Computing and Informatics, Vol. 31, 2012, 901–919

PAIR-WISE TEMPORAL POOLING METHOD FOR RAPID TRAINING OF THE HTM NETWORKS USED IN COMPUTER VISION APPLICATIONS

Svorad Štolc, Ivan Bajla

Institute of Measurement Science
Slovak Academy of Sciences, Bratislava, Slovakia
&
AIT Austrian Institute of Technology, GmbH
Seibersdorf, Austria
e-mail: Svorad.Stolc@ait.ac.at

Kristián Valentín, Radoslav Škoviera

Institute of Measurement Science
Slovak Academy of Sciences, Bratislava, Slovakia
&
Faculty of Mathematics, Physics, and Informatics
Comenius University in Bratislava, Slovakia

Communicated by Adam Csapo

Abstract. In the paper, several modifications to the conventional learning algorithms of the Hierarchical Temporal Memory (HTM) – a biologically inspired large-scale model of the neocortex by Numenta – have been proposed. Firstly, an alternative spatial pooling method has been introduced, which makes use of a random pattern generator exploiting the Metropolis-Hastings algorithm. The original inference algorithm by Numenta has been reformulated, in order to reduce a number of tunable parameters and to optimize its computational efficiency. The main contribution of the paper consists in the proposal of a novel temporal pooling method – the *pair-wise explorer* – which allows faster and more reliable training of the HTM networks using data without inherent temporal information (e.g., static images). While the conventional temporal pooler trains the HTM network on a finite segment of the smooth Brownian-like random walk across the training images, the proposed method performs training by means of the pairs of patterns randomly sampled (in a special manner) from a virtually infinite smooth random walk. We have conducted a set of experiments with the single-layer HTM network applied to the position, scale, and rotation-invariant recognition of geometric objects. The obtained results provide a clear evidence that the pair-wise method yields significantly faster convergence to the theoretical maximum of the classification accuracy with respect to both the length of the training sequence (defined by the maximum allowed number of updates of the time adjacency matrix – TAM) and the number of training patterns. The advantage of the proposed explorer manifested itself mostly in the lower range of TAM updates where it caused up to 10 % relative accuracy improvement over the conventional method. Therefore we suggest to use the pair-wise explorer, instead of the smooth explorer, always when the HTM network is trained on a set of static images, especially when the exhaustive training is impossible due to the complexity of the given task.

Keywords: Hierarchical temporal memory (HTM), temporal pooler, rapid learning, image explorer, position, scale, and rotation-invariant pattern recognition

Mathematics Subject Classification 2010: 68T05, 68T10, 68T45

1 INTRODUCTION

Hierarchical Temporal Memory (HTM), introduced by [1, 2, 3] and distributed as a free software package *NuPIC* by Numenta, Inc. [4, 5], is a recent development in the field of hierarchical Bayesian networks, which is ambitious in many, yet unresolved artificial intelligence problems. Promising results have been achieved in applications of HTM to various pattern recognition/classification problems in machine vision, voice recognition, etc. [6, 7, 8, 9, 10]. Up to now, most applications in machine vision have only dealt with the situation in which all recognized objects had position and scale within the network's field of view (retina). Application of HTM to problems requiring the position, scale, and rotation-invariant recognition represents another challenging step in a development of its functions. The prerequisite of a successful classifier in this domain, distinguished by extreme variability of a potential input, is to find an appropriate set of object features, which are sufficiently robust against expected transformations, but still provide sufficient specificity. Moreover, it is very important to use an appropriate and effective training algorithm, requiring neither too large training set nor unfeasibly long training process.

In the HTM model, an essential source of invariance are *temporal groups* which unite similar or otherwise related patterns. These groups are formed in course of the training in all network nodes in a phase called a *temporal pooling*. When dealing with static images, a traditional approach to the temporal pooling [4, 5] utilizes pattern sequences, also referred to as *temporal sequences*, which are generated by a smooth traversal (exploration) of the training images. However, this method often suffers from a slow convergence, causing that HTM requires quite long training process to deliver reasonable performance.

In the paper, we propose an alternative approach to the temporal pooling in the context of static image processing, that allows faster and more efficient training of the HTM networks. In order to demonstrate this ability, we conducted a set of experiments accounting for the position, scale, and rotation-invariant recognition of simple geometric objects. We considered objects such as rectangles, triangles, or circles (represented by smooth contours in gray-scale images, see Figure 1) in which the efficiency of both the traditional and proposed temporal pooling methods could be quantitatively compared.

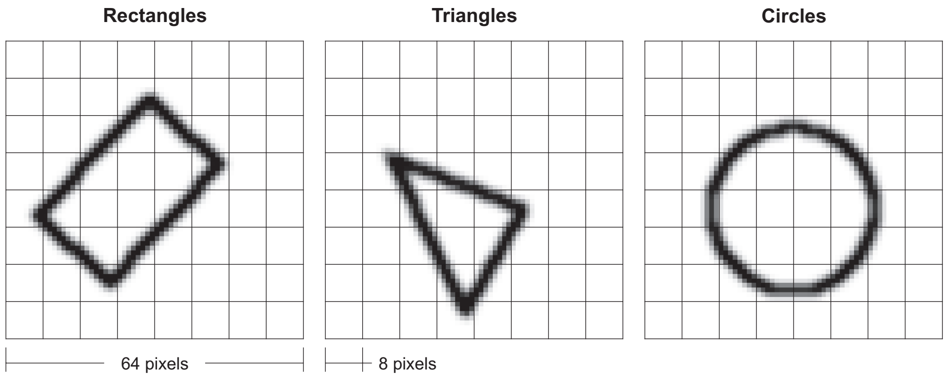


Fig. 1. Examples of simple geometric objects, which were used for testing both the traditional and proposed temporal pooling methods. The objects are presented in the context of a single-layer HTM network, which in this case consisted of 8 × 8 grid of nodes, each receiving input from its own receptive field – a local patch of 8 × 8 pixels. Note that all considered geometric objects were represented only as contours, therefore the gray-scale values (ranging from white = 0 to black = 1) could be seen as local outputs of a smooth edge detector applied to standard gray-scale images.

