Evaluating Shape Representations for Maya Glyph Classification

GÜLCAN CAN, JEAN-MARC ODOBEZ, and DANIEL GATICA-PEREZ, Idiap Research Institute and École Polytechnique Fédérale de Lausanne (EPFL)

Shape representations are critical for visual analysis of cultural heritage materials. This paper studies two types of shape representations in a bag-of-words based pipeline to recognize Maya glyphs. The first is a knowledge-driven Histogram of Orientation Shape Context (HOOSC) representation, and the second is a data-driven representation obtained by applying an unsupervised Sparse Autoencoder (SA). In addition to the glyph data, the generalization ability of the descriptors is investigated on a larger-scale sketch dataset. The contributions of this paper are four-fold: 1) the evaluation of the performance of a data-driven auto-encoder approach for shape representation; 2) a comparative study of hand-designed HOOSC and data-driven SA; 3) an experimental protocol to assess the effect of the different parameters of both representations; and 4) bridging humanities and computer vision/machine learning for Maya studies, specifically for visual analysis of glyphs. From our experiments, the data-driven representation performs overall in par with the hand-designed representation for similar locality sizes on which the descriptor is computed. We also observe that a larger number of hidden units, the use of average pooling, and a larger training data size in the SA representation all improved the descriptor performance. Additionally, the characteristics of the data and stroke size plays an important role in the learned representation.

CCS Concepts: •**Computing methodologies** → **Shape representations; Neural networks;** •**Applied computing** → **Arts and humanities;**

Additional Key Words and Phrases: HOOSC, sparse autoencoder, sketch, Maya glyph

ACM Reference Format:

Gülcan Can, Jean-Marc Odobez, Daniel Gatica-Perez. 2015. Evaluating Shape Representations for Maya Glyph Classification. *ACM Trans. Appl. Percept.* V, N, Article 1 (April 2016), 26 pages.
DOI: 0000001.0000001

1. INTRODUCTION

The ancient Maya civilizations flourished from around 2000 BC to 1600 AD and left behind a great amount of cultural heritage materials. These can be found in stone monument inscriptions, folded codex pages, or personal ceramic items. The common ground of all these materials are the Mayan hieroglyphs, in short glyphs, written on them. The Maya writing system is visually complex (see Fig. 1) and new glyphs are discovered with almost every new archaeological site study. This brings the necessity of better digital preservation and storage systems. Besides, the annotation of a small fraction of glyphs is still open to discussion between scholars due to either visual differences or semantic analysis. Some glyphs are damaged or have many variations due to artistic reasons and to the evolving language, i.e., differences with the era and place in which glyphs were produced. Fig. 2 shows the

This work is supported by SNSF through the MAAYA project ("Multimedia Analysis and Access for Documentation and Decipherment of Maya Epigraphy").
Authors' address: Idiap Research Institute, Centre du Parc, Rue Marconi 19, 1920 Martigny, Switzerland; G. Can, email: gcan@idiap.ch; J. M. Odobez, email: odobez@idiap.ch; D. Gatica-Perez, email: gatica@idiap.ch.

DOI: 0000001.0000001

variations of some glyphs in the top two rows. For instance, the last sample in the purple box is an impersonated glyph sample (glyphs designed as animal or human head figures), which makes it visually different than the other variations. The last sample from the dark blue box (second row, first box) is an example of a twin glyph variation (duplicated glyph). On the other hand, the boxes in the bottom row (red and yellow) show inter-class visual differences that can be quite subtle for some categories. Even though it is a challenging task to build a visual analysis tool for Maya glyph data, it would be quite useful for scholars as part of their annotation and deciphering tasks.

In this work, we collaborate with archaeologists and epigraphers to define potential use cases of glyph classification tools. For instance, for training purposes, when a novice archaeologist sketches a glyph, the system could provide the most likely categories that this glyph would belong to. Another potential use case is that the archaeologist could take a photograph of a glyph instance from a stone monument or folded codex and ask the system to categorize this instance. If the visual categorization system is reliable, the archaeologist would not need to search through the whole existing catalog to annotate the glyph instance. For damaged glyph instances, even though the opinions of archaeologists may differ due to the subjectivity of visual appearance, the system could be utilized by the archaeologists to get an additional quantitative assessment of visual similarity. Furthermore, for archaeologists, visual details at different scales are important to recognize a glyph instance. This points towards the necessity to study visual representations at different spatial contexts. Moreover, the glyph instances in a category may exhibit many variations (see Fig. 2) yet be consistent with respect to certain diagnostic details. In our methodology, a bag-of-words representation is a possible way to address this issue, as the shared diagnostic details of the glyph instances from the same category may correspond to specific bins in the histogram representation after clustering the patch-based local representations. Our work contributes to an essential component of visual categorization systems, namely how to represent the visual appearance of the glyphs.

This motivates the study of visual representations that accurately discriminate glyph categories, and in general arbitrary shapes. The focus of this work is the systematic comparative analysis of knowledge-driven visual descriptors and data-driven representations on binary shape datasets. Specifically, these are the previously proposed HOOSC descriptor for Maya glyphs [Roman-Rangel et al. 2011] and the shape representation learned by a single-hidden-layer sparse autoencoder [Hinton and Zemel 1994]. Similar to SIFT [Lowe 1999] and HOG [Dalal and Triggs 2005], which are quite common in the object recognition literature, HOOSC is a local descriptor that summarizes the shape regions as frequency histograms of line orientations. HOOSC is preferred in this study due to its competitive performance in hieroglyphic representation tasks [Roman-Rangel 2012; Franken and van Gemert 2013].

On the other hand, sparse autoencoder representations have attracted attention in computer vision due to their ability to capture hidden structures in the data in an unsupervised manner. The learned representation is expected to be a non-linear mapping that would cover edges in different angles and translations. Such data-driven representations may be more beneficial to capture data-specific patterns. Key research questions are how generic these representations are, and how well such representations perform compared to established handcrafted descriptors in the shape representation task.

In this work, we focus on single-hidden-layer sparse autoencoders, since it is worth understanding the limits of representation learning with “shallow” networks in comparison to hand-designed descriptors. Alternative “deep” networks such as Convolutional Neural Networks (CNN) with many layers require separate studies with tasks including: feature learning; adaptation of CNN models trained from different sources, i.e. natural images, sketches, or handwritten Asian characters and assessing which source is closer to Maya writings in a classification or retrieval system; pixel-wise document image binarization; and glyph localization.

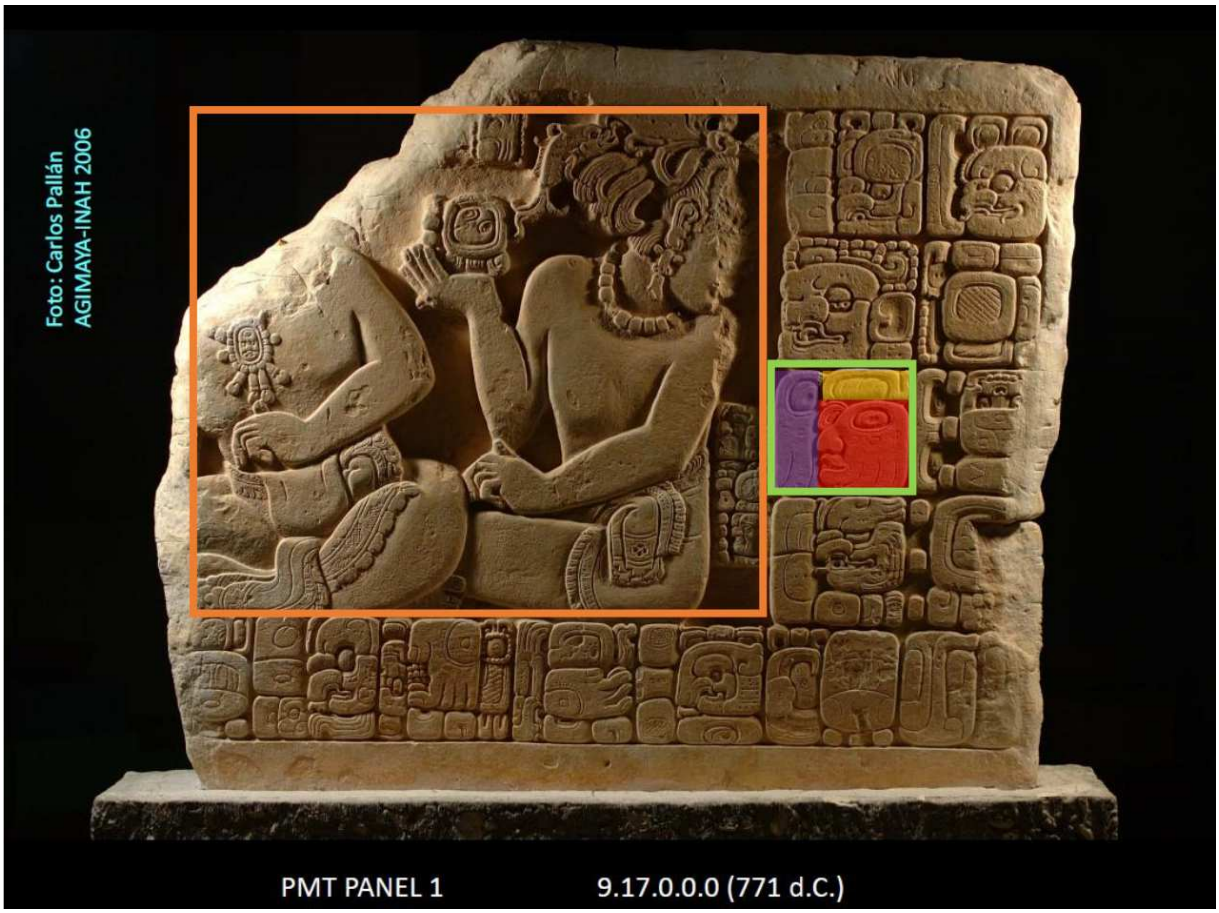

Fig. 1. A stone inscription found in Pomona, Tabsco (Mexico) Panel 1 from 771 AD. Photograph by Carlos Pallán Gayol for the AJIMAYA/INAH Project © 2006. Instituto Nacional de Antropología e Historia, Mexico. The icon area is marked with an orange bounding box. The rest of the inscription demonstrates Maya text (glyphs). One of the glyph blocks is marked with a green bounding box, and each glyph in this block is highlighted (yellow, purple, red).

In this paper, we investigate the effects of model parameters of single-hidden-layer sparse autoencoders (including regularization parameters, number of hidden units), and the scale of the dataset on the classification results, and whether they are able to give comparable results to traditional descriptors. In this respect, we quantify how effective and generic each of these representations are on a Maya monument dataset as well as on a much larger human sketch dataset. The contributions of this paper are the following:

(1) the evaluation of the performance of a data-driven auto-encoder approach for shape representation;
(2) a comparative study of hand-designed HOOSC and data-driven SA, which to our knowledge has not been conducted previously;
(3) an experimental protocol to assess the effect of the different parameters of both representations;
(4) the creation of a bridge between computer vision and machine learning and Maya studies, specifically for visual analysis of glyphs.

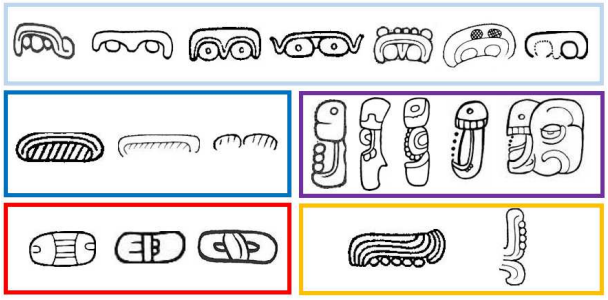

Fig. 2. Selected Maya glyph samples from several categories that illustrate the within-class variety and between-class similarity. The first two rows show samples from the same category in each box. The last row shows visually similar samples from different categories. See the text for details.

