Proceedings of the Federated Conference on Computer Science and Information Systems pp. 271–274
DOI: 10.15439/2016F133
ACSIS, Vol. 8. ISSN 2300-5963

Using Spatial Pooler of Hierarchical Temporal Memory for object classification in noisy video streams

Maciej Wielgosz, Marcin Pietroń, Kazimierz Wiatr
AGH University of Science and Technology
al. Mickiewicza 30, 30-059 Krakow, Poland
Academic Computer Centre 'Cyfronet' AGH
Nawojki 11, 30-950 Krakow, Poland
Email: wielgosz,pietron,wiatr@agh.edu.pl

***Abstract*—This paper focuses on analyzing a Spatial Pooler (SP) of Hierarchical Temporal Memory (HTM) ability for facilitating object classification in noisy video streams. In particular, we seek to determine whether employing SP as a component of the video system increases overall robustness to noise. We have implemented our own version of HTM and applied it to object recognition tasks under various testing conditions. The system is composed of a video preprocessing block, a dimensionality reduction section which contains SP, a histograms collecting module and SVM classifier.**

Our experiments involve assessing performance of two different system setups (i.e. a version featuring SP and one without it) under various noise conditions with 32–frame video files. In order to make tests fair and repeatable the videos of several 3–D geometric shapes were artificially generated. Subsequently, Gaussian noise of a different intensity was introduced to the videos making them more indistinct. Such an approach mimics real–life scenarios where the system is taught ideal objects and then faces in its normal working conditions the challenge of detecting noisy ones.

The results of the experiments reveal the superiority of the solution featuring Spatial Pooler over the one without it. Furthermore, the system with SP performed better also in the experiment without a noise component introduced and achieved a mean F1–score of 0.91 in ten trials.

I. Introduction

DESPITE the huge technological growth witnessed nowadays, there are still no autonomous machines available which would be capable of operating in the real world. Such machines would take over most of our tedious everyday duties and clear the way for a breakthrough in Artificial Intelligence. However, such robots need to be able to process inputs in real time, learn, generalize and react to events. This requires building an appropriate processing system which has human–like capabilities [1] [2].

A mammalian brain is an example of such a system which evolved over millions of years. Despite its apparent complexity there is only one algorithm [3] within the brain which governs the body functions. This allows for scalability of the solutions based on the algorithm since more complex systems may be built on a top of the simpler ones just by duplication of the basic structure.

The human brain as a whole has not been completely explored yet, making its artificial implementation and verification a very hard task. However, there are initiatives [4] which have taken up the challenge of simulating and modeling a brain as we know it today. Rather than model the brain, the authors of this paper have adopted a slightly different approach of gradually introducing selected components of HTM to the video processing system with the intention of enhancing its performance. By doing so we aim to develop a complete system working on the principles of the human brain as they were presented in [3][5] with our modifications making the algorithm suitable for hardware implementation.

Consequently, this paper attempts to take a step forward in examining the feasibility of using HTM for classifying objects in noisy video streams. The authors state the following hypothesis: employing Spatial Pooler in the video processing flow will improve the object classification ratio due to its beneficial reduction property of mapping to Sparse Distributed Representation (SDR). It is verified through a series of experiments.

The rest of the paper is organized as follows. Sections I-A and I-B provide the background and related work of object classification in video streams and Hierarchical Temporal Memory, respectively. The custom designed system used for the experiments is presented in Section II. Section III provides the results of the experiments. Finally, the conclusions of our research are presented in Section IV.

A. Object classification in video streams

Most state–of–the–art information extraction systems consist of the following sections: preprocessing, feature extraction, dimensionality reduction and classifier or ensemble of classifiers (Fig. 1). Their construction requires expertise knowledge as well as familiarity with the data that will be processed [6][7].

