Attacking image classification based on Bag-of-Visual-Words

A. Melloni #1, P. Bestagini #2, A. Costanzo *†3, M. Barni *†4, M. Tagliasacchi #5, S. Tubaro #6

# *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano*
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{1 amelloni/2 bestagini/5 tagliasa/6 tubaro}@elet.polimi.it

* *Department of Information Engineering and Mathematical Sciences,*
Università degli Studi di Siena, Via Roma 56, 53100 Siena, Italy
3 andreacos82@gmail.com, 4 barni@dii.unisi.it

† *National Inter-University Consortium for Telecommunications, Via di Santa Marta 3, 50139 Firenze, Italy*

***Abstract*—Nowadays, with the widespread diffusion of online image databases, the possibility of easily searching, browsing and filtering image content is more than an urge. Typically, this operation is made possible thanks to the use of tags, i.e., textual representations of semantic concepts associated to the images. The tagging process is either performed by users, who manually label the images, or by automatic image classifiers, so as to reach a broader coverage. Typically, these methods rely on the extraction of local descriptors (e.g., SIFT, SURF, HOG, etc.), the construction of a suitable feature-based representation (e.g., bag-of-visual words), and the use of supervised classifiers (e.g., SVM). In this paper, we show that such a classification procedure can be attacked by a malicious user, who might be interested in altering the tags automatically suggested by the classifier. This might be used, for example, by an attacker who is willing to avoid the automatic detection of improper material in a parental control system. More specifically, we show that it is possible to modify an image in order to have it associated to the wrong class, without perceptually affecting the image visual quality. The proposed method is validated against a well known image dataset, and results prove to be promising, highlighting the need to jointly study the problem from the standpoint of both the analyst and the attacker.**

I. Introduction

The widespread diffusion of multimedia content has determined the urgent need of easy and user-friendly search, browsing and filtering systems. To this end, image databases exploit the availability of tagged pictures to enable concept-based image retrieval. This means that sets of tags (i.e., textual representations of semantic concepts) are associated to each picture to describe its content. However, manual tagging is not a feasible solution as the database size increases. To address this issue, content-based image classification methods can be used instead.

Indeed, computer vision methods can automatically associate one or more tags to an image in order to describe its semantic content. For example, one might want to discriminate between adults-only pictures and family-safe pictures in order to filter downloadable content from different websites. This sort of classification is easily performed by humans, since we are trained to recognize objects/scenes in the real world. However, the automation of this procedure is not a trivial task, since it is not easy to train a classifier that is able to generalize and produce reliable results in challenging situations (e.g., complex illumination, severe perspective distortion, etc.).

Nowadays, state-of-the-art methods for image-based scene classification rely on the Bag of Visual Words (BoVW) representation [1]. That is, the image content is described by means of a set of visual descriptors (e.g., SIFT [2], SURF [3], HOG [4], etc.) extracted from the pixel-domain representation. These descriptors are then compared to those stored in a dictionary (previously trained) and assigned to one (or more, in the case of soft-assignment [5]) visual words, so as to map each image to a fixed-dimensional feature vector. Finally, a classifier is designed by means of supervised learning to identify the region of the vector space corresponding to images to be labelled with each tag of interest.

Such classification schemes are often considered as black-boxes, and little is known about which parts of an image actually contribute to the outcome of the classifier. Interestingly, it was recently shown that is it possible to reverse engineer the internal mechanisms of these classifiers [6], revealing the so-called *support regions*, i.e., sets of contiguous pixels that determine the outcome of the classifier. Inspired by this work, in this paper we show how to effectively attack a picture in order to fool a (binary) image classifier, yet introducing little distortion. On the one hand, this attack can be pursued by a malicious user, who wants to conceal the semantic content of some images, e.g., to avoid being detected by safe-search schemes. On the other hand, from the perspective of the forensic analyst, it provides interesting insights on the design of future attacker-aware image classification schemes, which are meant to guarantee security when facing this sort of attacks.