From our experiments, the data-driven representation performs overall in par with the hand-designed representation for similar locality sizes on which the descriptor is computed. We also observe that a larger number of hidden units, the use of average pooling, and a larger training data size in the SA representation all improved the descriptor performance. Additionally, the characteristics of the data and stroke size plays an important role in the learned representation.

The rest of the paper is organized in five sections. Section 2 gives a brief insight of related work on object recognition and discusses some of the representations in computer vision for this task. Section 3 explains the classification pipeline used in this paper, and describes the two studied representations. Section 4 summarizes the experimental set up, dataset details, and parameter setting. In Section 5, the results are discussed with respect to certain aspects and parameters of the pipeline. Finally, Section 6 concludes the paper by pointing out the potential and limitations of both representations, and hints about future directions.

2. RELATED WORK

Object recognition architectures are generally composed of three steps: low-level feature extraction, coding, and pooling. Low-level features of an image, such as color, intensity values, or orientation of the edges, can be extracted locally over windows sliding through the image or over a defined region. The coding step relies on the availability of a dictionary. It is often obtained by clustering low-level feature vectors of all images. The dictionary is a more abstract representation, aiming to group and characterize repetitive patterns in the features across images. Given a dictionary, low-level local features of each image are coded, for instance, by quantizing them to their nearest element in the dictionary or by extracting their sparse decomposition with respect to the dictionary vectors. Finally, in the pooling step, the coded features are combined spatially (e.g. through averaging or by taking the maximum value) and a global representation is obtained for the whole image.

Literature in the last ten years has investigated the different modules of this common pipeline. [Boureau et al. 2010] use the Caltech-101 object recognition and 15-scene recognition datasets as benchmarks. They utilize dense SIFT [Lowe 1999] as basic image features and propose “macro-features” in which neighboring features are encoded jointly. Macro-features improve only the object recognition results. Another similar study [Chatfield et al. 2011] analyzes feature encoding methods for bag-of-words image representations (BoW) [Sivic and Zisserman 2003] in object classification tasks for the PASCAL VOC and Caltech-101 datasets. The selected methods are locality-constrained linear encoding (LCL) [Wang et al. 2010], improved Fisher vector (FV) [Perronnin et al. 2010], super vector (SV) encoding [Zhou et al. 2010] and kernel codebook (KC) encoding [van Gemert et al. 2008]. Compared

to the traditional hard quantization, the above methods keep more information about the original feature vector. Fisher vector coding is reported to give superior results. Furthermore, in the empirical analysis, both larger vocabulary size and higher sampling density are noted as critical aspects for good performance.

There are several common practices in the literature for pooling local descriptors and obtaining a global image representation. Spatial pyramid matching (SPM) [Lazebnik et al. 2006] is a popular approach to incorporate spatial information into BoW representations. This approach treats the image on several spatial levels. For each level, it divides the image into a grid of different granularity, i.e. level-0 (whole image), level-1 (2x2 grid), and level-2 (4x4 grid). Then, the BoWs in each grid cell of each level are normalized according to the cell area and stacked together. The SPM approach might not be optimal for some object classes. Thus, instead of pre-defined regular grids, it has been proposed to learn the spatial cells from an over-complete set of arbitrary rectangles over the image and call these cells receptive fields [Jia et al. 2012]. Another study analyzes hierarchical bag-of-words for scene classification [Battiato et al. 2010]. Each scene is partitioned into subregions hierarchically and described via texton distributions, which yields good performance for this task.

However, in general, the basic way to operate pooling is by taking the sum or max of the features in the defined spatial neighborhood. In the case of deep neural networks, the spatial neighborhoods for pooling can be densely and regularly sampled over the image. As an alternative, the spatial neighborhoods can be simply defined around specific points of interest. Since the main interest of this paper is shape data, we use the latter and sample features around pivot points, which are points from the shape.

In the literature, gradient-based handcrafted features such as SIFT [Lowe 1999] and HOG [Dalal and Triggs 2005], and their variants [Roman-Rangel et al. 2011; Eitz et al. 2012] are popular and have been shown to work well in image recognition tasks. Roman-Rangel et.al. proposed the HOOSC shape descriptor, which is a combination of the HOG and Shape Context (SC) features, for shape retrieval tasks applied to Maya glyphs, ancient Chinese writing, and MPEG-7 shape dataset [Roman-Rangel 2012]. In a more recent study, it was shown that the performance can be boosted by adding the relative position of the pivot points where the HOOSC descriptors are computed [Hu et al. 2015]. Roman-Rangel et al. evaluated hard vector quantization and sparse coding methods and reported that hard quantization is superior to sparse coding approach with l_1 norm on a specific Maya glyph dataset from monuments [Roman-Rangel et al. 2012]. In a similar hieroglyph recognition task for Egyptian writing, HOG, SC [Belongie et al. 2000], and HOOSC [Roman-Rangel et al. 2011] are evaluated [Franken and van Gemert 2013]. All three descriptors were used with a bag-of-words representation, and performed similarly. By introducing spatial matching of the interest points via RANSAC, the results using HOG and HOOSC are further improved.

Recently, a sketch benchmark for 250-objects was released and analyzed in a similar recognition pipeline [Eitz et al. 2012]. The authors proposed a gradient-based fast histogram feature, similar to the SIFT feature, based on fast Fourier transform. It is reported that soft quantization and SVM outperforms hard quantization and k-nearest neighbor approach. As with traditional image recognition tasks, gradient-based shape descriptors work fine for shape recognition tasks. Another study about symbol classification examines the role of local and global shape descriptors [Battiato et al. 2015], where local SC features are clustered to obtain a BoW representation for aligning the shapes, and the Circular Blurred Shape Model (CBSM) [Escalera et al. 2011] is utilized to describe the shapes globally. The CBSM descriptor is defined via correlograms on radial sectors, and rings. The distances of each sampled point on the contour to the center of these radial regions are computed and normalized. The inverse of these distances compose the correlogram distribution. In [Battiato et al. 2015], this approach is applied on the 70-class MPEG-7 silhouette dataset, 17-class symbol dataset [Escalera et al. 2011],

and the aforementioned sketch dataset. Their approach, which includes local aligning of the shapes before describing them globally, is reported as superior compared to the approach in [Eitz et al. 2012], which does not include an aligning step.

Recently, an important trend in machine learning argues that features should be learned from the data rather than designed by hand [Bengio et al. 2013]. This has led to a development of methods that can leverage and enforce sparsity in the representation. The sparse autoencoder (SA), which is an unsupervised neural network, is one such approach. This includes several variants such as denoising autoencoder, contractive autoencoder, and their regularized stacked versions [Bengio et al. 2013]. There have been several promising applications of convolutional and stacked autoencoders to object recognition [Ranzato et al. 2007; Xie et al. 2015], 3D object retrieval [Leng et al. 2015], handwritten Chinese character recognition [Wang et al. 2014], and multiple organ detection [Shin et al. 2013]. A thorough study about single-layer sparse autoencoders was conducted in which SA is used to extract the representation of local patches in an image on the most often used natural image benchmarks [Coates et al. 2011]. This work suggests to use dense sampling, small stride (the distances between sampled patches), and small input patch size (so that the number of features can increase) for performance improvement. In [Chen et al. 2015], the authors discuss convolutional SA-based features for image matching task and compare them with SIFT features. The main finding is that even though SIFT performs better for the given configurations and parameter settings, SA-based features have potential to compete, and that with a larger number of hidden units, SA features gives better performance. On the other hand, assessing the performance of knowledge-driven descriptors compared to data-driven representations for shape data is still an open issue.

In this paper, we propose to investigate the use of SA methods for the recognition of complex shape images. This differs from the studies on MNIST hand-written digits or Chinese characters, because in those studies, shapes are simple and small enough that they can be handled as a whole rather than patches. In our case, SA features are generated on patches similar to [Coates et al. 2011] and [Chen et al. 2015]. However, their focus is on natural images and ours is on complex binary shape data that does not have color or texture.

3. METHODOLOGY

We focus on two representations: the previously proposed HOOSC descriptor; and a convolutional representation learned by a single-hidden-layer sparse autoencoder. Our aim is to study how a fully automatic learned representation competes with hand-designed features for shapes. This is done in a traditional bag-of-words (BoW) recognition scheme. In this section, we first describe our recognition scheme, and then explain the HOOSC descriptor and the SA representation.

3.1 Shape Classification Method

The classification method is illustrated in Fig. 3. The notation is summarized in Table I. Given an input shape image I to be classified, the following five steps are applied.

(1) Sample a set of positions $P = \{p_1, ..., p_i, ..., p_{N_P}\}$ over the image I, where N_P is the number of samples.
(2) At each position p_i, extract a feature vector f_{p_i} as described in either Section 3.2 or Section 3.3, and compose the overall feature vector $F_I = [f_{p_1}, ..., f_{p_i}, ..., f_{p_{N_P}}]$.
(3) Quantize each descriptor f using a dictionary D containing N_D elements to obtain the quantized indices $Q_I = [q_{p_1}, ..., q_{p_i}, ..., q_{p_{N_P}}]$, where each $q_p \in \{1, ..., N_D\}$.
(4) Compute the histogram b_I from Q_I, where $b_I(q)$ contains the number of times q appears in Q_I.
(5) Classify b_I into one of the N_c shape classes.

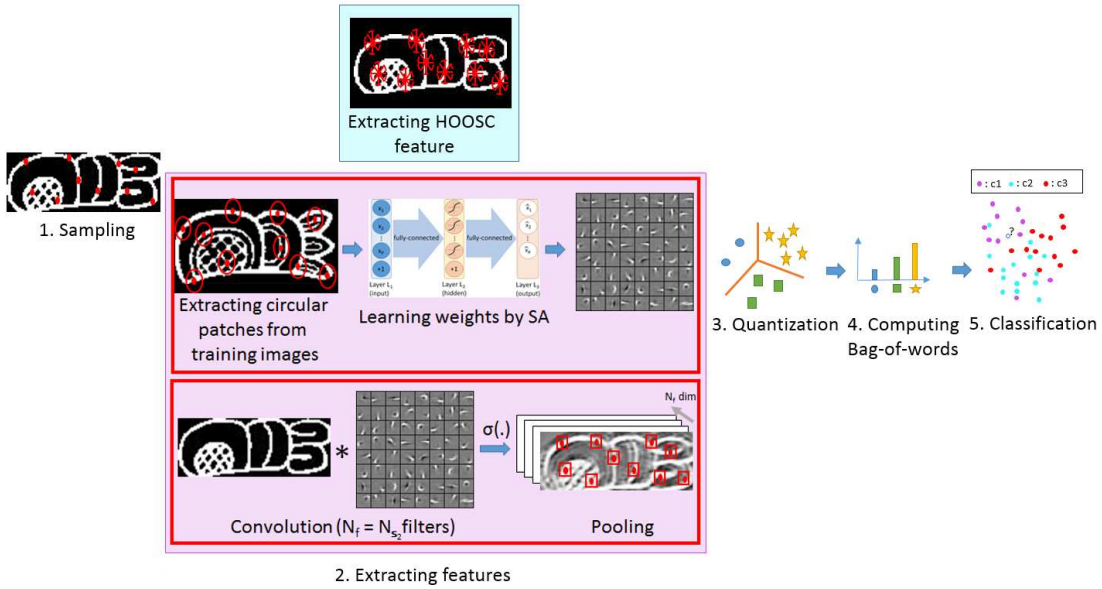

Fig. 3. Steps of the overall classification system. For feature computation, either the HOOSC (blue) or the SA (purple) methods can be used. The SA approach requires a learning phase.

