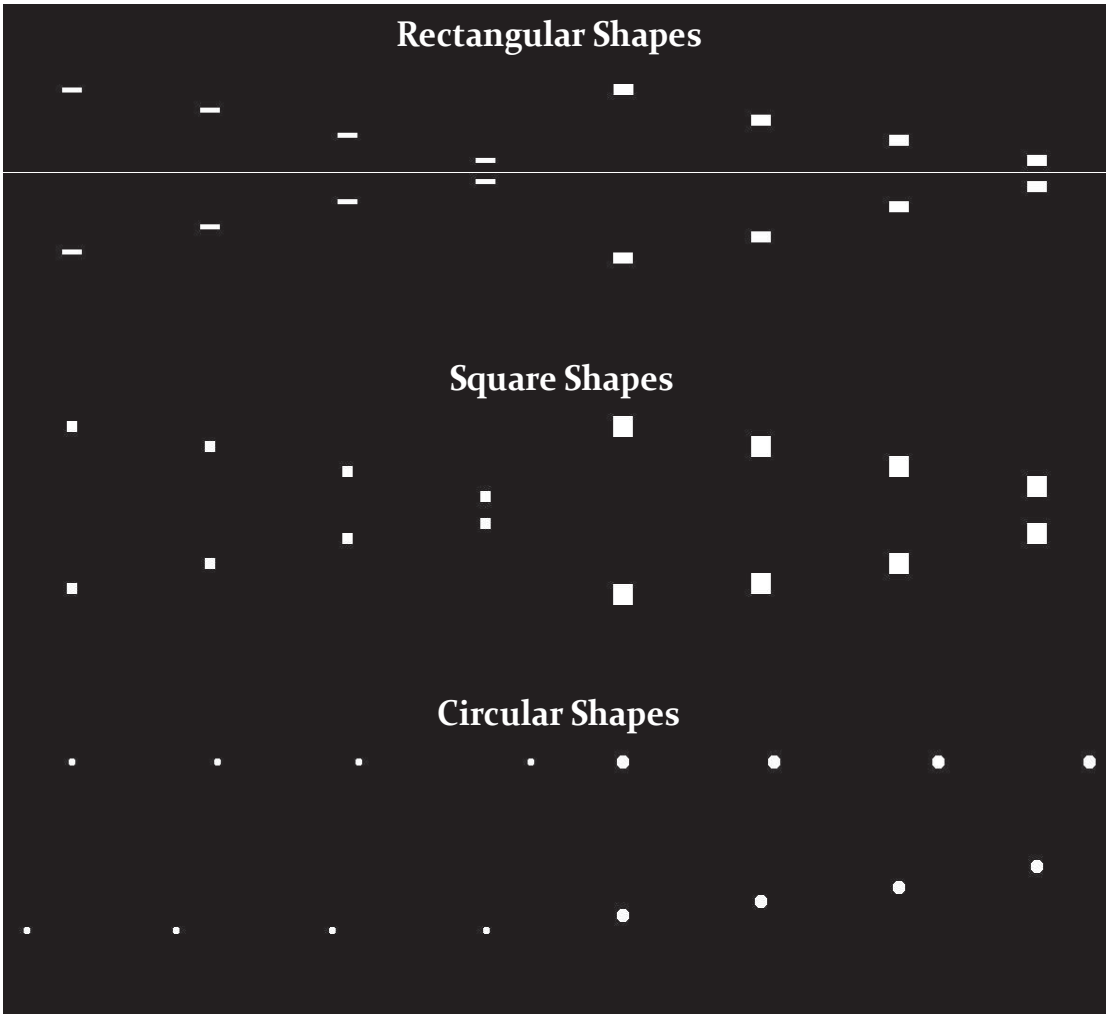

Figure 2. Images illustrating the rationale of the learning processes. Data is featured by shapes of different sizes located at different positions in the digital grid. With varying position, the top row and the second row from the top show the rectangular shape, the third and the fourth row from the top show square shapes, and the fifth row from the top and the bottom row show circular shapes. Each image consists of 147 x 168 pixels. Hence, the net receives 24,696 inputs. The pixel brightness is either 1000.0 or 0.0. The output neurons are 2; one for each class.

Number of Images - Learning Process	Rectangle Size (Pixels) – Learning Process	Square Size (Edge) – Learning Process	Correct Classification Rate (%) - Learning Process	First Separation revealed by HAA or Output Layer Separation. Hidden Layer Neuron (HID) or Output Layer (OUT)	Variables (Shape; Size; Position)	Number of Images - Test	Rectangle Size (Pixels) - Test	Square Size (Edge) – Test	Correct Classification Rate (%) - Test
38 (19 & 19)	20 x 10	10	100	HID 6; HID 8;	Shape; Position;	40 (20 & 20)	20 x 8	8	100
76 (19 & 19) (19 & 19)	20 x 10 14 x 7	15 10	100	HID 4; HID 9;	Shape; Size; Position;	100 (50 x 50)	20 x 7	15	95
84 (21 & 21) (21 & 21)	15 x 10 10 x 7	10 7	100	OUT;	Shape; Size; Position;	108 (54 & 54)	15 x 7	10	99
140 (35 & 35) (35 & 35)	20 x 5 14 x 3	10 7	100	HID 1;	Shape; Size; Position;	140 (70 & 70)	20 x 5	8	97
140 (35 & 35) (35 & 35)	20 x 5 14 x 3	15 10	100	HID 2;	Shape; Size; Position;	228 (114 & 114)	20 x 3	15	99
140 (35 & 35) (35 & 35)	20 x 5 14 x 3	5 3	100	HID 2; HID 4; HID 6; HID 8; HID 10;	Shape; Size; Position;	228 (114 & 114)	20 x 3	5	99.5
148 (37 & 37) (37 & 37)	15 x 5 10 x 3	10 7	100	OUT;	Shape; Size; Position;	236 (118 & 118)	15 x 3	10	99.5
148 (37 & 37) (37 & 37)	15 x 5 10 x 3	5 3	100	HID 2; HID 4; HID 6; HID 8; HID 10;	Shape; Size; Position;	148 (74 & 74)	15 x 5	4	100
160 (21 & 19) (21 & 19) (21 & 19) (21 & 19)	20 x 10 14 x 7 15 x 10 10 x 7	15 10 10 7	100	OUT;	Shape; Size; Position;	76 (38 x 38)	20 x 10	13	97
172 (43 & 43) (43 & 43)	10 x 5 7 x 3	8 5	100	HID 2;	Shape; Size; Position;	260 (130 & 130)	10 x 3	8	100
172 (43 & 43) (43 & 43)	10 x 5 7 x 3	5 3	100	HID 2; HID 4; HID 6; HID 8; HID 10;	Shape; Size; Position;	124 (62 x 62)	10 x 8	4	100
300 (19 & 21) (19 & 21) (35 & 35) (35 & 35) (21 & 19) (21 & 19)	20 x 10 14 x 7 20 x 5 14 x 3 10 x 5 7 x 3	15 10 10 7 5 3 x 3	100	OUT;	Shape; Size; Position;	92 (46 & 46)	20 x 8	12	95