More specifically, we show that by knowing the scheme

WIFS'2013, November 18-21, 2013, Guangzhou, China.
ISBN 978-1-4673-5593-3 ©2013 IEEE.

used by the classifier, it is possible for an attacker to tamper with an image in such a way that the classifier associates it to the wrong class. The key tenet is to alter the image in the pixel-domain so as to modify the BoVW representation used as input to the classifier. We consider two kinds of attack, depending on how local features are extracted from an image. In the case of *dense sampling*, local features are obtained by analyzing image patches regularly arranged according to a fixed grid. Therefore, we perform our attack by modifying the image locally, so that the descriptor corresponding to a given patch is modified accordingly. Instead, in the case of *sparse sampling*, local features are obtained by processing the image with a feature detector, which identifies salient keypoints (e.g., corner-like or blob-like structures), and descriptors are extracted from patches around these keypoints. Therefore, we attack the detector, following a strategy previously proposed in [7], [8] to fool copy-move detectors [9], by removing them in a convenient way to pilot the classification outcome.

Although the problem of attacking image classification systems based on local features was previously presented in the image forensic literature [10][11], to the best of the authors' knowledge this is the first time that is applied to a classifier relying on a BoVW representation.

The rest of the paper is structured as follows. Section II presents the typical scheme of an image classifier that relies on a BoVW representation. Section III shows the rationale behind the proposed attack, in the case of both dense and sparse feature sampling. Section IV reports the results obtained using our algorithm on a wide image dataset. Finally, in Section V we draw some conclusive remarks, and present possible future works.

II. Image classification

The BoVW representation has been successfully adopted to enable fast indexing and retrieval of large image collections [1] as well as content-based image classification [12]. In this paper we focus on the latter use, which has been pursued in the literature according to different implementations. In this section we present the typical scheme used for image classification based on BoVWs. This is not meant to be a comprehensive survey of content-based classification methods, but serves as background to introduce the problem we deal with, and to define the classifier targeted by our attack. Although other, more complex, methods have been developed in the literature, we decided to focus on a simple scheme for the sake of clarity. Indeed, since more complex classification schemes are usually derived from this one, it is straightforward to adapt the attack proposed in this paper to other classifiers.

The classification procedure consists of three steps: i) build the dictionary of visual words (VWs); ii) define a code (i.e., a fixed-dimensional feature vector) that describes each image in terms of visual words; iii) train a classifier by means of supervised learning. In the following we briefly illustrate these steps.

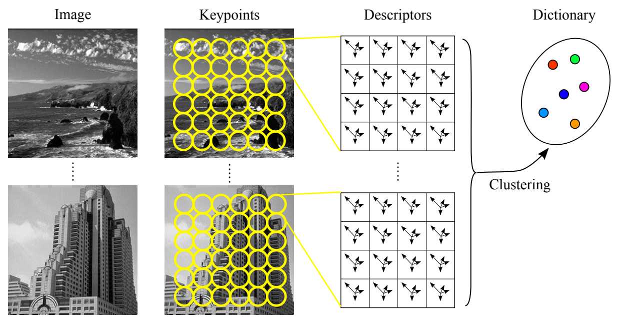

Fig. 1: Steps performed to build the dictionary of visual words.

A. Dictionary of VWs

The first step requires the computation of the dictionary of VWs, as illustrated in Figure 1. A training set of I images belonging to different classes is analyzed. For each image, a set of local descriptors (e.g., SIFT, SURF, HOG, etc.) is computed obtaining K-dimensional column-vectors $\mathbf{x}_i^n$, where $i = 1, \dots, I$, is the image index, and $n = 1, \dots, N_i$, is the descriptor index within the i-th image. In our work, we consider two commonly adopted sampling strategies. In the case of dense sampling, local descriptors are computed from (possibly overlapping) blocks equally spaced on a predefined grid. Alternatively, in the case of sparse sampling, the image is analyzed by means of a detector, which identifies a set of content-dependent salient keypoints corresponding, e.g., to corner-like or blob-like structures, possibly at different scales. Then, local descriptors are computed from image patches centered around these keypoints.

Regardless of the adopted feature sampling strategy, the descriptors are then clustered, e.g., using hierarchical k-means, into W clusters. The number of clusters W is equal to the number of visual words. The cluster centers $\overline{\mathbf{x}}^w$, $w = 1 \dots, W$, define the dictionary of VWs. Each visual word is therefore defined by the average descriptor of each cluster.

B. Code construction

Once the dictionary is trained, we need a way to describe each image according to the VWs of the dictionary. This step is performed building for each image a code vector $\mathbf{d}_i \in \mathbb{R}^W$ whose elements define the degree of similarity between the image and each visual word.

