A Revision of Image Classifiers for Scene Classification: from Handcrafted to Data-Driven Feature Learning

Martí Cobos*, Jonatan Poveda*
*Universitat Autònoma de Barcelona, Barcelona, 08193, Spain
{marti.cobos, jonatan.poveda}@e-campus.uab.cat

Computer Vision Center

***Abstract*—This document summarizes the research performed by our team regarding the Image Classification project developed during the *Machine Learning for Computer Vision* module in the framework of the *Master in Computer Vision Barcelona*. This document describes the algorithms developed, their evaluation, conclusions and further improvements. The objective of the project is to compare different machine learning techniques and approaches to classify images of landscape scenes into 8 different classes. Two different approaches were used to solve the classification problem: hand-crafting features and using data-driven learned features. The former consists on defining the features and applying standard machine learning classifiers, such as a SVM classifier. Later, Convolutional Neural Networks are implemented, as an end-to-end solution, to obtain classifier. At the end of the document, experimental results are shown, and all the assessed classifiers are compared.**

***Index Terms*—Computer vision, Image classification, Feature extraction, Supervised learning, Machine learning, Artificial neural networks, Multilayer perceptrons, Multi-layer neural network, Support vector machines**

I. Introduction

THIS document is a report of the project developed as the practical part of the *Machine Learning for Computer Vision* module from *Master in Computer Vision Barcelona*. The goal of the project is to implement and assess different machine learning techniques in order to compare their performance in a landscape classification algorithm.

The project was developed in Python using the following relevant open-source libraries: OpenCV, scikit-learn, Keras and TensorFlow. A small dataset of 2694 landscape images belonging to 8 classes (see Fig. 1) is used to train and evaluate the system.

In the following sections, the machine learning (subsection II-A) and deep learning (subsection II-B) techniques used for the classification are explained. The process of fine-tuning these techniques is also described. Then, performance results (section III) are explained and compared. Finally, the project's conclusions, as well as future work to improve the results, are presented on section **??**. The code developed during this course is fully published on GitHub [**?**] under a GPL-3.0 License.

Fig. 1. Scene types in the dataset.

II. Methodology

A. Hand-crafted methods

The first set of techniques used to solve the image classification problem consists on extracting several handcrafted image descriptors (either local or global image descriptors) and, afterwards, training a standard classifier, such as K-NN or SVM, in order to obtain a classification model. Fig. 2 summarizes this process in the case of the BoVW framework implementation.

In this sections several machine learning techniques using hand-crafted features will be described. First, the use of local descriptors is discussed on subsection II-A1. Then local features are replaced with the use of global image descriptors on subsection II-A2. Finally, the Bag of Visual Words framework [1] is discussed on subsection II-A3. Finally, the performance results of those techniques are shown on section III.

1) Basic classification with local features: The first approach to solve the scene classification problem consists on computing several local features on several points of the image

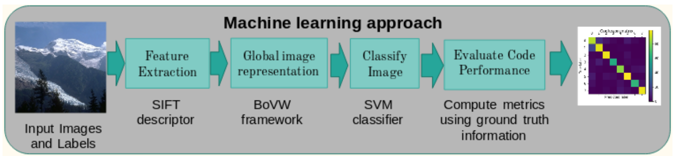
Machine learning approach
Input Images and Labels
Feature Extraction
SIFT descriptor
Global image representation
BoVW framework
Classify Image
SVM classifier
Evaluate Code Performance
Compute metrics using ground truth information

Fig. 2. Machine learning approach for image classification.

and classify the image based on the classification of those points. As well as most classification systems, this approach has two stages. Firstly a model must be trained using a train image database and later the classification algorithm must be evaluated by classifying the test image database using the trained model.

In order to obtain image local features, SIFT [2] descriptor and detector were used. As a classifier, the following techniques were tested:

1) K-NN
2) Random forest
3) Gaussian Bayesian Classifier
4) Bernouilli Bayesian Classifier
5) SVM
6) Custom Logistic Regression

The custom logistic regression implementation is based on the gradient descent algorithm with the following cost function:

$$\frac{\partial}{\partial \Theta_j} J(\Theta) = \frac{1}{m} \sum_{1}^{m} (h_j(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \Theta_j \tag{1}$$

The logistic regression has been implemented with the following features:

– Convergence detection.
– Selectable maximum number of iterations.
– Lasso regularization with selectable lambda parameter.
– One vs All multi-class classification.