Table 3. Training set and test set correct classification rates after the learning processes for rectangular and square shapes. For all the sessions, the value of the learning rate is 0.001, and the value of the smoothing factor is 0.005. The number of neurons in the hidden layer was consistently 10. After the training sessions, to verify the generalization power of the net, the three-layer perceptron confronted novel patterns not used during the learning process. For each experimental session reported in the table, new patterns were presented to the net, and the correct classification score was as low as 95%. The fifth column from the left shows the results of the Hidden Activation Analysis. This column shows the identifier (e.g. HID 1) of the neuron in the hidden layer that can solve the pattern recognition problem; or OUT which means that the network solves the pattern recognition task at the output layer.

3.3 Ablation, Comparison with Competitive Method, and Noise Study

This section reports on an ablation study and a comparison of the TLP with a competitive method developed to assist the performance evaluation. The ablation study was conducted on a three classes pattern recognition task involving the classification of three shapes at once. The shapes were circle, square and rectangle, and their position and size varied across the digital grid. The experiments were conducted in order 1. To determine the optimal number of neurons in the hidden layer of the TLP and 2. To compare the TLP with a competitive method. The competitive method is the TLP with growing hidden layer architecture while performing the learning process. The idea of the competitive method is to determine the neural network internal structure (number of neurons in the hidden layer) by making use of the learning process. Although equations (1) through (9) constitutes the core of the math implemented in the competitive method, details are here omitted because outside the scope of the present paper.

The comparison is illustrated in Figure 4a. The competitive method was trained with 6 neurons in the hidden layer. The TLP proposed by this research was ablated using 7, 8, 9, 10, 11, 12, 13, and 14 neurons in the hidden layer. Results of the ablation (see Figure 4a) show that the best architecture is archivable with 9, 10, and 12 neurons in the hidden layer. After the learning process, the recognition rate of the TLP is 100% versus 99% of the competing method (see yellow bar). Hence, the TLP shows to be superior versus the competitive method. Moreover, Figures 4b and 4c show the HAA performed on the TLP that has 9 neurons in the hidden layer (see bar with pattern fill in Figure 4a). HAA revealed that after the

learning process neuron 1 (HID 1) and neuron 9 (HID 9) can solve the three classes pattern recognition problem. Indeed, the charts in Figures 4b and 4c show the separation of the three classes, as shown by the spatial distribution of the numerical values of the hidden neurons' activations. The plot in Figure 4a shows also the robustness to the noise of the competitive method (see the three bars with gradient fill). The competitive method converged with 6 neurons in the hidden layer and had a classification rate of 99% for three experimental sessions with three different noise levels. The noise was introduced in the images by adding 1.0, 5.0, and 10.0 units to the pixel brightness values. Figure 4d shows the noise pattern which was used to contaminate the data.

Number of Images - Learning Process	Circle Diameter (Pixels) – Learning Process	Square Size (Edge) – Learning Process	Correct Classification Rate (%) - Learning Process	First Separation revealed by HAA or Output Layer Separation. Hidden Layer Neuron (HIID) or Output Layer (OUT)	Variables (Shape; Size; Position)	Number of Images - Test	Circle Diameter (Pixels) - Test	Square Size (Edge) – Test	Correct Classification Rate (%) - Test
60 (30 & 30)	15	15	100	HID 5; HID 9;	Shape; Position;	76 (38 & 38)	8	14	96
60 (30 & 30)	10	20	100	HID 1 through HID 10;	Shape; Position;	76 (38 & 38)	8	14	97
60 (30 & 30)	15	20	100	HID 1; HID 2; HID 5; HID 9;	Shape; Position;	68 (48 x 20)	6	18	100
76 (38 & 19) (19)	8	10 7	100	HID 2; HID 4; HID 6; HID 8; HID 10;	Shape; Size; Position;	68 (48 x 20)	6	18	100
76 (38 & 19) (19)	10	10 7	100	HID 1; HID 2; HID 8; HID 9;	Shape; Size; Position;	76 (38 & 38)	6	15	100
76 *(38 & 19)* *(19)*	*8*	*15* *10*	*100*	*HID 1 through HID 10;* *(see Figure 3)*	*Shape; Size; Position;*	76 (38 & 38)	6	15	100
76 (38 & 19) (19)	8	8 5	100	HID 2; HID 4; HID 8; HID 10;	Shape; Size; Position;	76 (38 & 38)	6	12	98
76 (30 & 19) (8 & 19)	15 20	25 17	100	HID 1; HID 5;	Shape; Size; Position;	78 (40 & 38)	15	23	100
120 (30 & 30) (30 & 30)	8 10	20 15	100	HID 2;	Shape; Size; Position;	138 (100 & 38)	9	17	98
120 (30 & 19) (30 & 11) (19) (11)	15 10	15 10 20 14	100	HID 2;	Shape; Size; Position;	138 (100 & 38)	8	14	98
152 (30 & 19) (8 & 19) (38 & 19) (19)	15 20 8	25 17 8 5	100	OUT;	Shape; Size; Position;	78 (40 & 38)	15	20	100
212 (30 & 19) (30 & 19) (38 & 19) (11) (19) (19)	15 10 8	25 17 20 14 8 5	100	OUT;	Shape; Size; Position;	78 (40 & 38)	15	20	100

Table 4. Training set and test set correct classification rates after the learning processes for circular and square shapes. For all the sessions, the value of the learning rate is either 0.001 or 0.01, and the value of the smoothing factor is either 0.005 or 0.05. The number of neurons in the hidden layer was consistently 10. The ability to correctly classify a novel pattern (unseen at the time of the learning process) both in size and position of the shape is relevant though to the after-the-learning-process generalization power of the three-layer perceptron (see right-most column in the table). The fifth column from the left shows the results of the Hidden Activation Analysis. This column shows the identifier (e.g. HID 1) of the hidden layer neuron that can solve the pattern recognition problem; or OUT which means that the network solves the pattern recognition task at the output layer.

