VFA-driven Hierarchical Temporal Memory Input for Object Categorization [*]

Csapó Ádám[†], Péter Baranyi

Department of Telecommunication & Media Informatics
Budapest University of Technology and Economics
H-1117 Budapest, Magyar Tudósok körútja 2., Hungary
csapo.adam@gmail.com

Abstract

Results in visual psychology have shown that the location and statistics of nodes, endpoints and corners carry essential and sufficient information for object recognition. In this paper, we present a method for object categorization which relies on the combination of the Visual Feature Array model and Hierarchical Temporal Memories. Experimental results show that even without taking into consideration statistics other than the spatial distribution of nodes, two categories can be told apart with a success rate of about 95%. The same results could not be achieved by simply taking into account grayscale pixel values. Efforts were also made to generalize the above results to a categorization task among 10 different categories.

1 Introduction

Object categorization and recognition has proven to be a difficult task in artificial intelligence for several decades. With the recent emergence of biologically inspired soft-computing methods, promising results in specialized application domains are more and more common. In this paper, we provide an experimental demonstration of Biederman's conjecture in his theory of recognition by components (RBC), according to which the statistics of nodes, endpoints and corners carry essential and sufficient information for object recognition [1]. In order to show the relevance of this conjecture, the VFA model is combined with the HTM theory in an object categorization task, and the performance of the network is evaluated given a wide variety of different input data structures. The first section of this paper consists of a brief introduction, in which we restate the formal definition of the VFA model, as well as present its node-filtering applications. This will be followed by a presentation of the HTM theory for size- and orientation-invariant object representation. Finally, we give a detailed case study in which a hierarchical temporal memory is used to distinguish between two, as well as several object categories.

---

[*]This research was supported by the János Bolyai Postdoctoral Scholarship
[†]Corresponding author

2 Theoretical background

2.1 The VFA model

2.1.1 The VFA data structure

The Visual Feature Array (VFA) was first proposed by the authors in [4]. It is an orthogonal arrangement of computational units with response characteristics tuned to different visual features, as is the case with neurons - simple cells - found in the primary visual cortex. The VFA contains values that represent the output of computational elements corresponding to a certain VFA location. $VFA(x, y, z)$ with dimensions x x y x z is generated from a two-dimensional u x v sized image represented by the function $f(u, v)$, where

$$f(u, v) \in \mathbb{N}, [0..255] \tag{1}$$

$$(u, v) \in \mathbb{N}^2, [0..U]x[0..V] \tag{2}$$

$$VFA(x, y, z) \in \mathbb{N}, [0..255] \tag{3}$$

$$(x, y, z) \in \mathbb{N}^3, [0..X]x[0..Y]x[0..Z] \tag{4}$$

$VFA(x, y, z)$ is the output of the computational unit that corresponds to the x and y image coordinates, and is sensitive to some image feature represented by z. The simplest version of the VFA structure, used in this paper, can be pictured as a collection of layers, stacked up one behind the other, in which each layer (a 2-dimensional array) represents iso-orientation contour elements. Neighboring layers contain contour elements with neighboring orientations. The VFA of a triangle can be seen in figure 1.

2.1.2 Lateral inhibition in the VFA model

Because filters used in biological neural networks are oftentimes fuzzy and are not not precise, values in the VFA are usually not crisp, but blurred between neighboring layers. To alleviate this problem, in a way similar to biological mechanisms, lateral inhibition can be introduced between compuational elements of each layer and those of neighboring layers. In general, a 3-dimensional mask array, $M(x, y, z)$, is defined:

$$M(x, y, z) = \begin{cases} 1 \text{ if } surr(VFA, x, y, z) = \max\limits_{i \in \{-1,0,1\}} surr(VFA, x, y, z+i) \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