In order to select the best classifier and set of hyper-parameters, cross-validation was implemented with stratified K-fold. This technique consists on splitting the train image set into small sets. Each set is used to validate the results of training the classifier with the rest of image sets. the best hyper-parameters are obtaining by analysing the average performance on all the dataset folds. Once the best configuration is obtained, the model with the full dataset.

All the techniques discussed on section II-A obtain the best configuration of the classifier by trying the above mentioned classifiers and several hyper-parameters using cross-validation.

When using SIFT as local feature descriptor, SVM with a radial basis function kernel was found that yield the best performance. However, the accuracy obtained using with this local feature and classifier lower than 20% and they are not shown on section III.

2) Image classification with global features: A first naive approach to obtain a global image was implemented by computing the image colour histogram. First the image is converted into the Luv colour space in order to separate the colour information (uv) from the illuminance information (L).

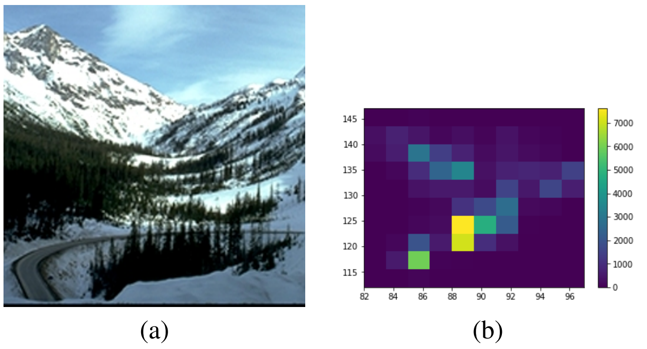

Fig. 3. Example of a scene converted to a colour histogram on the uv colour space: (a) initial RGB image, (b) colour histogram

Then the colour histogram is computed for the uv channels. The global image descriptor is the value of each one of the histogram bins. The size of the descriptor is equal to the histogram number of bins squared. Fig. 3 shows the colour histogram representation of an image.

As well as on section II-A1, the best configuration of classifier and hyperparameters is obtained by applying cross-validation. The optimal classifier found, for the colour histogram image descriptor, was an SVM with a radial basis function as the kernel. The performance of this system is shown on section III.

3) Bag of Visual Words framework: In order to improve results obtained on sections II-A1 and II-A2 the Bag of Visual Words (BoVW) is implemented. At First the standard BoVW, as described on [1] is implemented. Afterwards, several modifications are performed in order to further increase classification performance. The following BoVW variations are implemented and tested:

1) Standard BoVW with standard SIFT descriptor.
2) Standard BoVW with dense SIFT descriptor.
3) Extended BoVW (spatial pyramids and histogram intersection kernel) with dense SIFT descriptor.

The main idea behind BoVW framework is to aggregate several image local features into a global histogram representation of the image. First the SIFT descriptors of the images on the train set are computed. K-means clustering method is applied to all the descriptors in order to learn the vocabulary of the words contained on the images. Afterwards the histograms of words contained on the images are computed. This histogram is used as a global image representation for the classifier.

Once the vocabulary has been learnt and the model computed, an image can be classified. First, the image SIFT descriptors are computed. The image word histogram must be computed using the vocabulary learnt from k-means. Finally, this histogram is used as descriptor for the classifier.

One of the main drawbacks of this implementation is the fact that SIFT only computes descriptors on corners found by its detector. Plain regions (low contrast between pixels) of the image are not taken into account for the classification. This could lead to remove the background or uniform areas from the descriptor. In order to add information of the entire image, the dense detector is used. This detector places keypoints uniformly over all the image like a uniform sampling.

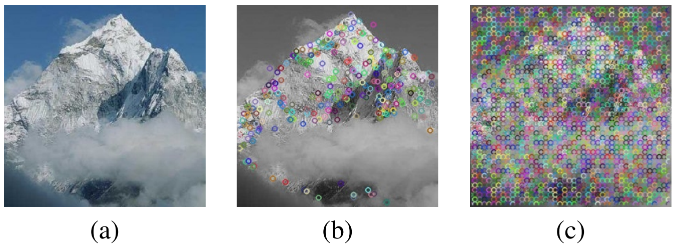

Fig. 4. Example of SIFT descriptures computed on a scene image: (a) initial RGB image, (b) standard SIFT, (c) dense SIFT

This allows the BoVW to have information relative to all the image.

A further improvement on this system is to extend the BoVW with the use of spatial pyramids [3] and the histogram intersection kernel for the SVM classifier. The spatial pyramids enhancement consists on several levels of splitting the image on smaller patches. Words histograms are computed for each patch on each level and all histograms are concatenated by applying a regularization factor at the end. Figure 5 shows an example of extending the BoVW histogram by using spatial pyramids.