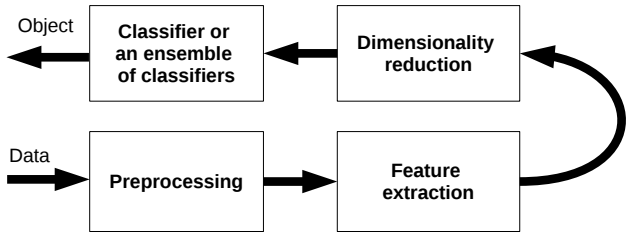

Fig. 1. Architecture of a video processing system

Usually, systems for object classification in video streams are also designed according this scheme. Consequently, the proper choice of the operations which constitute all the mentioned stages of the system is important and decides about the classification result [8]. One of the most challenging stages is feature extraction, which substantially affects the overall performance of the system.

There are also systems which take advantage of the spatial–temporal [5] profile of the data [1][9]. They are closer to the concept of the solution presented in this paper, which may be considered a hybrid approach since it features components of both schemes.

B. Hierarchical Temporal Memory

Hierarchical Temporal Memory (HTM) replicates the structural and algorithmic properties of the neocortex. It can be regarded as a memory system which is not programmed, but trained through exposing it to data flow. The process of training is similar to the way humans learn which, in its essence, is about finding latent causes in the acquired content. At the beginning, the HTM has no knowledge of the data stream causes it examines, but through a learning process it explores the causes and captures them in its structure. The training is considered completed when all the latent causes of data are captured and stable. The detailed presentation of HTM is provided in [5][10][11].

HTM is composed of two main parts, namely Spatial and Temporal Pooler. This paper focuses on Spatial Pooler (SP), aka Pattern Memory, which is employed in the processing flow of the system. Is contains columns with synapses connected to the input data [5]. The main role of SP in HTM is finding spatial patterns in the input data. It may be decomposed into three stages:

- Overlap calculation,
- Inhibition,
- Learning

The first two stages are very computationally demanding but can be parallelized, therefore the authors decided to implement them on GPU in OpenCL. The learning stage is implemented on CPU in Python. The detailed description of algorithms is provided in [5].

The overlap section computes *col.overlap* for every column in SP structure i.e. a number of active and connected synapses.

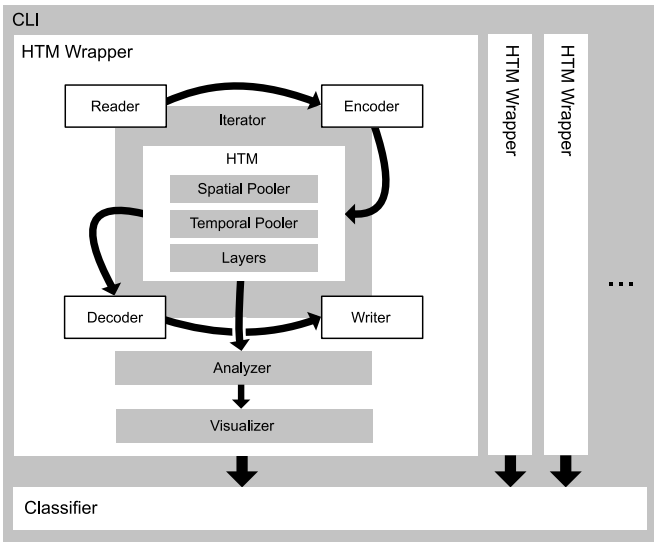

Fig. 2. Architecture of the implemented system

If the number is larger than *col.min_overlap* then it is boosted and passed on to the inhibition section.

The inhibition stage implements a winner–takes–all procedure where for each column a decision is made as to whether it belongs to a range of n columns of the highest values.

