Image Classification Challenge

Emile Mathieu
Department of Computer Science
Ecole des Ponts ParisTech
emile.mathieu@eleves.enpc.fr

Thomas Pesneau
Department of Computer Science
Ecole des Ponts ParisTech
thomas.pesneau@eleves.enpc.fr

1 Introduction

The challenge to be addressed is an image classification problem, in which 2000 images must be labeled with the help of 5000 labeled images. These are RGB images of size 32x32, each belonging to one of the 10 classes. Our approach is separated in two parts: feature extraction and classification. Rather than considering the images to be vectors in $\mathbb{R}^{32\times32\times3}$, smarter discriminative features of the images can be extracted.

2 Feature Learning

Images lie in a low-dimensional manifold, so we would like to build a feature extraction process which yield to a representation in such a manifold.

2.1 SIFT descriptors

SIFT descriptors represent local characterization of an image [11]. The descriptors are histograms of gradient orientations around key points of the images where the key point orientation was subtracted to get rotation invariance. We compute these descriptors for every image in X_{train}.
The usual kernels are useless since images may not have the same number of descriptors. The pyramidal match kernel [9] adresses this issue by separating the feature spaces in bins and computes histograms from these. It returns the weighted sum of the number of newly matched features at each scale.
The drawback of this approach is that the images are small and pre-processed. Experiments with the Matlab implementation of SIFT in the vl_feat library, and with our own implementation revealed that some images do not have descriptors.

2.2 Supervised feature learning

Convolutional Neural Network (CNN) [10] are known to be good at learning invariant features in images. Yet, as suggested in [7] and [13], CNN are not always optimal for classification while SVM are good at producing decision surfaces from well-behaved feature vector. These authors propose to use an hybrid system where a CNN is trained to recognize images, and a Gaussian-kernel SVM is trained from the features learned by the CNN.

This idea is appealing on a kernel point of view because integrating the feature extraction process $c(x)$ with a regular kernel $K(c(x), c(x))$ is equivalent to constructing a new kernel $K_c(x, x)$ on raw images. If K is a Mercer kernel, K_c also satisfies the Mercer condition. This means that a kernel K_c can be learned by cascading a learned feature extraction c with a fixed kernel function K.

Implementation We implemented a modular deep learning library inspired by pytorch. Modules can be combined so as to build deep neural networks which can then be trained by choosing a loss function and an optimizer method.

Computational speed-up The computational cost of pooling or 2d-convolution layers can be prohibitive if naively implemented. We followed the *im2col* approach which consists in rearranging image blocks into columns before performing matrix product with kernels, and used a *cython* `im2col` function.

Training speed-up A recent technique called batch normalization [8] has been developed so as to accelerate training, which has proved to be efficient in our case. We also found that the *RMSprop* optimizer [14] performs best in our case.

Preventing overfitting Overfitting is a serious issue with CNN since the number of parameters is usually large. This issue can be tackled by stopping training early but knowing when stopping is difficult, or with dropout [12]. Yet we chose the simpler L2 regularization which has the appealing property of encouraging the network to use all of its inputs a little rather that some of its inputs a lot.