2 DESCRIPTION OF THE HTM MODEL

As proposed in [1, 2, 4], the HTM network forms a tree-shaped hierarchy of layers consisting of basic operational units called *nodes*. Each HTM node works in two modes – *learning* and *inference*. In the learning phase the node performs two operations – *spatial pooling* and *temporal pooling*. Once these two steps are finished, the node can be switched to the inference mode.

Although, in principle, each node could be trained by its own pattern sequence and consequently could contain a unique learned information, in practice, all nodes at the same hierarchy level are considered as equivalent. Such an assumption comes from the fact that in case of position, scale, and rotation-invariant recognition, any pattern seen by any node in course of the training can later be observed by some other node at the same level; nevertheless, it should be recognized equally well. Therefore it is only natural to treat all nodes within one layer identically regarding their learned information. This, moreover, simplifies the whole training process significantly, as only one node needs to be trained at each network level, while the rest share the same learned information.

2.1 Spatial Pooling

In the course of spatial pooling, input patterns of each node are being quantized into representative clusters. All those clusters are characterized by their quantization centers (one for each cluster), which altogether form a codebook of spatial patterns approximating the input data. In the Numenta implementation of the spatial pooler, usually some kind of a smooth image explorer is used for collecting representative patterns. Their method depends strongly on an appropriate choice of the quantization parameter *maxDistance*, that specifies the radius of the quantization clusters. Such an approach suffers from several weaknesses which were addressed in [11].

In this study, we applied a different spatial pooling method that does not depend on any other parameters but the *codebook size*. This method randomly selects a required number of image patterns of a given size (e.g., 8×8 pixels) from the available training images which are afterward considered as the quantization centers. In order to construct the codebook out of patterns contained in the training images, trying to eliminate presence of irrelevant patterns (e.g., the empty pattern), one needs to define a suitable relative measure of the pattern relevance. Hereinafter, we call this function a *pattern likelihood*. Intuitively, the higher the likelihood of some pattern, the higher should be the probability of its presence in the codebook, and vice versa.

For generating the codebook via given pattern likelihood function, we adopted a well-known Metropolis-Hastings (M-H) algorithm [12, 13]. This algorithm is a Markov chain Monte Carlo method for drawing samples $\mathbf{s}$ from any probability distribution, requiring only a *likelihood function* $L(\mathbf{s})$ proportional to the desired probability density $P(\mathbf{s})$. The algorithm generates a Markov chain of random states (samples), in which acceptance of each new state $\mathbf{s}^{t+1}$ depends only on the previous accepted state $\mathbf{s}^t$ and a random factor α drawn from the uniform distribution $U(0,1)$. Furthermore, the algorithm also requires a *proposal density* $Q(\mathbf{s}'|\mathbf{s}^t)$, which serves for proposing new candidate states $\mathbf{s}'$ with respect to the current state $\mathbf{s}^t$. The proposal is accepted, i.e., $\mathbf{s}^{t+1} = \mathbf{s}'$, if and only if $\alpha < \frac{L(\mathbf{s}')\,Q(\mathbf{s}^t|\mathbf{s}')}{L(\mathbf{s}^t)\,Q(\mathbf{s}'|\mathbf{s}^t)}$. Otherwise, the current state is retained for the next step, i.e., $\mathbf{s}^{t+1} = \mathbf{s}^t$. If the proposal density is a symmetric function, $Q(\mathbf{s}'|\mathbf{s}^t) = Q(\mathbf{s}^t|\mathbf{s}')$, it can be omitted from the calculation, which leads to the simplified acceptance condition $\alpha < L(\mathbf{s}')/L(\mathbf{s}^t)$. The proposal process followed by accepting or rejecting the patterns is repeated until the required number of states is generated.

For the purpose of this study, we applied the M-H algorithm, so that the generated states $\mathbf{s}$ have been identified with the positions within the training images, i.e.,

$\mathbf{s} = (x, y, i)$, where x and y stand for the coordinate of a particular pixel within i^{th} training image. The proposal density $Q(\mathbf{s}'|\mathbf{s}^t)$ has been defined as an independent uniform distribution over all coordinates within the training images that implies the symmetry of this function. Therefore the simplified acceptance condition depending solely on the likelihood function could have been used. Most importantly, we defined the likelihood function as follows:

$$L(\mathbf{s}) = L(x, y, i) = E(X(x, y, i)^2)^k, \tag{1}$$

where $X(x, y, i)$ is a pattern extracted from the coordinate (x, y) of i^{th} training image, $E(\cdot)$ is the arithmetic mean over all elements of pattern $X(\cdot)$, and k is a tunable exponent (in our case, $k = 4$ worked well). It should be noted that Equation (1) is specifically designed for sparse input images containing contour objects, as shown in Figure 1. In such kind of images, each pixel value provides information about the presence or absence of the object's contour at that particular position, which gives in fact the relevance of individual pixels. Therefore, one is entitled to derive the total pattern relevance directly from the absolute pixel values. An example of the codebook generation process by means of the proposed M-H-based method is demonstrated in Figure 2.