3.4 Linear Classifier

This section presents results obtainable using the perceptron, a linear classifier used to solve the pattern recognition task consisting of linearly separate two images. The images pair were 1. Circle vs. Rectangle, 2. Circle vs. Square, and 3. Square vs. Rectangle. Three hundred iterations using 0.001 as learning rate and 0.005 as smoothing factor, were sufficient for the perceptrons to converge and learn both shape and position difference. Figure 5 shows the image maps of the weighting connections of the three images pairs obtained after the learning process. The weighting connections in (a) are obtained when separating the circle and the rectangle, in (b) the connections were obtained when separating circle and square, and in (c) the connections were

obtained when separating square and rectangle. Figure 5 shows that the perceptrons can encode the shape. The image maps of the weighting connections are binary as they show the white regions where the shapes are reconstructed (classified) and the black regions where the shapes are not reconstructed (negated). This binary behavior confirms that the shapes are separated by the classifiers.

Number of Images - Learning Process	Circle Diameter (Pixels) – Learning Process	Rectangle Size (Pixels) – Learning Process	Correct Classification Rate (%) - Learning Process	First Separation revealed by HAA or Output Layer Separation. Hidden Layer Neuron (HID) or Output Layer (OUT)	Variables (Shape; Size; Position)	Number of Images - Test	Circle Diameter (Pixels) - Test	Rectangle Size (Pixels) – Test	Correct Classification Rate (%) - Test
60 (30 & 30)	15	20 x 10	100	HID 5;	Shape; Position;	127 (39 & 88)	9	20 x 9	96
60 (30 & 15) (15)	10	15 x 10 10 x 7	100	HID 2; HID 4; HID 6; HID 8; HID 10	Shape; Size; Position;	127 (39 & 88)	9	20 x 9	99
76 (38 & 35) (3)	8	15 x 5 10 x 3	100	HID 1; HID 2; HID 4; HID 6; HID 8; HID 10;	Shape; Size; Position;	83 (43 x 40)	7	20 x 5	100
76 (38 & 35) (3)	10	20 x 5 14 x 3	100	HID 2; HID 8;	Shape; Size; Position;	65 (39 & 26)	9	20 x 14	98
76 (38 & 37) (1)	8	15 x 5 10 x 3	100	HID 2; HID 4; HID 6; HID 8; HID 10;	Shape; Size; Position;	95 (69 & 26)	5	20 x 14	98
76 (30 & 37) (8 & 1)	15 18	15 x 5 10 x 3	100	HID 1; HID 3; through HID 9;	Shape; Size; Position;	110	16	13 x 5	100
76 (30 & 38) (8)	15 18	10 x 5	100	HID 1; through HID 10;	Shape; Size; Position;	110 (50 & 60)	16	13 x 5	97
78 (33 & 22) (23)	17	20 x 10 20 x 6	100	OUT	Shape; Size; Position;	98 (9 & 35) (19 & 35)	20 29	20 x 7 20 x 5	97
136 (30 & 19) (30 & 19) (20) (18)	15 8	20 x 10 14 x 7 15 x 10 10 x 7	100	OUT;	Shape; Size; Position;	89 (59 & 30)	6	20 x 5	98
152 (30 & 19) (8 & 19) (38 & 21) (17)	15 18 8	20 x 10 14 x 7 15 x 10 10 x 7	100 100	HID 2;	Shape; Size; Position;	105 (59 & 46)	6	20 x 8	98
168 (30 & 19) (19) (30 & 35) (35)	10 15	20 x 10 14 x 7 20 x 5 14 x 3	100	OUT;	Shape; Size; Position;	89 (31 & 58)	11	20 x 6	100
255 (30 & 19) (19) (8) (30 & 35) (34) (38 & 21) (21)	15 20 10 8	20 x 10 14 x 7 20 x 5 10 x 8 15 x 10 10 x 7	100	OUT;	Shape; Size; Position;	82 (40 & 42)	14	20 x 9	97

Table 5. Training set and test set correct classification rates after the learning processes for circular and rectangular shapes. For all the sessions, the value of the learning rate is either 0.001 or 0.01, and the value of the smoothing factor is either 0.005 or 0.05. The number of neurons in the hidden layer was consistently 10. The correct classification rate of the images with shapes at unobserved positions and sizes (patterns not used for the learning process) was high; as low as 96% overall (see right-most column in the table). The fifth column from the left shows the results of the Hidden Activation Analysis. This column shows the identifier (e.g. HID 1) of the hidden layer neuron that can solve the pattern recognition problem; or OUT which means that the network solves the pattern recognition task at the output layer.

DISCUSSION

The Literature

Following the early developments of the sixties of the past century, in the early nineties three major learning rules were developed to train artificial neural networks. The learning rules were

the perceptron rule (Rosemblatt, 1957), the least square adaptation algorithm (Widrow et al., 1988; Widrow & Lehr, 1990), and the backpropagation algorithm (Werbos, 1974). At that time the problem of the adjustable yet complicated structure of the backpropagation based artificial neural networks was already recognized as the network complexity (De Villiers & Barnard, 1993) and it was measured by the Vapnik-Chervonenkis (VC) dimension. The VC dimension relates to the number of weighting connections of the ANN architecture (Baum & Haussler, 1989; Sontag, 1993). As recalled in Bezdek (1992), to characterize algorithms it takes the elucidation of complexity, convergence, stability, robustness, and performance validation.