II. System description

The implemented HTM version [12] follows description from [5], with the exception of synapses bias implementation being replaced with random connections. Its main purpose however is to emphasize the algorithm parallelism and to allow progressively replace parts of it with GPU–accelerated (and in the future FPGA–accelerated) fragments written in OpenCL. The system (Fig. 2) is highly configurable, with numerous parameters responsible for the core HTM's structure, the encoder behavior, statistics rendering, etc. The configuration is stored in a file written in JSON format, which allows it to maintain its readability while providing clear structure. In addition to the core module, a set of supporting modules has been developed. Most of them are used for feeding video data to the core module, and receiving and analyzing the results.

The complete processing flow of the system is presented in Fig. 3. The data is fed into the system in a frame–by–frame manner. In the first step the original frame is reduced and binarized using OpenCV procedures. During the encoding process, the original video frame is converted to a smaller binary image. Preliminary tests showed that reducing a 960x540 image by a factor of 16 (producing 60x33 images) has a low impact on the end results while significantly shortening the processing time (Fig. 4). After reduction, the color image is converted to grayscale, which later is turned into a binary one using adaptive thresholding (using a potentially different threshold value for each small image region).

Those operations constitute the encoding which allows the generation of input data for the SP processing stage. There-

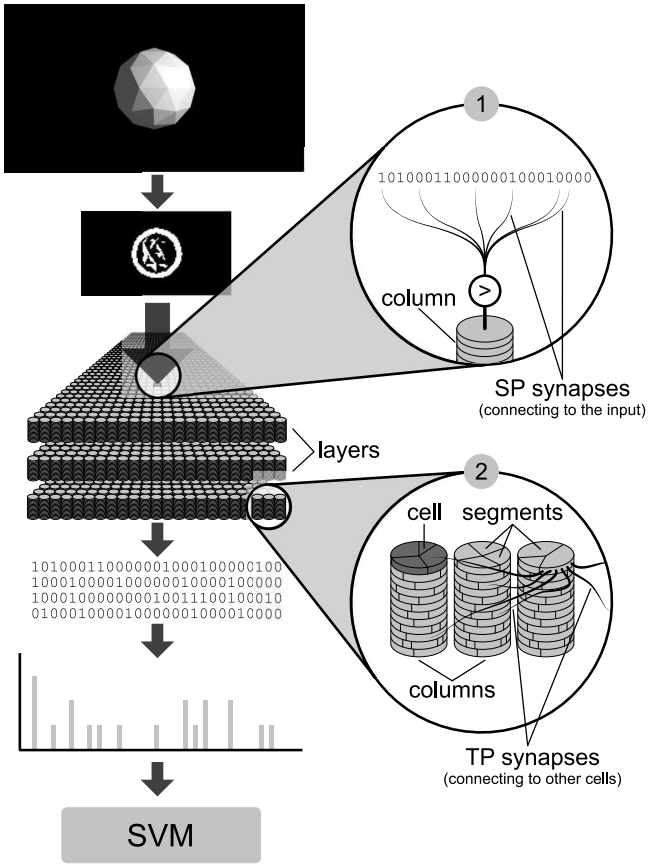

Fig. 3. Block diagram of the proposed approach

after, the data is mapped to SDR with SP and may be passed on to TP. Since our current work focuses on finding the optimal SP parameters values we do not use TP in this particular application, but the system in general has such capability. In the next step, histograms of consecutive processed frames are built on a per–video basis. The histograms are used as the input data for the SVM classifier which comes next.

There are two different working modes of the system, namely learning and testing modes. The system is trained in the learning mode with 80% of available data and then it is tested with the remaining 20% of the data in the testing mode.

III. Experiments and the discussion

A series of experiments (details of which are provided in Tab. I and Tab. II) was conducted to validate the hypothesis stated in the introduction of the paper. They allow a comparison of the performance of the system featuring Spatial Pooler in the processing flow with the one lacking it.