Where $surr(VFA, x, y, z)$ is a function that yields a neighborhood of point (x, y) on the image, the shape of which can depend on the feature represented by the layer in question. The original VFA can then be masked with $M(x, y, z)$, thus $VFA_{inhibited}$ is obtained:

$$VFA_{inhibited}(x, y, z) = VFA(x, y, z) * M(x, y, z) \tag{6}$$

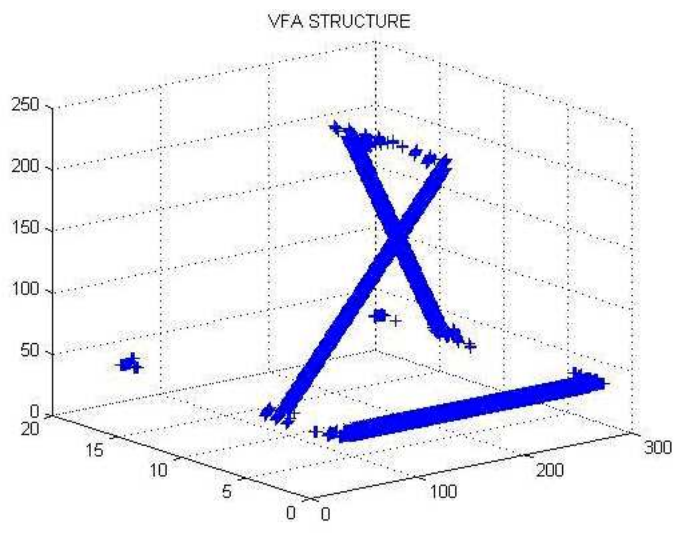

Figure 1: VFA of a triangle.

2.1.3 Node-filtering applications in the VFA model

The data representation used by the VFA model makes way for the fast detection of certain visual features with a very low computational load. One example is the detection of line segment intersections.

Let V^{18} be a vector containing 18 elements. Values of V^{18} correspond to the output of simple cells belonging to a given point of the image. This way, 180 degrees of orientation are covered in 10-degree increments.

$$V_{xy}(i) = VFA(x, y, \phi_i)$$
$$\phi_i = 1, 2, 3, ...18$$

Intersections can then be detected as follows:

$$N(x, y) = signum\{\sum_{j=1}^{18} V_{xy}(j) - c\}$$

where c is a threshold value. When a binary VFA is used, typically $c = 2$, and the meaning of the condition is that a point (x, y) contains a line intersection if it contains the intersection of two or more line segments.

2.2 The HTM theory

The theory of hierarchical temporal memories (HTMs) was originally proposed in [3],[2]. It is a hierarchical cortical model that accounts for invariant object representations, which are learned using time as a supervisor. The basic idea behind the HTM theory is that although objects can be seen from very different viewpoints - thus generating very different patterns on the retina, the cortex still manages to attribute such a variety of patterns to the same cause when necessary, because the patterns are seen one after the other, through a very short period of time. Processing units at each level of an HTM network categorize incoming patterns into different groups, and yield common output for input patterns belonging to the same group. Each node in an HTM network is comprised of two computational units: the spatial pooler and the temporal pooler. Input is fed to the spatial pooler, whose task it is to discretize the input space. Input vectors are stored if and only if the Euclidean distance between themselves and all the other previously stored vectors are sufficiently large (this can be controlled by an input parameter, maxDistance). After the spatial pooler stores as many input patterns as possible, the temporal pooler learns to attribute one common code to all stored input patterns that represent the same cause. The decision as to whether two input patterns are generated by the same cause (i.e., the same object in the world) is made based on information such as the relative frequency of the two input patterns occurring one after the other. In such a way, the HTM network learns to attribute one common 'name' to patterns caused by the same object. In its current version, the HTM platform developed by Numenta, Inc., uses only feedforward connections. Numenta promises to enhance the current version with lateral and feedback connections soon, which will allow for the incorporation of attention mechanisms, as well as predictive behavior.

3 Methods

In order to test HTM performance with input obtained from the VFA model, we used images from the Caltech 101 database. This is a collection of images from 101 categories.

