Orthogonal Local Image Descriptors with Convolutional Autoencoders

Edgar Roman-Rangel[1](✉) and Stephane Marchand-Maillet[2]

[1] Instituto Tecnologico Autonomo de Mexico, ITAM, 01080 Mexico City, Mexico
edgar.roman@itam.mx

[2] University of Geneva, 1227 Geneva, Switzerland
stephane.marchand-maillet@unige.ch

Abstract. This work proposes the use of deep learning architectures, and in particular Convolutional Autencoders (CAE's), to incorporate an explicit component of orthogonality to the computation of local image descriptors. For this purpose we present a methodology based on the computation of dot products among the hidden outputs of the center-most layer of a convolutional autoencoder. This is, the dot product between the responses of the different kernels of the central layer (sections of a latent representation). We compare this dot product against an indicator of orthogonality, which in the presence of non-orthogonal hidden representations, back-propagates a gradient through the network, adjusting its parameters to produce new representations which will be closer to have orthogonality among them in future iterations. Our results show that the proposed methodology is suitable for the estimation of local image descriptors that are orthogonal to one another, which is often a desirable feature in many patter recognition tasks.

Keywords: Local image descriptors · Orthogonal bases · Convolutional autoencoders

1 Introduction

The use of orthogonal bases in the estimation of local image descriptors was a widely used paradigm in many computer vision scenarios before the deep learning era [1], specially because this approach allows the definition of over-complete dictionaries for robust image description [1,2]. However, recent developments of deep architectures seem to disregard the potential of incorporating orthogonal bases in their models, perhaps because of the indisputable success that these deep models, and in particular Convolutional Neural Networks (CNN's), have already shown in solving several computer vision problems [3,4], even without the explicit consideration of orthogonality.

Since CNN's are often designed for end-to-end processing, the estimation of local image descriptors also seems to be unnecessary lately, i.e., the focus is in solving directly tasks like classification or localization [5,6]. Therefore, it has become common to overlook at intermediate representations during the image description process, as long as they get the task solved. Nonetheless, looking at intermediate representations of CNN's [9] might still prove beneficial for the estimation of local image descriptors, which might be desirable in at least two scenarios. First, when fine local details are highly relevant for a given task, as global descriptors risk overlooking at them. Second, when dealing with small datasets that could limit the capacity for properly training a large set of parameters, as is often the case with many deep neural networks architectures [7].

In this work, we propose the re-consideration of orthogonality as a constraint for the estimation of local image descriptors, which are computed using Convolutional Autoencoders (CAE's) [8]. We develop a methodology that readily inserts itself as another layer in a deep CAE architecture, and that allows to impose orthogonality constraints to intermediate representation of the network. We evaluate the impact of our model in the task of image reconstruction, and our results show that this approach is suitable for obtaining orthogonal local descriptors while still being able to reconstruct images at high precision rates.

The rest of this paper is organized as follows. Sect. 2, gives details about our proposed approach for generation of orthogonal local descriptors. Sect. 3 describes the protocol followed to evaluate our method. Sect. 4 discusses our results. Finally, Sect. 5, presents our conclusions.

2 Orthogonal Local Descriptors

This section explains the proposed deep learning architecture designed to compute orthogonal local descriptors, which corresponds to a type of convolutional autoencoder (CAE). This deep convolutional autoencoder simultaneously optimizes two objective functions: the reconstruction error of the autoencoder itself and an orthogonality constraint.

2.1 Architecture

An overview of our CAE-like architecture is shown in Fig. 1. As it happens with standard autoencoders, ours is composed of two stages: encoding and decoding, where the output of the central layer is considered an appropriate representation of the input image, as long as it allows the model to reconstruct its own input.