Let $\overline{\mathbf{X}} = [\overline{\mathbf{x}}^1, \overline{\mathbf{x}}^2, ..., \overline{\mathbf{x}}^W] \in \mathbb{R}^{K \times W}$ denote the matrix composed by the visual words in the dictionary. In the simplest BoVW representation, each descriptor $\mathbf{x}_i^n$ is assigned to the closest visual word, and the corresponding entry in the vector $\mathbf{d}_i$ is increased by one. Therefore, $\mathbf{d}_i$ represents a histogram, which counts the number of descriptors assigned to each visual word. In the literature, it was shown that soft-assignment of descriptors to visual words is often preferable [5] to hard-assignment. In this paper, we consider the soft-assignment method proposed in [13], which is also known as Locality-constrained Linear Coding (LLC). More specifically, for each image we compute the matrix $\mathbf{C}_i = [\mathbf{c}_i^1, \mathbf{c}_i^2, ..., \mathbf{c}_i^{N_i}] \in \mathbb{R}^{W \times N_i}$, in which each column vector $\mathbf{c}_i^n \in \mathbb{R}^W$ is the code associated to the n-th descriptor of the i-th image. The matrix $\mathbf{C}_i$ is obtained by solving the following constrained least squares problem:

$$\begin{aligned} \mathbf{C}_i = & \underset{[\mathbf{c}^1,\mathbf{c}^2,\ldots,\mathbf{c}^{N_i}]}{\operatorname{argmin}} \sum_{n=1}^{N_i} \|\mathbf{x}_i^n - \overline{\mathbf{X}}\mathbf{c}^n\| \\ & s.t.\ \mathbf{1}^\top \mathbf{c}^n = 1,\ n = 1, \ldots, N_i. \end{aligned} \tag{1}$$

Therefore, the W elements of $\mathbf{c}_i^n$ represent the degree of similarity between the descriptor $\mathbf{x}_i^n$ and each visual word.

However, the size of $\mathbf{C}_i$ depends on the number of descriptors in the i-th image. For this reason, an additional operation is applied to obtain a code whose size is not image-dependent. To this end, one of the most used operations is max-pooling, which consists in building a fixed-dimensional column vector $\mathbf{d}_i = [d_i^1, d_i^2, \ldots, d_i^W]^T$ as follows

$$\mathbf{d}_i = \max(\mathbf{C}_i), \tag{2}$$

where the $\max(\cdot)$ operator is applied row-wise, so that the w-th element of $\mathbf{d}_i$ is the maximum value of the w-th row of $\mathbf{C}_i$. Then, the vector $\mathbf{d}_i$ is normalized to have unit ℓ^2-norm.

C. *Classifier*

Support Vector Machines (SVM) have been successfully employed in content-based image classification. In the literature, it was shown that using a non-linear code (such as the one presented above) with a linear classifier provides better results than using a linear code with a non-linear classifier [13]. In this paper we consider a binary classifier, which is trained to distinguish images that belong to class $\mathtt{A}$ from the others (i.e., class $\overline{\mathtt{A}}$). A binary linear-SVM takes as input the set of training pairs $\langle \mathbf{d}_i, y_i \rangle$, where $\mathbf{d}_i \in \mathbb{R}^W$ is the BoVW representation of the i-th image, and $y_i \in \{\mathtt{A}, \overline{\mathtt{A}}\}$ is the class the image belongs to. During training, the hyperplane in $\mathbb{R}^W$ that maximizes the margin between the two classes is sought. Such hyperplane can be defined by means of its normal versor $\mathbf{u} \in \mathbb{R}^W$ and a bias term b that measures the signed-distance of the hyperplane from the origin.

The hyperplane serves as decision boundary between the two classes, splitting the W-dimensional space in two parts (see Figure 2). That is, during the classification step, points lying on one side of the hyperplane are associated to class $\mathtt{A}$, whereas points lying on the other side are associated to class $\overline{\mathtt{A}}$. Equivalently, the classification decision is taken based on the sign of the following value

$$\delta_i = \mathbf{d}_i \cdot \mathbf{u} - b, \tag{3}$$

which represents the signed-distance between $\mathbf{d}_i$ and the hyperplane. If $\delta_i > 0$ the i-th image is associated to class $\mathtt{A}$, otherwise it is associated to class $\overline{\mathtt{A}}$.

