Instance classification with prototype selection

Josip Krapac, Florent Perronnin, Teddy Furon, Hervé Jégou

Instance classification with prototype selection

Josip Krapac* Florent Perronnin† Teddy Furon* Hervé Jégou*

*INRIA †Xerox Research Center Europe

name.surname@[inria.fr|xrce.xerox.com]

ABSTRACT

We address the problem of instance classification: our goal is to annotate images with tags corresponding to objects classes which exhibit small intra-class variations such as logos, products or landmarks. We propose a novel algorithm for the selection of class-specific *prototypes* which are used in a voting-based classification scheme. We show significant improvements over two state-of-the-art methods, namely the Fisher vector and Hamming Embedding, on two challenging methods of logos and vehicles.

Keywords

Computer vision, image classification, feature selection.

1. INTRODUCTION

This paper deals with the task of image classification: given a query image, the goal is to predict the presence or absence of an object within a pre-determined set. We focus on *instance classification*, *i.e.* we consider objects with small intra-class variations such as logos, consumer products or landmarks. The applications are numerous, *e.g.* measuring brand exposure (logos), ensuring shelf compliance in the retail industry (products) or tagging consumer photos (landmarks). This problem is at the crossroad of two well-studied problems in multimedia retrieval: *object recognition* and *instance-level image retrieval*. As in the case of instance-level retrieval, we assume that the intra-class variability is mostly limited to extraneous factors such as viewpoint or lighting. However, as in object recognition, we assume that a small number of images is annotated with object labels to learn the object models.

We show that this instance classification is still not solved by current approaches to object recognition and instance-level search, and we propose a new method that outperforms the state-of-the-art. Our main contribution is a novel algorithm which automatically selects class-specific features which we refer to as *prototypes* (see Fig. 1). For each prototype we also learn a distance threshold, such that features below this threshold with high probability describe the same object as the prototype. In the query phase the prototypes are used in a voting-based mechanism to compute per-class scores. Our method is simple, efficient and flexible: we obtain very good performance without the use of geometric layout information, and we scale well with the addition of new training examples and object classes.

The paper is organized as follows. Section 2 reviews the state-of-the-art in object recognition and instance-level image retrieval. Section 3 describes the proposed approach for instance classification based on prototype selection. Section 4 finally shows that our method outperforms popular state-of-the-art approaches for object classification and instance retrieval on two datasets, namely the public FlickrLogos32 dataset [13] and a new dataset of vehicle images.

2. RELATED WORK

Since its introduction, the bag-of-words (BoW) image representation has been widely used for object recognition [4] and image retrieval [14]. The first step in the BoW pipeline is to represent an image with a set of image patches. The patches are selected either using interest point detectors [10] or sampled densely to cover the whole image. Each patch is then described by a *local descriptor* that captures the appearance of a patch, *e.g.*, a SIFT vector [8]. The distance in the space of local descriptors reflects visual similarity between patches. The descriptor space is partitioned using a quantizer (*e.g.* k-means) learned from a subset of local descriptors. Local descriptors are coded using the learned *visual dictionary*: they are mapped to the visual word index, *i.e.* the quantization cell they belong to. In the BoW, patches are compared via their codes: they are deemed similar when they are assigned to the same visual word. Extensions such as the Fisher Vector (FV) [11] use coarser vocabularies but encode more information per cell than just an index, leading to state-of-the-art results in classification [12, 3] and retrieval [6, 7].

Two main strategies have been adopted to deal efficiently with the fact that the image is represented as a collection of patches: *aggregation* and *inverted lists*. In aggregation the patch codes of the image are pooled into a vector representation, such as a histogram of the visual word indices assigned to image patches [4]. The image is thus embedded into a vector space and the similarity between images is computed by evaluating a *kernel function*, such as the cosine similarity. Such aggregated representations are often used for image classification to learn a parametric model of a class using, for instance, a support vector machine (SVM).

This work was done in the context of the FIRE-ID ANR Project (ANR-12-CORD-016). Preprint version published on February 5th 2014.

