TAG-SENSITIVE FEATURES FOR LARGE-SCALE SCENE CLASSIFICATION

Andre Filgueiras de Araujo

afaraujo@stanford.edu

ABSTRACT

Scene classification is a fundamental research problem in computer vision. However, performances of state-of-the-art systems are still far from acceptable. This motivates the search for improved descriptors and features to represent the scene. Multimodal learning approaches provide a way of building meaningful dictionaries that can take into account not only the visual structure, but also the tags associated to example scenes. In this work, we experiment with a 'Y-shaped network', and get encouraging preliminary results, comparable to very complex hand-crafted descriptors. Future work will explore promising new architectures, taking into account the full content of images.

1. INTRODUCTION

Unsupervised feature learning has been successfully applied to a variety of tasks in computer vision, audio, natural language processing, among others. In most cases, it outperforms hand-designed methods. Scene categorization is a fundamental research problem in computer vision. However, most work have been restricted to problems involving at most tens of classes.

Recently, the Scene Understanding (SUN) dataset [1] was released, comprising 899 classes and more than 130 thousand images. Modern methods, combining several carefully hand-designed complex features, achieve 38% classification accuracy with this dataset - which can be regarded as a satisfactory result given the difficulty of the problem. Some examples of scenes drawn from this dataset are given in Figure 1.

Concretely, the current state-of-the-art approach employs 12 different descriptors (GIST, HOG2x2, Dense SIFT, LBP, Sparse SIFT, SSIM, Tiny Images, Line Features, Texton Histograms, Color Histograms, Geometric Probability Map and Geometry Specific Histograms) by combining their SVM kernels in a weighted sum to obtain the final classification score. The construction of the descriptors involves, in most cases, dense extraction of patches, followed by a histogram assignment.

The main objective of this work is to design meaningful representations for scene classification in an unsupervised fashion. In such a highly complex scenario, the ability of a human to design efficient features can be hindered, making a case for the use of features that are learned automatically. We employ multimodal learning, with the objective of obtaining representations that take tags into consideration.

Fig. 1. Examples of scenes in the SUN dataset.

We considered the outcome of the work to be encouraging, given the very preliminary results that are presented here, and plan to continue on working on this problem by exploring novel machine learning architectures, as will be mentioned later.

This report is organized as follows. Section 2 introduces the utilized feature learning algorithms. Next, in Section 3, the classification scheme is presented. Experimental setup and results follow in Section 4. Finally, we conclude in Section 5.

2. FEATURE LEARNING

In this Section, we present the feature learning method that was employed to discover structure in images.

In this work, we employ one of the variations of Multimodal Learning, as introduced in [2], namely the "Y-shaped" configuration, in which one modality (in our case, image patches) is used to learn a shared representation, capable of efficiently describing two other modalities (in our case, image patches and tags). This architecture can be regarded as a generalization of a simple autoencoder, as will be seen below.

The architecture of this learning algorithm is shown in Figure 2. It consists of a fully-connected neural network with

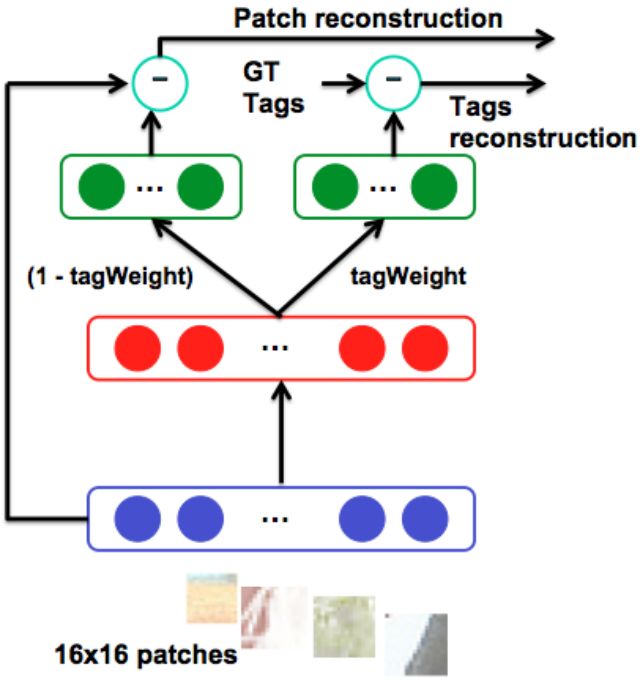

Fig. 2. Architecture of the Y-shaped network. 'GT Tags' stands for 'Ground-truth tags'.

one input layer (in which one modality is input), one hidden layer (which will learn a shared representation) and two parallel decoding layers (one for each modality). The balance between each modality's importance can be regulated by a weight (tagWeight, as shown in Figure 2).