III. Classification attack

The goal of this work is to demonstrate that is possible to attack a classifier based on the scheme presented in Section II. We consider an attack that aims at reverting the classification label determined by the classifier. That is, when an image belongs to class $\mathtt{A}$, it is attacked so that the outcome of the classifier is class $\overline{\mathtt{A}}$.

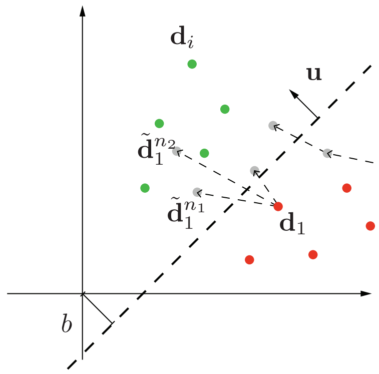

Fig. 2: Representation of a W-dimensional feature space, when $W = 2$ for illustration purposes. Red points are assigned to class $\mathtt{A}$, green points to class $\overline{\mathtt{A}}$. The dashed line described by the versor $\mathbf{u}$ and the bias b represents the decision boundary that separates the two classes. By attacking the image in the pixel domain, it is possible to move the BoVW representation from one side of the decision boundary to the other.

Since the classifier receives as input the BoVW representation $\mathbf{d}_i$, the attacker modifies the image in order to associate a different feature vector $\tilde{\mathbf{d}}_i$ to it. Thus, to modify $\mathbf{d}_i$ into $\tilde{\mathbf{d}}_i$, the attacker operates on those blocks from which descriptors are computed. In the following we consider two different attacks, depending on the specific feature sampling strategy adopted: dense or sparse.

A. *Attack to classifiers based on <u>dense</u> feature sampling*

When a classification scheme based on dense feature sampling is considered, the attacker operates on blocks defined on a regular grid. The attack consists in substituting a number of selected blocks with visually similar patches. To this end, we need to determine: i) which blocks need to be modified, and; ii) how to modify the selected blocks.

In order to identify the most promising blocks to modify, for every block used in the sampling grid, the attacker substitutes its pixel values with a constant, so that the corresponding visual descriptor is null. Then, the attacker computes:

- $\tilde{\mathbf{d}}_i^n$: the BoVW feature vector computed with max-pooling after replacing the patch with a constant. This is equivalent to removing the n-th column of $\mathbf{C}_i$, and then re-applying max-pooling.
- $\tilde{\delta}_i^n = \tilde{\mathbf{d}}_i^n \cdot \mathbf{u} - b$: the signed-distance between $\tilde{\mathbf{d}}_i^n$ and the SVM hyperplane. If positive, the image is associated to class $\mathtt{A}$, otherwise to class $\overline{\mathtt{A}}$.

Therefore, removing those blocks for which $\tilde{\delta}_i^n < 0$, the image changes class, as illustrated in the example in Figure 2.

When selecting the blocks to modify, the attacker considers them in increasing value of $\tilde{\delta}_i^n$. Indeed, a large negative value of $\tilde{\delta}_i^n$ indicates that the feature vector corresponding to the modified image is moved to the decision region associated to class $\overline{\mathtt{A}}$, and it is far from the decision boundary. For example, Figure 3 shows an image and the corresponding heat-map,

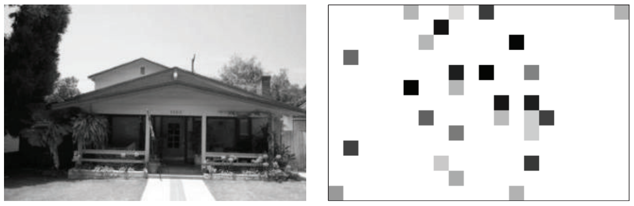
Fig. 3: Example image and corresponding heat-map. The darker the block, the largest the impact on the outcome of the classifier. Dark blocks are the most promising blocks to be substituted by the attacker.

which indicates the value of $\tilde{\delta}_i^n$ for each block in the image. Then, for a selected block, the attacker searches for a patch in a large dictionary of candidate patches. The block is replaced with a new patch satisfying the following conditions:

- *Classification Condition*: The new patch, when used to replace the n-th block, leads to the same BoVW feature vector $\tilde{\mathbf{d}}_i^n$ as the one obtained when setting the block to a constant value. To this end, the vector $\tilde{\mathbf{c}}_i^n$ associated to the new patch (i.e., the column of $\tilde{\mathbf{C}}_i$ associated to the new image) needs to have all the elements smaller than those of $\tilde{\mathbf{d}}_i^n$. This ensures that the new patch does not affect the output of max-pooling.
- *Visual Quality*: Among those patches satisfying the condition above, we are interested in selecting those that are visually similar to the original block. Visual similarity can be measured in terms of objective quality metrics, e.g., Peak Signal to Noise Ratio (PSNR) or Structural SIMilarity (SSIM) index [14] between the two blocks.