During experiments, we used 35 training images, and 20 test images from each of the categories. Images were first padded with fringes containing only zeroes, so that each image used for training and testing would be of the same size (512 x 512 pixels in this case). The VFA of each image thus obtained was then calculated. This was followed by a summation along the third dimension of the VFA, and normalization to obatain values between 0 and 1. Each map could then be pictured as a three-dimensional waveform (the first two dimensions are the dimensions of the image, and the third is defined by the values at each pixel), as shown in figure 2. Finally, every disjoint 8-by-8 pixel region was substituted by the average of its pixel values. This yielded 64-by-64 pixel images, which, when read pixel-by-pixel in a topological manner, produced row vectors with 4096 elements. Finally, every training map was shifted ten pixels, one pixel at a time in 4 directions (north, south, west and east). This shifting of images was performed in order to obtain a higher level of invariance when teaching the network. In this way, nodes in the network would learn

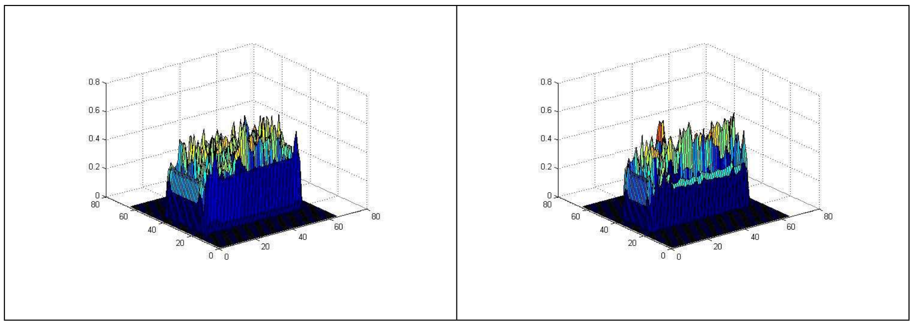

Figure 2: Two node maps of different instances of the same category. Such node maps can be thought of as three-dimensional waveforms.

that two shifted instances of the same image belong to the same cause.

In total, $n * (35 + 40 * 35) = 1435n$ training images and $20n$ test images were used altogether for n categories. The HTM network used for experiments was created using the Numenta platform, developed by Numenta, Inc. This is a programming framework, which has a Python API that can be used to set up custom experiments. The HTM network used had three layers, and processing units in each layer were arranged in a topographic manner. This means that neighboring areas in the image were covered by neighboring processing units. Processing units higher up in the hierarchy were also responsible for larger areas in the visual field. Figure 3 shows the topography of the HTM network. Figure 4 shows the order in which pixels were read out of the image in order to convert two-dimensional data into one-dimensional vectors, while ensuring that neighboring nodes would see regions that are neighboring in the image.

4 Experiments and results

4.1 Distinguishing between two categories

For the sake of simplicity, we first chose to use two categories only - airplanes and dolphins, and teach the HTM network to distinguish between the two. The choice of these two categories seemed justified because whereas for humans, airplanes and dolphins form very different categories, this is not so for object categorization software, which perceives both categories as having sleek bodies, for instance.

Two experiments were conducted. In the first experiment, greyscale pixels were supplied into the HTM network. This experiment was used as a reference, in order to verify that the use of node structures is more efficient. In the second experiement, the 3-dimensional waveforms described above were used.

In the first case, when grayscale images were supplied to the HTM network as is, a 55% hit rate was the best result that was achieved, even when varying network parameters to a large extent. This shows that grayscale pixel values, by themselves,

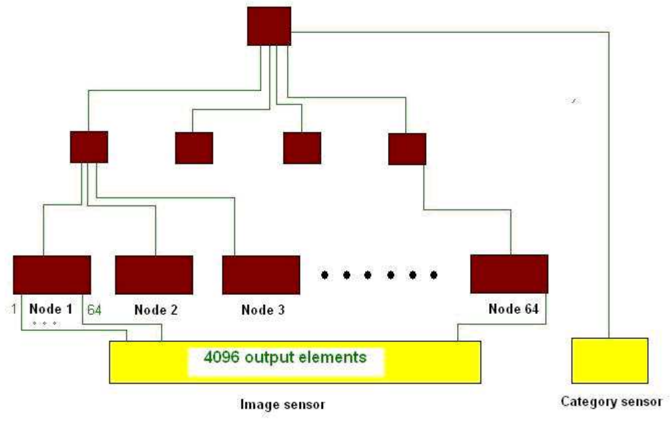

Figure 3: The HTM configuration used for training. Level 1 has 64 nodes, level 2 has 4 nodes, and the top level has a single node that uses supervised learning with the aid of a category sensor. Two-dimensional data was extracted from the node map in a topological manner.