In our case, the objective is to use randomly sampled 16x16 colored patches to learn efficient representations for the patches themselves and for the tags. To operate on image patches, it is important to, first, remove their means, in order for the data to be normalized. Also, it is common to apply PCA-Whitening in order to a) reduce the dimension of the data to the most important components, and to b) normalize the variance of each component to one. No pre-processing is performed on the tags, which are simply represented by a binary vector indicating the presence/absence of each tag.

Each node in the hidden layer (HL) performs a non-linear operation; specifically, the activation of the nodes in the hidden layer is given by:

$$a = f(W_H x + b_H) \tag{1}$$

where a is the activation of the hidden layer, $f(.)$ the sigmoid function, W_H and b_H the parameters and x the input vector (the vectorized 16x16 colored patches) to the layer.

For the output layer, we employ linear operations for the image patches and non-linear operations for the tags. The activation of the output layer is, then, given by:

$$y_i = W_i a + b_i \tag{2}$$

$$y_t = f(W_t a + b_t) \tag{3}$$

with a and $f(.)$ as above, y_i, W_i and b_i the output and the parameters of the image patch output layer, respectively, and, similarly, y_t, W_t and b_t for the tags output layer.

Each decoding branch is optimized independently, while the hidden layer takes into account both modalities to construct its weights. An optimization problem can then be formulated, targeting the minimization of the reconstruction error (J_{rec}). As is common practice in the field, we add to this optimization problem a regularization term for each nodes' weights, J_w, (to avoid overfitting) and a sparsity constraint on the hidden layers, J_s (to discover interesting structure). Mathematically, our overall cost function, J, is, then:

$$J = J_{rec} + \frac{\lambda}{2} J_w + \beta J_s \tag{4}$$

For the reconstruction error, we adopt a squared loss function for the image patches (J_{rec-i}) and a cross-entropy function for the tags (J_{rec-t}):

$$J_{rec} = (1 - TW) J_{rec-i} + (TW) J_{rec-t} \tag{5}$$

$$J_{rec-i} = \frac{1}{m} \sum_{k=1}^{m} \frac{1}{2} \|y_i(k) - t_i(k)\|^2 \tag{6}$$

$$J_{rec-t} = \frac{1}{m} \sum_{k=1}^{m} \sum_{j=1}^{V} [t_t(k,j) \log(y_t(k,j)) + (1 - t_t(k,j)) \log(1 - y_t(k,j))] \tag{7}$$

where m is the number of training examples, V is the vocabulary size, TW is the trade-off parameter tagWeight ($0 < TW < 1$), $y_i(k)$ and $y_t(k,j)$ are the image patch and tag outputs for the training example k (and tag j), $t_i(k)$ and $t_k(k,j)$ are the ground truth outputs for the training example k (and tag j).

The weight decay and sparsity terms are as usual, and due to the limited space, we don't rewrite their expressions here.

This specifies the cost function completely. To employ an optimization algorithm and minimize J in function of W_H, b_H, W_i, b_i, W_t, b_t, we need to calculate the gradient of J in terms of each of those parameters.

For that, we employ backpropagation. Again, for limitation of space, we won't rewrite all the equations for the calculation of the gradient. We note, however, that these correspond simply to a superposition of what is commonly employed for an usual autoencoder for each of the branches (the tag branch backpropagating from a sigmoid output, and the image patch branch backpropagating from a linear output), weighted accordingly with TW, for the tag branch, or with $1 - TW$, for the image patch branch.

It is important to note that the case where $TW = 0$ is identical to a common autoencoder setting.

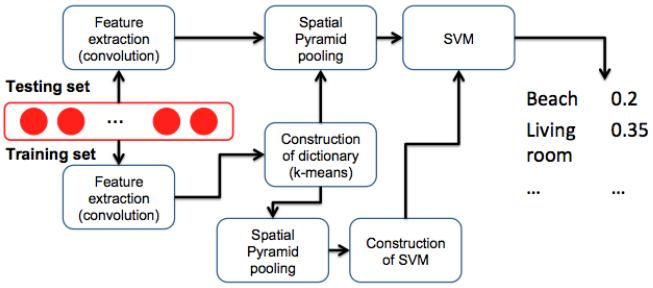

Fig. 3. Scheme utilized to predict the scene class.

3. CLASSIFICATION SCHEME

In this Section, we present the scheme that is employed to perform classification, shown in Figure 3.

