Weakly Supervised Adaptation to Re-sizing for Image Manipulation Detection on Small Patches

Ludovic Darmet[0000−0001−5445−9763], Kai Wang[0000−0003−0053−3352], and François Cayre[0000−0002−6634−9147]

Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, Grenoble, France
{ludovic.darmet, kai.wang, francois.cayre}@gipsa-lab.grenoble-inp.fr

Abstract. Basic image processing operations like median filtering and Gaussian blurring in general do not change the semantic content of an image, although they are commonly used to cover fingerprints of falsification that does alter image content such as copy-move and splicing. Therefore image forensics researchers are interested in detecting these basic operations. Some existing detectors track local inconsistencies in statistics of the image. However these statistics are very sensitive to image development process. Thus pre-processing operations can be damaging for performances of such detectors. In this paper, we focus on a very common pre-processing operation, *i.e.*, re-sizing, and study how it affects performance when trying to detect several image processing operations on small patches, with Gaussian Mixture Model (GMM) as feature extractor and a Dense Neural Network (DNN) as classifier. We first show performance drops. We then introduce an adaptation method which relies on better fit to testing data for the feature extraction and fine-tuning for the neural network classifier. Experimental results show that our method is able to improve results with very few labeled testing samples. We also present comparisons with an improved version of a recent CNN(Convolutional Neural Network)-based method.

Keywords: Image forensics · Gaussian mixture model · Neural network · Feature adaptation · Weakly supervised · Fine-tuning.

1 Introduction

Digital images now play a more and more important role in our decision-making processes of daily life, either personal or professional. However, people can be misled by falsified images which can now be created very easily even by non-experts. Under this context, it is necessary to build reliable tools to assess integrity of an image and to tell whether it is falsified or not. This is indeed the goal of image forensics research. In this work, we focus on a specific forensic problem, *i.e.*, the detection of basic image processing operations on very small patches of 8×8 pixels. Basic operations, *e.g.*, Gaussian blurring, median filtering and noise addition, are often used during the creation of a fake image to cover the traces of falsifications that do change the semantic meaning of the image.

Detection of such operations on very small patches of 8×8 pixels is challenging due to lack of information, but can make it possible to reliably detect operations applied within a small spatial extent.

In an image development process it is quite common to re-size an image to fit some layout or displaying constraints or else to reduce storage. So this operation is not suspicious in general, as it is a realistic scenario to assume that an image could have been re-sized before being manipulated. This pre-processing operation should not harm performances of forensic detectors. It is not practical to assume that a sufficiently large number of labeled samples are available to re-train models from scratch. From a computing time and power perspective it would be exhaustive and very demanding to perform many times of re-training. A similar issue has been raised in [11] in the field of computer vision, which underlines the importance and benefit of being able to train a model when incomplete or only few labeled samples are available. Training with few labeled samples is also called "few-shot learning" in the machine learning literature [16,15]. In digital image forensics, recently authors of [4] have proposed an interesting method to tackle a related yet different problem of weakly supervised learning. Both considered problem and adopted approach are different from ours. Classifier in [4] is able to distinguish synthetic images from natural ones, and the "target" domain means a new class of samples (*e.g.*, tested on synthetic images created by a new algorithm). The good performance under weakly supervised scenario is mainly due to the design of disentangled latent variables in an auto-encoder-based detector. In this paper, we propose a new and different weakly supervised approach for a classifier based on image statistical models, for the forensics of manipulations on images which have undergone re-sizing pre-processing. In the related field of steganalysis, researchers have expressed some concerns about similar issues of performance drop under the so-called cover-source mismatch [9,7,8]. In general, popular solutions in steganalysis are to train a classifier with more samples of big diversity, or to train multiple classifiers and later use the most suitable one during specific testing. By contrast, in this paper, we propose a light-weight *adaptation* method to image pre-processing for the detection of basic manipulation operations. Our contributions are summarized as follows:

- We raise the problem of forensic performance drop under re-sizing pre-processing, which until now seems ignored and underestimated;
- For image statistical models that have been trained on samples of original size, we develop a simple *weakly supervised* (making use of around 2000 labeled patches) adaptation method to re-sized testing samples;
- Our method takes into account both feature and classifier adaptation and is very quick and straightforward so as to provide a real shortcut.