1	3	9	11	33	35	41	43
2	4	10	12	34	36	42	44
5	7	13	15	37	39	45	47
6	8	14	16	38	40	46	48
17	19	25	27	49	51	57	59
18	20	26	28	50	52	58	60
21	23	29	31	53	55	61	63
22	24	30	32	54	56	62	64

Figure 4: The order in which pixels were read out of the image. The above scheme was continued recursively.

are not sufficient to characterize a whole category of objects. On the other hand, using the data set described in section 3, a 97.77% success rate was achieved on training images, and a 95% success rate was achieved on testing images.

The parameters of the HTM network used were as follows:

Layer 1:

- levelSize = 64
- pooler algorithm: gaussian; sigma = 0.4

- maxDistance = 5
- maxGroupSize = 1435
- grouper algorithm: sumProp

Layer 2:

- levelSize = 4
- pooler algorithm: product
- maxGroupSize = 1435
- grouper algorithm: sumProp

Layer 3:

- levelSize = 1
- pooler algorithm: product
- mapper algorithm: sumProp

$maxDistance$ on the first level defines the minimum value the squares of the Euclidean distances between an input (x) and all the previously memorized inputs (y_i) have to take in order for x to be considered novel. $maxGroupSize$ sets an upper limit for the number of quantized inputs that can form a group in the temporal pooler. The pooler algorithm used by the spatial pooler of higher levels is $product$, which means that the belief that an input during inference is similar to a given vector (previously memorized by the spatial pooler) is calculated as follows:

$$belief_i = \prod_{j=1}^{nchildren} y_i[child_j] * x[child_j] \tag{7}$$

where $nchildren$ is the number of children the node has, x is the input vector, y_i are the vectors previously stored by the spatial pooler, and $a[child_n]$ is the part of vector a that is received from the n^{th} child.

This approach tends to be less forgiving for differences between input and stored vectors than the *dot* algorithm, which uses the scalar product of the supports for the two vectors instead and is therefore additive:

$$belief_i = \sum_{j=1}^{nchildren} y_i[child_j] * x[child_j] = x * y_i \tag{8}$$

Finally, the temporal pooler at each level uses the $sumProp$ algorithm, which takes the highest belief from each group to generate a distribution of beliefs over temporal groups during inference.

Figure 5 shows the patterns learned by a typical first-level node. The frequency rating below each group represents the number of times the pattern was encountered. The stability of a group shows how often members of the same group were encountered one after the other. It is worth noting that in the case of this node, the number of coincidences encountered and the number of groups formed are equal in all cases. This means that all inputs to this node were pooled into 1 of 4 categories. Indeed, with $maxDistance = 5$, of 64 sensors per node, in order for an input x to be pooled into its own category,

$$5 < \sum_{i=1}^{64} (x_i - y_i)^2 \tag{9}$$

for all previous input categories y. This means, that on average:

$$5/64 = 0.078 < (x_i - y_i)^2 \tag{10}$$

which implies that on average, corresponding coordinates of x and y hold very different values:

$$0.2795 < |x_i - y_i| \tag{11}$$

Figures 6 and 7 show testing images which were categorized correctly and incorrectly, respectively.

The difference in performance between the two approaches is empirically clear, because even objects of the same category can have very different contrasts in images, and thus the effect of noise is much worse in the case where grayscale values are used than in the case where node statistics are used - due to the lateral filters used by the VFA model.

4.2 Scaling up the network to differentiate among more categories

A second experiment was conducted to see whether the highly acceptable performance encountered in the categorization task between two categories could be scaled up to more categories. We chose examples from 10 different categories from the same Caltech 101 database. Thus we obtained $1435 * 10 = 14350$ training images, and $20 * 10 = 200$ test images.