Figure 1: Prototypes with high patchAP. *Left*: images from the *FlickrLogos32* dataset. *Right*: images from the *Vehicles29* dataset. The selected prototypes reveal discriminative parts of the objects.

Inverted lists [14, 6] on the other hand do not collapse the descriptor codes, but instead use them for indexing. This allows more precise matching, which is required for applications such as instance-based image retrieval.

Boiman *et al.* [2] argue that the main drawbacks of the BoW methods for object recognition are the quantization and the image-to-image distances. They showed good classification performance without quantizing descriptors and using an image-to-class distance. Several extensions of this model have been proposed, targeting class inbalance [1] and efficiency [9]. Our work is related to these Nearest Neighbor (NN) classifiers. We also do not quantize descriptors. However, instead of using all training set descriptors, we identify and use only a subset that allows reliable class prediction. Different from NN classifiers which use the distance to the class descriptor as a vote, the votes for the class are based on estimated class posteriors in neighborhood of the prototypes. The main advantage of our method *wrt.* both NN classifiers and BoW methods is that the score for the image is influenced only by the patches that are close to prototypes, therefore achieving robustness to varying object backgrounds.

3. PROPOSED ALGORITHM

This section describes our method to select prototypes and class scoring based on votes given to prototypes.

Notations. Suppose the training set contains M images of C classes. The i-th image of the training set associated to class label y_i is represented by its set of local features $\mathcal{X}_i = \{\boldsymbol{x}_i^1, \cdots \boldsymbol{x}_i^{N_i}\}$, where N_i denotes the number of the extracted patches. Given a class y, we form its positive set by taking all patches from the M_y associated training images: $\mathcal{X}_y = \cup_{y_i=y} \mathcal{X}_i$. The patches from the other training images form the negative set of class y. Note that this is weakly supervised because we do not require any bounding box around objects. Thus, the positive set usually contains also some features from the background.

Prototype selection. The prototypes for a given class y are selected from $\mathcal{X}_y$. We require that their neighborhood contains a high proportion of descriptors coming from different positive images, while the descriptors of the negative set are far away. To quantify this property we propose a new measure which we call *patchAP*. We use distances between the prototype candidate and train images to rank the images and we define patchAP as average precision of this ranking (*c.f.* left part of the Fig. 2). This corresponds to an area under the PR curve that displays image precision against image recall.

Formally, for a given feature $\boldsymbol{x}_i^\ell$ of the i-th image, we first compute the distance d_j to all images $j \neq i$: $d_j = \min_{\boldsymbol{x} \in \mathcal{X}_j} \|\boldsymbol{x} - \boldsymbol{x}_i^\ell\|_2$. These $M-1$ images are sorted by increasing distances: $j_1, \ldots, j_{M-1}$. We define the class indicator function $I(k) = 1$ if $y_{j_k} = y_i$, and else 0, and the cumulative sum $J(r) = \sum_{k=1}^{r} I(k)$. The patchAP is:

$$\text{patchAP} = \frac{1}{M_{y_i} - 1} \sum_{k=1}^{M-1} \frac{1}{k} I(k) J(k). \qquad (1)$$

A patch is selected as a prototype of the class if its patchAP is larger than a predefined threshold t_{pAP}. The distribution of patchAP values varies across classes, so different classes are represented by a different number of prototypes.

If the local neighborhood of a feature contains a large proportion of patches coming from different positive images the patchAP of the feature is high. If the patches come from a small number of positive images, patchAP is small, since we consider only one patch per image. This way, we enforce that prototypes are both *class-specific* and that they occur in *many different positive images*, *i.e.* that they generalize well across images. As shown in Fig. 1, the selected prototypes are mostly located on the objects. This, as a side-product of our approach, enables to discover objects' discriminative parts and their locations *automatically*.