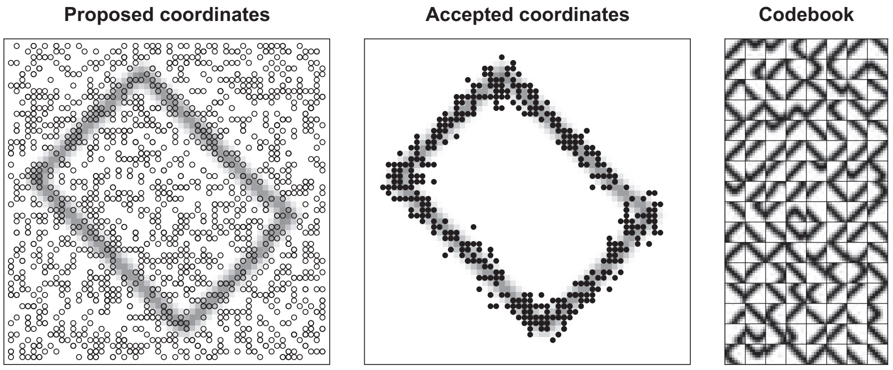


Fig. 2. The figure demonstrates the principle of the codebook generation by means of the M-H algorithm. In each step of the algorithm, a single image coordinate is proposed, drawn randomly from an independent uniform distribution (left). Afterward, some of the proposed coordinates are accepted, if the M-H acceptance rule is met (middle). The codebook of image patterns is then constructed by extracting the local image patches (e.g., of 8 × 8 pixels) appearing at the accepted coordinates (right). Note that duplicate patterns are not accepted in the codebook.

2.2 Temporal Pooling

Once the spatial pooling is over, the HTM learning continues with the temporal pooling. In this step, the collected quantization centers are grouped according to their temporal coherence within the training sequence of patterns. The resulting disjunctive subsets of the codebook patterns are called *temporal groups*.

The HTM theory postulated several conditions which are to be satisfied by any pattern sequence used for training HTM networks. The most crucial is the condition of a smooth translation of objects within the networks retina, meaning that the position, rotation, scale, or illumination of objects alter smoothly in time.[1] Up to now, when dealing with static images containing no inherent temporal information (unlike video streams), the temporal pooling has been accomplished using some sort of a smooth traversal of the training images, in course of which a sequence of image patterns has been generated. Hereinafter, we refer to this type of image exploration as a *smooth explorer*. Typical smooth explorers proceeded either along straight lines (e.g., image rows or columns), or performed a smooth Brownian-like random walk (see Figure 3, left). The generated pattern sequence then serves for estimating the temporal statistics that reflect temporal coherence of the codebook patterns.

The concept of the smooth image exploration, however, can be implemented in various ways, which may result in significantly different temporal groups in terms of their invariance. Usually, the reason is that different temporal pooling approaches may provide differently accurate temporal statistics, though based on the same training data (images). Since the temporal learning is a key element of the HTM invariance, an efficient temporal pooling algorithm is essential for its proper functionality. The construction of a novel, more efficient temporal pooler has been the main objective of our research and it will be described in details in Section 3.

2.3 Inference

In the inference mode, each HTM node produces a vector of beliefs for all memorized temporal groups, given an arbitrary input pattern X. Afterward, the resulting belief vectors from all nodes in the same network layer are passed to the next (superior) layer as inputs.

The first step in producing the node's output belief vector is a *spatial inference*, in which the beliefs for all the codebook patterns are calculated. In the original Numenta formulation of the spatial inference, the belief $P(C_i|X)$ is estimated according to a Gaussian function

$$P(C_i|X) = \exp \frac{-\|C_i - X\|^2}{\sigma^2}, \tag{2}$$

[1] Be aware that such a condition does not imply generation of pattern sequences, which are smooth in terms of inter-frame Euclidean distances. The patterns, which appear nearby in time, usually look perceptually similar; nevertheless, their L_2 distances are not necessarily small.

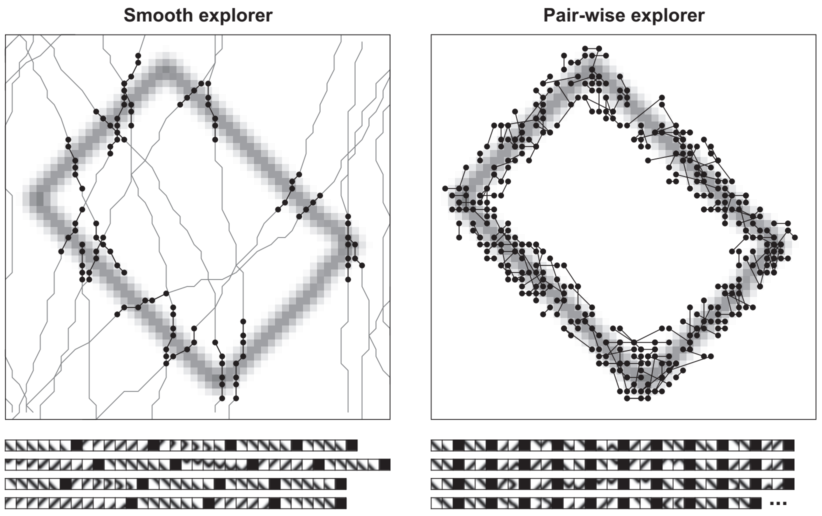


Fig. 3. An illustration of the function of the smooth explorer (left) and a novel pair-wise explorer (right). The smooth explorer used in this study has been implemented via a Brownian-like random walk (thin curves) in the course of which a presence of a nonempty pattern at the consecutive positions is being evaluated. If the valid pattern is detected at some position, the pattern sequence used for the temporal pooling is initiated (black spots connected by thick curves). The sequence is terminated by the first occurrence of the empty pattern. An example of the resulting pattern sequence is shown below the image. Black squares stand for interruptions of the sequence due to the empty patterns. On the contrary, the illustration of the pair-wise explorer shows a more efficient image traversal given a constant number of TAM updates. Namely, the pairs of patterns are sampled from a virtually infinite random walk, which hypothetically crosses each training image coordinate in each direction infinite number of times. Each generated pair of patterns can be seen as an extremely short fragment of the infinite temporal sequence, which produces a single TAM update of a unit increment. Note that both explorers are asymptotically equivalent, although the pair-wise explorer captures the infinite training sequence in a more efficient way, and therefore requires fewer TAM updates to provide sufficiently accurate temporal statistics.

