Spatial Pooling as Feature Selection Method for Object Recognition

Murat Kirtay, Lorenzo Vannucci, Ugo Albanese,
Alessandro Ambrosano, Egidio Falotico and Cecilia Laschi *

The BioRobotics Institute, Scuola Superiore Sant'Anna, Pontedera (PI), Italy

Abstract. This paper reports our work on object recognition by using the spatial pooler of Hierarchical Temporal Memory (HTM) as a method for feature selection. To perform the recognition task, we employed this pooling method to select features from COIL-100 dataset. We benchmarked the results with state-of-the-art feature extraction methods while using different amounts of training data (from 5% to 45%). The results indicate that the performed method is effective for object recognition with a low amount of training data in which state-of-the-art feature extraction methods show limitations.

1 Introduction

Object recognition is a crucial task in many robotic applications. A key component of an object recognition system is a feature extractor, capable of detecting useful pieces of information that can improve the actual recognition. In particular, object recognition is employed in industrial environments, where computational efficiency and storage benefits must be taken into consideration, to perform visually guided manipulation. In such cases, it could be that the image has a very low resolution or a uniform background where there are no sharp local changes in the contrast. This can impair some state-of-the-art feature extraction methods that rely on these image properties, such as scale-invariant feature transform (SIFT), and speeded up robust features (SURF) [1]. In this study, we employed a feature selection method that can work without such assumptions, employing the spatial pooler phase of Hierarchical Temporal Memory, a Neo-cortically inspired machine learning algorithm [2]. This method is employed in a full object recognition pipeline that includes a multi-class support vector machine (M-SVM) [3] to perform the actual recognition starting from the features extracted. The whole process is tested on the Columbia University Image Library (COIL-100), a collection of images taken in a controlled indoor environment with an artificially created uniform background. The employed method is benchmarked against other feature extraction methods capable of dealing with such limitations, such as: using raw pixels, 3D color histograms and Histogram Oriented Gradients (HOG) [1]. The obtained results show that, with this method, the recognition accuracy is improved even with a smaller amount of training data, therefore proving the usefulness of the approach.

*This project has received funding from the European Union Horizon 2020 Research and Innovation Programme under Grant Agreement No. 720270 (HBP SGA1).

2 Methods

The recognition pipeline consists of three distinct processing parts: preprocessing, feature selection and recognition. We note that the main purpose is to benchmark the recognition performance of feature extraction methods by using a lesser amount of training data.

2.1 Preprocessing

In this part, the images are transformed and downscaled into preprocessed vectors to be inputs of either feature extraction phase or directly to M-SVM. These steps are unique for each feature extraction method. For instance, when employing raw pixels or 3D color histograms, only normalization of the image was performed. Whereas, HOG requires normalization of grayscaled images and the computation of horizontal and vertical gradients of the images in a predefined size of pixel length. Finally, for the spatial pooler (cortical activations), the images were grayscaled and thresholded to be binarized. As the last step, the generated input vectors were randomly grouped into training, validation and test sets. The size of training and validation sets were chosen to be the same (from 5% to 45%), while the remaining data were used as the test set.

2.2 Feature Selection: Spatial Pooler Method

To derive features as cortical activations from a given visual pattern, we follow the formalization provided in the study [4] by using the neuroscientific concepts in [2] and implementation in [5]. Constructing a spatial pooler region requires predefining a number of parameters which are n as the total number of patterns (7200), p as the vectorized length of a pattern (1024), m as a total number of cortical columns (1024) and q as number of proximal dendrite synapses (100). To perform spatial pooling operations, $\mathbf{U} \in \{0,1\}^{n \times p}$ represents the binarized input matrix, $\mathbf{c} \in \mathbb{N}^{1 \times m}$ defines the indices of constructed cortical columns and $\Phi \in \mathbb{R}^{m \times q}$ is used for a set of randomly initialized synaptic permanence values for each columns. The connections between cortical columns and an input pattern elements were formed by using set of proximal synapses, $\Lambda \in \{0, \ldots, p-1\}^{m \times q}$. In Figure 1, the cortical columns were shown as cylinders, a row-vector of the $\mathbf{U}$ shown as a binary input vector, some of the proximal synapses for Λ_0 and Λ_8 are illustrated as solid and dashed lines. Additionally, the