Radius selection. For a given prototype, we determine a distance threshold that is a radius of a prototype-centered ball. We require that the precision, *i.e.* the proportion of the positive patches within the ball, is higher than a threshold. Instead of just one radius we have a pre-determined set of ball precisions which yield a set of radii: the balls corresponding to lower precisions are larger. Notice that each prototype ends with its own set of radii. The precisions range from t_{bp} to 1, where t_{bp} denotes the minimum required ball precision.

This is a per-prototype set of radii: prototypes in different parts of the feature space have a different ball radius for a given precision depending on the local density of the class (*c.f.* Fig. 3). This is different from NN classifiers based on distances that do not change across feature space.

Class scoring. Given a set of class prototypes equipped with their own set of radii, we compute the class scores for a query image as follows. Each patch extracted from the query image casts a vote. We determine into which balls the patch falls by comparing the distances to the prototypes with their radii. We select only the smallest ball for each prototype. Out of these, the patch votes only for the ball with highest precision. If there are several such balls, the patch vote is divided between them equally (*c.f.* right part of the Fig. 2).

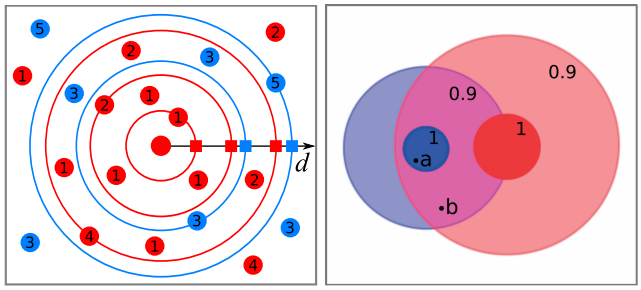

Figure 2: Illustration of the method. *Left*: Prototype selection. The patches from the positive set are red, the ones from the negative set blue. The numbers denote the images they are extracted from, the circles and the squares the distances to the closest patches from different images. The patchAP corresponds to average precision of the ranking of the squares. The patchAP is equal to 1 when all red squares are ranked before the blue ones. *Right*: Ball assignment. Two patches (a,b) from a query image, scoring two prototypes (red, blue). The precision of the ball is denoted up left. Although both patches fall into two balls, patch a is assigned only to the ball with higher precision (dark blue), while the contribution of the patch b is divided between two balls having equal precision. *Best viewed in color.*

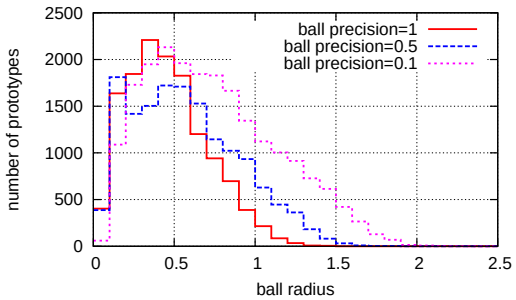

Figure 3: Histogram of ball sizes for 3 precisions. Balls associated to the same precision vary in size considerably. Balls with the same radius contain different proportions of patches of the same class.

The ball vote is determined as the sum of all given patch votes. Next, each ball votes for its prototype with a weight that equals its associated precision. Finally, the score for a class is detemined as the sum of the class prototypes' votes.

A scalable implementation. Since the number of patches in the training set is large ($\approx 10^8$ on FlickrLogos32), the patchAP of a given feature is computed only from the k nearest neighbor patches. The Product Quantization scheme (PQ) provides a fast approximate nearest neighbors search [7]. We used it to shortlist k closest points from the training set and re-rank them by computing the exact distances. The ball radii corresponding to the precision thresholds are computed in the same k-neighborhood.

PQ is also used in the test phase, to compute the class scores for the query image: the prototypes for all classes selected in the training phase compose the database. Then, for each patch of a query image, PQ finds the k nearest prototypes, and we compute the exact distances from these shortlisted prototypes, which we use to obtain class scores.

Notice that the scores for *all classes* are derived at a *constant time cost* of querying PQ and computing the shortlisted distances. This setup scales well with addition of new labeled data: adding classes or training images amounts to adding the newly computed prototypes into the database.