where C_i is the i^{th} codebook pattern, X is the inferred input pattern and σ is a tuning parameter. The parameter σ should be chosen in such a way that the belief values close to 1 are assigned to the codebook patterns similar to X, while the dissimilar patterns receive the values close to 0. Afterward, to support a discrimination between similar and dissimilar patterns, the principle of strong lateral inhibition called *"winner-takes-all"* is applied to the spatial belief vector. According to this rule, only a constant number (typically 1) of components with the largest beliefs are preserved, while the remaining components are set to 0.

After the spatial inference is finished, a *temporal inference* takes place. In this step, the beliefs for all memorized temporal groups G_j are calculated as a sum of all spatial beliefs for the codebook patterns C_i, which constitute this temporal group:

$$P(G_j|X) = \sum_{C_i \in G_j} P(C_i|X). \tag{3}$$

The vector consisting of beliefs for all temporal groups represents an output of the HTM node in the inference mode.

It should be noted that in many practical applications when one operates in high-dimensional vector spaces, the use of the Gaussian-like spatial inference (see Equation (2)) is not necessary and can be replaced by a simpler and a more efficient method. The objection against the Gaussian function in high-dimensional spaces consists in impossibility to find any value of the parameter σ, that would provide a reasonable discrimination between similar and dissimilar patterns [11]. Theoretically, if the data (i.e., patterns) are assumed to be approximately normally distributed within the d-dimensional vector space, then L_2 distances between arbitrary pairs of data points should follow χ distribution with d degrees of freedom. It follows from the nature of χ distribution that the higher the data dimensionality, the smaller is its coefficient of variation, meaning that the vector data become more and more concentrated around some nonzero distance. As the Gaussian inference function corresponds with a special case of χ distribution with exactly 2 degrees of freedom, it is only capable of an accurate discrimination between close and distant patterns in low-dimensional vector spaces. In high-dimensional spaces (such as, e.g. $d = 64$, 256, or more), in which the data points tend to concentrate at a certain distance from each other, the Gaussian function becomes eventually too flat with respect to the actual distance distribution and, regardless of the σ value, provides almost equal beliefs for any activated codebook pattern. Therefore, the parameter σ has almost no influence on the final inference performance.

Exploiting this knowledge, we have reformulated the original spatial inference method, in order to dispose of the useless parameter σ and to simplify the method from the computational point of view. As the Gaussian function is monotonically decreasing with respect to L_2 distance of patterns, one can be sure that the codebook patterns which are closest to the inferred pattern in terms of L_2 distance would also receive the highest belief values. It is clear from the above reasoning, that all these beliefs will be almost equal, the parameter σ influences only the absolute value of this constant. Therefore we took the liberty of assigning the belief of 1 to the required number of the nearest codebook patterns and assigning 0 to the remaining ones:

$$P(C_i|X) = \begin{cases} 1 & \text{if } C_i \in \mathrm{NN}(X,k), \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $\mathrm{NN}(X,k)$ is a set of k nearest codebook patterns to the inferred pattern X. In this study, we considered $k = 1$.

2.4 Supervised Classification

The HTM model, as described so far, consists of only HTM nodes, which are trained in an entirely unsupervised manner. However, such a network alone cannot be directly applied to any classification task, where objects are categorized into a predefined set of classes. To enable this functionality for the HTM model, a supervised classifier is typically placed at the very top of the HTM hierarchy. This classifier is responsible for mapping between predefined object categories/classes and the belief vectors generated by the topmost HTM nodes during the inference. In the literature, one can find various supervised classifiers such as k-Nearest Neighbor (k-NN) [14] or Support Vector Machines (SVM) [7] being used for this purpose.

In order to investigate qualities of the two temporal pooling algorithms, regardless of the capabilities of any particular supervised classifier, we restricted ourselves to the k-NN approach (with $k = 1$). The classification accuracy provided by this method could then be used as a relative measure suitable for the comparison of different temporal pooling algorithms. This claim is supported by the fact that the generalization power of k-NN solely depends on the distribution of different data categories within the input space, which, in our case, was the space of belief vectors produced by the topmost HTM level. It should be noted that the structure of this space was entirely dependent on the capabilities of the employed temporal pooling algorithm provided that all other parameters of the model remained constant. If the temporal pooler worked efficiently during the training, the data within the generated belief vector space would be well organized and the classification accuracy achieved by k-NN would always be higher than in the case of less efficient temporal pooler method.

Since each HTM node at the topmost network level provides its own belief vector, one needs to combine all these vectors to a single belief vector in order to perform a supervised classification. In this study, we explored two alternative approaches to the construction of the joint belief vector out of the set of subordinate belief vectors.