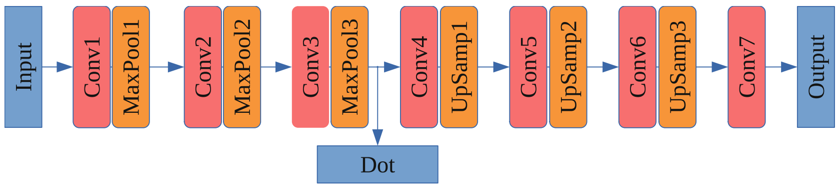

Fig. 1. Model architecture. Orthogonal convolutional autoencoder.

The encoding part of our model starts with a set of convolutional and max-pooling layers packed into blocks, whose tasks is that of applying a cascade of low-level filters that identify or enhance the visual information relevant for local description. There are three such convolutional-pooling blocks, as suggested by previous works [4], in which it has been observed that three blocks suffice for learning to detect edges, corners and contours, and object parts, respectively [9].

Likewise, we also implement three blocks with convolution filters and up-sampling operations for the decoder stage, i.e., the reconstruction section of the model. This decoding section has the only task of recovering the initial visual information from whatever representation had resulted after the initial encoding process. There is an additional final convolutional layer whose purpose is to smooth out the output of the last up-sampling operation, this is, it corrects for the abrupt zero-order hold extrapolation produced by the up-sampling layer.

In this model, all convolutional operations in the encoding and decoding stages are performed by filters of size 5×5, and are followed by $ReLU$ activation functions. Regarding the max-pooling layers, all of them operate over 2×2 pixels neighborhoods. Similarly, the up-sampling steps correspond to zero-order hold interpolation processes performed in localities of 2×2 pixels.

The central layer in our model, indicated by the name "Dot" in Fig. 1, is the one where the orthogonality between convolutional filters is optimized. See Sect. 2.2, for further details about this process.

2.2 Orthogonality

For the purpose of measuring orthogonality, we consider individually, the output of each convolutional filter in the central layer (*latent representation*). This is, we optimize for each convolutional unit to produce outputs that are orthogonal to one another. Concretely, we measure the orthogonality between pairs of outputs of the layer *maxpool3*, and from there, we backpropagate a loss measurement indicating a notion of *lack of orthogonality.*

More formally, the orthogonality between two convolutional outputs corresponds to the dot product computed between their respective vectorized forms v_i and v_j, i.e., the output of the convolutional filter is a matrix, however, vectorizing it has no impact for the optimization process. Namely, the orthogonality score $\hat{o}_{i,j}$ between vectors v_i and v_j is computed as,

$$\hat{o}_{i,j} = \langle v_i, v_j \rangle \, . \tag{1}$$

Computing this dot product for each pair of the C convolutional outputs from a given layer, results in a matrix $\hat{O}$ of $[C \times C]$ elements, which indicates the degree of correlation between pairs of outputs, and thus some sort of similarity between the convolutional filters that generated them themselves. Moreover, having the off-diagonal elements of $\hat{O}$ all equal to zero, indicates that the operand vectors that originated them are pair-wise orthogonal.

Using this notion of orthogonality, we compute the orthogonality loss $\mathcal{L}^o$ as,

$$\mathcal{L}^o = \frac{1}{M} \|O - I\|_1, \tag{2}$$

where, O is the orthogonal matrix normalized through,

$$o_{i,j} = \frac{\hat{o}_{i,j}}{\max_{i,j} \hat{O}}, \tag{3}$$

and I is the identity matrix of size $[C \times C]$. This is, Eq. (2) computes the element-wise mean absolute error between O and I.

In practice, max-pooling layers contain no parameters that can be optimize via gradient descent. Therefore, the orthogonality loss $\mathcal{L}^o$ is back-propagated directly to the previous convolutional layer (i.e., *conv3*), whose parameters are updated to maximize the orthogonality of future outputs.