The challenging part involved generation of sample videos for testing (available from [13]. The videos had to meet a series of requirements such as object location, camera location and object–camera distance. Consequently, a dedicated application was used to generate the videos (i.e. Blender [14]). Blender provides Python API, which was used to automate and randomize the video generation. Each video contains a single, centered shape and a randomly positioned light source which brightness is picked from predefined range. During the course of a video the camera randomly changes position. Since noise addition at the runtime proved to be a very time consuming process, a separate script introducing noise to a large set of generated videos was created. Embedding noise also ensures equal conditions for all the experiments and test setups.

TABLE I
VIDEO PARAMETERS

Parameter		Value
Size of a single video frame		960x540
Reduction level		16
Frame size after reduction		60x33
No. of frames in a single video		32
Object classes		cone, cube, cylinder, monkey, sphere, torus
No. of classes		6
Total no. of videos	all	6000
	training	4800
	testing	1200
Videos per class	all	1000
	training	800
	testing	200
Videos per trial	all	100
	training	80
	testing	20

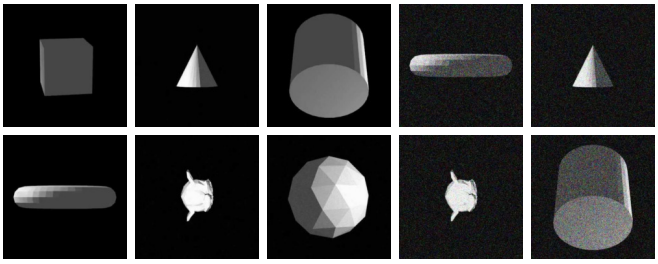

Fig. 4. Sample frames of different shapes and noise levels

The F1–score is used as a quality evaluation of the experiments' results presented in this paper.

Experiments conducted for higher noise levels than presented in Tab. III resulted in an unacceptably low F1–score (below 0.75). Therefore, the authors decided not to include the results of those experiments in the table.

It is worth noting there there was a huge difference in the calculation time of a single experiment across setups, see Tab. IV. This is the result of an HTM algorithm complexity,

TABLE II
SP PARAMETERS

Parameter	Value
No. of columns	2048
No. of synapses per column	64
Perm value increment	0.1
Perm value decrement	0.1
Min overlap	8
Winners set size	40
Initial perm value	0.21
Initial inhibition radius	80

TABLE III
EXPERIMENTS RESULTS FOR DIFFERENT NOISE LEVELS

	F1–score : mean and variance (10 repetitions)	
Noise level (σ)	**SVM**	**SVM + SP**
0	0.82 (0.001)	0.91 (0.001)
4.25	0.81 (0.001)	0.89 (0.001)
8.5	0.78 (0.001)	0.88 (0.003)

TABLE IV
EXECUTION TIME OF A SINGLE EXPERIMENT FOR DIFFERENT SETUPS

Setup	**Time [hours]**
SVM	0.23
SVM + SP	28.5

object–oriented programming approach and high level scripting language used in implementation. Code optimization and introducing more hardware–accelerated fragments should improve execution time. The tests were run on Intel(R) Core(TM) i5–4210M CPU @ 2.60GHz, and Nvidia GeForce GT 730M (selected sections of SP). Each experiment consisted of 10 trials.

According to the authors knowledge it is hard to find papers which directly correspond to the research conducted in this work (i.e. video classification in noisy video streams). Nevertheless, we examined the following papers : [15], [16], [17] which presents results of video classification using UCF-101 dataset. The best systems presented in those papers are based on various architectures of Convolutional Neural Networks (CNNs) and achieve accuracy of 80% and more. It is worth emphasising that despite similar performance in terms of the quality results our test setup is different mostly in a process of the video generation.

IV. CONCLUSIONS AND FUTURE WORK

This paper presents the preliminary experimental results of using an HTM–based system for object classification in video streams. The authors showed that using SP in the video processing flow improves the object classification ratio by approx. 10%. In future work, the authors are going to modify the preprocessing stage of the video processing flow and introduce TP. The authors are going to implement the most computationally–exhaustive routines in OpenCL and deploy the system on platforms equipped with GPU– or FPGA–based acceleration. This will enable conduction of experiments with video of a lower image reduction ratio.

