Influence of input sparsity to Hierarchical Temporal Memory Spatial Pooler noise robustness

Damir Dobric
University of Plymouth - Faculty of Science and Engineering, Node Frankfurt am Main
November 21, 2019

Hierarchical Temporal Memory - Spatial Pooler is a Neocortical Learning Algorithm inspired by a biological functioning of neo-cortex. It is responsible for sparse encoding of spatial pattern typically used as an input for further processing inside of algorithm. This paper presents results, which show how robust is Spatial Pooler to noise in dependence on the sparsity of an input pattern and proposes an implementation of a new boosting of the input.

1. Introduction

Hierarchical Temporal Memory - Spatial Pooler is a Neocortical Learning Algorithm inspired by biological functioning of neo-cortex. It is responsible for the sparse encoding of spatial pattern typically used as an input for further processing inside of Hierarchical Temporal Memory (HTM). HTM is known a theoretical model that forms many algorithmic properties of the neocortex. A major module of a HTM is Spatial Pooler, which is responsible for continuous encoding of sensory input streams and other sources of information, from different regions and layers of the neocortex [1]. One of most important features of Spatial Pooler is its capability to encode streams of zeros and ones as SDR (sparse distributed representation). Spatial Pooler groups similar spatial patterns represented as active neurons into highly sparse representation of cortical mini columns. SDR produced by Spatial Pooler typically occupies a small percent (i.e.: 2%) of defined column space. One of capabilities of the Spatial Pooler is a robustness against noise. Robustness of Spatial Pooler and sparse representations was documented in the previous works [1] and [2] respectively. By varying amount of the noise in input [1], sensitivity of SP output representation was measured. Adding more and more noise to the input, trained output for specified input keeps stable for relatively large fraction of input noise. This result approves that trained SP is robust against noise in contrast to untrained SP.

Motivation in this work is to analyse how encoding of input pattern itself influence robustness of SDR against noise when using Spatial Pooler. Previous work does not show the influence of a spatial input pattern on the noise robustness of SP. Existing experiments do not suggest what kind of input can be used to achieve higher robustness by passing input thorough SP and getting output without or with less noise than input. Knowing this is essential to easier implement reliable real-world applications based on Spatial Pooler, which are robust to noisy input.

Experiments in this work show that robustness against noise directly depends on sparsity of the spatial input pattern. Higher number of non-zero bits in the input stream, leads to more robustness against noise and better memorizing of the pattern.
Assuming that sparseness is a metric defined as the number of active neurons divided by number of available bits [1], it implies that higher sparseness leads to better robustness. Results presented in this paper demonstrate how the noise robustness depends on sparseness of the input.
This finding suggest that input pattern can easily be boosted (stroked out) to achieve better memorizing of the reference pattern and produce a more stable representation of SDR with very high level of noise.

2. Methods

As known, the Spatial Pooler (SP) performs fast memorizing of spatial pattern [2] and has a good robustness against noise [1]. This was shown by measuring the sensitivity of the SP to a varying amount of input noise [3]. In this work, robustness was measured by varying sparsity of the input vector by trained and untrained

Spatial Pooler [1]. To illustrate this, two images of 28x28 pixels have been taken from the MNIST database (see Figure 1). Images are binarized to 0-1 vectors and used as input during training process.