2.3 Loss Function

The training of the proposed model consists in minimizing simultaneously a reconstruction loss $\mathcal{L}^r$ and the orthogonality loss $\mathcal{L}^o$ defined in Eq. (2). This is,

$$\mathcal{L} = \mathcal{L}^r + \lambda \mathcal{L}^o, \tag{4}$$

where, λ is a coefficient that weights the contribution of the orthogonality loss $\mathcal{L}^o$ to the total loss function, and,

$$\mathcal{L}^r = \mathcal{L}^r(X, X; \Omega), \tag{5}$$

indicates a notion of the error obtained when using X (an image patch) as input to our model, and trying to reconstruct it using the set of parameters Ω. Note that the specific form of $\mathcal{L}^r$ might vary depending on the nature of the data and the problem that is being addressed.

3 Experiments

This section provides information regarding the evaluation of the proposed orthogonal local descriptor. Concretely, it describes the dataset used for evaluation, provides several implementation details, and presents the two types of experiments we performed to validate our methodology.

3.1 Data

We used a dataset of binary images [10] containing arrangements of hieroglyphs from the ancient Maya culture. These arrangements of hieroglyphs are known as glyph-blocks, or simply blocks. Figure 2 shows a few examples of them.

This glyph-blocks dataset is formed by 5000 images, each containing from 1 to 6 individual signs visually located at arbitrary positions. In turn, each individual sign belongs to one of 255 semantic classes (a notion of word)[1]. It is indeed, the

[1] Approximately, 1000 different Mayan glyph-signs have been identified thus far by archaeologists, but our dataset only contains instances from 255 of them.

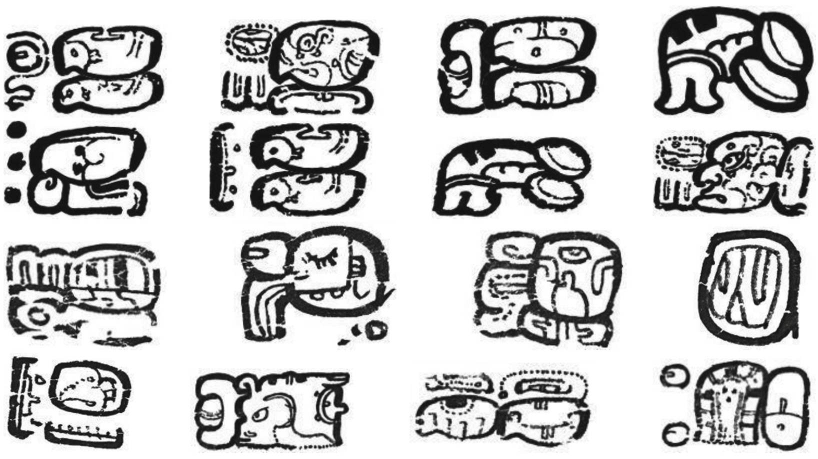

Fig. 2. 16 examples of glyph-blocks in our dataset.

relative small size of this dataset that has motivated the investigation on the use of convolutional autoencoders to generate local image descriptors.

During our experiments, we randomly chose 80 glyph-blocks for training and 20 for validation, leaving 4900 aside for testing. For the generation of the local image descriptors, we input square image patches uniformly sampled from the complete glyph-blocks. The size of these patches is 64×64 pixels, sampled at strides of 16 pixels. Since glyph-blocks images are of varying size, this segmentation resulted in 39,125 training and 10,251 validation patches.

3.2 Implementation Details

We implemented our proposed model using python 3.7 and the keras module of the tensorflow 2.0 library.

Conceptually, our model is a branched network as depicted in Fig. 1. Its main branch consists of 15 layers organized in three types of blocks: convolution plus max-pooling, convolution plus up-sampling, and only convolution blocks. Its purpose is to process input images extracting relevant information into a compact latent representation, and then reconstruct the original image starting from such latent representation.

With the exception of its last convolutional layer, which implements the *sigmoid* activation function for recovering pixel values within the interval $[0, 1]$, all other convolutional layers use the $ReLU$ activation function.