Note that the feature vector f_p around each point p is computed from a local circular patch x_p centered at p, and whose size is defined by the scale sc, defined as a ratio. That is, $sc = 1/1$ means that the patch size is equal to the input image size (defined by the longest edge of the image), $sc = 1/2$ means that it has half the size, and so on. In addition, some of the steps require training. More precisely, in step (2), while the HOOSC computation follows from its definition (see Section 3.2), the SA representation requires learning (as described in Section 3.3). Similarly, before step (3), the dictionary D is learned by clustering a set of features f_p sampled from training images using the k-means algorithm. The learned dictionary D is used in step (3) to quantize each feature f in the overall feature vector F_I. Finally, in step (5), we classify the b_I representation of the image into a shape class using the k-nearest neighbor method.

3.2 Handcrafted Features: Histogram of Orientation Shape Context

The Histogram of Orientation Shape Context (HOOSC) is a shape descriptor previously proposed for Maya glyphs [Roman-Rangel et al. 2011]. The HOOSC feature is computed in two main steps as illustrated in Fig. 4. First, the orientation of all image points is computed. This is done using the software by [Kovesi 2015] ("featureorient" function), which extracts orientations from the local tensor of gradient directions, after image smoothing using a Gaussian filter. Note that this method differs from the original HOOSC definition [Roman-Rangel et al. 2011], in which the orientations are computed using simple pairwise pixel differential operations, for which the shapes are thinned to their skeletons by morphological operations. Our method improves the local orientations around thick strokes, which avoids the unwanted branches that are usually brought in by the thinning process (the pivots on these branches end up having different orientations even if they are on the same thick contour area and have similar orientations).

Table I. Notation used in the paper.

Notation	Explanation	Applicable to / Used for
I	Input image	HOOSC, SA
p_i	i^{th} position	HOOSC, SA
P	A set of positions	HOOSC, SA
F_I	Overall feature vector	HOOSC, SA
f_{p_i}	Feature extracted for p_i	HOOSC, SA
N_f	Number of dimensions of feature vector	HOOSC, SA
D	Dictionary	HOOSC, SA
N_D	Number of elements in the dictionary D	HOOSC, SA
Q_I	Quantized indices for all sampled positions in image I	HOOSC, SA
q_{p_i}	Quantized indices for p_i in image I	HOOSC, SA
N_P	Number of samples	HOOSC, SA
b_I	Histogram of bag-of-words for image I	HOOSC, SA
sc	Spatial context (spatial extent), see Section 3.1 for definition	HOOSC, SA
N_c	Number of classes	Classifier
k	Number of neighbors in k-NN classification	Classifier
N_a	Number of bins for discretizing oriantation angles	HOOSC
N_r	Number of rings	HOOSC
N_s	Number of spatial slices	HOOSC
$a_i^{(l)}$	Activation of unit i in layer l	SA
L_i	The i^{th} layer	SA
N_l	Number of layers l	SA
N_{sl}	Number of units in layer l	SA
$W^{(l)}$	Weights of layer l	SA
$b^{(l)}$	Bias of layer l	SA
$\sigma(.)$	The activation function (sigmoid)	SA
$h_{W,b}(x)$	Estimated target values for x with W and b parameters	SA
x_i	Value of i^{th} example (a pixel value of a sampled patch)	SA
N_{tr}	Number of training examples	SA
$KL(\rho\|\|\hat{\rho_j})$	KL divergence between sparsity and estimated sparsity	SA
ρ	Sparsity parameter (target mean activation of SA)	SA
β	Coefficient for sparsity cost term	SA
α	Weight regularization parameter (weight decay)	SA

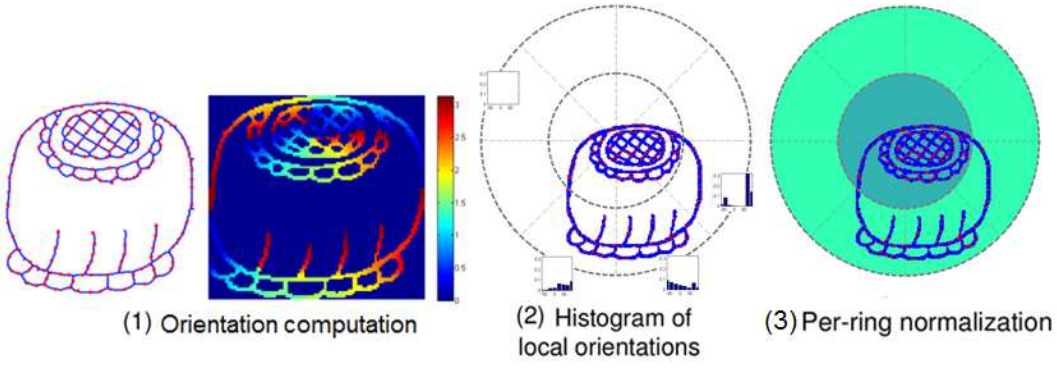

Fig. 4. HOOSC computation at a sample position p of the shape, which includes: computation of 1) pixelwise orientations, 2) histogram of local orientations in each spatial bin, and 3) per-ring normalization of the histograms.

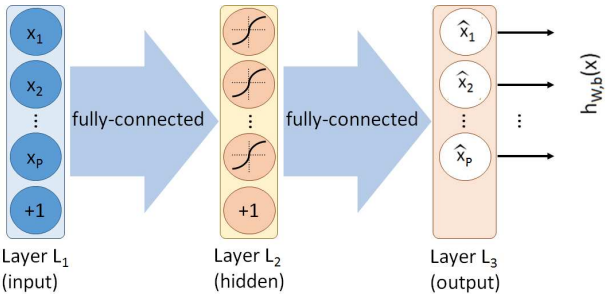

Fig. 5. Sparse autoencoder model. Layer 1 is the input layer (each blue input unit holds the value of one pixel of an input local patch in our application), Layer 2 is the hidden layer, and Layer 3 is the output layer where the reconstructed output activations (which should match the input vector size) are computed (drawing modified from [Ng 2013]).

Secondly, a histogram of local orientations is computed using N_a angle bins within each spatial bin of a circular grid centered at p_i. The HOOSC descriptor is obtained by concatenating all the histograms, and by applying a L_1 normalization for each of the two rings. Note that HOOSC is characterized by several parameters: the distance sc of the spatial context defining the extent of the spatial partition; the number of rings N_r; and the number of slices in a ring N_s. In our experiment, we use $N_a = 8, N_r = 2, N_s = 8$, which resulted in a HOOSC descriptor of $N_f = 128$ dimensions.

3.3 Feature Learning: Sparse Autoencoder

The SA model and its application to shape classification are described below.

3.3.1 *Sparse Autoencoders.* We introduce the SA model following the notation and formulation provided in [Ng 2013]. An autoencoder is a feed-forward unsupervised neural network. Neural networks are composed of three types of layers: an input layer, hidden layers, and an output layer. Units are the building blocks of the neural network architecture; in each layer, they compute an output value from outputs of the previous layer, and are connected to all units in the next layer. Let ${a_i}^{(l)}$ be the output value of unit i in layer l.

For a given input x, the single hidden layer autoencoder is illustrated in Fig. 5 and defined as follows. The first layer L_1 is the input layer. The units in L_1 give the input values as their output. These input values are low-level image features, such as pixel values in general. Thus we have:

$$a^{(1)} = x. \tag{1}$$

The second layer L_2 takes the output values of $a^{(1)}$, applies a linear transformation (multiplication by a weight matrix $W^{(1)}$ and addition of a bias vector $b^{(1)}$), and then passes these values through a linear or nonlinear function, such as the sigmoid function $\sigma(z) = \frac{1}{1+exp(-z)}$, leading to:

$$a^{(2)} = \sigma(W^{(1)}a^{(1)} + b^{(1)}). \tag{2}$$

In general, each element of the weight matrix $W_{ji}^{(l)}$ is the parameter that connects unit i in layer l to unit j in layer $l+1$. Finally, the third layer L_3 is the output layer and the units in L_3 outputs the reconstructed input values $h_{W,b}(x)$ by following the same linear transformation (but not applying the sigmoid function) as in the case of the second layer units. We have thus:

$$h_{W,b}(x) = W^{(2)}a^{(2)} + b^{(2)} \tag{3}$$

When the weights are tied, the transpose of the weight matrix in L_2 is used for the weights in L_3, $W^{(2)} = W^{(1)T}$. We followed this approach, and hence the autoencoder is parameterized by $W = W^{(2)} = W^{(1)T}$, and $b = (b^{(1)}, b^{(2)})$. The weight matrix is learned during optimization by back-propagation.

An autoencoder differs from a traditional neural network, which outputs a class label. The autoencoder, instead, approximates its input x (layer L_1 in Fig. 5) by reconstructing the output $\hat{x}$ (layer L_3 in Fig. 5). The aim of this set up is to capture structure of the data in the hidden layers of the autoencoder (layer L_2 in Fig. 5). In another perspective, SA can be stated as composed of two parts, namely encoder and decoder. The coder maps the input data to the "codes" in the hidden units, i.e., the representation, and the decoder maps back these codes to the output data by minimizing the reconstruction error.

Sparse autoencoders (SA) are regularized autoencoders with an additional sparsity penalty term in the cost function. Let N_{tr} be the number of training examples x_i. Then the SA cost function can be expressed as follows:

$$J_{sparse}(W,b) = J(W,b) + \beta \sum_{j=1}^{N_{s_2}} KL(\rho \parallel \hat{\rho}_j), \tag{4}$$

$$J(W,b) = \left[\frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} \left(\frac{1}{2} \left\| h_{W,b}(x_i) - y_i \right\|^2 \right) \right] + \frac{\alpha}{2} \sum_{i=1}^{N_{s_2}} \sum_{j=1}^{N_{s_3}} \left(W_{ji} \right)^2, \tag{5}$$

where $J(W,b)$ stands for the energy term for neural networks parameterized by the weight matrix W and biases $b = (b^{(1)}, b^{(2)})$. It computes the average reconstruction error and includes a regularization term on the weight matrices. In Eq. 4, $KL(\rho \parallel \hat{\rho}_j)$ denotes the Kullback-Leibler divergence [Kullback and Leibler 1951] between the a priori distribution of the activation unit $a_j^{(2)}$ modeled by a Bernoulli distribution with mean ρ, and the empirical distribution whose parameter $\hat{\rho}_j$ is computed from the training data ($\hat{\rho}_j = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} \left[a_j^{(2)}(x_i) \right]$). The KL divergence is given by $KL(\rho \parallel \hat{\rho}_j) = \rho \frac{\log \rho}{\log \hat{\rho}_j} + (1 - \rho) \frac{\log 1-\rho}{\log 1-\hat{\rho}_j}$.

The divergence measures how distant two distributions are. Thus, by choosing empirically the sparsity parameter ρ to be small and close to zero, we can enforce the codes $a_j^{(2)}(x)$ to be inactive (zero or close to zero) most of the time. In Eq. 4 and 5, several parameters control the importance of the different cost terms: β controls how strong the sparsity penalty term is, while α controls the regularization level.

Training. We utilized the libORF machine learning library [Firat 2015] written in MATLAB. A sigmoid function is utilized as nonlinear hidden unit activation function in the model. The model is learned by backpropagating the parameters through the network. For optimizing the parameters, the mini-batch Stochastic Gradient Descent (SGD) method is utilized. The batch size is empirically chosen as 100, and the optimization is performed for 200 epochs. In order to get away from local minima and saddle points and for faster convergence, momentum and adaptive learning rate methods (namely, Adadelta [Zeiler 2012]) are used. The momentum parameter controls how much the new gradient is affected by the direction and the magnitude of the gradient in the previous step in the optimization, and it is empirically chosen as 0.9. We note that while the limited Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm might provide better-converged weights (smaller test reconstruction error) than SGD [Ngiam et al. 2011], it is computationally demanding (in terms of memory usage), so it was discarded as a choice.