2.3 Unsupervised feature learning

A typical approach taken in [5] is to use an unsupervised learning algorithm to train a model of the unlabeled data and then use the results to extract interesting features from the data. The idea is to learn the most represented features in the set of images, by clustering for instance. The unsupervised feature learning algorithm may be a sparse auto-encoder, a sparse RBM, a Gaussian mixture, or simply a K-means algorithm.
First, one randomly crops patches of size $p \times p \times 3$ from images of both X_{train} and X_{test}. These patches are standardized and whitened (to remove the correlation between nearby pixels). Then one applies an unsupervised learning algorithm on the patches. We used K-means since it is rather simple compared other feature learning algorithms and was shown to achieve better performances. One obtains a set of k centroids, denoted c^k. One designs an *activation* function $f : \mathbb{R}^N \to \mathbb{R}^k$ (with $N = 32 \times 32 \times 3$). We used $f_k(x) = \max(0, \mu(z) - z_k)$, where x is a patch, $z_k = \|x - c^k\|_2$, and $\mu(z)$ is the mean of the elements of z. This function is referred to as *K-means triangle*.
For each image in X_{train} and X_{test} one extracts every patches of size $p \times p \times 3$ with a step between each patch (or stride) w. One then applies the activation function on each of these patches. The number of features is reduced by pooling (such as average or max operations) over the four quadrants of the image to get $4k$ features. These features are the ones used for classification.
Such model requires a lot of tuning as the final score usually depends more on the features than on the learning algorithm [4]. Typically, the size of the patches p and the number of centroids k can greatly affect the performance of the learning. For the given images, $p = 6, 8$ is a good value. The performance are better for a stride w as small as possible ($w = 1$), however one can increase this value for computational power requirements with little effects on the score (minus 2-3%). The number of centroids k must be set as large as compute resources allow. This parameter has a very significant influence on the results. The larger k gets, the more patches one needs to extract. In our case, the performance asymptotes for 400000 patches and $k = 4000$. As to the K-means algorithm, 10 iterations are usually enough.

3 Classifier

SVMs [2] have became a standard tool in the classification toolbox for several main reasons: theoretically, they can learn any training set perfectly with the right choice of kernel; the loss to be minimized is convex; the maximum margin criterion gives raise to sparse solutions; once the kernel has been chosen, the only parameter to be tuned is the maximum cost of misclassified samples.

Implementation First, we implemented SVM by solving the dual problem which is a quadratic problem with the help of `cvxopt` [6]. This approach is not computationally competitive and does not lead to sparse solution. That is why we then followed the same approach as `LibSVM` [3], which implements Sequential Minimal Optimization (SMO). As detailed in [1], the idea is to proceed by updating only two coefficients at each iteration and leaving the others unchanged.

We observed in practice, that as suggested, a significant speed-up can be obtained by caching last kernel computations (which we implemented using an ordered dictionary with limited size). We achieve the same performance as `LibSVM` with only a x5 slow-down.

Choice of kernel The best choice of kernel depends on the distribution and the structure of the data. With raw grey-scale images, a quadratic kernel is competitive. For features obtained from the CNN, a radial basic function kernel works best since features are not high-dimensional. Finally, with features extracted from the K-means method, a linear kernel achieves the best score.

4 Conclusion

By using features learned from the CNN along with a Gaussian-kernel SVM we reach a score of $\sim 52\%$. With features extracted from centroids learned on both X_{train} and X_{test}, along with a linear SVM, we attain our best score of $\sim 62\%$ which placed us equal eighth in the challenge leaderboard.

References

[1] Lon Bottou and Chih jen Lin. Support vector machine solvers, 2006.

[2] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, June 1998. ISSN 1384-5810. doi: 10.1023/A:1009715923555.

[3] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199.

[4] A. Coates, H. Lee, and A.Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudk, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR Workshop and Conference Proceedings*, pages 215–223. JMLR W&CP, 2011.

[5] Adam Coates and Andrew Y. Ng. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 561–580. 2012. doi: 10.1007/978-3-642-35289-8_30.

[6] Joachim Dahl and Lieven Vandenberghe. CVXOPT: A python package for convex optimization, 2008.

[7] Fu Jie Huang and Yann LeCun. Large-scale learning with svm and convolutional for generic object categorization. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 284–291, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0. doi: 10.1109/CVPR.2006.164.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456, 2015.

[9] Trevor Darrell Kristen Grauman. The pyramid match kernel: discriminative classification with sets of image features. pages 1–8, Cambridge, MA, USA, 2005. IEEE.

[10] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[11] David Lowe. Distinctive image features from scale-invariant keypoints. 2004.

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435.

[13] Yichuan Tang. Deep learning using support vector machines. *CoRR*, abs/1306.0239, 2013.

[14] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.