Spatially specific features – The resulting belief vector is created by a simple concatenation of the belief vectors produced by all the topmost HTM nodes. This approach is spatially specific, since the final vector implicitly preserves information about positions where (in which node) particular temporal groups were found activated. Obviously, for the geometrically transformed input images, the spatial specificity may cause the decrease of the *classification accuracy* (CA), especially in cases when the classifier is trained on a small number of sample images.

Bag of features – The resulting belief vector is calculated as the element-wise sum of all individual belief vectors coming from the HTM nodes. Afterward, the accumulated belief vector is normalized to the unit length. As the spatial information is deliberately discarded in this case and only some sort of common statistics is being kept, the classification performance should be improved in the recognition problems, where the position invariance is required. On the other hand, when applying this approach to the classification tasks with a high number of similar categories, the classifier's specificity, and consequently also its accuracy, might be reduced due to the inability to exploit spatial relationships characteristic for the classified objects.

3 ALTERNATIVE APPROACH TO TEMPORAL POOLING PROVIDING MORE ACCURATE TEMPORAL STATISTICS

As already claimed, the main drawback of the Numenta-like smooth explorer is a slow convergence to the theoretical maximum of CA with respect to the length of the training sequence. When the problem domain is large, pattern sequences produced by the smooth explorer need to be rather long to capture the data in its entirety, assuring sufficiently accurate temporal statistics.[2]

Let us now describe in more detail how the temporal pooling using the smooth explorer actually works and what is the reason for its slow convergence. When processing the training sequence, codebook patterns, which occur nearby in the sequence (representing a virtual temporal sequence), generate increments of a structure called the *time adjacency matrix* (TAM)[3]. Each increment of TAM, which takes place in the course of training, has an individual strength given by the update function defined as follows[4, 5]:

$$U(d) = \begin{cases} TM - d + 1 & \text{if } 0 < d \leq TM, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $d \in \mathbb{N}$ is the temporal distance between the two processed patterns (i.e., the number of temporal transitions separating given two patterns in the training sequence) and *TM* is the parameter *transition memory*, which defines the maximum temporal distance between any two patterns which generate nonzero increment of TAM. It should be noted that in each training step depending on the parameter TM, TAM is being updated multiple times with a different weights given by Equation (5). Obviously, using such an approach, the same computational effort is put to the higher-weighted updates which carry out large TAM increments, as well as to the lower-weighted ones. Considering the fact that there is usually quite a large amount of useful (pseudo)temporal information stored in the training images which can only be unveiled by a rather long smooth training sequence, collecting the accurate TAM values usually takes considerable number of processing steps, since most of them are spent on the less important TAM updates.

[2] Note that the overtraining effect does not apply to the temporal pooling. The longer training sequence is taken, the more stable and accurate temporal statistics is collected.

[3] TAM is a square matrix, where each row and column corresponds to a single codebook pattern. Thus, each coordinate in TAM has a unique association with a particular pair of codebook patterns. In our experiments, TAM was always updated symmetrically.

To overcome this drawback of the smooth explorer, we propose a novel method of the temporal learning, the so-called *pair-wise explorer*. Instead of generating smooth random walks through images, our explorer performs the HTM training using pairs of patterns sampled from a virtually infinite random walk, which hypothetically crosses each coordinate of all training images in each direction, infinite number of times. These pairs of patterns are randomly sampled from the training images, such that distances $\hat{d}$ of their coordinates follow the probability distribution $P(\hat{d})$ proportional to the update function $U(d)$ (see Equation (5)), given a reasonable conversion between $\hat{d}$ and d (e.g., $d = \lceil \hat{d} \rceil$, where $\lceil \cdot \rceil$ stands for the round up operator). Each generated pair of patterns can be seen as an extremely short fragment of the infinite temporal sequence, which produces exactly one TAM update of a unit increment. The whole training is finished when the requested number of the TAM updates (i.e., processing steps) is undertaken. An example of such a training process is shown in Figure 3, right.

We argue that the pair-wise explorer helps the HTM to converge faster to the theoretically accurate TAM, especially in case of a limited number of TAM updates (which is a common constraint regarding the computational complexity). The algorithm generates fewer patterns with the low temporal coherence, and therefore fewer lower-weighted TAM updates are performed.

Once the temporal learning is over, the requested number of temporal groups is generated by means of the *agglomerative hierarchical clustering* (AHC). We achieved good results using the *UPGMA*[4] linkage and the dissimilarity measure $D_{i,j}$ given as:

$$D_{ij} = \begin{cases} 1 - \frac{\mathrm{TAM}_{ij}}{\sqrt{\max\limits_i(\mathrm{TAM}_{ij}) \max\limits_j(\mathrm{TAM}_{ij})}} & \text{if } i \neq j, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

4 CLASSIFICATION EXPERIMENTS WITH SIMPLE GEOMETRIC OBJECTS

In order to investigate and compare properties of the pair-wise and the smooth explorer, we conducted experiments with the position, scale, and rotation-invariant recognition of geometric primitives of three classes – circles, triangles, and rectangles. Instances of these three classes, denoted by their contour lines, were arbitrarily scaled, rotated and translated within the network's retina represented as a gray-scale image of 64×64 pixels (similarly as in [11, 8, 3]; see Figure 1).

The explored HTM network consisted of a single layer of non-overlapping nodes, each looking at the patch of 8×8 pixels of the retina. In accordance with the image size and types of the used patterns, we set the codebook size to 512, the requested temporal group count to 64, and the transition memory parameter TM to 4. The level of CA has been investigated with respect to two variable parameters: the number of training images, varied from 10 to 300 per each object class, and the number

[4] Unweighted Pair Group Method with Arithmetic Mean (UPGMA)

of the TAM updates, ranging between 1 024 and 32 768. These numbers represented the length of training of the HTM network for both temporal pooling methods, serving thereby as a unified evaluation basis. The number of testing images was set to 300 per class. In order to minimize the influence of randomly generated sets of images used for training and testing, each CA estimate was evaluated 10 independent times for every combination of the variable parameters, each time with newly generated training and testing sets. For final evaluation, the average CA values have been used.