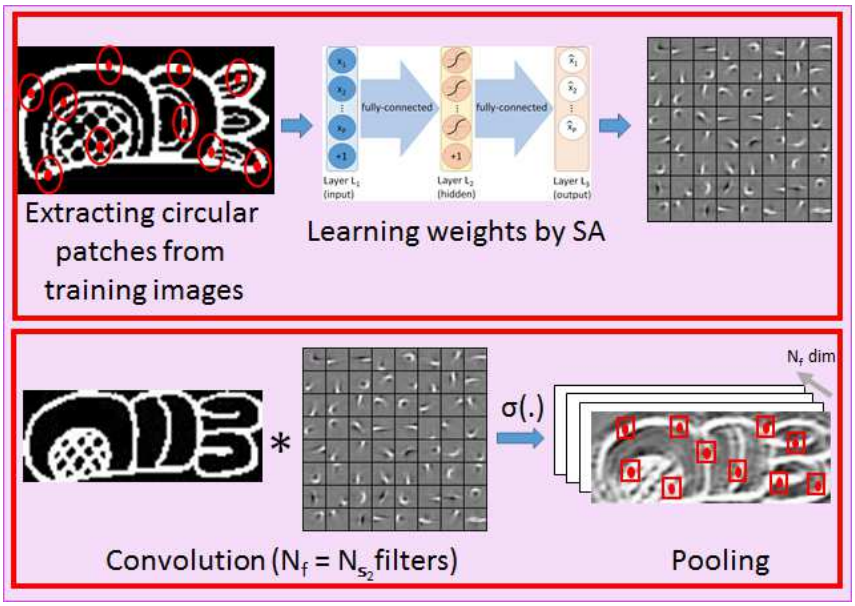

Fig. 6. Applying SA for shape classification. Top box: learning the autoencoder parameters. Bottom box: applying the autoencoder filters to a new shape image.

3.3.2 *Application to Shape Classification.* Applying the autoencoder model to the shape classification task requires two steps: how we learn the encoder parameters (learning step) and how we compute the features for new shape images (feature extraction step). These steps are illustrated in Fig. 6. Applying the SA model requires its input area to be the same as HOOSC's. As we use the same input patches, the SA representation can then be fairly compared to HOOSC.

Learning. To learn the weights W, circular patches centered around sampled positions p and spanning the defined spatial context are cropped from the images. These patches are used as input to the single-layer SA and the weights are obtained as described in the previous subsection. To understand the representation learned by SA, the learned weights are visualized in Fig. 6 (top, rightmost). Each square j of the 8x8 matrix displays the normalized weight values $\hat{W_{ji}}$ connecting the input image x to the activation function $a_j^{(2)}$ of the hidden unit j defined as:

$$\hat{W_{ji}} = \frac{W_{ji}^{(1)}}{\sqrt{\sum_{i=1}^{N_p}(W_{ji}^{(1)})^2}}. \tag{6}$$

Note that the activation unit $a_j^{(2)}$ is maximal when the input patch x indeed corresponds to $\hat{W_{ji}}$. We can thus interpret the visualization of each small square as the type of structure that the corresponding hidden unit will be responsive to. Accordingly, we treat each of these squares as spatial filters that can be applied to an input image. The representation is not only the filter, but involves going through the sigmoid function $\sigma(.)$ to get the activation $a_j^{(2)}$.

Feature extraction. Given a shape image, we need to compute the SA representation at a set of positions p_i of the image (see Section 3.1). Rather than extracting a patch x_{p_i} around each point and

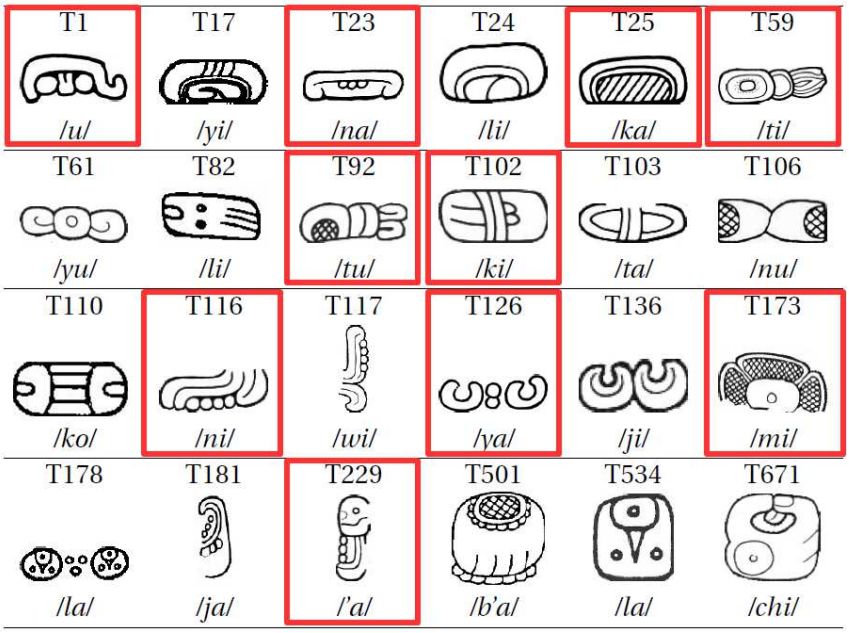

Fig. 7. Sample glyph images, corresponding Thompson annotations and syllabic values (sounds) from 24-class syllabic Maya glyph dataset. The selected 10 classes are outlined with red.

computing its representation f_{p_i} (which is provided by the activations vector $(a_1^{(2)}, ..., a_{N_{s_2}}^{(2)})$), we proceed as follows. To compute the feature map f_{m_j} corresponding to activation unit j, we convolve the image with the filter W_j, and pass the output through the sigmoid function $\sigma(.)$. The second step is to pool (i.e, to aggregate) the feature map responses of each filter. In pooling, the responses within a small neighborhood around the sampled positions p_i are converted to feature vectors f_{p_i} by taking their sum, max, or just taking the response at the sample positions.

4. DATASETS AND EXPERIMENTAL PROTOCOL

While our main interest is in analyzing Maya glyphs, we also want to understand the generalization ability of the representations to other shape categories. With this in mind, we have assessed the HOOSC and SA representations on two datasets, namely the Maya monument glyph dataset and the sketch dataset. This section describes the datasets and the experimental settings.

4.1 Datasets

The data in our experiments are binary shapes, more specifically contours. Due to the lack of color and textural information, this data source has other challenges compared to natural images. The main issues are the visual variations due to missing or extra parts across samples, different writing styles across regions or time (see Fig. 2), or different viewpoints (in the case of sketch data).

4.1.1 *Maya Glyphs.* This dataset contains 24 classes of frequent syllabic glyphs inscripted on monuments, which were collected from four subregions of the Maya area, i.e., Petén, Usumacinta, Motagua, and Yucatán. It is non-trivial to produce this type of data. Epigraphers follow strict quality and fidelity standards to generate hand-drawn glyphs from the original sources. As an additional source, around

300 glyph samples are taken from existing catalogs [Macri and Looper 2003; Thompson and Stuart 1962]. Since this dataset is not balanced (i.e. several glyph categories have very few examples), we only kept the 10 classes with the largest number of examples (marked with red in Fig. 7) for the classification experiments.

All the images are resized to be enclosed in a 128x128 pixel area, i.e. the largest glyph side is set to 128 pixels. To characterize the shapes, we measured the stroke thickness using morphological operators. For this dataset, the mean average stroke thickness is 2.2 pixels, and the mean of the maximum stroke thickness in each image is 5 pixels.

For Maya glyph classification, it is challenging to obtain a large dataset with enough samples for classification task due to the limited amount of sources and the amount of expert work needed to produce data. Even though there are other image sources of glyphs such as glyphs from folded codices or image crops from catalogs, these sources contain few examples for many of the classes. Hence, these sources might require exploring classification methods which work with few examples. In this study, as the focus is on the evaluation of the representation for classification purposes, only the previously collected dataset from [Roman-Rangel et al. 2011] is studied as a source of Maya glyph dataset.

4.1.2 *Sketches.* Eitz et. al. provide a crowdsourced human sketch dataset over 250 object classes [Eitz et al. 2012]. Each class is composed of 80 sketches. For our experiments and to compare with the Maya glyph case, we use two versions of this dataset: a 10-class subset, and the full dataset. For the first case, we randomly picked the 10 classes. These classes are: ashtray, axe, banana, binoculars, camera, diamond, ear, guitar, pipe (for smoking), and present. For this dataset, the mean average stroke thickness is 1.2 pixels and mean maximum stroke thickness is 2 pixels. In other words, these sketches are roughly twice as thin as the Maya glyphs, and due to the way they were produced (digital pen of fixed thickness), contain almost no thickness variations (see Fig. 8). The original samples are provided as square images (1111x1111) in which the shapes are in the center and padded by 120 pixels (on average) on each side. We rescale the images to 128x128 size in our experiments.

4.2 Classifier

Given the BoW representation of each image, we used a k-nearest neighbor (k-NN) classifier with L_1 distance metric. This is illustrated in the fifth step of Fig. 3, where samples from various classes are shown in different colors, and the label of the test sample (shown in question mark) needs to be inferred from the labels of its k nearest neighbors. In the case of a tie between two or more classes, any of the tied classes is randomly assigned. The motivation for choosing k-NN as classifier is two-fold: the risk of overfitting problem due to small amount of data, and the emphasis of the comparison being on the visual representations (HOOSC vs. SA) and not necessarily on the machine learning schemes.

4.3 Performance Measure

The classification results are evaluated using average accuracy across data splits. The classification accuracy is computed as the ratio of the number of correctly classified samples (both positive and negative samples) over the total number of test samples.

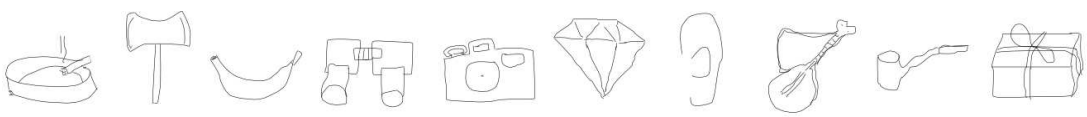

Fig. 8. Representative images of the randomly selected 10 classes from sketch dataset [Eitz et al. 2012].

4.4 Evaluation Protocol

For the glyph dataset, we randomly selected 25 image samples from each category for training, and the rest (413 samples) are used for testing. We repeated this process three times, and the average accuracy of the three data splits is reported. The SA representation has been trained on patches extracted from a total of 630 images, coming from the training samples of the 10 classes (250 images) as well as the unexploited images of the other 14 classes of the dataset (380 images), since training such a network requires as many samples as possible.

For the 10-class sketch dataset, we randomly selected 50 image samples from each class for training, and used the remaining 30 images for testing per class. The SA representation for sketch experiments are trained using 50 patches from each of the 50 image training samples of each class of the full set, thus resulting in 250x50x50 = 625000 patch samples.

For the 250-class experiments, we followed the experimental procedure described in [Eitz et al. 2012] (3 times run of 3-fold experiments) with 48 training samples per class, and reported the average accuracy.

4.5 Parameter Setting

In this section, we describe the parameter choices selected to conduct the classification experiments.