In fact, contemporary research uses statistical theory to model the output layer, within the effort to develop a general model for multilayer neural networks. This theory can be generalized to networks that use any form of learning rule or optimization procedure (Kleyko et al., 2023). Studying three-layer and four-layer neural networks accomplished the result which ascertains that the latter is not superior to the former. Moreover, the four-layer neural networks were more difficult to train and prone to get trapped in a local minimum of the cost function (De Villiers & Barnard, 1993). More than a single hidden layer and a neurons' overfull hidden layer usually degrade performance yielding to an increased number of false minima, and overfitting which in terms translate to poor generalization ability of the net (Macukow, 2016; Park & Lek, 2016). Moreover, recent research studied the optimal hidden layers' architecture as defined by the number of layers in a multilayer perceptron, and it showed that 3 layers were optimal for the network architecture (Uzair & Jamil, 2020).

The present research work finds its place in the literature in between the establishment of ANNs that were built based on the perceptron original model, and the years of the new century when we have witnessed the expansion of ANNs to deep learning approaches that make use of CNNs (Hinton et al., 2006; Hinton & Salakhutdinov, 2006), and the evolutional neural networks (Yao, 1999; Al-Mahasneh et al., 2017). The widespread use of CNNs has been witnessed by their application to various pattern recognition tasks such as facial classification (Fasel, 2002; Tivive & Bouzerdoum, 2004). Handwritten classification (Simard et al., 2003), and object detection (Liu et al., 2015). Also, the calculation of the Fourier transform between hidden layers was able to classify human electroencephalographic (EEG) activity, hence was able to analyze the frequency domain of the EEG (Cecotti & Graeser, 2008). Speech recognition (Abdel-Hamid et al., 2014), and more generally, copious vocabulary continuous speech recognition (LVCSR) (Sainath et al., 2013). Other applications include the parallelization of CNNs using graphic processing unit (GPU) to achieve face recognition and pose estimation (Nasse et al., 2009), or image classification (Ciresan et al., 2011; Krizhevsky et al., 2012). CNNs often benefit from the advantage of identifying the object pattern in an input image of arbitrary size (Tivive & Bouzerdoum, 2004).

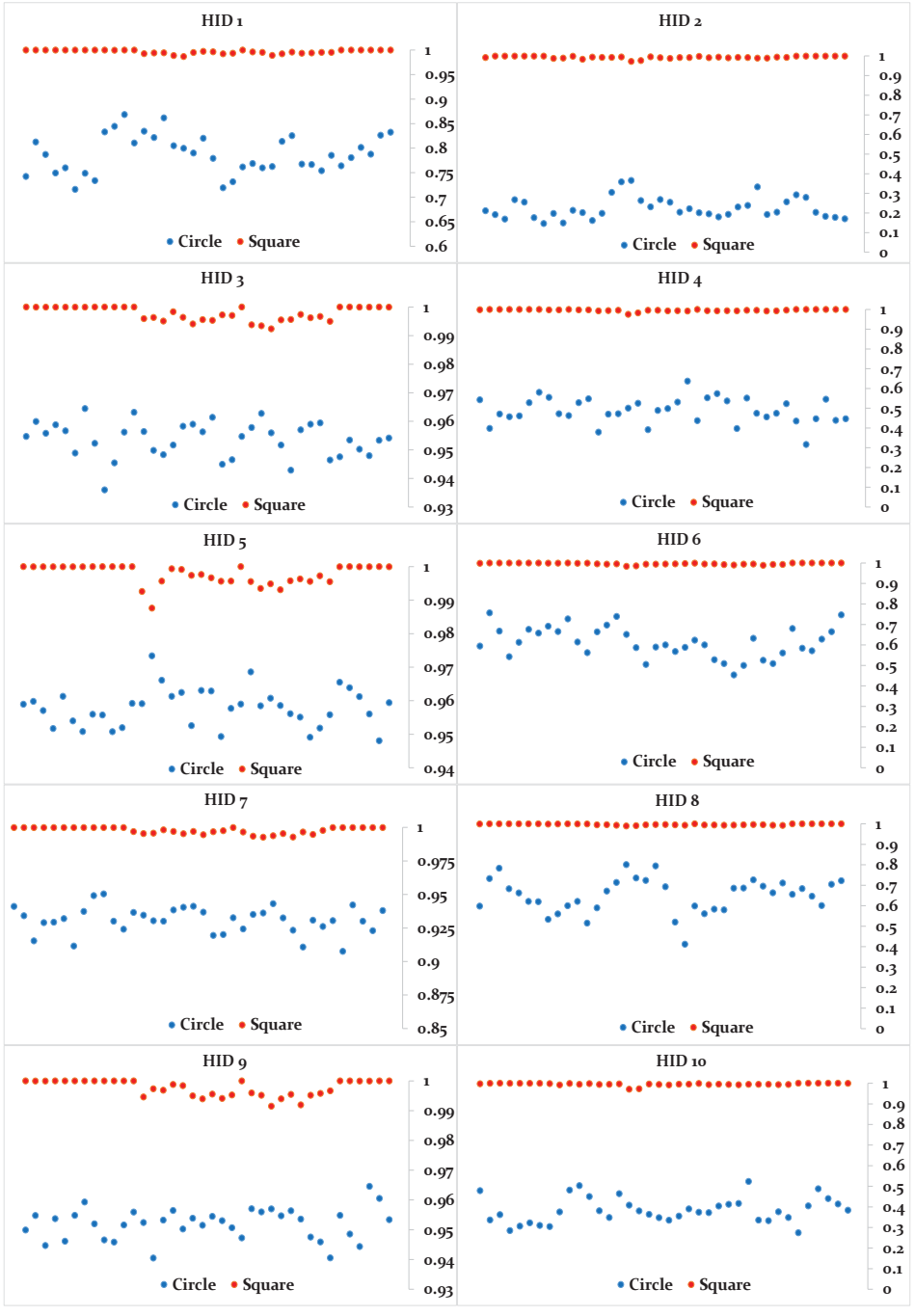

Figure 3. Hidden Activation Analysis (HAA). Experiment involving 76 images (38 circles with 8 pixels diameter and 38 squares with 15 pixels edge). These experiments are summarized in Table 4 (see row with italics font). This figure plots the value of the sigmoidal output of each of the 10 hidden neurons

(each plot is for a specific neuron, such as HID 1 – the first) for all the 76 images. Along the x direction, each pair of dots (red and blue) represent the two shapes located at a given position and given a specific size in the digital grid. Hence, encoding of position, size and shape of the objects is successfully determined by each of the neurons of the hidden layer because of the clear separation between red dots (square shape) and blue dots (circular shape) across the horizontal dimension (which represents the variables position and size).