In some cases, the attacker needs to replace more than one block to change the class assigned to an image. In this case, the procedure above is iterated in a greedy fashion considering the blocks in increasing order of $\tilde{\delta}_i^n$. For example, Figure 2 shows that it is necessary to replace three blocks to successfully attack image I_2. We will refer to this strategy as *substitution attack*.

An alternative strategy for the attacker consists in re-synthesizing the texture of each candidate block interpolating the values of neighbouring pixels. In order to achieve a result which is visually pleasing, an inpainting algorithm is employed (for example, in our experiments we used [15]). The attacker proceeds iteratively replacing each block with its inpainted version, computes the corresponding BoVW representation $\tilde{\mathbf{d}}_i^n$, and the signed distance $\tilde{\delta}_i^n$ from the decision boundary. If there is at least a block for which $\tilde{\delta}_i^n < 0$ (i.e., the image is misclassified) the attack is completed. Otherwise, the attacker selects the block that leads to the smallest value of $\tilde{\delta}_i^n$, and re-iterates the attack selecting the second block to modify. We will refer to this strategy as *inpainting attack*.

B. Attack to classifiers based on <u>*sparse*</u> *feature sampling*

In the case of sparse feature sampling, the descriptors are computed only for local patches centered around the detected keypoints. First, the attacker determines, for each keypoint, the impact of removing the keypoint and the corresponding descriptor on the outcome of the classifier. More specifically, the values $\tilde{\delta}_i^n$ are computed as illustrated before, where in this case n is the index of the detected keypoint. Then, the removal attack proposed in [8] is applied, incrementally removing the keypoints in increasing value of $\tilde{\delta}_i^n$, until the outcome of the classifier is changed. In a nutshell, the algorithm in [8] identifies different types of keypoint and uses an ad-hoc attack for each type. By doing so, performance is maximized both in terms of removal effectiveness and perceptual quality of the forgery. The type of keypoint is determined based on the analysis of the histogram of grey-levels computed on a squared neighborhood around every keypoint. On the basis of the histogram shape, three different types are defined: unimodal, bimodal and multimodal. For each of them, the most suitable attack is selected (Gaussian Smoothing, Removal with Minimum Distortion [10] or Collage). The procedure is iterated until all the targeted keypoints are removed or a maximum iteration limit is reached. The iterative approach is required for two reasons: i) manipulations can accidentally introduce new keypoints, which are eliminated in the subsequent iterations; and ii) some keypoints may survive to an iteration, thus requiring less forgiving parameter assignments. Notice that the scheme of [8] cannot be employed to attack a classifier based on dense feature sampling, since the detector is bypassed.

IV. Results

In order to test our algorithm, we used the dataset presented in [16], which includes natural images with slightly different sizes in the range of 256×256 pixels[1]. Images are tagged according to 15 different categories (e.g., coast, forest, highway, etc.), each of which composed by more than 200 images. For each category, we selected 50 images to build the dictionary, 50 images to train the SVM, and 100 images for testing. The linear SVM follows the implementation in [17]. Although the proposed approach is general, in our experiments we adopted SIFT visual features, which are computed either on non-overlapping patches (dense sampling) and by resorting to the canonical DoG (Difference-of-Gaussian) detector (sparse). The edge and peak thresholds of the DoG detector were set respectively to 10 and 4, to obtain descriptors that are as close as possible to the original implementation by Lowe [2]. Descriptors were computed using the VLFeat SIFT implementation [18].

With this setup, class $\mathtt{A}$ represents one of the 15 categories, and class $\overline{\mathtt{A}}$ includes images randomly selected from all other categories. Table I summarizes the classification accuracy obtained for all the image categories, when using either one of the two feature sampling schemes. Notice that, in the case of dense sampling, the classifier accuracy varies depending on the class and the block size used.