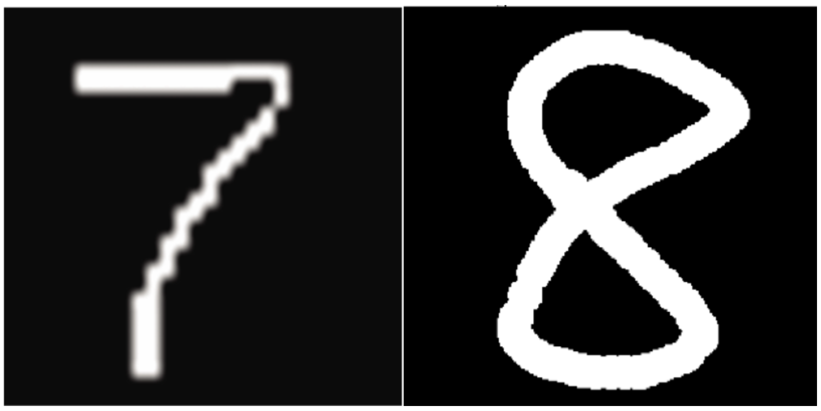

Figure 1
Digits "7" and "8", 28x28 pixels obtained from MNIST database. These images were originally used to measure noise robustness of trained and untrained Spatial Pooler. Image 8 and image 7 have sparseness of 17% and 6% respectively. Sparseness is defined as a ratio between non-zero bits and total number of bits.

All experiments in this paper were executed on .NET core implementation of Spatial Pooler (see [1]). For every image, the same instance of Spatial Pooler was used with parameters shown in Table 1. In these tests, the global inhibition was used. Column Bosting was set to high cycle period to ensure that Spatial Pooler doesn't enter column boosting while the experiment is running. Boosting is the feature of Spatial Pooler, which ensures that all columns uniformly participate in learning process. Disabling boosting during experiment execution makes sure, that eventually activated boosting will not clear memorized spatial pattern during the learning process.

The whole learning process was set to take ten iterations steps, which ensured that Spatial Pooler entered stable representation of both images in active columns set. With listed parameters in Table 1 and given input vectors, Spatial Pooler enters stable state in usually 2-3 steps. Ten iteration steps were chosen for practical reasons.

As next, the level of noise (5%, 10%, 15%,..,100%) was added to the original reference image. For every noise level, the existing instance of Spatial Pooler was trained with the noise-deformed representation of reference input. After every training step, the input distance and output distance between vectors are calculated. The input vector with noise is compared with the reference input vector without noise. The same calculation of distance is done on output vectors. The distance of the output vector of the input with noise is compared with the reference output value calculated for input without noise. After all, input and output distances are compared and the relation between them is used as a metric, which shows resistance against noise. The model is robust against noise if smaller output distance is notified for higher input distance. In other words, for relatively significant fraction of noise in input, the output keeps almost unchanged.

The distance in this context represents a metric, which defines a similarity between compared vectors.
As defined in equation (1), first the hamming distance between vectors is calculated.

$$o_k = \sum_{i=0}^{N} \| a_0(i) \circ a_k(i) \|_0 \tag{1}$$

$$l_k = \sum_{i=0}^{N} a_k(i) \tag{2}$$

$$d_k = \frac{l_k - (N - o_k)}{(N - o_k)} 100 \mid o_k < N \tag{3}$$

a_0 is a reference vector (input or output) without noise and a_k is a vector (input or output) with specific noise level defined by noise level index k . Both comparing vectors must have the same number of bits N. Every input vector with noise is bitwise compared with reference input by using of L0-norm, which defines the total number of non-zero elements in the vector. The overlap o between two vectors is calculated as the number of non-zero bits at the same position. In opposite, $N - \mathrm{o}_k$ represents the number of different bits.
Value l_k in equation (2) represents the number of non-zero bits of the noised vector k. The distance d_k in the equation $\mathrm{d}_k = \frac{\mathrm{l}_k-(N-\mathrm{o}_k)}{(N-\mathrm{o}_k)} 100 \mid \mathrm{o}_k < N$ (3) defines the metric used in this experiment, which shows how similar is vector a_k to a compared reference vector a_0. This metric is referred as a *distance* and it is used in all experiments described later in this paper.