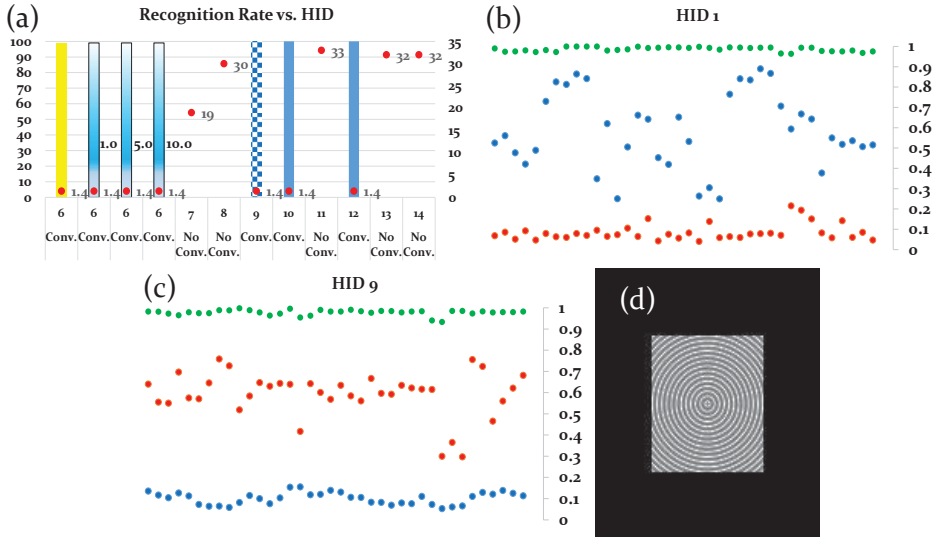

Figure 4. Ablation, comparison with competitive method, and noise study. The pattern recognition task is the classification of the three types of shapes, circles, squares, and rectangles, with varying size and position in the digital grid. The chart in (a) shows that the competitive method with 6 neurons in the hidden layer achieves 99% accuracy after the learning process (see yellow bar). The TLP proposed in this paper achieves 100% accuracy with 9, 10, and 12 neurons in the hidden layer. The left Y axis shows the accuracy values. The right Y axis shows the residual error of fit which is also numerically displayed by the numbers in the chart (e.g. 1.4 was the error obtained when the learning process converged (Conv.). The charts in (b), and in (c), show HAA of the TLP indicated by the bar with pattern fill when TLP had 9 neurons in the hidden layer. The charts in (b) and in (c) show the sigmoidal activation values in the neurons of the hidden layer for the square (green), the circle (blue), and the rectangle (red). The meaning of the HAA is to reveal that the three classes pattern recognition task is solved by neurons 1 and 9 in the hidden layer. This solution could not be found by the competitive method because none of the 6 neurons in the hidden layer was able to separate the three classes. On the other hand, the competitive method was able to achieve 99% accuracy when data was contaminated by noise. These experiments are illustrated by the three bars with gradient fill in (a). The three sessions used three different noise level contaminations, all additive to the pixel brightness of 1.0, 5.0, and 10.0 pixel brightness values. The image in (d) shows the noise pattern when the additive value was 1.0 pixel brightness units. The other patterns were topographically the same but with

higher noise contamination. So, the competitive method shows to be superior to the traditional three-layer perceptron in handling noise in the dataset.