Initially, using the hidden layer weights that were discovered by the methods introduced in Section 2, the features corresponding to each position in each image in the training set are extracted (in other words, we multiply each patch by W_H, as is done in Equation 1). This generates a 'response map' for each hidden layer node, for each image.

Then, with all these responses, a codebook is built using k-means, to enable capturing the diverse types of outputs. In the next step, the response maps are pooled using a spatial pyramid, constructing a histogram for each section, at each pyramid level. As is common practice, these histogram-like vectors are concatenated to form the final feature vector for the image. An SVM is then trained based on these final feature vectors.

For the testing images, similarly, the response maps are constructed and pooled using the dictionary generated in training stage. The resulting histogram-like feature vectors are input to the previously constructed SVM, to generate scores for each class (in Figure 3, these are illustrated by scores '0.2' and '0.35' to the classes 'Beach' and 'Living Room').

4. EXPERIMENTS AND RESULTS

In this Section, the experimental setup and obtained results are given. We start with feature learning results, then present results for classification on the SUN dataset.

4.1. Multimodal Learning

We used more than 300,000 images crawled from Flickr. These images comprise all sorts of variations which are inherent in images, such that we believe the learned features will be representative enough to be used in scene classification problems. Also, as they present tags associated to each image, they allow for the learning of patches sensitive to textual data.

16x16 patches were extracted at random positions in randomly selected images, making up a total of 300,003 patches, which were saved together with their tags, and used as inputs to the scheme presented in Section 2. For each patch, we made sure there were at least two tags associated to the image it was in. A vocabulary of 2000 tags was used, corresponding to the most frequent words present in the dataset.

In order to make our results directly comparable to the best method presented in [1] (namely, HOG 2x2), we decide to employ 124 hidden layers (in HOG 2x2, 124 features are extracted from 16x16 patches) per color and patch size of 16x16x3 (in other words, to be clear: input layer and patch output layer of size 768 and hidden layer of size 372). As the optimization algorithm, we employ L-BFGS.

Results are shown in Figure 4, which gives the learned weights W_H for each value of tagWeight, TW, as introduced in Section 2. As previously mentioned, the case in which $TW = 0$ reduces to an usual autoencoder scenario.

We observe that many edge-like structures are captured, bearing strong resemblance to usual Gabor filters. Also, it is interesting to notice that the majority of bases do not present color, despite being trained with color inputs.

The case in which $TW = 1$ does not learn a useful basis. This can be seen by the fact that the standard deviation for each of the bases is very low - this indicates the algorithm could not optimize the weights, due to very large error.

It is important to say that this result agrees with what was expected: since the patches are not very informative of the tags (e.g., the patches are usually similar for very different scene classes), the result for $TW = 0.25, 0.5, 0.75$ does not change much from $TW = 0$. This can be understood since the error in the tag branch is extremely high, due to the inefficacy of predicting tags based on single patches.

As previously mentioned, this configuration represents an intermediate level between an usual autoencoder and other neural network-based unsupervised learning architectures, which represents the final goal of this work, to be continued as a research project.

4.2. Scene Classification

All experiments are performed with respect to the exact same training and testing partitions as used in [1], so that all results are directly comparable.

Our codebook was constructed using k-means with 300 centroids, and the spatial pooling is performed in 3 levels (resulting in $1 + 4 + 16 = 21$ partitions), as is common practice. The SVM is implemented using libSVM [3] with a histogram intersection kernel.

Initially, we compare the results for classification on SUN Dataset for all the different configurations of tagWeight, using 5 training examples per class. This serves to understand which one is more useful to the task and, as the dataset is very large, this experiment can give results in a reasonable amount