4.5.1 *Spatial Context.* One key parameter to study is the amount of spatial support that is used to build a descriptor (HOOSC or SA) around each point sample (termed as "spatial context"). In our experiments, the spatial context is defined as a fraction $sc \in \{1/1, 1/2, 1/4, 1/8\}$ of the longest image edge in both datasets. Since all images are scaled to 128 pixels while their aspect ratio is kept the same during preprocessing, the circular patches have fixed diameters (128, 64, 32, and 16, respectively) in each spatial context level. Fig. 9 illustrates the regions covered by these spatial context levels for the two different examples. Fig. 10 shows 100 example patches for each spatial context level and for both datasets.

4.5.2 *Sampling Strategy.* To build the BoW representation of an image, we need to sample the image positions where the local descriptors (the HOOSC or the SA representation) are to be computed and quantized. Two sampling strategies are evaluated. In the first one, following the previous work on shape analysis [Belongie et al. 2000], the points (pivots) are sampled randomly along the contour. We

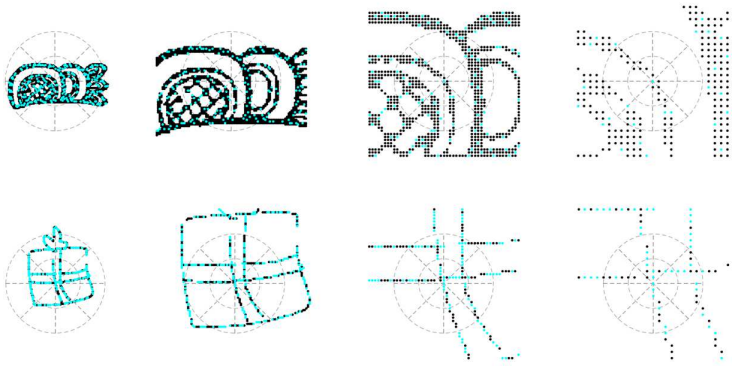

Fig. 9. Region used to compute the patch descriptor (HOOSC or SA) around a sample point (cyan) for the four different spatial context levels (from left to right, sc of $1/1, 1/2, 1/4,$ and $1/8$) for a glyph and a sketch sample.

Table II. Receptive field (patch) sizes while learning weights in SA.

Spatial context (sc)	sc = 1/1	sc = 1/2	sc = 1/4	sc = 1/8
Patch size in image	128	64	32	16
Actual patch size as input to SA (after rescaling)	32	32	32	16

work with 300 pivots in our experiments. Furthermore, for fairness in the experiments comparing the HOOSC and the SA representations, the same exact pivots have been used. As the second strategy, following [Coates et al. 2011] and [Eitz et al. 2012], dense sampling is applied. For dense case, the stride is set to 4 pixels, and 10-pixel offset is kept from the edges. In total, 28x28=784 regularly sampled pivots are used.

4.5.3 *HOOSC Parameters.* We follow the setting utilized in [Roman-Rangel et al. 2013], which was shown to perform best in the large majority of cases. The descriptor is built using 2 rings, 8 spatial bins, and 8 orientation bins.

4.5.4 *Sparse Autoencoder Parameters.* Training the SA representation using input patches of large sizes quickly leads to a large number of parameters to be learned. Roughly speaking, using circular patches of diameter N pixels, and N_D hidden units results in $O(N^2 N_D)$. For $N = 32$ and $N_D = 64$, the number of parameters is 65536 as noted in Table V. Thus, in practice, while training SA, we keep the patch size to be the same for different spatial context levels (except $sc = 1/8$) due to the increasing model complexity and the data shortage limitation. When dealing with patches of spatial context $sc = 1/1, 1/2$ levels, we rescaled their content to the actual patch size of sc = 1/4. For both datasets (Glyph and Sketch), the input to SA for $sc = 1/1, 1/2, 1/4$, is provided as patches of 32 pixels of diameter; and for $sc = 1/8$, the patch size is 16 pixels.

To train a SA representation, we need to set three parameters introduced in Section 3.3.1, namely the sparsity parameter ρ, the sparsity cost term coefficient β, and the weight regularization parameter α.

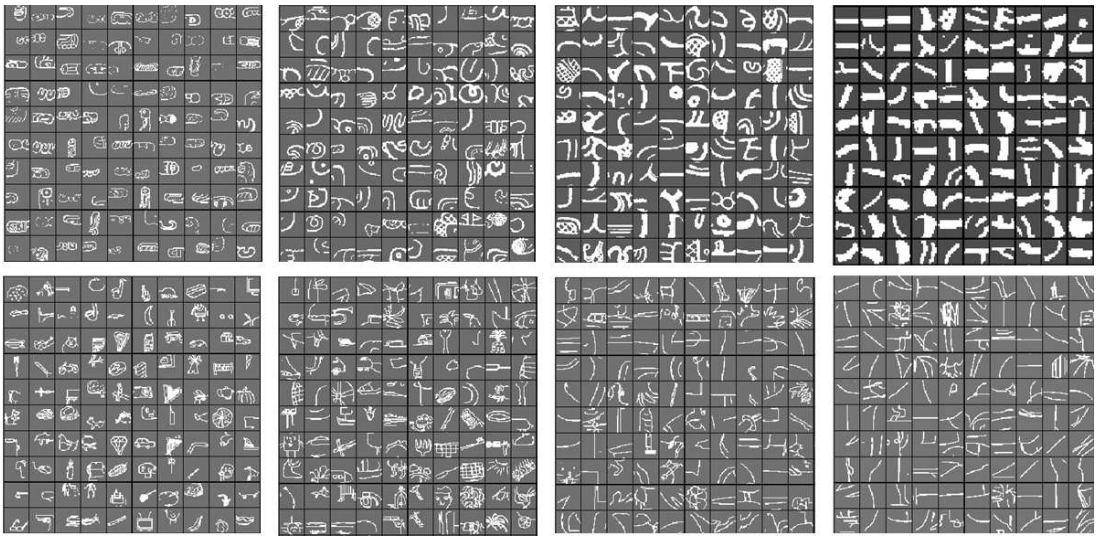

Fig. 10. Examples of circular glyph patches (top row) and sketch patches (bottom row) given as training data to SA when using a spatial context sc of $1/1, 1/2, 1/4, or 1/8$ (from left to right).

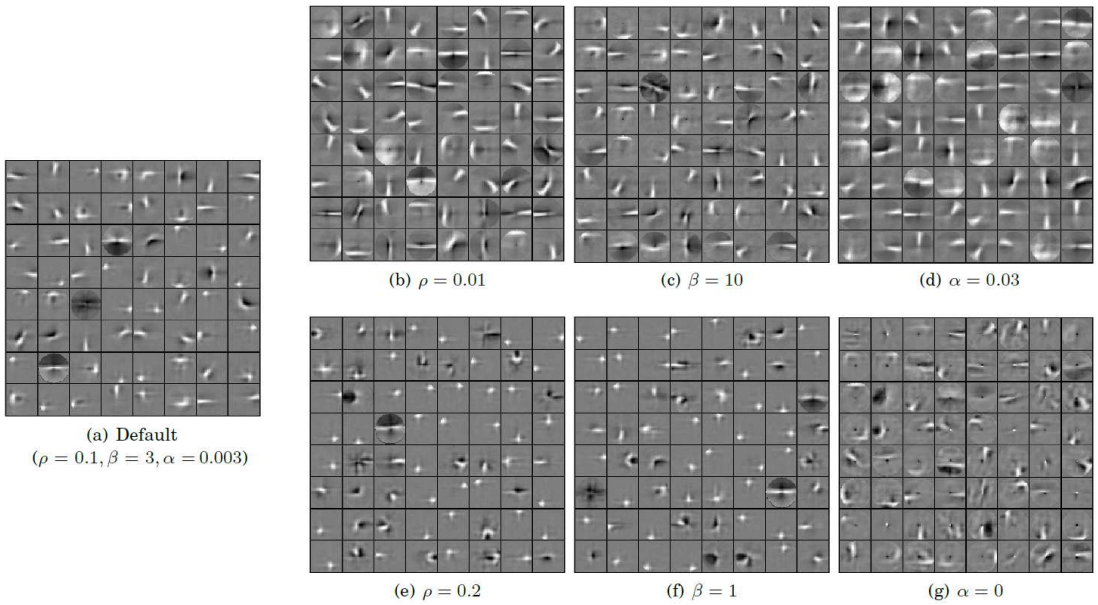

Fig. 11. Impact of the SA parameters on the learned weights (filters) when using 64 hidden units and the glyph dataset. On the left, the weights learned with the default parameters are shown. On the right, the impact of each single parameter keeping the others fixed is illustrated in each column (the rest of the parameters have default values).

Table III. Selected parameters for learning weights in SA.

Parameters	Sparsity parameter (ρ)	Sparsity cost term coefficient (β)	Weight regularization parameter (α)
Value	0.1	3	0.003

To analyze the impact of these parameters on training, we set the spatial context sc to 1/4 and consider 64 hidden units for the glyph data. Fig. 11 shows the weights learned with different parameter values.

On the left side of Fig. 11, the chosen weights are illustrated. Then, the weights learned with two other values of the parameters are shown in the following three columns. In each column, only one parameter is changed (stated below each subfigure) and the other parameters are set to their default values (see Table III). Looking at the filters, we observe a similar effect when a larger sparsity parameter ρ or a smaller sparsity cost term coefficient β are utilized. Since this corresponds to putting less weight on the sparsity term in the objective function, we obtain more localized filters. In other words, the reconstruction of the input data with these filters would be as if putting pixelwise responses from each hidden unit together. On the other hand, as ρ is smaller and β is larger, each filter is implicitly required to account for more spatial scope and therefore becomes less localized, in other words, more "blurry". As the regularization parameter α gets larger, we observe less variation between the weight values (less sparse weight per filter).

To derive a good set of parameters, we analyzed the correlations of the weights. We computed the correlation matrix of the filters (the inputs that would give the highest activation from the learned weights of each hidden unit) for each parameter configuration. Our aim is to select the configuration that results in a correlation matrix whose eigenvalues are small [Bengio and Bergstra 2009], i.e., decorrelated weights. The L1 distance of the eigenvalues of the correlation matrix of the weights to the identity matrix is shown as a quantitative measure in Table IV. In this table, we observe that this

Table IV. Correlation values of the weights learned with different parameters (shown in Fig. 11).

Parameter values	L1 distance of eigenvalues of the correlation matrix to identity
(a) Default ($\rho = 0.1, \beta = 3, \alpha = 0.003$)	45.4
(b) ($\rho = 0.01, \beta = 3, \alpha = 0.003$)	59.8
(c) ($\rho = 0.1, \beta = 10, \alpha = 0.003$)	52.2
(d) ($\rho = 0.1, \beta = 3, \alpha = 0.03$)	87.1
(e) ($\rho = 0.2, \beta = 3, \alpha = 0.003$)	39.9
(f) ($\rho = 0.1, \beta = 1, \alpha = 0.003$)	41.2
(g) ($\rho = 0.1, \beta = 3, \alpha = 0$)	39.2

measure gives smaller values for more localized filters (such as cases (e) and (f) in Fig. 11) compared to other configurations. In view of this, we set the parameter configuration such that this measure gives a small value and the filters are not too localized. Table III shows the chosen parameters. These parameters were used for all experiments with SA on both datasets.

5. RESULTS AND DISCUSSION

This section describes and discusses the performance according to variations of model parameters and evaluation frameworks. We first report results on the 10-class experiments for the Maya and sketch dataset. We then consider the extension to the 250 classes of the sketch data.