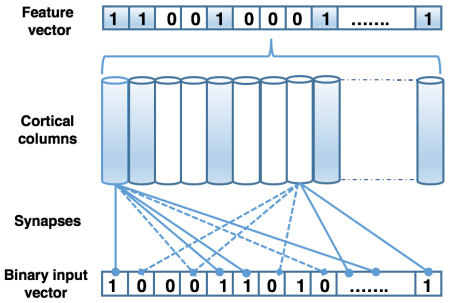


Fig. 1: Feature selection flow from binarized inputs to the cortical activations.

operations of spatial pooler require defining several functions as follows. In that, I(ω) is an indicator function which yields 1 if ω is true, otherwise 0. kmax(S, k) returns kth largest element of S and $max(\mathbf{v})$ returns maximum value of the given vector ($\mathbf{v}$). A clipping function denoted as clip($\mathbf{M}$, lb, ub), defined to return a bounded matrix with $[lb, ub]$ where values smaller than lb assigned as lb, and values larger than ub assigned as ub. Finally, $\odot$ and $\oplus$ indicate element-wise multiplication and sum, respectively.

The spatial pooler computations are performed into three distinct phases: overlap, column inhibition and learning. The purpose of overlap is to derive the total number of active proximal synapses (shown as $\alpha \in \mathbb{R}^{1\times m}$) for each column to decide whether a column should be active. To derive α, we obtained the number of synaptic permanence values which are above the threshold, as $p_s = 0.5$. $\mathbf{X} \in \{0,1\}^{m\times q}$ represents set of inputs for the columns. $\mathbf{Y}$ is a matrix that contains bitmasked values for synaptic permanence values which computed as $(\mathbf{Y} \equiv I(\Phi_i \geq p_s) \ \ \forall_i)$. $\hat{\alpha} \in \mathbb{N}^{1\times m}$ holds sum of the active connected proximal synapses as $\hat{\alpha}_i = \mathbf{X}_i\mathbf{Y}_i$. In Figure 1, an active synapse shown a solid line with circular end point while dashed line with diamond endpoint indicated inactive one. To encourage less frequently active columns to compete for activation, boosting values (as $\mathbf{b} \in \mathbb{R}^{1\times m}$) are assigned to the associated columns (i.e, $\mathbf{b}_i$ is boosting value for the column $\mathbf{c}_i$). As a final step of this phase, the elements of α vector was derived by $\hat{\alpha}_i\mathbf{b}_i$ where $\hat{\alpha}_i \geq p_d$ where $p_d = 10$ refers to the proximal dendrite activation, otherwise it is assigned to be 0. Column inhibition phase aims to compute the set of active columns, blue colored cylinders in Figure 1, while achieving desired column activity which is shown as p_c and set to be 20% of all columns. The neighborhood mask matrix ($\mathbf{H} \in \{0,1\}^{m\times m}$) indicates that $\mathbf{H}_i$ are the neighborhood columns for $\mathbf{c}_i$. In particular, if $\mathbf{H}_{i,j}$ is 1 the $\mathbf{c}_j$ column assumed to be a neighbor of $\mathbf{c}_i$. To determine which columns should be active after inhibition, the lower-bounded k^{th} largest overlap ($\gamma \in \mathbb{N}^{1\times m}$) was computed by $\gamma \equiv max(kmax(\mathbf{H}_i \odot \alpha, p_c), 1) \ \ \forall_i$. Lastly, $\hat{\mathbf{c}} \in \{0,1\}^{1\times m}$ is an indicator vector constructed for the active columns by $\hat{\mathbf{c}} \equiv I(\alpha_i \geq \gamma_i) \ \ \forall_i$. Learning phase consists of three subphases: synaptic permanence adaptation, boosting operations and updating inhibition radius. To begin with the first subphase, the permanence matrix (Φ) is adjusted by $\Phi \equiv clip(\Phi \oplus \delta\Phi, 0, 1)$ where $\delta\Phi$ is calculated by $\delta\Phi \equiv \hat{c}^\intercal \odot (\phi_+\mathbf{X} - (\phi_-\neg\mathbf{X}))$. Performing this subphase provides the adapted values for the synapses (Φ_i) that exist on a column ($\mathbf{c}_i$). We employed values of $\phi_+ = 0.001, \phi_- = -0.001$ for this subphase. In the next subphase, the columnar and synaptic level boosting operations carried out to determine how frequently a column become active and boost the less frequently active columns to compete. In that, $\eta^{(a)} \in \mathbb{R}^{1\times m}$ refers to set of active duty cycle for all columns where $\eta^{(min)} \in \mathbb{R}^{1\times m}$ as the minimum active duty cycles for all columns derived by $\eta^{(min)} \equiv k_a max(\mathbf{H}_i \odot \eta^{(a)}) \ \ \forall_i$ where $k_a = 0.001$ is minimum activity level scaling factor. Based on the comparison of $\eta_i^{(a)}$ and $\eta_i^{(min)}$ the boosting value of columns is determined by $\mathbf{b} \equiv \beta(\eta_i^{(a)}, \eta_i^{(min)}) \ \ \forall_i$. This boosting function returns 1 if the $\eta_i^{(a)} > \eta_i^{(min)}$, returns $\beta_0 = 10$ as maximum amount of boosting if $\eta_i^{(min)} = 0$, otherwise it returns $\eta_i^{(a)}(1-\beta_0)/\eta_i^{(min)} + \beta_0$.