In the case of attacks to image classifiers based on dense feature sampling, we considered non-overlapping square patches

[1]The dataset is freely available for download at http://www-cvr.ai.uiuc.edu/ponce_grp/data/

TABLE I: Image classification accuracy for each of the 15 image categories. First four rows: dense feature sampling, when using different block sizes; fifth row: sparse feature sampling.

Size	C1	C2	C3	C4	C5	C6	C7
16	0.60	0.84	0.74	0.74	0.67	0.65	0.74
20	0.70	0.86	0.93	0.80	0.82	0.76	0.84
26	0.73	0.89	0.95	0.77	0.82	0.84	0.84
Sparse	0.85	0.72	0.92	0.87	0.76	0.79	0.74

Size	C8	C9	C10	C11	C12	C13	C14	C15
16	0.72	0.67	0.67	0.67	0.61	0.71	0.61	0.61
20	0.66	0.82	0.83	0.73	0.55	0.71	0.71	0.69
26	0.58	0.81	0.85	0.70	0.59	0.75	0.71	0.70
Sparse	0.73	0.71	0.73	0.72	0.61	0.70	0.79	0.79

with side equal to 16, 20, and 26 pixels, centered on a dense grid. Figure 4 (b) - (c) shows an example of attacked images, together with the original version (Figure 4 (a)). Figure 5 shows, for two exemplary image categories (C1 and C5), the percentage of images that are successfully attacked as a function of the number of pixels that need to be changed. In the case of classifiers based on dense feature sampling, we show the impact of changing the patch size and the attack strategy (substitution vs. inpainting). As an example, the inpainting attack needs to modify less than 5% of the pixels of each image in order to misclassify 80% of the images belonging to class C1, when using 20×20 blocks.

Table II reports comprehensive results for a larger number of image categories, when setting the patch size to 16 × 16 and constraining the attacker to change at most 15% of the pixels within each image. For each class, only the images correctly classified before the manipulation were considered. Notice that the attack to the classifier based on dense sampling leads almost always to 100% of misclassified images. Even though the local PSNR and SSIM (i.e., the quality metrics calculated on modified blocks only) might be low, the corresponding values computed on the overall image are very high, denoting that the attacked image is visually similar to the original one.

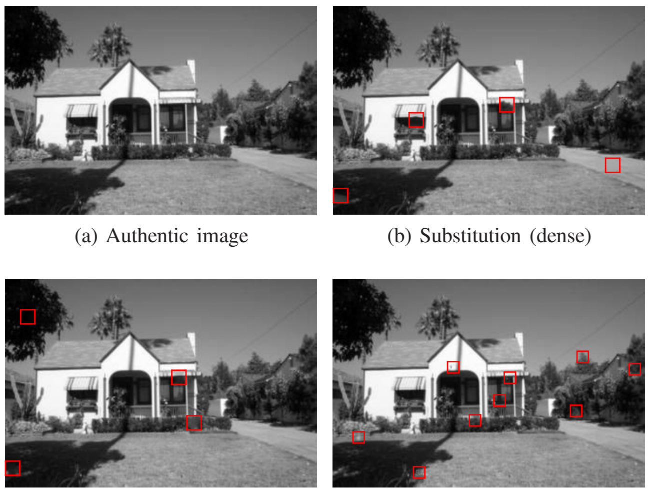
(a) Authentic image
(b) Substitution (dense)
(c) Inpainting (dense)
(d) Keypoint removal (sparse)

Fig. 4: Examples of attacked images. Red squares indicate the blocks affected in order to change the image class.

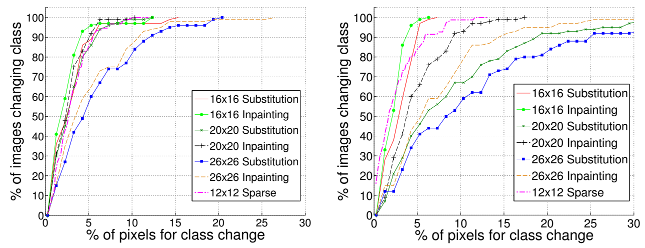

Fig. 5: Percentage of pixels changed in each image to change class. Left: image category C1 (suburb). Right: image category C5 (inside city).