5.1 10-class Experiments

5.1.1 *Overall Results.* Fig. 13 shows the 10-class classification results for the 4 different spatial context levels of the representations for both glyph (left) and sketch (right) datasets. Fig. 13 also illustrates the comparative analysis of two different representations, namely HOOSC and SA with different number of hidden units (SA-64 and SA-256 with average pooling). Unless stated otherwise, for building the dictionary D via k-means clustering, the number of elements in the dictionary (vocabulary size)

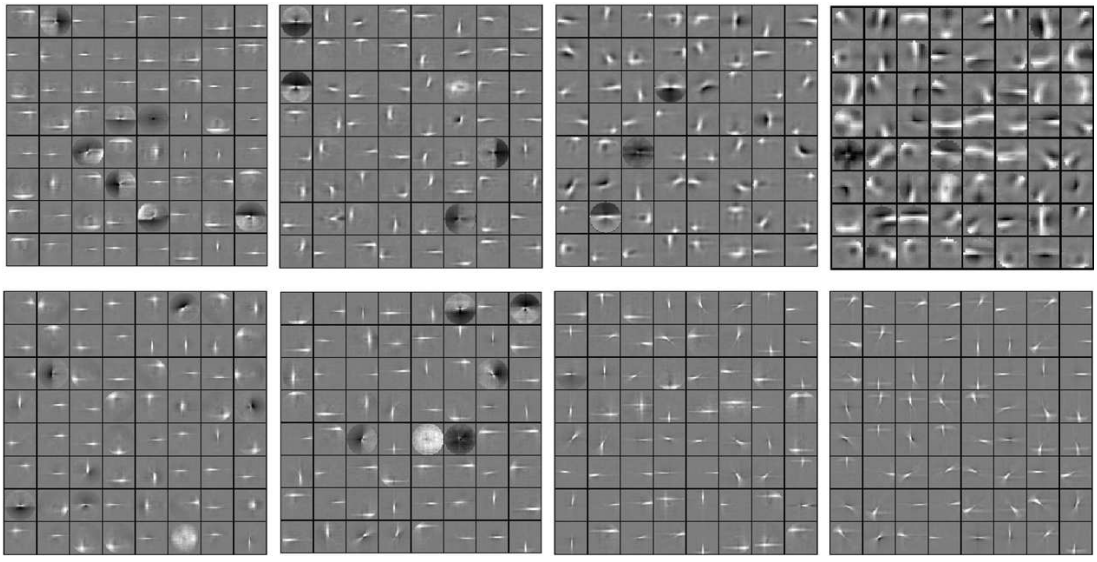

Fig. 12. Tied weights learned by SA for the patches with spatial context sc of 1/1, 1/2, 1/4, and 1/8 (from left to right). The top row is for glyph data and the last row is for sketch data.

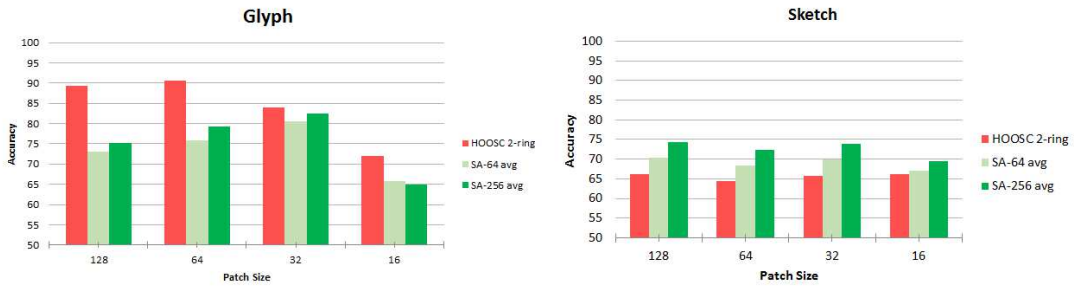

Fig. 13. 10-class classification results of the SA (64 or 256 hidden units), and the HOOSC descriptors with four spatial context levels for glyph (left) and sketch datasets (right). The number of words in quantization is 500, and $k = 5$ for k-NN classification.

N_D was set to 500, and for k-NN classification, 5 nearest neighbors ($k = 5$) were used. The experiments are repeated three times with random splits of the samples and the average accuracy is reported.

HOOSC vs SA Representations. The HOOSC descriptor and the SA representations have different trends in the classification accuracies for both datasets. In Fig. 13 (left), we observe that the HOOSC descriptor outperforms the SA representations in the glyph case, but the SA representation performs better than the HOOSC descriptor for the sketch data. This might be due to the HOOSC descriptor being a knowledge-driven representation. Since the HOOSC representation was originally proposed and designed for the glyph data, the parameters of the HOOSC descriptor are chosen optimally for glyph samples, even though those parameters might not be the optimal choice for the sketch data.

Glyph vs Sketch Data. Fig. 12 shows the filters learned with the patches collected on four spatial context levels for glyph (top row) and sketch (bottom) datasets. We observe curvy patterns on some filters, especially at the last two smaller spatial context levels in the glyph case. Other than that, all filters are quite similar (corresponding to lines in various orientations), which suggests that these two datasets are not so different from each other with respect to local shape and that the local shapes do not exhibit much data-specific correlation or repetitions.

The main difference between these two datasets is the stroke thickness as observed from the filters learned from patches with smallest spatial context ($sc = 1/8$). Glyph filters learned from the smallest patches are thicker than sketch filters learned from the same spatial context level. Indeed, with glyph data, at smaller spatial context levels ($sc = 1/4$ and $sc = 1/8$), more circular and curvy activation functions are learned for SA. These curvy patterns can be observed for spatial context level $sc = 1/4$ in the sample patches in Fig. 10, although for the glyph with $sc = 1/8$, there are still curvy patterns, they are visually more similar to sketch patches with $sc = 1/8$ except for the stroke size. This kind of curvy details is not captured at the larger levels ($sc = 1/1$ and $sc = 1/2$). This is probably due to loss of details (like small circles) during rescaling of the patches. We also hypothesize that as the region where patches are collected gets larger, the continuity and angle of the contour become more prominent features rather than small details. Besides, the repeatability of small, curvy patterns at

Table V. Number of parameters while learning weights in SA.

Dataset	Number of training samples	Patch Size	Number of hidden units	Number of parameters (Tied)
Glyph	630*300=189000	32	64	32x32x64=65536
Glyph	630*300=189000	32	256	32x32x256=262144
Sketch	250*50*50=625000	32	64	32x32x64=65536
Sketch	250*50*50=625000	32	256	32x32x256=262144

the same location in the training samples when using larger spatial context might be lower than in the smaller context cases. On the other hand, the weights learned on the sketch data do not contain any such small circular strokes, but mostly horizontal/vertical line strokes or intersection of these. We hypothesize this is due to the large variation in the sketching styles.

Even though glyph data is complex, it follows the Maya language rules. For instance, small circular patterns are useful parts to categorize some glyph classes. Based on the location or configuration of these parts with respect to others, the category of the glyph may change. For instance, the main visual difference of /*ya*/ (T126) and /*ji*/ (T136) are the small circles between half-moon shapes (see Fig. 7). Another example can be subtle differences between /*li*/ (T82), /*ki*/ (T102), and /*ko*/ (T110). The distinct points of /*li*/ from /*ki*/ are the small circles which are in the middle of the contour but not connected to any vertical "band" and the inner lines connected to the outer contour rather than the "band" as in /*ki*/. Similarly, /*li*/ has an eliptic element which is connected to the outer contour. This element is also visible in /*ko*/. The only difference between /*li*/ (T24) and /*ka*/ (T25) is the number of inner slanted lines, and for /*ni*/ (T116) and /*wi*/ (T117), the only distinctive hint is the direction of the small part. There are also compositionally similar glyph categories, even though visually they may seem not so close, as in the case of /*ti*/ (T59) and /*tu*/ (T92), which have three main parts. This means that there are shared, small patterns encoded in the glyph data. However, in the sketch case, shapes are by construction over-simplified or contain very few detailed samples. Furthermore, the shared patterns between sketch classes are not as obvious as in the glyph case. Therefore, only the lines with different orientations and their intersections are learned as common patterns by the SA architecture.

One point to recall is that the number of training samples are different in the two datasets. In the glyph dataset, there are 25 training samples, while in the sketch dataset we use 50 samples (see Table V). We have also a larger quantity of data to train SA for the sketch case and less variability in stroke thickness. This suggests that the higher performance of the SA representation compared to the HOOSC descriptor on sketch data might be related to the larger amount of training data, which is also noted in [Coates et al. 2011; Eitz et al. 2012]. Even though both algorithms perform worse on the sketch data as compared to the glyph case, performance of the HOOSC descriptor degrades much more as compared to the SA representation. Therefore, we hypothesize that the SA representation gains some performance by learning from a larger training set.

Impact of Spatial Context. The spatial context, i.e., the patch size where the representation is extracted, affects the HOOSC and SA representations differently in the two datasets (see Fig. 13). In the glyph case, as the patches get smaller, the performance of the HOOSC descriptor degrades, and it marginally increases in the sketch case. For the SA representations, we obtain the highest performance (78 % with SA-256 avg representation) with 32x32 patch size ($sc = 1/4$) in the glyph case. However, for the sketch data, the trend is not so clear, as the performance values are close and minimal changes are observed across patch sizes. The learned SA filters (Fig. 12) play an important role on these performance values. Since the filters learned on sketch patches are similar, the performance values turn out to be close to each other. On the other hand, the filters learned on 32x32 ($sc = 1/4$) and 16x16 glyph patches ($sc = 1/8$) are the most characteristic filters (such as curvy patterns); we obtain the highest performance for the representations on 32x32 glyph patches, although the SA representations over 16x16 glyph patches are not competitive (as what the representation capture over the region becomes unspecific as can be seen in Fig. 9).

5.1.2 *Analysis of the SA Representation.* We now analyze the performance of the SA representation according to the number of hidden units during the learning step, and to the pooling strategy at the feature extraction step.

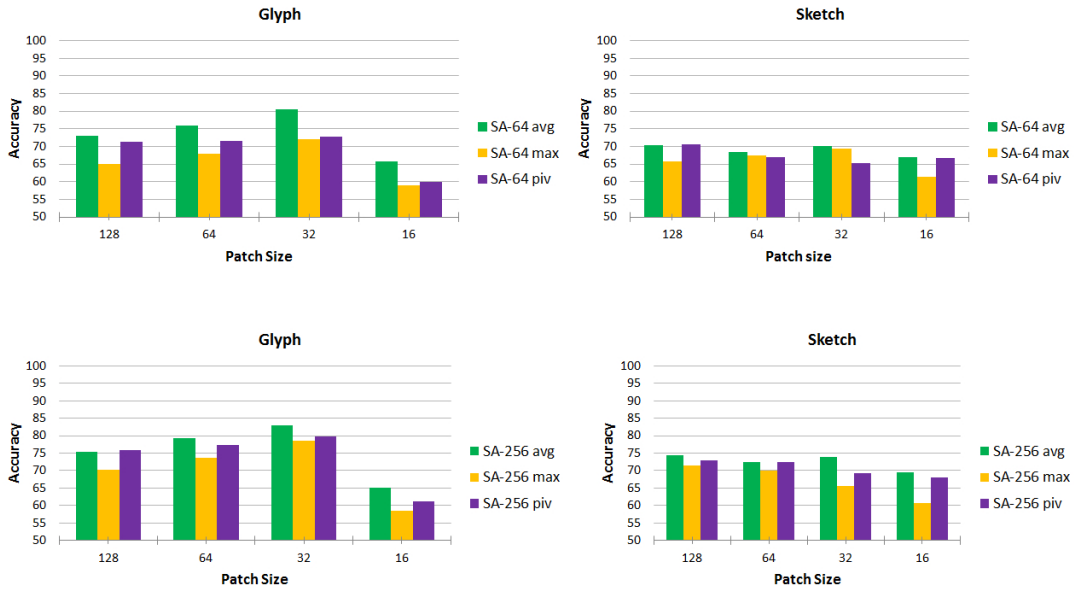

Fig. 14. 10-class classification results of the SA with 64 (top) or 256 hidden units (bottom) for three different pooling strategies and four spatial context levels, for glyph (left) and sketch datasets (right).