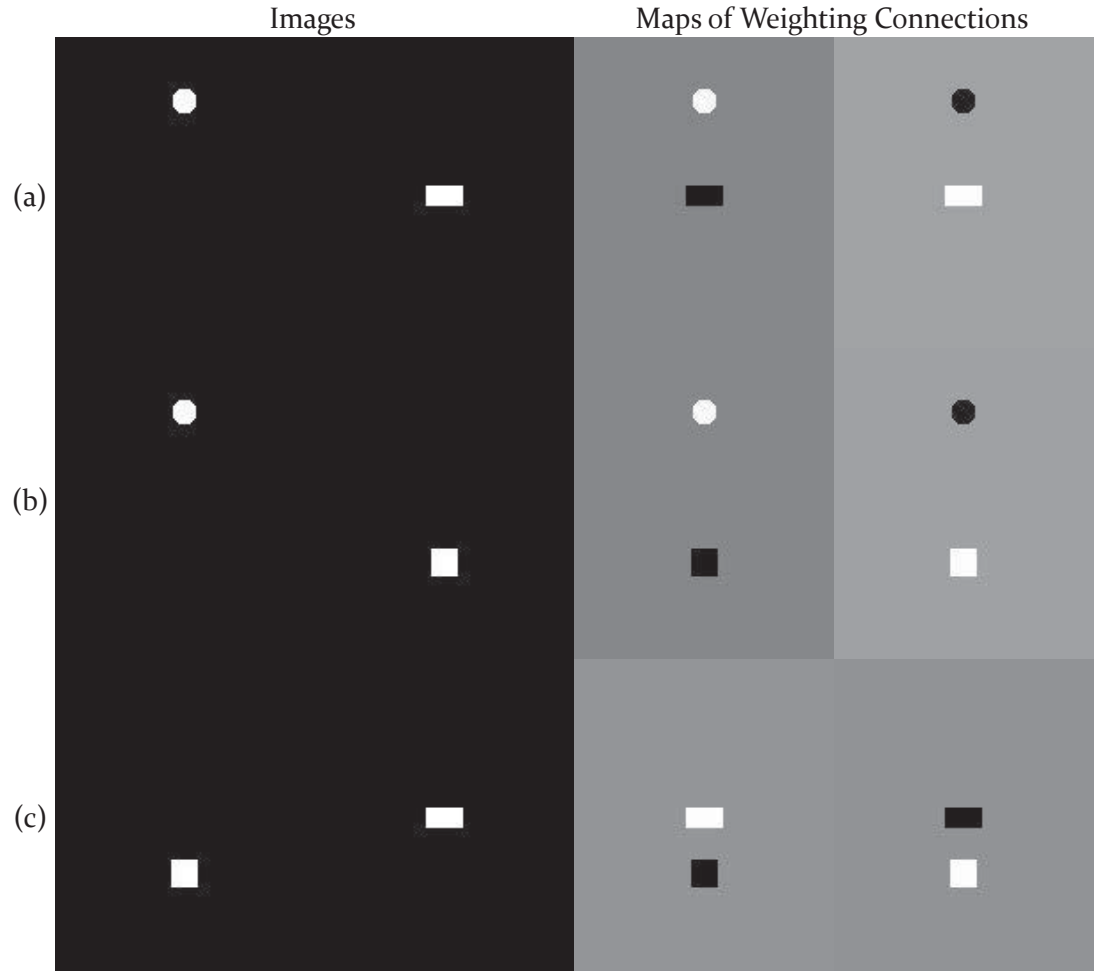

Figure 5. Shape difference encoding by the perceptrons. The images are shown on the two left-most columns. The image maps of the weighting connections are shown on the two right-most columns. Each image map shows the weighting connections obtained after the learning process consisting of 300 iterations using 0.001 as learning rate and 0.005 as smoothing factor. Each image map was calculated by two perceptrons, one used to discern the shape (see white) and the other used one to negate the other shape (see black). Hence, each image map is relevant to a couple of perceptrons acting by binary behavior. The pattern recognition tasks are 1. Circle vs. Rectangle (see (a)), 2. Circle vs. Square (see (b)), and 3. Square vs. Rectangle (see (c)). The binary behavior of the image maps of weighting connections confirms the ability of the perceptrons to encode both shape and position of these three patter recognition tasks.

This characteristic of CNNs is due to their ability to use feature extraction filters that code translational invariant features. These filters are implemented as hidden layers and it is possible to optimize their parameters using gradient-based methods (Szarvas et al., 2005). However, in some cases, one disadvantage of this coding approach is that the CNNs pooling architecture captures only translational invariance (Le et al., 2010). Hence, it follows that ANNs can be trained to detect their own translational and rotational invariances from non-labelled data (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Le et al., 2010). Recent reviews of CNNs architectures and optimization strategies are provided in Cong & Zhou (2023), and in Zhao et al. (2024).

The Contribution

A study of two-layer networks and three-layer networks showed in the year 1987 that two-layer networks trained with backpropagation provide good performance in forming both convex and disjoint decision regions, however, the three-layer network can form complex decision regions (Huang & Lippmann, 1987). The role of the present research work is to provide evidence obtained by HAA that the hidden layer of the three-layer perceptron can successfully encode the variables of a pattern recognition problem in such level of details that allow the solution of the problem.

In this work the pattern recognition problem is the classification of geometrical shapes, which is based on encoding the position and the size in the digital grid. This work is relevant to deep learning because it demonstrates the ability of the hidden layer of the neural network to encode both location and size of the shapes placed in the Cartesian digital grid.

CONCLUSION

This paper presents evidence obtained by HAA in support of the revitalization of the three-layer perceptron used for geometrical shapes classification tasks. More specifically, this research shows that the hidden layer alone can successfully encode the three variables: shape, size and position. This evidence is relevant to more complex structures like convolutional neural networks used nowadays in deep learning. Despite the evolution brought by convolutional neural networks, this piece of research shows that the simple hidden layer of the three-layer perceptron can successfully separate classes within the context of pattern recognition problems that would be solved by deep learning structures. Hence, the conclusion of this work is a message to use in deep learning. That is, to look closely at the level of separation between classes at each layer of the neural network architecture, because, after the learning process, each layer might at least in principle able to achieve separation of the classes. In conclusion, through its ability to encode shape, size and position, the hidden layer of the three-layer perceptron can solve the pattern recognition task at hand.

STATEMENTS AND DECLARATIONS

Funding - This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflicts of Interest Declaration - The author declares to have no conflicts of interest.
Author contributions - The author contributed to the study conception and design, material preparation, software, data collection and analysis, manuscript editing and review.
Data Availability - The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request. The software is available on the internet for free download at https://github.com/cxc2728/artificial-neural-networks

REFERENCES

Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional Neural Networks for Speech Recognition. IEEE/ACM Transactions on audio, speech, and language processing, 22(10): 1533-1545.

Al-Mahasneh, A. J., Anavatti, S. G., & Garratt, M. A. (2017). The Development of Neural Networks Applications from Perceptron to Deep Learning. In: International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA) (pp. 1-6). IEEE.

Bao, C., Pu, Y., & Zhang, Y. (2018). Fractional-Order Deep Backpropagation Neural Network. Computational intelligence and neuroscience, (1): 7361628.

Baum, E., & Haussler, D. (1989). What Size Net Gives Valid Generalization? Neural Computation, 1, 151-160.

Bezdek, J. C. (1992). On the Relationship between Neural Networks, Pattern Recognition and Intelligence. International Journal of Approximate Reasoning, 6(2): 85-107.

Billings, S. A., Jamaluddin, H. B., & Chen, S. (1991). A Comparison of the Backpropagation and Recursive Prediction Error Algorithms for Training Neural Networks. Mechanical Systems and Signal Processing, 5(3): 233-255

Cecotti, H., & Graeser, A. (2008). Convolutional Neural Network with Embedded Fourier Transform for EEG Classification. In: 19th International Conference on Pattern Recognition (pp. 1-4). IEEE.