In the case of attacks to image classifiers based on sparse feature sampling, the parameter values of the attack were set as suggested in [8], with the exception of the attack support size, increased to 12. An example of counterfeited image is provided in Figure 4 (d). Figure 5 shows the fraction of successfully attacked images for categories C1 and C5, whereas Table III summarizes the results for a wider set of classes, for which the highest performance was achieved.

The attack was generally successful in hindering the correct classification and such an effectiveness did not come at the expense of the perceptual quality of the altered image. Indeed, local PSNR and SSIM, averaged on all the 12×12 attacked patches, range from 30 dB to 37 dB and from 0.71 to 0.98, respectively.

To better highlight the effect of the attack, in Figure 6 we plot the distance from the separation hyperplane for authentic (blue circle marker) and attacked (red square marker) images. More specifically, we provided two examples: one for the best performing class, i.e. C1, and one for C3, not summarized in Table III above (63.4% misclassification, local PSNR 31.7 dB, local SSIM 0.978).

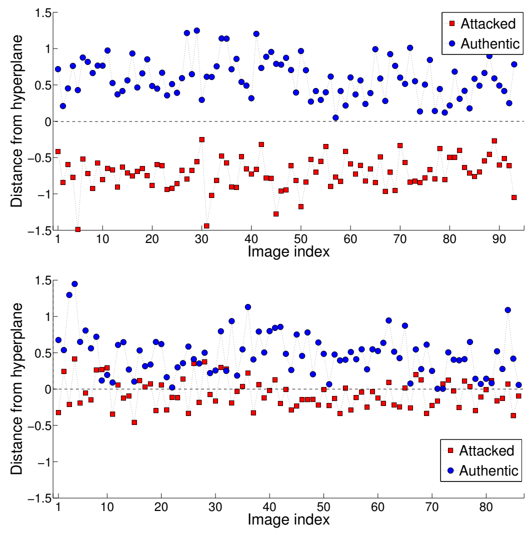

Fig. 6: Distance from the separation hyperplane for authentic and attacked images: class C1 (top) and class C3 (bottom).

In the former case, all examples are clearly separated while in the latter they are more mixed up (although the percentage of misclassification remains satisfactory). The reason behind

TABLE II: Results for attacks to classifiers based on dense feature sampling: true positives (TP) preceding and following the attacks; percentage of misclassification; local and full-frame quality. Top: 16×16 Substitution; bottom: 16×16 Inpainting. Local metrics are computed only on modified blocks, total ones on the overall image.

Class	TP before	TP after	Mis-classified	Local PSNR	Local SSIM	Total PSNR	Total SSIM
C1	70	0	**100 %**	22.4	0.646	40.2	0.987
C2	88	11	**87.5 %**	26.0	0.728	39.0	0.975
C4	70	2	**97.1 %**	28.8	0.808	45.4	0.992
C5	76	0	**100 %**	20.1	0.578	36.8	0.986
C9	84	7	**91.7 %**	21.5	0.592	35.5	0.968
C10	84	0	**100 %**	26.4	0.754	45.5	0.994
C11	66	0	**100 %**	25.2	0.706	43.3	0.990
C12	69	0	**100 %**	20.6	0.529	39.2	0.991
C13	82	0	**100 %**	22.4	0.633	39.3	0.986
C14	79	0	**100 %**	22.3	0.632	39.4	0.987

Class	TP before	TP after	Mis-classified	Local PSNR	Local SSIM	Total PSNR	Total SSIM
C1	70	0	**100 %**	24.0	0.629	42.8	0.995
C2	88	4	**95.5 %**	24.9	0.729	38.9	0.982
C4	70	0	**100 %**	29.0	0.801	46.9	0.995
C5	76	0	**100 %**	22.3	0.609	40.2	0.994
C9	84	4	**95.2 %**	22.9	0.640	37.5	0.985
C10	84	0	**100 %**	28.2	0.780	47.5	0.997
C11	66	0	**100 %**	28.8	0.765	47.5	0.996
C12	69	0	**100 %**	28.5	0.665	48.7	0.997
C13	82	0	**100 %**	26.2	0.702	44.2	0.995
C14	79	0	**100 %**	26.3	0.711	44.2	0.995

TABLE III: Results for attacks to classifiers based on sparse feature sampling: true positives (TP) preceding and following the attack; percentage of misclassification; local and full-frame quality.