Boosting for permanence values was applied on a vector $\eta^{(o)} \in \mathbb{R}^{1\times m}$ where $\eta_i^{(o)}$ is the overlap duty cycle for each $\mathbf{c}_i \in \mathbf{c}$. The permanence values boosted by $\Phi \equiv clip(\Phi \oplus k_b p_s I(\eta_i^{(o)} < \eta_i^{(min)}), 0, 1)$ where $k_b = 0.1$ refers to the permanence boosting scaling factor. In the last subphase, the inhibition radius should be updated to adjust the average receptive size (input space) between columns and their corresponding connected synapses, yet we prefer to use fixed desired column activation level (p_c) to represent a feature vector with a constant number of active bits. To generate a feature vector, the aforementioned spatial pooler operations will be performed for each input that exists in $\mathbf{U}$.

2.3 Recognition: Multi-Class Support Vector Machine

A multi-class support vector machine was trained to optimize the Hinge loss via gradient descent [3]. The generated feature vector annotated with x_i and a bias node and bias vector were concatenated to x_i and the weight matrix (W), respectively. Then, the score function for an object j ($f(x_i, W)_j$) was computed by taking the j-th column of the result of the multiplication between x_i and W. The loss value for i-th feature vector is defined as $L_i = \sum_{j\neq y_i}^{M} (max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)) + \lambda||W||_2$, and was computed by adding 1 as a soft margin (Δ) parameter to the object score of the actual object y_i ($f(x_i, W)_{y_i}$), and by summing the losses over all the object represented in the dataset (M), except the actual one. To avoid overfitting, we added a L2 regularization factor with a hyperparameter λ. The calculated loss for each input is averaged over the dataset to get the full loss, $L_{full} = \frac{1}{N}\sum_{i=1}^{N} L_i$. To find values for the hyperparameters such as learning rate for the gradient descent and λ, cross-validation employed for model selection, with an early stopping criterion. In particular, we stopped the training if the accuracy rate of the validation set remained below a threshold of 0.01 for 20 iterations. Then, the best model (with hyperparameters) was used to recognize the objects in the test set images. To further analyze the recognition performances of each feature extraction methods, the recognition procedures were performed with different training set sizes and different recognition metrics, such as average accuracy and F_1 scores, were computed.

3 Dataset description and reproducibility of the study

The object recognition task was carried out on the Columbia University Image Library (COIL-100) [6]. The dataset consists of 100 objects with 72 color images (7200 in total) for each object that captured by rotating 5 degrees on a motorized turntable. Note that, several preprocessing operations were performed on the images such as cropping images from the background and resizing them to 128×128. The images downsized to 32×32 for our recognition pipeline. The related source code for this study can be found on the repository named *ESANN2018*[1].

[1] www.github.com/muratkirtay/ESANN2018

4 Results

This section provides the results for the object recognition performed by the proposed pipeline with the four different feature vectors. The results were benchmarked by increasing the training set size by 5% (from 5 to 45%) and using a validation set having the same size of the training set. Then, we used the remaining data for the test sets to obtain recognition performance metrics including accuracy, precision, recall, and F_1 scores. The accuracies on the test sets at different training percentages are illustrated in Figure 2, where cortical activations, raw pixels, histogram oriented gradients and 3D color histograms can be compared. By comparing the increment trends in these curves, we observe that the use of cortical activations as input yielded better accuracy rates than using other feature vectors while employing less amount of training data. For instance, recognition accuracy by using a training percentage of 5% obtained an 82.0454% accuracy for cortical activations, 63.3939% for raw pixels, 56.5909% for HOGs and 60.3787% for 3D histograms. To further analyze the results we constructed