Chen, T., & Zhong, S. (2009). Privacy-Preserving Backpropagation Neural Network Learning. IEEE Transactions on Neural Networks, 20(10): 1554-1564.

Ciresan, D., Meier, U., Masci, J., Gambardella, L.M. and Schmidhuber, J. (2011). Flexible, High Performance Convolutional Neural Networks for Image Classification. Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Spain, 1237-1242.

Cong, S., & Zhou, Y. (2023). A Review of Convolutional Neural Network Architectures and Their Optimizations. Artificial Intelligence Review, 56(3): 1905-1969.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 2(4): 303-314.

Dampfhoffer, M., Mesquida, T., Valentian, A., & Anghel, L. (2023). Backpropagation-based Learning Techniques for Deep Spiking Neural Networks: A survey. IEEE Transactions on Neural Networks and Learning Systems.

De Villiers, J., & Barnard, E. (1993). Backpropagation Neural Nets with One and Two Hidden Layers. IEEE Transactions on Neural Networks, 4(1): 136-141.

Elbaz, G., Avraham, T. & Fischer, A. (2017). 3D point cloud registration for localization using a deep neural network auto-encoder. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4631-4640).

Fasel, B. (2002). Robust Face Analysis Using Convolutional Neural Networks. In: 2002 International Conference on Pattern Recognition (Vol. 2, pp. 40-43). IEEE.

Fekri-Ershad, S., & Alsaffar, M. F. (2023). Developing a tuned three-layer perceptron fed with trained deep convolutional neural networks for cervical cancer diagnosis. Diagnostics, 13(4): 686.

Frasconi, P., Gori, M., & Tesi, A. (1993). Successes and Failures of Backpropagation: A Theoretical Investigation. Progress in Neural Networks, 5, Intellect Ltd.

Günther, F., & Fritsch, S. (2010). Neuralnet: Training of Neural Networks. The R Journal, 2(1): 30.

Hagan, M. T., & Demuth, H. B. (1999). Neural Networks for Control. In: Proceedings of the 1999 American Control Conference (cat. No. 99CH36251) (Vol. 3, 1642-1656). IEEE.

Hecht-Nielsen, R. (1992). Theory of the Backpropagation Neural Network. In: Neural Networks for Perception (pp. 65-93). Academic Press.

Hinton, G.E., Osindero, S., & Teh, Y.W. (2006). A Fast Learning Algorithm for Deep Belief Nets, Neural Computation. 18(7): 1527-1554.

Hinton, G.E., & Salakhutdinov, R.R. (2006). Reducing the Dimensionality of Data with Neural Networks. Science, 313(5786): 504-507.

Hornik, K., Stinchcombe, M. & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2(5): 359-366.

Huang, W., & Lippmann, R. P. (1987). Neural Net and Traditional Classifiers. In: Neural Information Processing Systems.

Islam, M.A., Jia, S. & Bruce, N.D. (2020). How much Position Information Do Convolutional Neural Networks Encode? In International Conference on Learning Representations.

Jin, R., Wu, M., Wu, K., Gao, K., Chen, Z. & Li, X. (2022). Position encoding based convolutional neural networks for machine remaining useful life prediction. IEEE/CAA Journal of Automatica Sinica, 9(8): pp.1427-1439.

Kalogirou, S. A. (2000). Applications of Artificial Neural-Networks for Energy Systems. Applied energy, 67(1-2): 17-35.

Khotanzad, A., & Chung, C. (1998). Application of Multi-Layer Perceptron Neural Networks to Vision Problems. Neural Computing & Applications, 7: 249-259.

Klemmer, K., Safir, N.S. & Neill, D.B. (2023). Positional encoder graph neural networks for geographic data. In International conference on artificial intelligence and statistics (pp. 1379-1389), PMLR.

Kleyko, D., Rosato, A., Frady, E.P., Panella, M., & Sommer, F.T. (2023). Perceptron Theory Can Predict the Accuracy of Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, 35(7): 9885-9899.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in neural information processing systems, 25.

Le, Q.V., Ngiam, J., Chen, Z., Chia, D., Wei Koh, P., & Ng, A.Y. (2010). Tiled Convolutional Neural Networks. In: Advances in Neural Information Processing Systems, Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., & Culotta, A., Eds. Vol. 23.

Li, Y., Si, S., Li, G., Hsieh, C.J. & Bengio, S. (2021). Learnable Fourier features for multi-dimensional spatial positional encoding. Advances in Neural Information Processing Systems, 34: 15816-15829.

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., & Hinton, G. (2020). Backpropagation and the Brain. Nature Reviews Neuroscience, 21(6): 335-346.

Liu, B., Wang, M., Foroosh, H., Tappen, M., & Pensky, M. (2015). Sparse Convolutional Neural Networks. Proceedings of the IEEE conference on computer vision and pattern recognition, Boston, MA, U.S.A., 806-814.

Lv, C., Xing, Y., Zhang, J., Na, X., Li, Y., Liu, T., Cao, D., & Wang, F. Y. (2018). Levenberg–Marquardt Backpropagation Training of Multilayer Neural Networks for State Estimation of a Safety-Critical Cyber-Physical system. IEEE Transactions on Industrial Informatics, 14(8): 3436-3446.

Magoulas, G. D., Vrahatis, M. N., & Androulakis, G. S. (1997). Effective Backpropagation Training with Variable Stepsize. Neural Networks, 10(1): 69-82.

Mai, G., Janowicz, K., Hu, Y., Gao, S., Yan, B., Zhu, R., Cai, L. & Lao, N., (2022). A review of location encoding for GeoAI: methods and applications. International Journal of Geographical Information Science, 36(4): pp.639-673.

Macukow, B. (2016). Neural Networks–State of Art, Brief History, Basic Models and Architecture. In: Computer Information Systems and Industrial Management: 15th IFIP TC8 International Conference, CISIM 2016, Vilnius, Lithuania, Proceedings 15 (pp. 3-14). Springer International Publishing.

Marmarelis, V. Z., & Zhao, X. (1997). Volterra Models and Three-Layer Perceptrons. IEEE Transactions on Neural Networks, 8(6): 1421-1433.