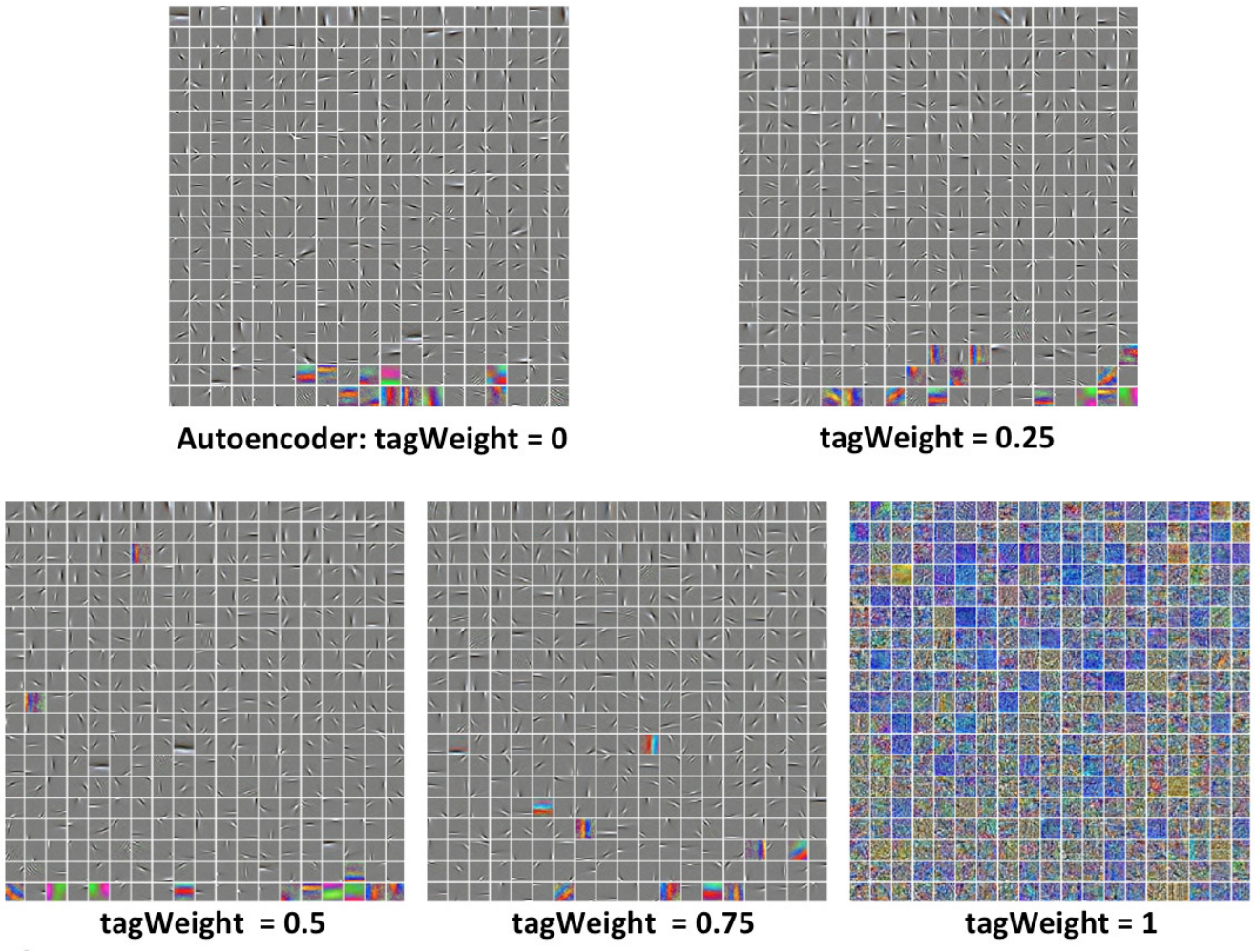

Fig. 4. Dictionary of learned weight parameters W_H for the hidden layer, for each tagWeight (= TW) configuration. The bases are sorted in ascending order of standard deviation. The bases for the configuration tagWeight = 1 are almost constant, indicating no effective learning - however, as the visualization is normalized to the full range of the input, the bases have this noisy appearance

of time. We also note that, in [1], no crossing of the learning curves of the different methods is observed - which indicates that it is very likely that the method which performs better for 5 training examples will perform better with more training data.

The results for this first classification experiment are given in Table 1. We observe that the features learned with the autoencoder present higher results in terms of classification accuracy. This agrees with the results from Subsection 4.1: the tags do not help the learning of meaningful representations of the data, since the patches cannot be good examples of the observed tags.

Next, we run the full experiment to obtain the learning curve and compare the results with what is currently state-of-the-art for this dataset. Based on the results from the previous experiment, we use only the weights from the configuration

Method	Accuracy (%)
tagWeight = 0 (AE)	4.7105
tagWeight = 0.25	4.2445
tagWeight = 0.50	4.0439
tagWeight = 0.75	4.1293

Table 1. Classification results for the different learned features on SUN dataset using 5 training examples per class.

$TW = 0$ (i.e., the autoencoder configuration). Due to the very high processing time required for the full experiment to run, we were not able to perform parameter tuning, and the presented results are still quite preliminary.