TABLE V
EXAMPLE CONFUSION MATRIX FOR SVM–ONLY SETUP AND GAUSSIAN NOISE WITH $\sigma = 8.5$

	Predicted classes					
	cone	**cube**	**cylinder**	**monkey**	**sphere**	**torus**
cone	19	0	0	0	0	1
cube	0	10	8	0	2	0
cylinder	0	3	16	0	1	0
monkey	0	1	2	14	2	1
sphere	0	3	3	0	13	1
torus	0	1	0	0	0	19

ACKNOWLEDGMENT

I would like to thank my wife Urszula Wielgosz for her huge contribution to the preparation of the paper.

REFERENCES

[1] S. Sengupta, H. Wang, W. Blackburn, and P. Ojha, "Spatial information in classification of activity videos," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds., vol. 5. IEEE, oct 2015. doi: 10.15439/2015F382 pp. 145–153.

[2] J. F. Sowa, "The Cognitive Cycle," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds., vol. 5. IEEE, oct 2015. doi: 10.15439/2015F003 pp. 11–16.

[3] V. Mountcastle, "The columnar organization of the neocortex," *Brain*, vol. 120, no. 4, pp. 701–722, apr 1997. doi: 10.1093/brain/120.4.701

[4] "The Human Brain Project - Human Brain Project," (Accessed on 10.04.2016). [Online]. Available: https://www.humanbrainproject.eu

[5] J. Hawkins, S. Ahmad, and D. Dubinsky, "Hierarchical temporal memory including HTM cortical learning algorithms," Numenta, Inc, Tech. Rep., sep 2011. [Online]. Available: http://numenta.org/resources/HTM_CorticalLearningAlgorithms.pdf

[6] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, sep 2009. doi: 10.1109/TKDE.2008.239

[7] P. Zhang, X. Zhu, and L. Guo, "Mining Data Streams with Labeled and Unlabeled Training Examples," in *2009 Ninth IEEE International Conference on Data Mining*, IEEE. Miami, USA: IEEE, dec 2009. doi: 10.1109/ICDM.2009.76 pp. 627–636.

[8] R. N. Hota, V. Venkoparao, and A. Rajagopal, "Shape Based Object Classification for Automated Video Surveillance with Feature Selection," in *10th International Conference on Information Technology (ICIT 2007)*, IEEE. Rourkela, India: IEEE, dec 2007. doi: 10.1109/ICIT.2007.57 pp. 97–99.

[9] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, aug 2013. doi: 10.1109/TPAMI.2013.50

[10] X. Chen, W. Wang, and W. Li, "An overview of Hierarchical Temporal Memory: A new neocortex algorithm," in *Modelling, Identification & Control (ICMIC), 2012 Proceedings of International Conference on*. Wuhan, China: IEEE, 2012, pp. 1004–1010.

[11] D. Rachkovskij, "Representation and processing of structures with binary sparse distributed codes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 2, pp. 261–276, 2001. doi: 10.1109/69.917565

[12] "Hierarchical temporal memory implementation," (Accessed on 12.04.2016). [Online]. Available: https://bitbucket.org/maciekwielgosz/htm-hardware-architecture

[13] "HTM Test Datasets," (Accessed on 02.07.2016). [Online]. Available: http://data.wielgosz.info

[14] "Blender project - Free and Open 3D Creation Software," (Accessed on 12.04.2016). [Online]. Available: https://www.blender.org/

[15] Joe Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4694–4702, jun 2015. doi: 10.1109/CVPR.2015.7299101

[16] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2014. doi: 10.1109/CVPR.2014.223 pp. 1725–1732.

[17] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov, "Exploiting Image-trained CNN Architectures for Unconstrained Video Classification," *ArXiv e-prints*, mar 2015. [Online]. Available: http://arxiv.org/abs/1503.04144