Parameters	Value
POTENTIAL_RADIUS	-1 (all inputs are connected to every column)
POTENTIAL_PCT	1
LOCAL_AREA_DENSITY	-1
NUM_ACTIVE_COLUMNS_PER_INH_AREA	0,02*64*64 (2% of columns)
STIMULUS_THRESHOLD	0,5
SYN_PERM_INACTIVE_DEC	0,00
SYN_PERM_ACTIVE_INC	0,01
SYN_PERM_CONNECTED	0,1
MIN_PCT_OVERLAP_DUTY_CYCLES	0,001
MIN_PCT_ACTIVE_DUTY_CYCLES	0,001
DUTY_CYCLE_PERIOD	100
MAX_BOOST	10

Table 1
Spatial Pooler parameters used in experiments. Parameters shown in this table are commonly used in most HTM experiments. However, DUTY_CYCLE relevant parameters are chosen to prevent Spatial Pooler from early boosting. When Spatial Pooler starts column boosting, it will "forget" learned pattern. To avoid "forgetting" of learned pattern column boosting is deactivated during experiment. The distance shown in previous equations is calculated during inactive column boosting. For more detailed information about meaning of all parameters please see [4].

2 Results

The Figure 2 shows the *distance* of trained Spatial Pooler for two coincidently chosen MNIST images. According to the orange line, Image "7" (sparseness 6%) holds stable set of active columns till approx. 30% of noise. Similarly, image "8" (sparseness 17%) is stable till approx. 40% of noise. Both results are obviously noise resistant, but different. It can be concluded that image "8" is more resistant to noise than image "7". Hypothesis is that that the different results depend on sparseness of the spatial input (in this case MNIST image) is encoded.

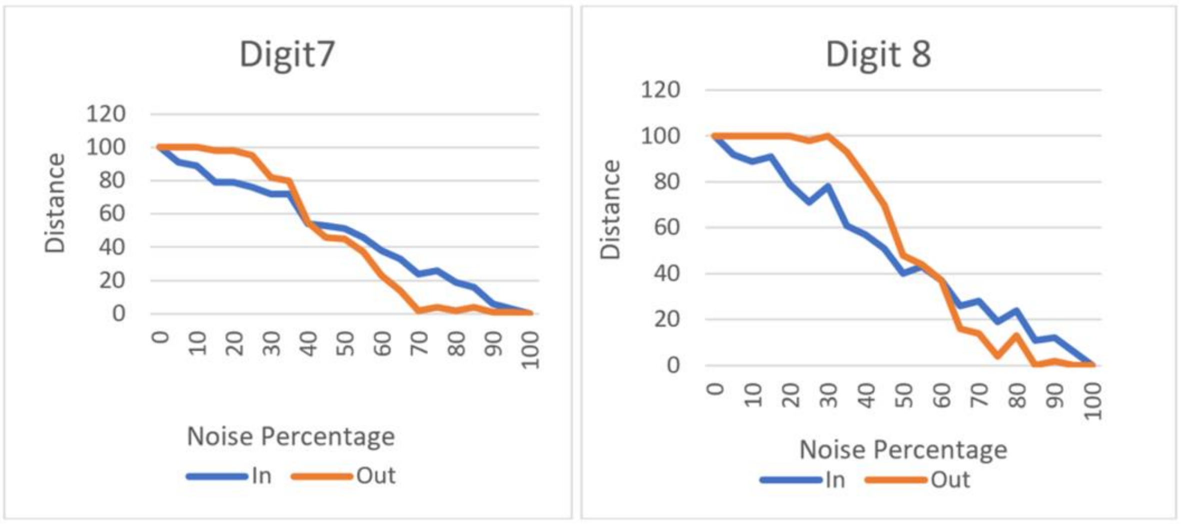

Figure 2
Blue line shows calculated hamming distance between input and noised Input after adding noise. Orange line shows the distance between active columns for the reference input and a given noised input vector.

To validate this hypothesis, a robustness of two box-images with 1024x1024 pixels have been created (see Figure 3). First image referred as "Box1" consists of 240 and second image referred as "Box2" consists of 110 non-zero bits. This corresponds to sparsity of 0.24 and 0,11 respectively. Box2 is sparser than Box1. For both images in experiment, the noise levels of 5, 10, 15,..,100 percent have been added to the reference image. Then, Spatial Pooler was trained with reference images and noised images.