As the number of descriptors is greatly increased, execution time of the training and evaluation must be reduced. Applying an Histogram Intersection (HI) reduces computational cost of training and evaluating the SVM when using spatial pyramids and allows to develop the system on a feasible time. The histogram kernel is computed using the following equation:

$$K_{int}(A,B) = \sum_{i}^{m} min\{a_i, b_i\} \tag{2}$$

Best configuration of the classifier and hyper-parameters are obtained by applying cross-validation for each BoVW variation. Table I shows accuracy cross-validation accuracy. As expected the system which implements dense SIFT, spatial pyramids and a histogram intersection kernel for the SVM yielded best results, due to the fact that takes into account information of key-points placed uniformly into the image and its location. The performance of this system is also shown on section III.

TABLE I
ACCURACY RESULTS FOR BOVW VARIATIONS

Method	Accuracy
Standard SIFT	.660
Dense SIFT	.793
Spatial Pyramids and HI kernel	**.844**

B. Data driven methods

The second approach to build classifiers is use data driven methods. In contrast to the former method, we are not defining descriptors, we are going to learn this descriptors from the training dataset.

As the following methods use a massive number of parameters, we should consider the size of the classifiers for different reasons. First is the learning time of our classifier, and second the over-fitting effect. The size of a Artificial Neural Network (ANN) must match the complexity of our problem. In classification, the complexity refers to how hard is to separate our dataset into different classes. The more complex our problem the more expressive classifier we need. Note that having a too expressive classifier leads it to over-fitting. That means our classifier is going to memorize the input data and not going to generalize well. This effect can be seen comparing the training and validation accuracy curves. When the gap between training and validation accuracy is too large, we can state our classifier is over-fitting. How to avoid this effect is shown further in this section.

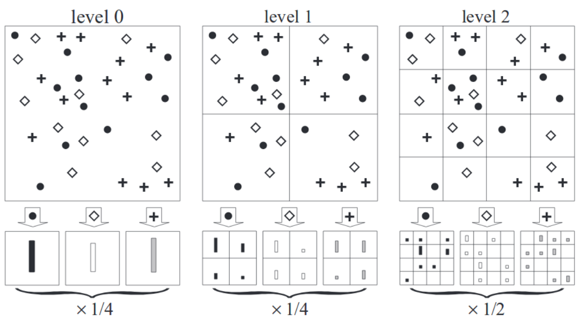

Fig. 5. Spatial pyramids for BoVW framework.

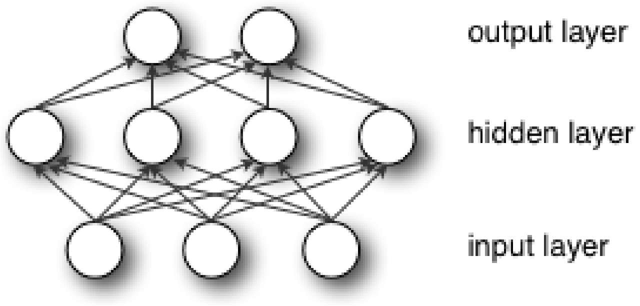

Fig. 6. A Multi-layer Perceptron with one hidden layer

1) From hand-crafted features to learnt features: As done in II-A, we start with a naive approach. We try to learn the features from a Multi-layer Perceptron (MLP) (Fig.6). A MLP is an ANN with an input layer, one or several hidden layers and an output layer. The size of the layers depend on our problem and designing parameters. The input layer contains as many neurons as input values (in our case, the number of pixels of a dataset's image). The output layer contains as many neurons as number of classes to classify, eight in our case. The number of neurons of the hidden layers as well as how many layers we use it depends on our design. As more layers and more neurons we can achieve a more expressive neural network which could solve more complex problems. All the layers in a MLP are fully-connected.

Our approach with MLP consist in resizing the input image, three hidden layers in decreasing size and an final layer of 8 neurons to have 8 predictions, one for each landscape class (Fig. 7). Two experiments were done, one with only the MLP and the second one extracting the features on the last

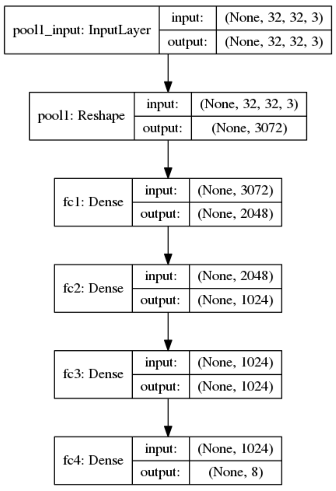

Fig. 7. Our MLP diagram

FC layer and training a SVM with them. Both experiments resulted in similar results. As shown in II the training time is significantly large due to the number of parameters is more than 18 millions. Although it larger size the accuracy is much lower than our BoVW. Analysing our problem, which is an image classifying problem we must consider to exploit space correlation images have. Using this naive method we are discarding this information from the start due to the reshape applied on the input layer. Each pixel of the image is treated independently as an one dimensional signal.