An empirical approach was used to obtain an estimate for parameters that would cause the network to perform similarly well. $transitionMemory$ was incremented to 3 on each layer (this means that when the temporal pooler keeps track of which

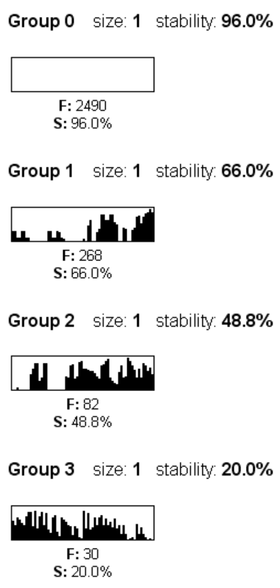

Figure 5: Groups of input vectors formed by a level 1 node.

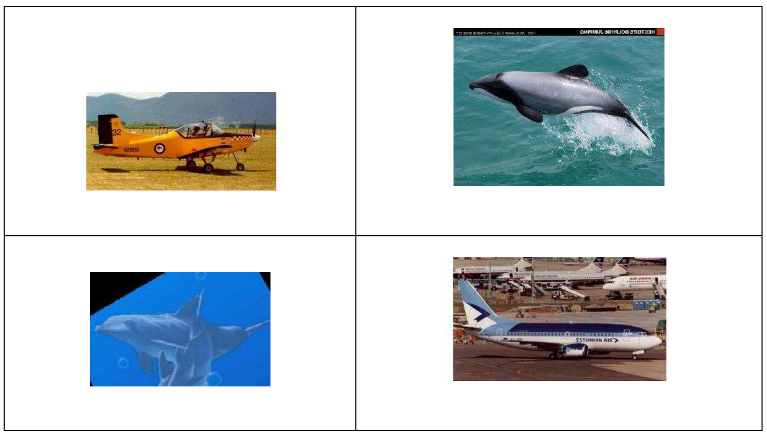

Figure 6: Images that were successfully categorized.

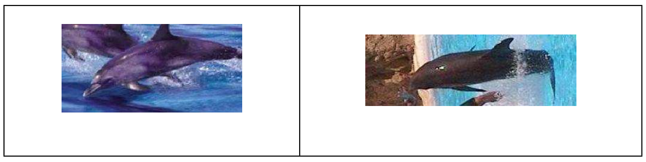

Figure 7: Images that were miscategorized.

inputs follow which, a queue of length 3, with decayed memory is used). When estimating the value for $maxDistance$ in the first level, we tried to ensure that 5 times the number of groups could be formed in the temporal pooler of level 1 as before. The formation of groups depends on several parameters, but one empirical way to somehow bias this process is by learning more coincidences in the spatial pooling phase. Thus, in order for an input to be considered as novel, we set $maxDistance$ to 1. Obviously - even if our goal is simply to generate 5 times as many quantization points at the first level - this approximation only works if the structure of the testing data truly forms a uniform distribution over the $[0..5]$ interval, that is, on average:

$$\sqrt{(1/64)} = 0.125 < |x_i - y_i| \tag{12}$$

for all x and y vectors which represent the same category. This was not verified for the testing data used.

Results obtained are not as satisfying as in section 4.1; a hit rate of 65% was achieved. While this is still way higher than the 10% reference value, evolutionary or other methods should be developed to find optimal HTM configurations, to see whether or not this deterioration is inevitable.

5 Conclusion

Using a combination of the visual feature array model and the hierarchical temporal memory model for invariant object representation, an object categorization algorithm was obtained. Experimental results demonstrate that:

- Not all types of input to HTMs are equally efficient in object categorization tasks.
- Grayscale pixels do not adequately represent visual structures, however, the spatial arrangements of nodes is less sensitive to noise and can be used as input to object categorization tasks.
- Results for categorization between two categories can be scaled up to 10 categories using a simple empirical approach, and similar results are obtained, albeit some deterioration in performance can be observed. The obtained results suggest that given an optimization technique, higher hit rates might be achieved.

References

[1] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[2] J. Hawkins and S. Blakeslee. *On Intelligence*. Henry Holt Co., 2004.

[3] J. Hawkins and D. George. Hierarchical temporal memory concepts, theory and terminology. *Numenta, Inc. whitepaper*, 2006.

[4] B. Resko, A. Roka, and P. Baranyi. Visual cortex inspired intelligent contour detection. *JACIII*, 10(5):761–768, 2005.