For the sake of verification of the positive effect of the temporal pooling, which is a crucial part of the HTM model, the supervised (k-NN) classification has been accomplished restricting itself to various intermediate products of the HTM inference. Firstly, to account for the CA baseline, the classification has been performed in the source image space, where the obtained accuracy can only grow with an increased number of training samples. Secondly, to examine the isolated influence of the vector quantization accomplished by the spatial pooler, the k-NN classifier has been applied to the belief vectors produced by the spatial pooler as if they were outputs of the HTM nodes; and finally, the ultimate classification took advantage of the full-fledged HTM inference which includes both the spatial and the temporal inference step. If the HTM parameters were specified correctly and the temporal pooling had a definite positive impact on the invariant classification, the final CA should be significantly higher than the accuracy obtained in the image space or the one assuming only the spatial inference. Furthermore, we expected that the pair-wise explorer is capable of providing a considerably higher CA compared with the smooth explorer, given an equal number of TAM updates.

5 RESULTS

First of all, let us concentrate on the verification of the HTM network functionality, if it yielded a desired improvement of the invariant recognition of the geometric primitives. Since the pair-wise explorer is nothing else but a more efficient implementation of the same learning concepts as used by the smooth explorer, these two approaches should converge to the identical results, provided a sufficient number of TAM updates. An overview of the recognition rates obtained by k-NN performed in different classification spaces can be found in Table 1. Comparing the columns corresponding with the pair-wise and the smooth explorer, one can see that the obtained recognition rates are almost equal. This observation strongly supports the asymptotic equivalence of these methods. Furthermore, in all the evaluated cases, the CA values obtained in the image space and the CA values for HTM that used just the spatial pooler are inferior to those when also the temporal pooler was included. The actual CA gain due to using the temporal pooler ranged between 5 and 26 % for the spatially-specific-features approach and between 18 and 35 % for the bag-of-features approach, depending on the number of training images per object class.

In Figure 4, the recognition rates obtained by the pair-wise and the smooth explorer are presented with respect to both the number of TAM updates and the number of training images per object class. In this view, one can clearly see that the proposed pair-wise explorer outperforms the smooth explorer in the vast majority of evaluated parameter combinations. As expected, the advantage of the pair-wise explorer is most visible in the initial range of TAM updates, where the increase in accuracy is much larger than produced by the smooth explorer. This behavior is clearly documented by the course of the accuracy ratio between the pair-wise and the smooth explorer (see Figure 5 and Table 2). Regardless of the approaches to the construction of the joint belief vector used for classification – the spatially specific features and the bag of features, the predominance of our method peaked when TAM updates took values 2 896 and 4 096, yielding up to 10 % relative accuracy improvement over the conventional method.

		Spatially specific features			**Bag of features**		
Number of trn. images per class	**Image space**	**Spatial pooler**	**Pair-wise explorer**	**Smooth explorer**	**Spatial pooler**	**Pair-wise explorer**	**Smooth explorer**
10	0.378	0.354	0.428	0.424	0.417	0.679	0.654
20	0.398	0.366	0.475	0.472	0.440	0.741	0.724
40	0.439	0.396	0.549	0.542	0.481	0.792	0.781
60	0.477	0.419	0.600	0.594	0.496	0.818	0.806
80	0.510	0.436	0.641	0.632	0.518	0.825	0.815
100	0.540	0.452	0.674	0.663	0.531	0.847	0.830
150	0.594	0.495	0.722	0.708	0.557	0.858	0.839
200	0.637	0.514	0.771	0.769	0.576	0.873	0.857
250	0.671	0.536	0.795	0.793	0.578	0.879	0.872
300	0.701	0.558	0.814	0.808	0.595	0.887	0.879

Table 1. An overview of the recognition rates obtained by k-NN performed in different classification spaces, given the number of training images. The *"image space"* column gives the accuracy values of k-NN applied in the source image space excluding the influence of the HTM network. Two columns labeled *"spatial pooler"* contain accuracy values obtained when k-NN used the spatial belief vectors only. Finally, the remaining columns contain the results of the pair-wise and the smooth explorer, obtained for the maximum number of TAM updates (i.e., 32 768).

It should be noted that in the presented experiment of the position, scale, and rotation-invariant recognition of simple geometric objects, the bag-of-features approach provides a considerably higher classification accuracy than the spatially specific features. Such a result can be explained by the position invariance that is inherent to the bag-of-features approach (see Section 2: Supervised classification). As this approach only preserves common statistics about the beliefs for the temporal groups over several HTM nodes, the unwelcome consequence might be a decrease of the classifiers specificity, typically resulting in a low recognition rate. Nevertheless, considering the fact that the performance of the bag-of-features approach always exceeded the performance of the spatially specific features, we conclude that the mentioned reduction of the specificity did not actually take place in our experiments.