Min, E., Guo, X., Liu, Q., Zhang, G., Cui, J., & Long, J. (2018). A survey of clustering with deep learning: From the perspective of network architecture. IEEE Access, 6: 39501-39514.

Mirus, F., Blouw, P., Stewart, T.C. & Conradt, J. (2019). An investigation of vehicle behavior prediction using a vector power representation to encode spatial positions of multiple objects and neural networks. Frontiers in Neurorobotics, 13, p.84.

Mukundan, A., Tolias, G. & Chum, O. (2019). Explicit spatial encoding for deep local descriptors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9394-9403).

Nasse, F., Thurau, C., & Fink, G. A. (2009). Face Detection Using GPU-based Convolutional Neural Networks. In: Computer Analysis of Images and Patterns: 13th International Conference, CAIP 2009, Münster, Germany. Proceedings 13 (pp. 83-90). Springer Berlin Heidelberg. Newton, I. (1736). Method of Fluxion, J. Colson Ed., London, U.K.

Newton, I. (1736). Method of Fluxion, J. Colson Ed., London, U.K.

Park, Y. S., & Lek, S. (2016). Artificial Neural Networks: Multilayer Perceptron for Ecological Modeling. In: Developments in environmental modelling (Vol. 28, 123-140). Elsevier.

Pineda, F. (1987). Generalization of Back Propagation to Recurrent and Higher Order Neural Networks. In: Neural Information Processing Systems.

Rana, A., Rawat, A. S., Bijalwan, A., & Bahuguna, H. (2018). Application of Multilayer (Perceptron) Artificial Neural Network in the Diagnosis System: A Systematic Review. In: International conference on research in intelligent and computing in engineering (RICE) (pp. 1-6). IEEE.

Rojas, R., & Rojas, R. (1996). The Backpropagation Algorithm. Neural Networks: A systematic Introduction, pp. 149-182.

Rosemblatt, F. (1957). The Perceptron – A perceiving and Recognizing Automation. Report 85-460-1. Cornell Aeronautical Laboratory, U.S.A.

Rumelhart, D. E., Hinton, G.E., & Williams, R.J. (1985). Learning Internal Representations by Error Propagation. ICS Report 8506, Institute of Cognitive Science, University of California at San Diego, U.S.A.

Rußwurm, M., Klemmer, K., Rolf, E., Zbinden, R. & Tuia, D. (2024). Geographic Location Encoding with Spherical Harmonics and Sinusoidal Representation Networks. The Twelfth International Conference on Learning Representations - Vienna, Austria.

Sainath, T.N., Kingsbury, B., Mohamed, A-R., Dahl, G.E., Saon, G., Soltau, H., Beran, T., Aravkin, A.Y., & Ramabhadran, B. (2013). Improvements to Deep Convolutional Neural Networks for LVCSR. In: 2013 IEEE workshop on automatic speech recognition and understanding, IEEE, 315-320.

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. ICDAR, 3(2003).

Sontag, E. D. (1993). Neural Networks for Control. In Essays on Control: Perspectives in the Theory and its Applications (pp. 339-380). Boston, MA, Birkhäuser Boston, U.S.A.

Szarvas, M., Yoshizawa, A., Yamamoto, M., & Ogata, J. (2005). Pedestrian Detection with Convolutional Neural Networks. IEEE Proceedings. Intelligent Vehicles Symposium, 2005. (224-229). IEEE.

Thimm, G., Moerland, P., & Fiesler, E. (1996). The Interchangeability of Learning Rate and Gain in Backpropagation Neural Networks. Neural Computation, 8(2): 451-460.

Tivive, F. H. C., & Bouzerdoum, A. (2004). A Face Detection System Using Shunting Inhibitory Convolutional Neural Networks. IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541) (Vol. 4, pp. 2571-2575). IEEE.

Uzair, M. & Jamil, N. (2020). Effects of Hidden Layers on the Efficiency of Neural Networks. IEEE 23rd International Multitopic Conference (INMIC), Bahawalpur, Pakistan, 1-6.

Wang, H., Huang, L., Du, C., Li, D., Wang, B. & He, H. (2020). Neural encoding for human visual cortex with deep neural networks learning "what" and "where". IEEE Transactions on Cognitive and Developmental Systems, 13(4): 827-840.

Werbos, P.J. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. Thesis, Applied Mathematics, Harvard University.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10): 1550-1560.

Widrow, B., Winter, R. G., & Baxter, R. A. (1988). Layered Neural Nets for Pattern Recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing, 36(7): 1109-1118.

Widrow, B., & Lehr, M. A. (1990). 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. Proceedings of the IEEE, 78(9): 1415-1442.

Wythoff, B. J. (1993). Backpropagation neural networks: a tutorial. Chemometrics and Intelligent Laboratory Systems, 18(2): 115-155.

Xu, Y., Piao, Z. & Gao, S. (2018). Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5275-5284).

Yao, L., Liu, T., Qin, J., Lu, N. & Zhou, C. (2021). Tree counting with high spatial-resolution satellite imagery based on deep neural networks. Ecological Indicators, 125, p.107591.

Yao, X. (1999). Evolving Artificial Neural Networks. Proceedings of the IEEE, 87(9): 1423-1447.

Yegnanarayana, B. (1994). Artificial Neural Networks for Pattern Recognition. Sadhana, 19, 189-238.

Zajmi, L., Ahmed, F. Y., & Jaharadak, A. A. (2018). Concepts, Methods, and Performances of Particle Swarm Optimization, Backpropagation, and Neural Networks. Applied Computational Intelligence and Soft Computing, (1): 9547212.

Zhang, J., & Qu, S. (2021). Optimization of Backpropagation Neural Network Under the Adaptive Genetic Algorithm. Complexity, (1): 1718234.

Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. (2024). A Review of Convolutional Neural Networks in Computer Vision. Artificial Intelligence Review, 57(4): 99.

Zhou, X. (2018). Understanding the Convolutional Neural Networks with Gradient Descent and Backpropagation. In Journal of Physics: Conference Series (Vol. 1004, p. 012028). IOP Publishing.