The three layers in the encoding stage consists of 32, 16, and C convolutional filters, respectively. Note the parameter C (the number of filters in the third convolutional layer is an hyper-parameter, which determines the number of layers that allow appropriate orthogonality rates). Similarly, the last four layers in the decoding stage are formed by 8, 16, 32, and 1 convolutional filters. All convolutional filters in all convolutional layers are of 5×5 pixels. Also, all max-pooling and up-sampling layers were fixed to pooling size equals 2×2, thus generating outputs half, or double, the size of their inputs, accordingly.

The second branch of our model corresponds to the "Dot" layer, which is connected after the third max-pooling layer. This layer receives as input a tensor of size $[W, H, C]$, where W and H indicates the *width* and *height* of the image response after the previous convolution-pooling block, and C corresponds to the *number of responses* (channels or filters) resulting from that previous stage. The activation function of this layer is the dot product applied on its input tensor, channel-wise, which produces matrix O as output (as explained in Sect. 2.2). The reference data used to compare the output of this second branch, and therefore to calculate the error function defined in Eq. (2), corresponds to the identity matrix of size $C \times C$, as we are using C filters in the third convolutional layer.

We trained our model during 64 epochs using batches of 32 local patches and the *adam* optimizer with default parameters [11]. Since our data consists of binary images, we optimize the *binary cross entropy* as reconstruction loss $\mathcal{L}^r$.

3.3 Evaluation

We evaluated two aspects that the proposed orthogonality constraint could induce in the estimation of local image descriptors via convolutional autoencoders. First, its impact in the reconstruction loss with respect to its contribution to the whole optimization process, as dictated by the coefficient λ. And second, its impact in the reconstruction loss in relation with the dimensionality of the generated local descriptor, i.e., the impact that the number C of convolutional units used to generate the local image descriptor.

4 Results

This section presents the results obtained during the evaluation of the proposed orthogonal local descriptors. Our evaluation focuses on the impact induced in the reconstruction loss of the CAE, by the addition of the orthogonality constraint and by the length of the resulting local descriptor.

4.1 Orthogonality Impact

Table 1 shows the impact of enforcing the orthogonality constraint into the loss function, evaluated on the validation set. These results correspond to the reconstruction loss $\mathcal{L}^r$ (binary cross entropy, bce), the orthogonality loss $\mathcal{L}^o$ (mean average error, mae), and the total loss $\mathcal{L} = \mathcal{L}^r + \lambda\mathcal{L}^o$, as detailed in Sect. 2.3. Moreover, this evaluation corresponds to the use of 512 elements in the real-valued vectors used to compute orthogonality, which, given the fact that the output of our model is of size 8×8 pixels after the third max-pooling layer, implies that there are $C = 8$ filters in the third convolutional layer.

From Table 1, one can see a clear opposite tendency between the reconstruction and orthogonality losses. In general, all of our experiments showed that the

Table 1. Comparison of different values of λ, used for enforcing the orthogonality constraint into the loss function, evaluated on the validation set.

$\mathcal{L}^o$ contribution (λ)	3.00	0.90	0.60	0.30	0.09	0.06	0.03
$\mathcal{L}^r$ (bce)	0.126	0.115	0.108	0.108	0.102	0.102	0.098
$\mathcal{L}^o$ (mae)	0.071	0.080	0.091	0.104	0.118	0.129	0.138
$\mathcal{L}$	0.169	0.163	**0.163**	0.162	0.173	0.179	0.181

more strict the orthogonality penalty λ, the higher the reconstruction loss (bce). Since their combination attains its minimum when using $\lambda = 0.6$, we kept this value fixed for subsequent evaluations.

To validate that, effectively this approach is able to generate local descriptors that are orthogonal to one another, in Fig. 3 we show the visual representations of the orthogonal matrix O (Eq. 3) for different values of λ. Although all three matrices show a diagonal-like patter, it is clear that $\lambda \geq 0.6$ enforces better orthogonality between pairs of convolutional outputs.