Parameter selection. The method has two parameters: the threshold on patchAP t_{pAP} and the minimum ball precision t_{bp}. The former determines the number of selected prototypes, while the t_{bp} determines the number of balls per prototype. We perform a grid search and select the parameters t_{pAP} and t_{bp} that yield the best accuracy on the training data.

4. EXPERIMENTS

We first describe the datasets, the features and the experimental setup. We then provide a comparative evaluation of the proposed approach based on the mean average precision (mAP), which is commonly defined as the mean of the per-class average precisions.

Datasets. The *FlickrLogos32* dataset [13] contains 8,240 images of 32 logo classes. The dataset is divided into three sets. The first set contains 10 images per class of close-up logos with limited background. The second and third sets both contain 30 images of logos per class and 3,000 distractors (*i.e.*, images that do not contain any logos) of various sizes displayed against various backgrounds. We use the first two sets to extract the prototypes and for parameter tuning, and the remaining set to test the method. So far, this dataset has been tested only for image retrieval, but we believe it is also very relevant for instance classification.

We also introduce the *Vehicles29* dataset which we collected by querying 29 car models of 7 brands to a second-hand ads website. Each manufacturer is represented by 3 to 6 models. A class may contain multiple submodels corresponding to the years of production. In total, there are 10,622 images, divided into 5,266 training and 5,356 test images. On average, there are 150 training and testing images per car model. The dataset is very challenging as some cars are very similar and differ only in small details like front mask, door handles, lights or position of wiper.

Local features are SIFT descriptors [8] extracted from points detected by the the rotation-invariant Harris-Hessian-Laplace interest point detector [10]. Following prior work [5], the descriptors are ℓ_2-normalized to the unit ball and square-rooted as these normalization steps were shown to improve matching performance.

Experimental setup. For each candidate prototype, the patchAP is computed from a k-neighborhood of size $20,000$. The radii for these prototypes are computed for a set of 20 precisions in range $[0.05, 1]$ with a step size of 0.05. The scores for the query image are computed from $10,000$ nearest prototypes for each detected patch.

Justification of the proposed approach. Fig. 3 shows that the ball precision (the proportion of the class patches inside the ball) varies considerably for a given radius. Patch votes should reflect class membership whose degree of confidence we capture by the ball precision. This justifies our approach: determine offline the radii at which the required class precision is attained, so that, online, patches falling into such balls cast a vote directly linked to the degree of confidence. This is better than determining the patch vote based on its distance to the class prototype.

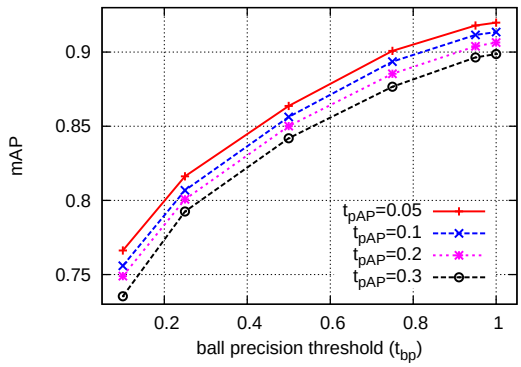

Figure 4: Influence of the patchAP threshold t_{pAP} and the ball precision threshold t_{bp} on the accuracy on FlickrLogos32. The behavior is similar for Vehicles29. Low t_{pAP} and high t_{bp}, which correspond to more prototypes and fewer high precision balls per prototype, yield better performance.

Parameter analysis. Fig. 4 shows the influence of the parameters on the performance on the FlickrLogos32 dataset. The behavior is similar on Vehicles29 dataset, but at the lower range of patchAP thresholds. Logos are, by design, discriminative and unique and therefore match well over many training images, resulting in prototypes with high patchAP. On the other hand, car classes can contain visually different vehicles, so patches may be matched across a smaller number of training images, which means that selected prototypes have lower patchAP values. Lower t_{pAP}, corresponding to larger number of prototypes, and high t_{bp} corresponding to small number of high precision balls per prototype, gives good performance on both datasets.