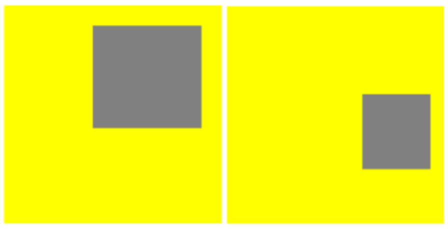

Figure 3
Two boxes 1024x1024 pixels (in yellow) with different sparsity (0.24 on left and 0.11 on right) have been used for training. Comparing of two images 1024x1024 pixels with different number of non-zero bits.

Figure 5 shows the results of five noise levels for each of two boxes defined in Figure 3. Every image in Figure 5 on left represents input used for Spatial Pooler. The Output of Spatial Pooler is shown at the right as a sparse output of active columns. For example, *0%* represents reference input without noise followed by inputs with 5%,10%, 15% and 20% of noise.

By following results in Figure 4, Box1 is still recognized with a distance of near to 100% by the given input distance of approx. 40% (Figure 4 on left).

Results for Box2 look slightly different. Image with Box2 gets well recognized for inputs with approx. up to 25% of noise. That means Box1 with the higher number of nonzero bits show better results than Box2.

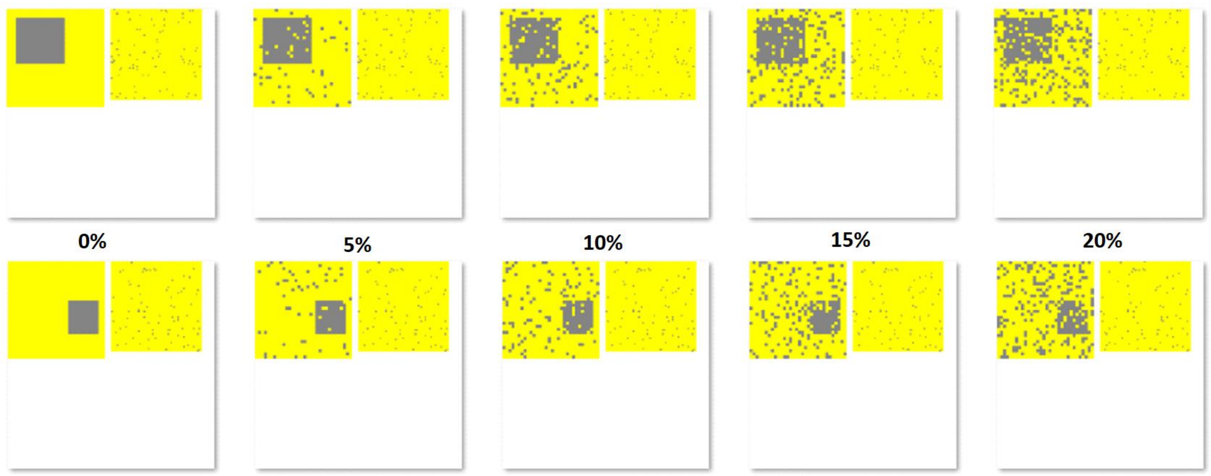

Figure 5
Sparse representation of images Box and Box2, their noised input. Images on top are related to Box1 and images on bottom are related to Box2. Each image represents the input vector with specific noise level and their sparse representation of active columns.

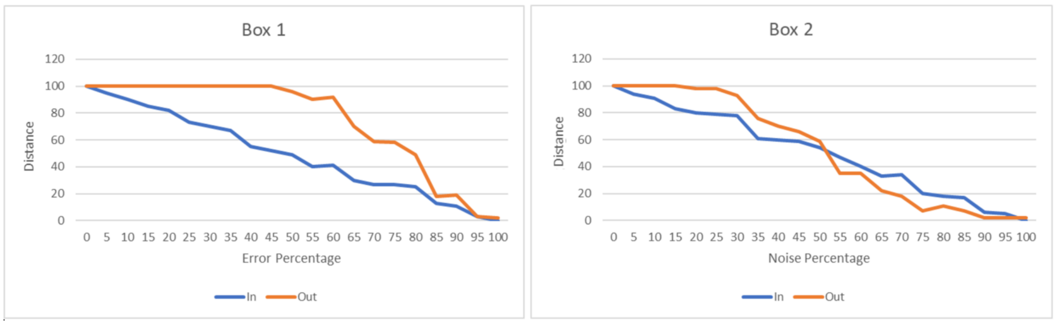

Figure 4
Noise resistance of two boxes. Blue line shows distance between noised input vectors and reference input vector without noise. Orange line shows hamming distance between output of trained Spatial Pooler of noised input and output of reference input.