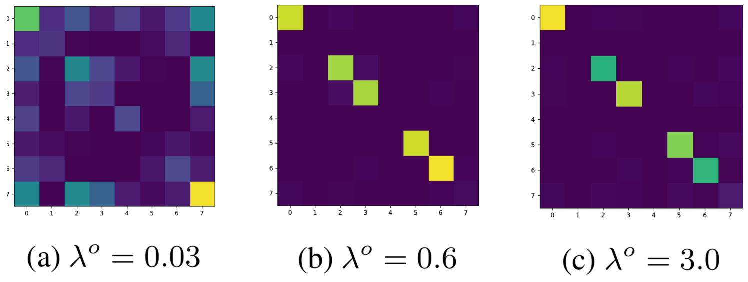

Fig. 3. Visual representations of the orthogonal matrix O for different values of λ. The more the off-diagonal elements are close to zero, the more orthogonal are the corresponding vectors, which are nothing but the responses to the convolutional filters.

For better understanding the impact of the orthogonality penalty λ, Fig. 4 shows examples of patches that have been reconstructed using our methodology, with $C = 8$ filters in the third convolutional layer and $\lambda = 0.6$ (as suggested by the previous analysis). Images in the first row are input local patches segmented from the glyph-blocks, while images in the second row correspond to their reconstructed counterparts. All images in Fig. 4 are well defined in visual terms, and highly similar to their original counterparts. Moreover, only a few pixels have changed their real value. This indicates that allowing a reconstruction loss of around 0.1 in terms of binary cross entropy (induced by $\lambda = 0.6$), brings no serious damage to the reconstruction process.

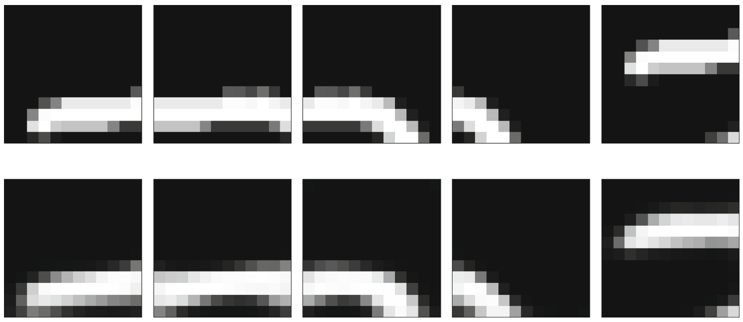

Fig. 4. Examples of local image patches reconstructed by our model, using $C = 8$ filters in the third convolutional layer and $\lambda = 0.6$. First row corresponds to the original image patches. Second row corresponds to the reconstructed patches.

Our results also showed that the proposed approach has good generalization behavior, as the validation error is only slightly higher than the training error for both types of losses, as shown in Fig. 5.

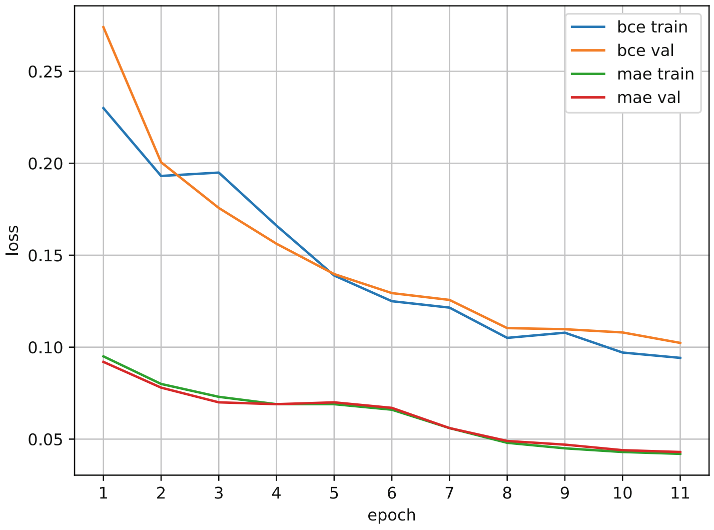

Fig. 5. Training an validation performance for the reconstruction and orthogonal losses.