Class	TP before	TP after	Mis-classified	Local PSNR	Local SSIM	Total PSNR	Total SSIM
C1	92	0	**100 %**	35.1	0.985	44.5	0.997
C2	44	0	**100 %**	29.7	0.717	51.1	0.998
C4	78	23	**70.5 %**	30.4	0.692	50.8	0.999
C5	76	1	**98.6 %**	35.5	0.981	45.4	0.999
C9	53	3	**94.3 %**	33.8	0.921	46.8	0.998
C10	75	2	**97.3 %**	38.5	0.963	51.7	0.999
C11	63	6	**90.5 %**	37.5	0.945	48.8	0.998
C12	57	4	**93.0 %**	33.4	0.976	46.7	0.998
C13	80	13	**83.7 %**	36.9	0.973	48.9	0.999
C14	77	13	**83.1 %**	36.9	0.982	48.1	0.998

such a behavior is related to the visual content of class C3, depicting forests. Highly textured regions, in fact, generate considerable amounts of sparse keypoints, which, combined to the fact that the employed attack does not always allow total removal (see [8]), have a detrimental effect on performance.

Given these results, we notice that the attack based on dense SIFT representation often allows to change class to a higher number of images. On the other hand, the attack based on sparse SIFT representation generally introduces less distortion.

V. Conclusions

In this paper we demonstrate that it is possible to attack image classifiers based on BoVWs. To this end, we propose a set of algorithms to deal with classification methods that make use of either sparse or dense feature sampling. The algorithms were validated on a common dataset showing high attack success rate. This highlights the need for studying countermeasures that prevent possible malicious attacks to such classifiers. Future works will be devoted to the study of anti-counterfeiting techniques.

Acknowledgment

The project REWIND acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 268478.

References

[1] J. Sivic and A. Zisserman, "Video google: a text retrieval approach to object matching in videos," in *2003 9th IEEE International Conference on Computer Vision (ICCV)*, 2003, pp. 1470–1477 vol.2.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, 2008.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *International Conference on Computer Vision & Pattern Recognition*, 2005.

[5] L. Lingqiao, W. Lei, and L. Xinwang, "In defense of soft-assignment coding," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011.

[6] L. Lingqiao and W. Lei, "What has my classifier learned? visualizing the classification rules of bag-of-feature model by support region detection," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012.

[7] R. Caldelli, I. Amerini, L. Ballan, G. Serra, M. Barni, and A. Costanzo, "On the effectiveness of local warping against sift-based copy-move detection," in *Communications Control and Signal Processing (ISCCSP), 2012 5th International Symposium on*, 2012.

[8] I. Amerini, M. Barni, R. Caldelli, and A. Costanzo, "Counter-forensics of SIFT-based copy-move detection by means of keypoint classification," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, p. 18, 2013.

[9] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "A SIFT-based forensic method for copy-move attack detection and transformation recovery," *IEEE Trans. on Information Forensics and Sec.*, vol. 6, pp. 1099–1110, 2011.

[10] T.-T. Do, E. Kijak, T. Furon, and L. Amsaleg, "Deluding image recognition in SIFT-based CBIR systems," in *Proceedings of the 2nd ACM workshop on Multimedia in forensics, security and intelligence*, ser. MiFor '10. New York, NY, USA: ACM, 2010, pp. 7–12.

[11] A. Piva, "An overview on image forensics," *ISRN Signal Processing*, vol. 2013, p. 22, 2013.

[12] L. Yizhi, L. Shouxun, T. Sheng, and Z. Yongdong, "Adult image detection combining bovw based on region of interest and color moments," in *Intelligent Information Processing V*. Springer Berlin Heidelberg, 2010.

[13] W. Jinjun, Y. Jianchao, Y.and Kai, L. Fengjun, T. Huang, and G. Yihong, "Locality-constrained linear coding for image classification," in *2010 IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[14] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, 2004.

[15] M. Burger, L. He, and C. Schonlieb, "Cahn-Hilliard inpainting and a generalization for grayvalue images," *SIAM Journal on Imaging Sciences*, vol. 2, pp. 1129–1167, 2009.

[16] 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*, 2006.

[17] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[18] A. Vedaldi and B. Fulkerson, "VLFeat: an open and portable library of computer vision algorithms," in *Proceedings of the international conference on Multimedia*. ACM, 2010, pp. 1469–1472.