2) Fine tuning of pre-trained CNN: In order to take into consideration the space correlations images have, we implement Convolutional Layers. These layers can be explain as a 2D filter which scans the full image with a shifting window and doing a convolution on each frame. In Convolutional Neural Networks (CNN) the size of the window is and the shift are called Receptive Field and Stride, respectively.

The first approach using CNN is to use an already existent one and adapting it to our problem. The model VGG16 [4], presented on Large Scale Visual Recognition Challenge in 2014 (LSVRC-2014) and in the second position in classification and localization, is selected for the experiment. This CNN is designed to classify over 1000 classes. In order to use it for our problem an adaptation is done (Fig.8). The last block of layers were replaced by a Pooling to reduce the number of parameters, then adding two FC layers and a prediction layer to have 8 outputs which are going to provide us of eight class probabilities. For the purpose of using a pre-trained classifier, the data normalization done in the original network must be known and applied to our dataset. Otherwise we are going to feed the classifier with data so different from the training and validation that the pre-training would be useless.

In order to do the best of the VGG16, the parameters learn by its team are loaded in memory in read-only basis. Therefore the training process only updates the new parameters introduced, skipping the loaded ones. This allows to reduce drastically the training time cause back-propagation is done in a small part of the net. The size of the layers were found using a random search on the hyper-parameter space in order

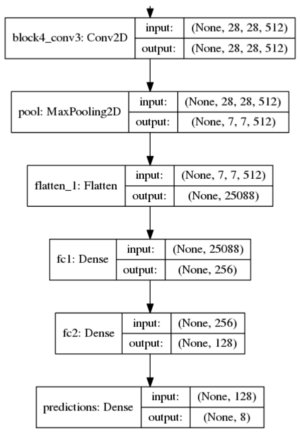

Fig. 8. The modification to VGG16

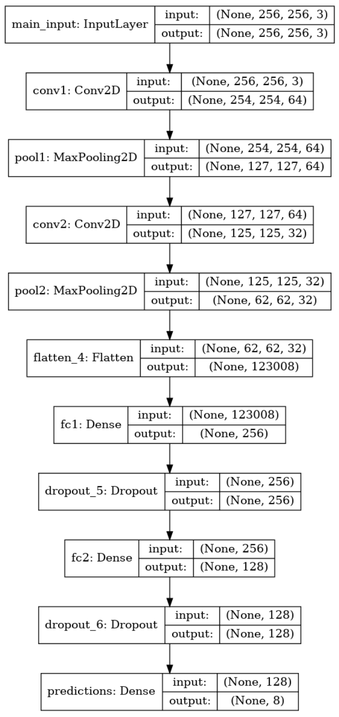

Fig. 9. Our CNN diagram

to find the optimal. This takes a considerable amount of time hence the values are sub-optimal. Our last experiment with the VGG16 was to allow modifying the original parameters for the first part of the net once after the last block of layers were trained during some epochs. This caused an slight improvement due to an small adaptation of the overall network to our dataset. The best performance found is much better than our last experiment using MLP (Table.II).

3) Train a CNN: As the last group of experiments we design our model from the scratch with all what we learn in the process.

The motivation is clear, the VGG16 model is pre-trained using a general dataset and it is able to classify over 1000 classes. Our problem has an smaller scope. Our dataset is specific of landscapes and only has 8 different classes. Therefore, it seams feasible to build an small CNN for our purpose.

We did several tests with mainly two convolutional and two FC layers, changing its size and inserting Max Pooling and Dropout [5] between layers. A Max Pooling reduces the number of parameters, propagating only the *predominant* ones. The dropout technique consist in randomly deactivate some of the neurons. The idea behind the dropout is that when some neurons are deactivated others have to learn the same features, reinforcing the feature learning over all the layer and reducing the super-specialization of the neurons, in other words, reducing the over-fitting. We explored this hyper-parameters as well as shrinking the input image, which reduces the number of parameters dramatically.