Impact of Number of Hidden Units. We can observe from Fig. 13 that a larger number of hidden units tend to increase the recognition accuracy. This is the case for both datasets and most spatial context levels sc, as shown in Fig. 13.

Impact of Pooling Strategy. The pooling operation is necessary to aggregate the convolutional filter responses and to obtain a reasonably-sized feature dimension. During pooling, the spatial information is also implicitly embedded, since the values are aggregated in 5x5 neighborhoods around the sampled pivots. We used three ways to get information from the circular region of interest: average pooling, max pooling, or taking only the response over the pivots. Results are shown in Fig. 14. We observe that in the great majority of the cases, average pooling outperforms or is on par with the other strategies. Furthermore, taking only the pivot response gives higher accuracy than max pooling except for the sketch SA-256 case. This is in contrast to recent work in the literature [Boureau et al. 2010], which empirically found that max pooling is robust to subtle changes in the pooled spatial neighborhood. We hypothesize that it is due to the nature of the data and our sampling strategy. Since the data consists of binary shape images sampled along contours, average pooled features might be more robust at representing and taking into account the contour variations around points compared to the max-pooled features.

5.1.3 *Analysis of Bag-of-Words Model.* Here, we analyze the bag-of-words model according to the dictionary size and the value of k in k-nearest neighbor classification.

Dictionary Size. In our experiments, the dictionary size does not follow a single trend. In Fig. 15, the performance values for glyph data are presented for $N_D = 500$ (light colors) and $N_D = 1500$ (dark colors) dictionary sizes. We observe 2 to 4 percent increase when using a larger vocabulary size as the patch sizes are large. HOOSC classification can reach a highest value of 93.46 % accuracy. However,

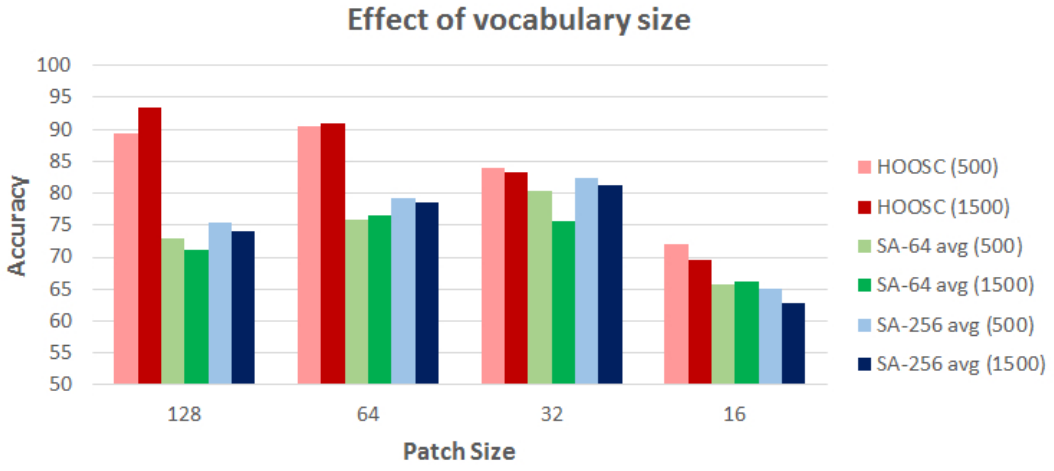

Fig. 15. 10-class classification results of the HOOSC and SA representations for 500-word and 1500-word dictionary size, with four spatial context levels, for the glyph dataset.

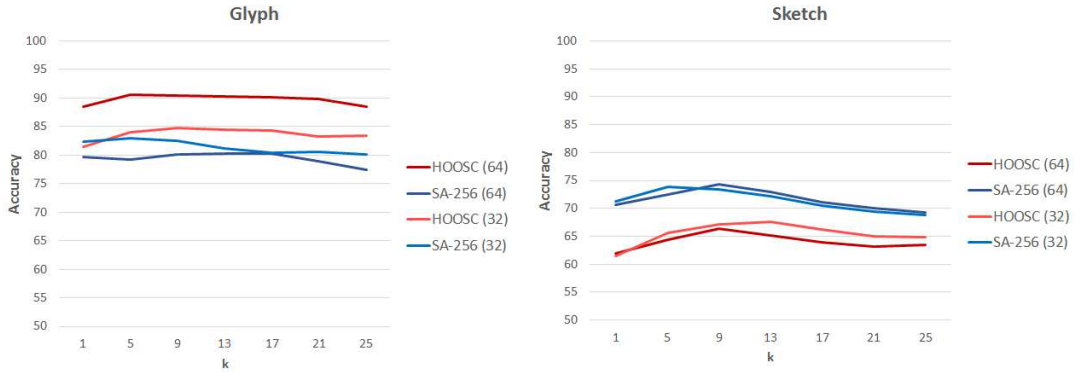

Fig. 16. 10-class classification results for the HOOSC and SA representations, for 64x64 ($sc = 1/2$) and 32x32 ($sc = 1/4$) patch sizes, with $k = 1, 5, 9, 13, 17, 21, 25$ during k-NN classification for glyph (left) and sketch (right) datasets.

for smaller patch sizes, a larger vocabulary size generally degrades or does not affect the results. Note that we also observed a similar trend in the sketch dataset.

Value of k in k-NN Classification. Fig. 16 shows the effect of the value of k in k-nearest neighbor classification for the HOOSC and SA-256 representations. To be brief, only the results of the representations defined on 32x32 and 64x64 patches are shown. Results are relatively stable overall with respect to k. We can however notice that for the representations defined on 32x32 glyph patches (light lines on left plot), the classification performance decreases as k increases. Similarly, the performance degrades for 64x64 sketch patches (darker lines on right plots) as k increases beyond a certain point.

5.2 Generalizing the Results: 250-class Experiments

We also study the generalization capability of our classification model over all 250 sketch classes. As stated earlier in the paper, we designed a second set of experiments with a larger shape dataset in terms of numbers of classes and examples. Unfortunately, such dataset is not publicly available for Maya hieroglyphs, and so we use the sketch data as data source. For the future, we anticipate that large datasets of glyphs could be generated by experts. We followed the experimental procedure described in [Eitz et al. 2012] (3-fold experiments) with 48 training samples per class, and reported the average accuracies. Besides, we compare our pivot sampling strategy with dense sampling.

How Do the Representations Generalize? The performance results of all the representations for 250 sketch classes are provided in Fig. 17, with vocabulary size equeal to 500, $k = 25$ during k-NN classification, and the same values in Table III used for SA parameters. As in the 10-class case, we observe that the SA representations clearly outperform the HOOSC representation (red) in the 250-class case (with 20.73 % for HOOSC and 29.36 % for SA). But there is also a significant drop in performance for all methods given the larger complexity of the task. The highest accuracy is of 29.36 %. As in the 10-class case, using more hidden units (256) leads to better results with most of the pooling strategies and patch sizes. Also similar to the 10-class case, average pooling performs better for the SA-64 representation. However, contrary to the 10-class results, the SA-256 pivot pooled representation is the most competitive of the SA-256 representations. The impact of patch size is not prominent in the 250-class case. Even though the HOOSC results degrade quite marginally as the patch size gets smaller, the SA results are close to each other in terms of the patch size they are defined on.

Value of k in k-NN Classification. Fig. 18 shows the corresponding plot of the right plot in Fig. 16 as the task is generalized to 250 sketch classes. This time, we see a clear trend with respect to the k value in k-NN classification. For both representations defined on both 32x32 and 64x64 patches, the performance increases with k value and comes to a stable state, reflecting that with more classes assessing the label with more neighbors is a better strategy.

Pivot vs Dense Sampling. The recent literature suggests that as the sampling of points around which descriptors are computed gets denser, the larger amount of produced descriptors results in a better model (here represented by the bag-of-words representation) of the overall image content [Coates et al. 2011]. We test the dense regular grid sampling (784 pivots) utilized in [Eitz et al. 2012] for the HOOSC and SA representations as compared to random sampling on contours (300 pivots). The results are illustrated in Fig. 19. From this plot, we observe that the densely-sampled HOOSC

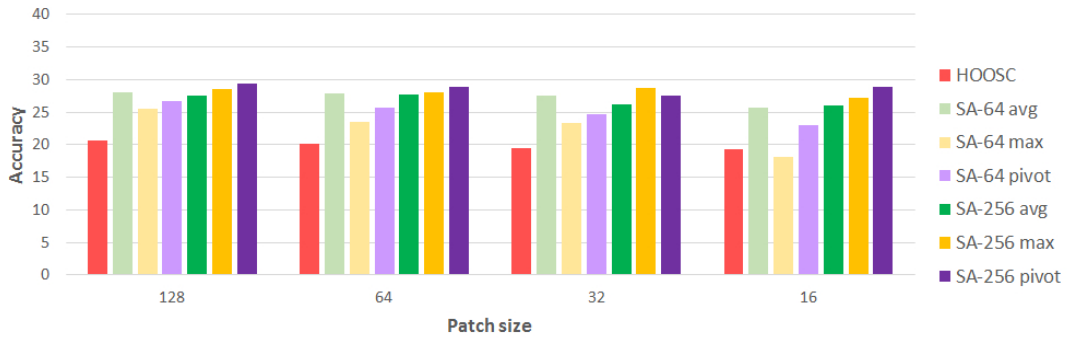

Fig. 17. Accuracy results of the HOOSC and SA representations with pivot-sampling and four spatial context levels for 250-class sketch classification.

descriptor defined over large regions is much better (about 15% absolute increase in accuracy) than the pivot-sampled HOOSC descriptor. However, the densely-sampled HOOSC results get worse than pivot-sampled HOOSC results as the patch size gets smaller. This is due to the nature of the shape data. Since shapes have large empty regions, as the patch size is smaller, the number of empty descriptors increases when using a regular grid sampling. This effect is also seen in the SA representation results with the smallest patch size (the dense sampling results are 7% lower than the pivot sampling results). Interestingly though, the SA representations are affected by this cause only at the smallest spatial context level (16x16). Furthermore, we observe that the SA results, regardless of the sampling, are quite similar (around 27-28% accuracy) for larger patch sizes.

Comparison with the state-of-the-art. We compare the studied representations with the local orientation-based descriptor in [Eitz et al. 2012]. In their work, they use a square patch of around 90 pixels in size for 256x256 scaled images, which compares in terms of setting to our circular patches of radius 64. The accuracy obtained by the densely-sampled HOOSC over 64x64 patch regions is 35.13% which shows a marginal increase (around 0.63%) to [Eitz et al. 2012] with their descriptor that uses hard quantization with 500 number of words.

6. CONCLUSION

This work assessed a data-driven, non-linear filter bank representation against a hand-crafted, orientation-based histogram representation for Maya hieroglyphic shape data. We investigate the impact of the different parameters of the representations and the most important aspects to consider while working with such image representations. The main conclusions are the following.

—**Sparse Autoencoder modeling.** Appropriate parameters to learn the filters could be obtained by following principles in the literature, and few empirical choices. The learned filters reveal the data content, and are tuned to the inherent motifs in shape data. With respect to classification, both the 10- and 250-class experiments show that using a larger number of hidden units improved the performance in most of the settings on both datasets. In addition, average pooling tends to perform best.