The learning curve is given in Figure 5, together with results for other methods, as in [1]. Our method obtains results

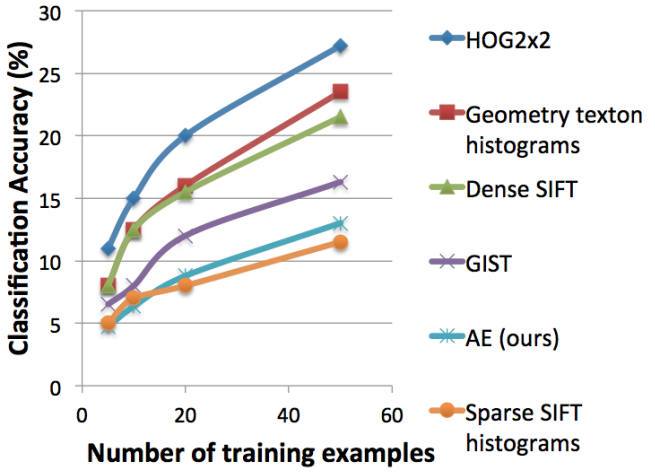

Fig. 5. Learning curve for our method, together with previously reported results.

which are still not comparable to state-of-the-art. However, it is encouraging to observe that it can perform comparably to highly non-linear and manually designed approaches such as SIFT.

5. DISCUSSION

In this work, we employed unsupervised feature learning techniques with the objective of discovering meaningful structure in scenes, which would be helpful to perform classification. The final results are encouraging: our algorithms construct descriptors that perform comparably to highly non-linear, hand-crafted descriptors, such as SIFT.

We experimented with multimodal learning, to be able to discover representations that would be relevant to describe not only segments of images, but also textual tags. However, our results show that the tags do not help learning - this was not surprising, since they are not descriptive of the image patches in general. At the limiting case where only the tags are taken into consideration, no learning occurs.

In this scenario, an usual autoencoder performs best - i.e., the case where the tags are not taken into consideration. Gabor-like dictionaries, showing very well-defined edges, were discovered directly from a database containing more than 300,000 images crawled from Flickr.

Classification was implemented by applying usual schemes, as spatial pyramid pooling and SVM. As expected, the autoencoder performs best among the other methods, which included nonzero TW. Our best results are still not comparable to the state-of-the-art results given by HOG2x2 in [1], but are interesting in the sense that they are comparable to complex descriptors without any parameter tuning.

Future work is intended to take the next step and learn representations based on different network architectures, such as

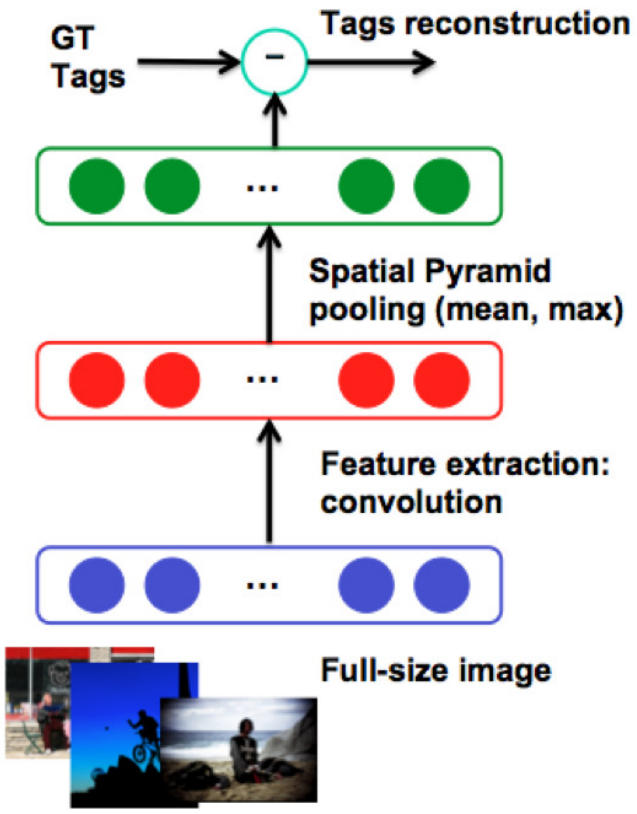

Fig. 6. Architecture to be employed in future work: the full image is considered as the input, and the discovered structure aims at representing the tags accurately.

the one in Figure 6. In this case, the full image is considered, and thus we expect it to perform much better, since in this scenario the tags should be coherent with the image under consideration.

6. ACKNOWLEDGEMENTS

The author acknowledges Andrew Maas and Maurizio Caligaris for the help with discussions and implementations.

7. REFERENCES

[1] J. Xiao, J. Hays, K. Ehinger, A. Oliva, A. Torralba "SUN Database: Large Scale Scene Recognition from Abbey to Zoo," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* San Francisco, CA, June 2010

[2] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, A. Ng "Multimodal Deep Learning," *International Conference on Machine Learning* Bellevue, WA, 2011

[3] C.-C. Chang, C-J. Lin "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, 2011.