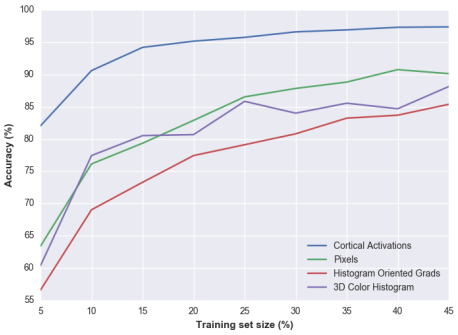
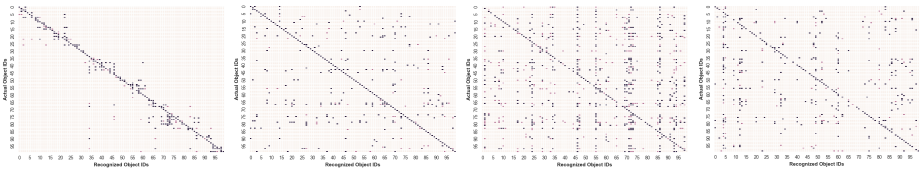


Fig. 2: Accuracy rates for different training set sizes.



(a) CA, F1:0.9060 (b) Pixel, F1:0.7613 (c) HOGs, F1:0.6903 (d) 3D Hists, F1:0.774

Fig. 3: Confusion Matrices and Average F_1 Scores for Cortical Activations (CA), Raw Pixels (Pixels), Histogram Oriented Gradients (HOGs) and 3D Color histograms (3D Hists).

confusion matrices as heat maps for each feature vector at 10% training size. These matrices are depicted in Figure 3 and the actual object IDs and the recognized object IDs are presented, respectively, as the rows and the columns of the matrices. These figures can be interpreted in this manner: the densely populated main diagonal indicates the success in recognition while denotes a sparsely populated matrix lower performances. In these visualizations, it can be observed that the elements in the matrices for Figure 3(b), 3(c), 3(b) are more sparse. Conversely, in the case of cortical activations in Figure 3(a), the elements tend

to be more concentrated on the main diagonal. In order to numerically explain this observation, we report the average F_1 scores, computed as the harmonic mean of precision and recall. Consistent with the confusion matrices, the F_1 scores obtained for the recognition algorithms follow the same trend, ($CA_{F_1} > 3DHist_{F_1} > Pixel_{F_1} > HOG_{F_1}$) that we previously observed in the accuracy values ($CA_{accuracy} > 3DHist_{accuracy} > Pixel_{accuracy} > HOG_{accuracy}$).

5 Conclusions

The use of spatially pooled features (cortical activations) gives rise to a significant improvement in recognition accuracy rates while employing less amount of training data, compared to the other feature extraction algorithms considered. Moreover, the performed feature selection is capable of working with the dataset composed of images with low resolution and few changes in the contrast. Thus, our ongoing studies aim to utilize this method on the dataset that constructed by employing the iCub humanoid robot for recognition and grasping tasks where environmental changes are controlled, and the state-of-the-art methods show limitations or poor performance [7]. The generality of the computational and representation aspects of the Neo-cortically inspired spatial pooling could be further exploited to develop a hierarchical framework, in a realistic simulation platform [8], which allows for multimodal (e.g., vision and tactile) sensory representations of the objects for recognition.

References

[1] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[2] Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in Neural Circuits*, 10:23, 2016.

[3] J Weston and C Watkins. Support Vector Machines for Multi-Class Pattern Recognition. *Proceedings of the 7th European Symposium on Artificial Neural Networks (ESANN-99)*, (April):219–224, 1999.

[4] James Mnatzaganian, Ernest Fokoue, and Dhireesha Kudithipudi. A mathematical formalization of hierarchical temporal memory's spatial pooler. *Frontiers in Robotics and AI*, 3:81, 2017.

[5] Murat Kirtay, Egidio Falotico, Alessandro Ambrosano, Ugo Albanese, Lorenzo Vannucci, and Cecilia Laschi. *Visual Target Sequence Prediction via Hierarchical Temporal Memory Implemented on the iCub Robot*, pages 119–130. Springer International Publishing, Cham, 2016.

[6] Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Object image library (coil-100). Technical report, 1996.

[7] M. Kirtay, U. Albanese, L. Vannucci, E. Falotico, and C. Laschi. A graspable object dataset constructed with the icub humanoid robot's camera. Technical report, 2017.

[8] Egidio Falotico and et al. Connecting artificial brains to robots in a comprehensive simulation framework: The neurorobotics platform. *Frontiers in Neurorobotics*, 11:2, 2017.