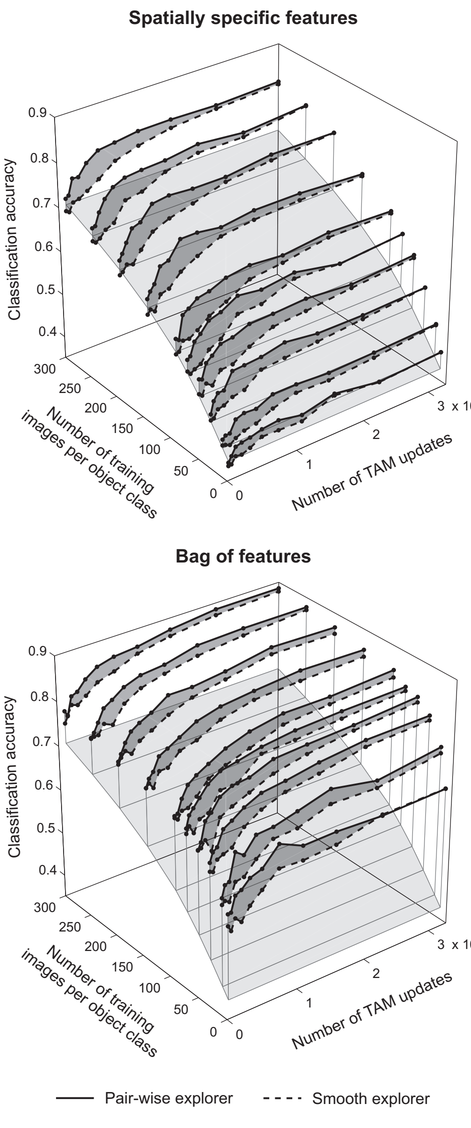
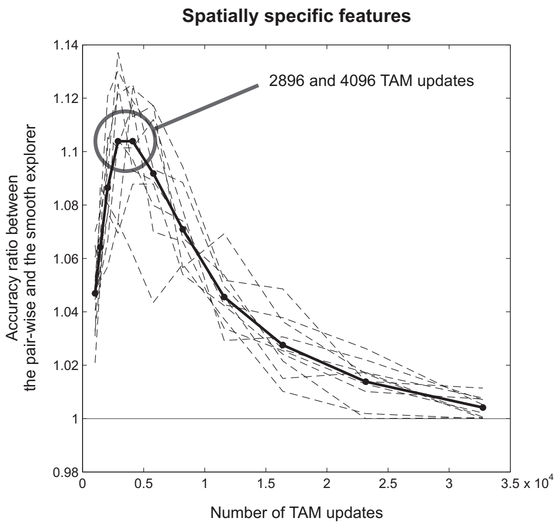


Fig. 4. The plots demonstrate that the pair-wise method outperforms the Numenta-like smooth method in terms of faster convergence, mostly in the lower range of TAM updates. The gray surface in the background represents CA achieved by k-NN classifier performed in the input data space.



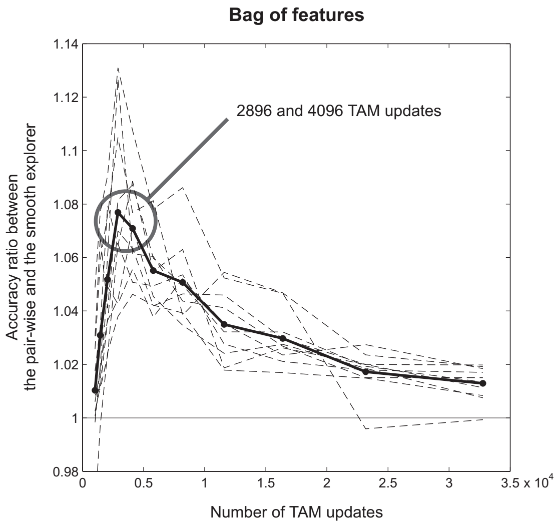


Fig. 5. The CA ratios of the pair-wise explorer over the smooth explorer. The dashed lines represent trials with different numbers of training examples while the solid line is their arithmetic mean.

Number of TAM updates	Spatially specific features	Bag of features
1 024	1.047	1.010
1 448	1.064	1.031
2 048	1.086	1.052
2 896	1.104	1.077
4 096	1.104	1.071
5 793	1.092	1.055
8 192	1.071	1.051
11 585	1.046	1.035
16 384	1.028	1.030
23 170	1.014	1.017
32 768	1.004	1.013

Table 2. The actual numerical values of the CA ratio of the two examined temporal pooling methods. The provided numbers are averages over the ratios obtained for different numbers of training images. The maxima of the accuracy ratio are emphasized.

6 CONCLUSIONS

In the paper, the functions of the single-layer HTM network with non-overlapping nodes have been explored with regard to the problem of the geometric object recognition, invariant to the position, scale, and rotation transformations.

Firstly, we have introduced essentials of the HTM theory and described peculiarities of our implementation of the HTM network. An alternative spatial pooling method has been introduced which makes use of a random pattern generator based on the Metropolis-Hastings algorithm. We have also reformulated the original HTM inference algorithm proposed by Numenta, in order to reduce a number of tunable parameters and to optimize its computational efficiency.

The main contribution of the paper consists in the proposal of a novel temporal pooling method – the *pair-wise explorer*. The obtained results show that, in contrast to the conventional smooth explorer, our method yields significantly faster convergence to the theoretical maximum classification accuracy with respect to both the length of the training sequence (defined by the maximum allowed number of TAM updates) and the number of training samples. The advantage of the pair-wise explorer manifests itself in all evaluated cases, though the most pronounced accuracy gain has been obtained in the lower range of TAM updates. In this range, up to 10 % relative accuracy improvement over the conventional method has been achieved. These results justify the conclusion that the proposed pair-wise temporal pooling method is more accurate and reliable, especially when applied to the large complex problems where the exhaustive training is not feasible. Therefore we suggest to use our method, instead of the smooth explorer, always when the HTM network is trained on a set of static images which lack of inherent temporal information.