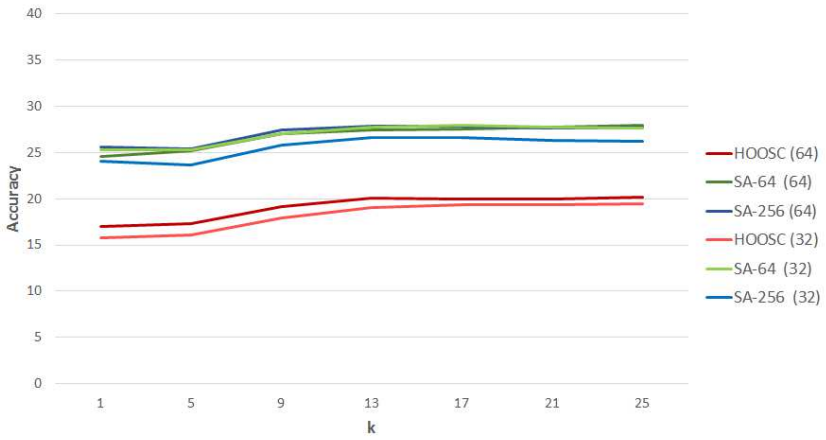

Fig. 18. 250-class sketch classification results of the HOOSC and SA representations, for 64x64 ($sc = 1/2$) and 32x32 ($sc = 1/4$) patch sizes with $k = 1, 5, 9, 13, 17, 21, 25$ during k-NN classification.

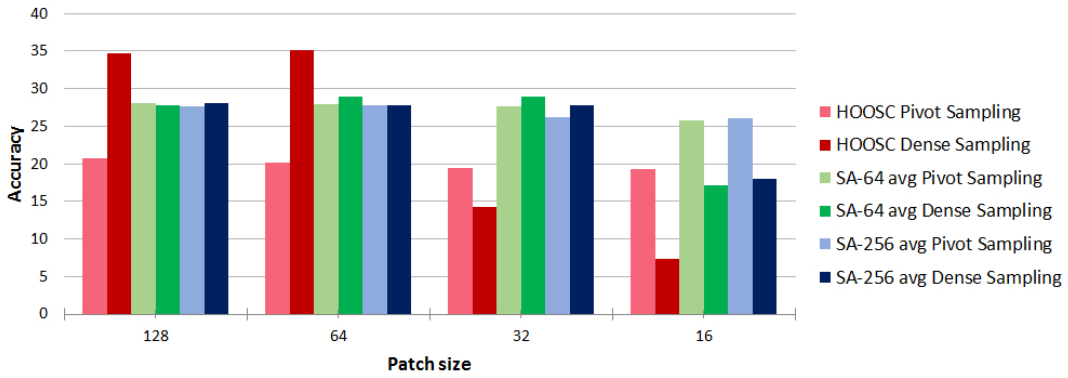

Fig. 19. 250-class HOOSC and SA results with densely-sampled and pivot-sampled patches (left) with four spatial context levels for sketch dataset.

—**Knowledge-driven vs. data-driven features.** Our work showed that the HOOSC descriptor performed better than the SA representation on the smaller dataset (glyph), for which the HOOSC descriptor was originally designed for. In contrast, when considering more data (10-class experiments on sketches) and more classes (250-classes), with the pivot sampling-based approach, the SA representation was able to surpass the HOOSC descriptor (29.36 % vs. 20.73 % on the 250-class experiments). This shows the importance (and the necessity) of the amount of available data for data-driven approaches. Keeping that in mind, we plan to apply data augmentation for learning more elaborate representations with multi-layer convolutional networks in future studies.

—**Dense sampling.** With dense sampling, the HOOSC descriptor improved substantially when considering large enough spatial regions, reaching 35% accuracy on the 250-class case. On the other hand, the SA representation did not benefit from this factor, staying behind HOOSC's accuracy.

—**Comparison with the state-of-the-art.** The HOOSC performance is in par with that of [Eitz et al. 2012] for dense sampling using big patches of sketch data. A probable reason why SA was not able to leverage the dense sampling on large patches is that increasing the patch size resulted in two issues. First, the low amount of stroke points that are present in many large patches obtained by dense sampling would make the learning of efficient representation difficult, as most contours would not tend to repeat in the training data. The second issue is that the number of model parameters grows quadratically with the patch size, making it infeasible to learn them from raw data. To avoid this problem, we downscaled the patches in the experiments, but at the cost of losing some useful information present in the original data. The main conclusion is that a single-layer SA is not sufficient to efficiently learn shape characteristics, and that deeper networks with more layers might allow a hierarchical and gradual learning of shape elements without the quadratic increase of parameters. We plan to explore such representations in the future.

—**Use for cultural heritage.** We hope that our study, using both Maya hieroglyphics and generic hand-drawn shapes, can inform about the possible performance trends of the studied methods for other cultural heritage visual shape sources. This could include future Maya datasets (e.g. from codices that we are collecting as part of ongoing work) and other sources (e.g. Egyptian hieroglyphs).

7. ACKNOWLEDGEMENTS

This work was funded by the Swiss National Science Foundation as part of the MAAYA project. We thank our partner Carlos Pallán Gayol (University of Bonn) for his help with the glyph data preparation, and Rui Hu (Idiap) for her help editing the paper and the discussions.

REFERENCES

Sebastiano Battiato, Giovanni Maria Farinella, Giovanni Gallo, and Daniele Ravì. 2010. Exploiting textons distributions on spatial hierarchy for scene classification. *Journal on Image and Video Processing* 2010 (2010), 7.

Sebastiano Battiato, Giovanni Maria Farinella, Oliver Giudice, and Giovanni Puglisi. 2015. Aligning shapes for symbol classification and retrieval. *Multimedia Tools and Applications* (2015), 1–19.

Serge Belongie, Jitendra Malik, and Jan Puzicha. 2000. Shape context: A new descriptor for shape matching and object recognition. In *Conference on Advances in Neural Information Processing Systems*, Vol. 2. 3.

Yoshua Bengio and James S. Bergstra. 2009. Slow, Decorrelated Features for Pretraining Complex Cell-like Networks. In *Conference on Advances in Neural Information Processing Systems 22*. 99–107.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.

Y-L Boureau, Francis Bach, Yann LeCun, and Jean Ponce. 2010. Learning mid-level features for recognition. In *Conference on Computer Vision and Pattern Recognition*. IEEE, 2559–2566.

K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. 2011. The devil is in the details: an evaluation of recent feature encoding methods. In *British Machine Vision Conference*.

L Chen, F Rottensteiner, and C Heipke. 2015. Feature Descriptor by Convolution and Pooling Autoencoders. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 1 (2015), 31–38.

Adam Coates, Andrew Y Ng, and Honglak Lee. 2011. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*. 215–223.

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*, Vol. 1. IEEE, 886–893.

Mathias Eitz, James Hays, and Marc Alexa. 2012. How Do Humans Sketch Objects? *ACM Transactions on Graphics (Proceedings SIGGRAPH)* 31, 4 (2012), 44:1–44:10.

Sergio Escalera, Alicia Fornés, Oriol Pujol, Josep Lladós, and Petia Radeva. 2011. Circular blurred shape model for multiclass symbol recognition. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 41, 2 (2011), 497–506.

Orhan Firat. 2015. libORF: A Machine Learning Toolkit for Deep Learning, Probabilistic Graphical Models and Structured Prediction. (2015). Available from: <http://www.ceng.metu.edu.tr/ e1697481/libORF.html>. Accessed: 2015-06-03.

Morris Franken and Jan C van Gemert. 2013. Automatic egyptian hieroglyph recognition by retrieving images as texts. In *ACM International Conference on Multimedia*. ACM, 765–768.

Geoffrey E Hinton and Richard S Zemel. 1994. Autoencoders, minimum description length, and Helmholtz free energy. *Conference on Advances in Neural Information Processing Systems* (1994), 3–3.

Rui Hu, Gulcan Can, Carlos Pallan Gayol, Guido Krempel, Jakub Spotak, Gabrielle Vail, Stephane Marchand-Maillet, Jean-Marc Odobez, and Daniel Gatica-Perez. 2015. Multimedia Analysis and Access of Ancient Maya Epigraphy. *Signal Processing Magazine* 32, 4 (July 2015), 75–84.

Yangqing Jia, Chang Huang, and Trevor Darrell. 2012. Beyond spatial pyramids: Receptive field learning for pooled image features. In *Conference on Computer Vision and Pattern Recognition*. IEEE, 3370–3377.

P. D. Kovesi. 2015. MATLAB and Octave Functions for Computer Vision and Image Processing. (2015). Available from: <http://www.peterkovesi.com/matlabfns/>. Accessed: 2015-01-16.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Conference on Computer Vision and Pattern Recognition*, Vol. 2. IEEE, 2169–2178.

Biao Leng, Shuang Guo, Xiangyang Zhang, and Zhang Xiong. 2015. 3D object retrieval with stacked local convolutional autoencoder. *Signal Processing* 112 (2015), 119–128.

David G Lowe. 1999. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, Vol. 2. Ieee, 1150–1157.

Martha J Macri and Matthew George Looper. 2003. *The New Catalog of Maya Hieroglyphs: The Classic Period Inscriptions*. Vol. 1. University of Oklahoma Press.

Andrew Ng. 2013. Sparse Autoencoders Lecture Notes. https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf. (2013). Accessed: 2015-07-30.

Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc V Le, and Andrew Y Ng. 2011. On optimization methods for deep learning. In *International Conference on Machine Learning*. 265–272.

Florent Perronnin, Jorge Sánchez, and Thomas Mensink. 2010. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision*. Springer, 143–156.

Marc Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. 2007. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.

Edgar Roman-Rangel. 2012. *Statistical Shape Descriptors for Ancient Maya Hieroglyphs Analysis*. Ph.D. Dissertation. École Polytechnique Fédérale de Lausanne.

Edgar Roman-Rangel, Jean-Marc Odobez, and Daniel Gatica-Perez. 2012. Assessing Sparse Coding Methods for Contextual Shape Indexing of Maya Hieroglyphs. *Journal of Multimedia* 7, 2 (April 2012), 179–192.

Edgar Roman-Rangel, Jean-Marc Odobez, and Daniel Gatica-Perez. 2013. Evaluating Shape Descriptors for Detection of Maya Hieroglyphs. In *Mexican Conference on Pattern Recognition*.

Edgar Roman-Rangel, Carlos Pallan, Jean-Marc Odobez, and Daniel Gatica-Perez. 2011. Analyzing ancient maya glyph collections with contextual shape descriptors. *International Journal of Computer Vision* 94, 1 (2011), 101–117.

Hoo-Chang Shin, Matthew R Orton, David J Collins, Simon J Doran, and Martin O Leach. 2013. Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1930–1943.

Josef Sivic and Andrew Zisserman. 2003. Video Google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*. IEEE, 1470–1477.

John Eric Sidney Thompson and George E Stuart. 1962. *A catalog of Maya hieroglyphs*. University of Oklahoma Press Norman.

Jan C van Gemert, Jan-Mark Geusebroek, Cor J Veenman, and Arnold WM Smeulders. 2008. Kernel codebooks for scene categorization. In *European Conference on Computer Vision*. Springer, 696–709.

Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. 2010. Locality-constrained linear coding for image classification. In *Conference on Computer Vision and Pattern Recognition*. IEEE, 3360–3367.

Meng Wang, Youbin Chen, and Xingjun Wang. 2014. Recognition of Handwritten Characters in Chinese Legal Amounts by Stacked Autoencoders. In *International Conference on Pattern Recognition*. IEEE, 3002–3007.

Guo-Sen Xie, Xu-Yao Zhang, and Cheng-Lin Liu. 2015. Efficient Feature Coding Based on Auto-encoder Network for Image Classification. In *Asian Conference on Computer Vision*. Springer, 628–642.

Matthew D Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

Xi Zhou, Kai Yu, Tong Zhang, and Thomas S Huang. 2010. Image classification using super-vector coding of local image descriptors. In *European Conference on Computer Vision*. Springer, 141–154.

Received August 2015; revised December 2015; accepted March 2016