These results show that more non-zero bits in the input vector lead to more resistant sparse representation of active columns (output). To validate this hypothesis multiple random input vectors have been used by varying percentage of non-zero bits with given noise level. Results are shown at Figure 6.

By comparing results with enabled learning (Figure 6 on left) and not enabled learning (Figure 6 on right), it can be concluded that Spatial Pooler keeps stable SDR output with enabled learning. Using of Spatial Pooler with disabled learning does not memorize the pattern and it is not resistant against noise. As shown at figures on right for all experiments, the output distance changes even faster than the input distance, especially for lower noise levels, which are more applicable in the real-world applications. Also, with enabled learning by using less than 10% of non-zero bits, the Spatial Pooler does not show good resistance (first figure on left). In this case, even slightly deforming of an input immediately deforms the output, which is not the wanted result.

Fortunately, with enabled learning and a/the higher number of non-bits input, the distance between output of noised vector and a reference output with zero-noise keeps almost constant for relatively wide range of noise. This indicates that Spatial Pooler “remembers” reference value even under noisy conditions.

For example, by increased number of non-zero bits, like 20% (second figure on left) the output keeps output stable with more than 90% of output distance by almost 25% of noise in the input. With 30% of non-zero bits, Spatial Pooler will keep almost unchanged output by up to 50% of destroyed input. This is a good indicator for the robustness of Spatial Pooler. It clearly shows that an increased number of non-zero bits in input leads to a higher robustness. Unfortunately, higher number of non-zero bits decreases sparsity and reduces capacity [2] of Hierarchical Temporal Memory.