4.2 Length Impact

The number of elements in the resulting local descriptor is also an important parameter that must be evaluated. On one hand, we would like to obtain descriptors that are large enough to facilitate distributing relevant visual information among their elements, thus favoring orthogonality and sparsity. On the other hand, short descriptors could be desirable for further processing, e.g., indexing or aggregation.

Given the architecture of our model, the output after the third convolution-pooling block is fixed to 8×8 pixels. Therefore, the only varying parameter to modify the total number of elements of this intermediate output is the number C of filters (channels) located in the third convolutional layer. We evaluated the impact on the different losses of our model when varying the number of filters in the set $C = \{1, 2, 4, 8, 16, 32, 64\}$, which respectively correspond to having $64, 128, 256, 512, 1024, 2048, 4096$ elements in the local descriptor.

Table 2 shows evidence of the relation that the length of the local descriptor has with the reconstruction and orthogonality losses, using a fixed $\lambda = 0.6$ as suggested by the results shown in Table 1.

Table 2. Reconstruction and orthogonal losses with respect to the length of the local image descriptor, using fixed $\lambda = 0.6$.

Length	64	128	256	512	1024	2048	4096
$\mathcal{L}^r$ (bce)	0.121	0.109	0.162	0.108	0.112	0.117	0.171
$\mathcal{L}^o$ (mae)	0.148	0.132	0.130	0.091	0.129	0.131	0.162

This evidence shows that local patches are well described using 512 elements, as this length provides the lowest reconstruction loss (bce). This, however, is but a consequence of the design of the model, which we have set to receive inputs of size 64×64 pixels, and process them through three convolution-pooling blocks, which results in responses of 8×8 elements. Different combination of these hyper-parameters might require different numbers of filters in the third convolution, thus resulting in varying lengths for the local image descriptor.

5 Conclusions

We proposed the re-consideration of orthogonality as a constraint for the unsupervised estimation of local image descriptors using Convolutional Autoencoders (CAE). For this purpose we presented a methodology based on the use of a hidden layer that computes the dot product between intermediate outputs, and uses it as a secondary model output, which is subject to a loss estimation, and therefore allows to adjust the network parameters to induce orthogonality.

Our results show that the proposed methodology is suitable for the estimation of local image descriptors that are orthogonal to one another, this without hurting the reconstruction process of the CAE. We also investigated the impact of the length of the resulting local descriptor in the process of reconstructing binary images, and noticed that reconstruction of high visual quality are possible.

Currently we are investigating whether the proposed method might also have an impact in the degree of sparsity of the resulting local descriptors, or serve as a new type of regularizer. Additionally, this approach can be tested in specific tasks like image classification or image retrieval.

Acknowledgments. This research was supported by the Asociación Mexicana de Cultura A.C.

References

1. Aharon, M., Elad, M., Bruckstein, A.: K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans. Signal Process. **54**(11), 4311–4322 (2006)
2. Roman-Rangel, E., Odobez, J.-M., Gatica-Perez, D.: Assessing sparse coding methods for contextual shape indexing of Maya hieroglyphs. J. Multimedia **7**, 179–192 (2012)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2012)
4. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations, ICLR (2014)
5. Szegedy, C., et al.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2015)
6. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2016)
7. Roman-Rangel, E., et al.: Transferring neural representations for low-dimensional indexing of Maya hieroglyphic art. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9913, pp. 842–855. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46604-0_58
8. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) ICANN 2011. LNCS, vol. 6791, pp. 52–59. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21735-7_7
9. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
10. Gatica-Perez, D., et al.: MAAYA: Multimedia Methods to Support Maya Epigraphic Analysis. Arqueologia computacional: Nuevos enfoques para el analisis y la difusion del patrimonio cultural (2017)
11. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. In: International Conference for Learning Representations, ICLR (2014)