Qualitative analysis. Our method succeeds in selecting discriminative parts of the objects. Although the prototypes are extracted in a weakly-supervised manner, *i.e.*, without any bounding boxes, the vast majority of them lies on the object, as can be seen from Fig. 1. Notice that the supervised setting (*e.g.*, bounding boxes) would not help in the case of cars since only a subset of patches detected on the object is important to discriminate the classes.

Performance comparison. In the second set of experiments presented in Table 1, we compare the proposed method to baselines. The proposed method is most related to NBNN [2], as we also do not quantize features. However, we use aggregate class posteriors rather than distance to obtain class scores and only a selected subset of all training features contributes to the class score. This significantly influences the performance. We compare also to the Fisher Vector (FV) [12] which is the state-of-the-art for patch-based object recognition [3]. We experimented with two settings: a "standard" setting where, following [12], we used a 256-component mixture-of-Gaussians and a "costly" setting where we used 4,096 components (thus leading to 524,288-dim FVs). A linear one-*vs*-all support vector machine (SVM) is learned for each class. For logo classification, we also compare to Hamming Embedding (HE), the state-of-the-art for image retrieval. In this case, we use a k-NN classifier with k=40. For each test image, we get HE scores for logo images in the train set using a Hamming threshold $h_t = 20$. All compared methods use all query image features to obtain class scores. The proposed method computes the similarities between query features and class prototypes. Therefore only the query features in the neighborhood of the class' prototypes influence the class score.

Method	classification mAP	
	FlickrLogos32	Vehicles29
NBNN	5.25	5.36
HE + kNN (k=40)	76.9	-
Fisher Vector (256)	80.5	44.6
Fisher Vector (4,096)	86.6	49.7
Prototype voting	**91.4**	**55.7**

Table 1: Comparison to the state-of-the-art. Using precision rather than distance, and selecting prototypes improves results significantly *wrt.* NBNN. k-NN classifier using Hamming Embedding (HE) scores performs comparably to FVs with 256 Gaussian components. Increasing the number of Gaussians improves performance, but with diminishing returns.

5. CONCLUSION

This paper introduced a method for visual instance classification based on the automatic selection of class representatives – *class prototypes*. The method performs favorably to state-of-the-art approaches in object classification and image retrieval, as demonstrated experimentally on two challenging datasets of logos and vehicles. Our method selects the prototypes independently, so they are redundant. In addition, all prototypes are weighted equally. Learning their relative importance may improve the classification performance. To this end, the combination with sparse linear models, *e.g.* [15], is a promising research direction.

Acknowledgments. This work was done in the context of the FIRE-ID ANR Project (ANR- 12-CORD-016).

6. REFERENCES

[1] R. Behmo *et al.* Towards optimal naive Bayes nearest neighbor. In *ECCV*, 2010.
[2] O. Boiman *et al.* In defense of nearest-neighbor based image classification. In *CVPR*, 2008.
[3] K. Chatfield *et al.* The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.
[4] G. Csurka *et al.* Visual categorization with bags of keypoints. In *ECCV SLCV*, 2004.
[5] M. Jain *et al.* Hamming embedding similarity-based image classification. In *ICMR*, 2012.
[6] H. Jégou *et al.* Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.
[7] H. Jégou *et al.* Aggregating local images descriptors into compact codes. *IEEE TPAMI*, 2012.
[8] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.
[9] S. McCann and D. Lowe. Local naive Bayes nearest neighbor for image classification. In *CVPR*, 2012.
[10] K. Mikolajczyk *et al.* A comparison of affine region detectors. *IJCV*, 2005.
[11] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.
[12] F. Perronnin *et al.* Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010.
[13] S. Romberg *et al.* Scalable logo recognition in real-world images. In *ICMR*, 2011.
[14] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.
[15] Z. Wang *et al.* Learning class-to-image distance via large margin and L1-norm regularization. In *ECCV*, 2012.