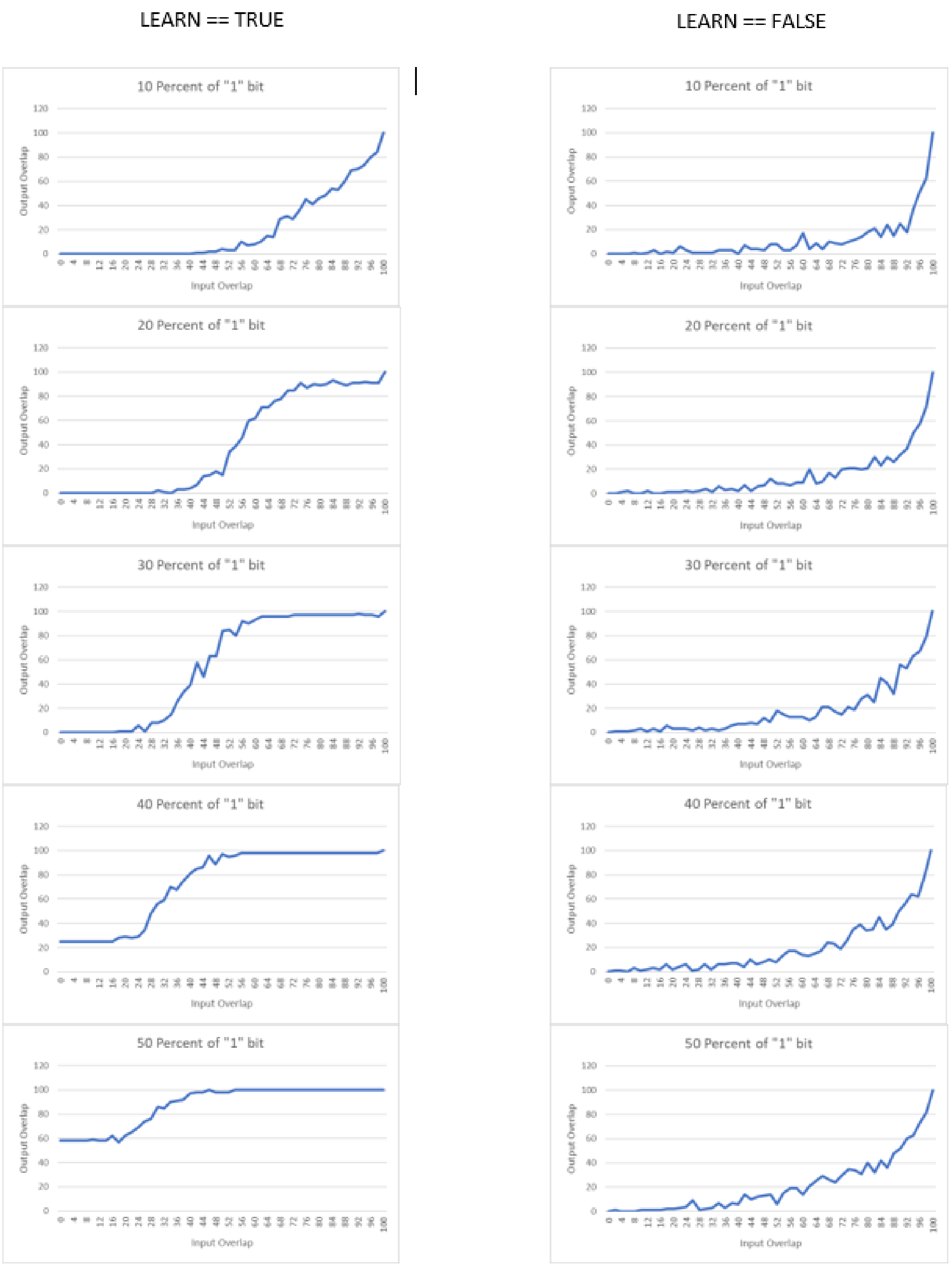

Figure 6
Influence of number of non-zero to SDR representation of active columns with training enabled (on left) and training disabled (on right)

3 Conclusion

As results show, the number of non-zero bits in input vector, which encodes a specific spatial pattern, has a massive influence on noise robustness of the Spatial Pooler. To better remember some spatial pattern and achieve better robustness on the noise, the higher number of non-zero bits should be used when encoding an input. Biologically, it means that more energy in sensory input leads to be higher robustness. However, the increase of the number of non-zero bits will also lead to a decrease in the overall capacity of the memory and higher probability of false positives. This is a trade-off, which needs to be considered when building applications.

In a case of encoding of input by encoders (i.e. scalar encoder, category encoder, etc.) as proposed by HTM framework, encoders can mostly be configured to strike-out encoding values. However, if using images or similar spatial patterns, some other technique must be used. To achieve better results for such kind of inputs, it is proposed to implement a new spatial boosting input layer or a boosting encoder. Note that this is not related to existing column boosting algorithm, which has a different purpose. The proposed spatial boosting input layer should receive encoded input from encoder, appropriately (by keeping same semantical meaning of input) increase the number of non-zeros by bolding (striking out) the encoded input and then passing it to Spatial Pooler. Finally, this component should also make sure that all semantically different patterns have a similar sparseness, because different sparseness of inputs will lead to different robustness against noise.

References

[1] Yuwei Cui, Subutai Ahmad, and Jeff Hawkins, “The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding,” *Frontiers in computational neurosince,* vol. , no. 111, p. 15, 2017.

[2] Jeff Hawkins, Subutai Ahmad, “Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex,” *Frontiers in Neural Circuits,* vol. 10, p. 13, 2016.

[3] Maciej Wielgosz, Marcin Pietroń, Kazimierz Wiatr, “Using Spatial Pooler of Hierarchical Temporal Memory for object classification in noisy video streams,” *Frontiers in Neural Circuits,* vol. 10, p. 13, 2016.

[4] D. Dobric, “NeoCortexApi,” 2019. [Online]. Available: https://github.com/ddobric/neocortexapi/blob/master/README.md.

https://www.edas.info/uploadPaper.php