In order to push further the training, some data augmentation is introduced in our training and validation dataset. For example, we apply horizontal flipping, horizontal and vertical displacement, an a slight rotation. This technique increases the performance some points.

It is also recommended for many authors to do some kind of normalization to both the input image and its features, like zero meaning and normalize their standard deviation.

The best model we got is represented in Fig. 9. The results in Table. II are after a 50 epoch training, applying some data augmentation and data normalization. As an back-propagation optimizer we used Adadelta [6] with the parameters *learning rate* to 1 and *rho* equal to 0.95. The dataset split is 70% for training, 18% for validation and the reminding for the test. The batch-size is formed by 128 images and each epoch consisting of 300 iterations. We set the accuracy as the metric to maximize.

III. Results

The classifier assessment has been done using the following metrics:

$$\textit{Accuracy} = \frac{TP + TN}{P + N} \tag{3}$$

$$\textit{Precision} = \frac{TP}{TP + FP} \tag{4}$$

$$\textit{Recall} = \frac{TP}{TP + FN} \tag{5}$$

$$\textit{F1 score} = \frac{2TP}{2TP + FP + FN} \tag{6}$$

In order to compare the different techniques implemented, the following image classification pipelines have been evaluated:

1) Colour histogram with 32 bins as a global image descriptor with an SVM classifier.
2) Bag of Visual Words (BoVW) framework improved with Dense SIFT, Spatial Pyramids and histogram intersection kernel [7] in conjunction with a the SVM classifier.
3) Multilayer perceptron (MLP) with 3 fully connected layers applied to an SVM.
4) Neural network based on VGG16 [4], fine-tuned and adapted for 8 classes.
5) Custom end to end Convolutional Neural Network (CNN) described in section II-B3.

Evaluation results of these methods are shown in Table II.

TABLE II
Performance results

Method	Accuracy	Precision	Recall	F1	CPU time[s]
Colour H.	.265	.274	.267	.270	1.17
BoVW	.844	.853	.844	.849	1104
MLP	.660	.674	.667	.670	15172
VGG16 Mod.	.747	.780	.729	.784	2269
Our CNN	**.853**	**.867**	**.846**	**.856**	**21695**

Due to the fact of GPU acceleration in Tensorflow [8] the training time was cut down more than seven times using one GPU (Nvidia Titan Xp). This library was used in the three last methods but for the sake of comparison we run them once using the same hardware. All the experiments were run in an 8 cores Intel Xeon E5-2683 v4 at 2.10GHz and 256GB of RAM.

IV. Conclusions

In this study case, an 8-class classification, a random classification would stochastically result in an accuracy of 12.5%, considering a balanced dataset. In our experiments we can see that the naive technique of using a global colour histogram does not work, since its accuracy is just a 26%. On the other hand, using a more complex system using hand-crafted features, as our Method 2, demonstrate that in the case of a small dataset the resulting classifier is similar to using a small CNN. It deserves to mention that the training time is nearly 20 times less in BoVW than in our CNN. Also note that the reduced training time for the VGG16 modified is due to back-propagation is only computed on the additional layers instead of all the network.

In our experiments we note that introducing Max Pooling after convolutional layers and Dropout in FC layers improves the accuracy while reduces the number of parameters of the net. We also analyze how the image size affects the performance. It results that using smaller images, in our case 64x64 pixels, improve the performance. This effect leads us to conclude that fine details of images of landscape are not important for classification. Moreover, it highly increases the number of parameters of the net and increases the potential over-fitting on training.

V. Future work

Owing to time constrains we could not test more classifiers and achieve a plateau on loss and accuracy curve. Therefore we are sure we could get a better performance increasing the number of epochs and adding some modifications to the architecture of our CNN. Furthermore, the hyper-parameter space could be expanded in order to fine-tune more the model. It is in our roadmap to implement Visualization to detect weaknesses in any part of the network as well as give a try to the novel Capsule Networks, based on CNN. As on any problem, it is always good to have more data to train and validate a classifier. For this reason and due to there is a lot of landscape photographies over the web and social media, it is feasible to improve our model crawling more images and building a larger dataset.

References

[1] J. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary," January 2005. [Online]. Available: https://www.microsoft.com/en-us/research/publication/object-categorization-by-learned-universal-visual-dictionary/

[2] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157 vol.2.

[3] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 2169–2178.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] N. Srivastava, G. Hinton *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[6] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701

[7] A. Barla, F. Odone, and A. Verri, "Histogram intersection kernel for image classification," in *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, vol. 3, Sept 2003, pp. III–513–16 vol.2.

[8] M. Abadi, A. Agarwal *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/