Acknowledgement

This work has been supported by the Slovak Grant Agency for Science (VEGA research project No. 2/0019/10) and partially by the Foundation of Tatra banka, through the program E-talent for young scientists in the field of applied informatics (project No. 2010et019).

REFERENCES

[1] Hawkins, J.—Blakeslee, S.: On Intelligence. Henry Holt and Company, New York 2004.

[2] George, D.—Hawkins, J.: Towards a Mathematical Theory of Cortical Micro-Circuits. Plos Computational Biology, Vol. 5, 2009, No. 10. Doi 10.1371/Journal.Pcbi.1000532.

[3] George, D.—Hawkins, J.: A Hierarchical Bayesian Model of Invariant Pattern Recognition in the Visual Cortex. In: D. Prokhorov (Ed.): Proceedings of the International Joint Conference on Neural Networks (IJCNN), IEEE, 2005, Vol. 3, pp. 1812–1817.

[4] Numenta. Hierarchical Temporal Memory, concepts pages 157–168, theory, and terminology, June 2008, document version 1.8.0.

[5] Numenta. Numenta node algorithms guide, NuPIC 1.7, October 2009.

[6] Csapo, A.—Baranyi, P.—Tikk, D.: Object Categorization Using VFA-Generated Nodemaps and Hierarchical Temporal Memories. In Proceedings of the 5th IEEE International Conference on Computational Cybernetics 2007, pp. 257–261.

[7] Sassi, F.—Ascari, L.—Cagnoni, S.: Classifying Human Body Acceleration Patterns Using a Hierarchical Temporal Memory. In: R. Serra, R. Cucchiara (Eds.): AI*IA 2009: Emergent perspectives in artificial intelligence, Springer-Verlag 2009, pp. 496–505.

[8] Lorenzi, D.—Vaidya, J.: Identifying a Critical Threat to Privacy Through Automatic Image Classification. In Proceedings of the 1st ACM Conference on Data and Application Security and Privacy, San Antonio 2011, pp. 157–168.

[9] van Doremalen, J.—Boves, L.: Spoken Digit Recognition Using a Hierarchical Temporal Memory. In Interspeech 2008, pp. 2566–2569.

[10] Rozado, D.—Rodriguez, F.—Varona, P.: Optimizing Hierarchical Temporal Memory for Multivariable Time Series. In: L. Iliadis (Ed.): Proceedings of the 20th International Conference on Artificial Neural Networks, Part II, Springer-Verlag 2010, pp. 506–518.

[11] Štolc, S.—Bajla, I.: On the Optimum Architecture of the Biologically Inspired Hierarchical Temporal Memory Model Applied to the Handwritten Digit Recognition. Measurement Science Revue, Vol. 10, 2010, No. 2, pp. 28–49, DOI 10.2478/v10048-010-0008-4.

[12] Metropolis, N.—Rosenbluth, A. W.—Rosenbluth, M. N.—Teller, A. H.—Teller, E. et al.: Equation of State Calculations by Fast Computing Machines. Journal of Chemical Physics, Vol. 21, 1953, pp. 1087–1092.

[13] Hastings, W. K.: Monte Carlo Sampling Methods Using Markov Chains and Their Applications. Biometrika, Vol. 57, 1970, pp. 97–109.

[14] Kostavelis, I.—Gasteratos, A.: On the Optimization of Hierarchical Temporal Memory. Pattern Recognition Letters, Vol. 33, 2012, No. 5, pp. 670–676.

Svorad Štolc studied artificial intelligence and parallel computing at the Faculty of Mathematics, Physics and Informatics of the Comenius University in Bratislava where he achieved his Master's Degree in 2001. In 2009, he accomplished his Ph. D. research work at the Institute of Measurement Science of the Slovak Academy of Sciences, Bratislava on the topic of Epo image analysis in doping control. Since 2008 he is a research fellow with the High-Performance Image Processing business unit of the Safety & Security Department at the AIT Austrian Institute of Technology, GmbH. His research interests comprise artificial intelligence, image processing, pattern recognition and visual object classification using biologically inspired approaches.

Ivan Bajla received his Ph. D. degree in measurement science (biomeasurement and image processing) in 1977 from the Institute of Measurement Theory of the Slovak Academy of Sciences in Bratislava. His research interests include gray-scale and local transformations of digital images, geometry-driven diffusion methods of nonlinear image filtering, image segmentation methods in the field of MRI and electrophoretic gel image analysis. Between 1998 and 2010 he was with the High-Performance Image Processing Department of the AIT (Austrian Institute of Technology, GmbH, formerly ARC (the Austrian Research Center GmbH). He was the scientific leader of two research projects: GelMaster – a system for the analysis of electrophoretic gel images, and GASepo – a system for the analysis of Epo images in the doping control on Epo. He teaches courses in image processing at the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava. He is currently a Professor at this university. Since 2010 he has been conducting the VEGA research project at the Institute of Measurement Science of the Slovak Academy of Sciences, Bratislava.

Kristián Valentín received his Bachelor Degree in applied informatics and his Master's Degree in cognitive science from the Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava. His main research interests include neural networks, image processing and pattern recognition using biologically motivated algorithms. He is currently a Ph. D. student at the same faculty and a researcher at the Institute of Measurement Science, Slovak Academy of Sciences, Bratislava under supervision of Ivan Bajla.

Radoslav Škoviera received his Bachelor Degree in applied informatics and his Master's Degree in cognitive science from the Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava. His main research interests include image analysis and pattern recognition using biologically motivated algorithms with the focus on content-based image retrieval. He is currently a Ph. D. student at the same faculty and a researcher at the Institute of Measurement Science, Slovak Academy of Sciences, Bratislava under supervision of Ivan Bajla.