We will first introduce in Section 2 the research problem. Our approach of weakly supervised adaptation is described in Section 3. Experimental results are presented in Section 4. Finally, we draw the conclusion in Section 5.

2 Background and Research Problem

Image forensics. Various problems have been considered by image forensics researchers [12], such as camera identification [2], identification of synthetic images [4,14], detection of falsification such as splicing and copy-move [3,17], and detection of image manipulation [13,5,10,1]. We are interested in the last topic of detecting image manipulations. Here we distinguish between falsifications, *i.e.*, modifications that alter the semantic content of the image, and *manipulations*, *i.e.*, modifications with basic image processing operations which in general do not change the image's semantic meaning. Regarding manipulation detection, in the literature researchers first focused on building specific and targeted detector for one particular manipulation, *e.g.*, median filtering, JPEG compression, *etc.* Then after achieving successful results with these methods, the community tried to build so-called "universal" detectors. They are detectors not focusing on one particular operation but capable of detecting several ones with same analysis pipeline. Our work follows this trend. Existing universal detectors can be classified into three categories:

1. Explicit statistical modeling of the image (using the Gaussian Mixture Model, GMM) to spot statistical discrepancy [5];
2. Extraction of steganalytic features, *e.g.*, based on SPAM (Substractive Pixel Adjacency Matrix) or SRM (Spatial Rich Model), combined with classifier training [13,10];
3. "End-to-end" deep-learning-based method [1].

As mentioned earlier, we consider that it is crucial to be able to forensically analyze very small patches, so as to be spatially more accurate and capable of detecting manipulation of a very small region. This has driven us to focus on the first GMM-based approach mentioned above [5], as it appears to be the most promising method for very small patches. The explicit statistical modeling can be effectively conducted on patches of 8×8 pixels [5], while the other two approaches, as described in their original papers [13,10,1], work on larger patches. These two approaches seem not specifically designed to cope very well with small patches. In particular, the second approach involves an occurrence accumulation step which is more reliable with more contributing pixels, and tends to have decreasing accuracy as the size of patch decreases [13,10]; and CNNs in the third approach [1] normally contain pooling layers, which would cause loss of information and thus is not necessarily very suitable for small patches.

Performance drop under re-sizing. Considered manipulations are borrowed from [5] and listed in Table 1. Most of these manipulations are also used in [1], but here we consider a slightly more challenging setting (*i.e.*, distortion introduced by manipulation is smaller) than that in [1]. We decide to focus on these basic operations as they are among the most common in image processing. Moreover, they can be used to cover more complex tampering. For instance, one can use Gaussian blurring to smooth boundary between a spliced part and the rest of

Table 1: List of considered manipulations.

ORI	No image modification
GF	Gaussian filtering with 3×3 kernel and $\sigma = 0.5$
MF	Median filtering with window size of 3×3
USM	Unsharp masking with Laplacian filter of window size 3×3 and strength factor of 0.5
WGN	White Gaussian noise addition with $\sigma = 2$
JPEG	JPEG compression with quality factor $Q = 90$

Table 2: Testing accuracy (in %) without any adaptation for GMM-based method using log-likelihood ratio (details in Section 3.1). The first column gives the re-sizing factors. The performance drop compared to the case without re-sizing (*i.e.*, the row of $\times 1$) is given in parentheses. We do not use ratios like 0.5 to avoid the potential special side effect of such ratios. The last column of "AVG" gives the average accuracy and performance drop of the 5 classification problems.

	GF	MF	USM	WGN	JPEG	AVG
$\times 1$	91	86	97	98	89	92
$\times 0.51$	64 (-27)	75 (-11)	73 (-24)	69 (-29)	79 (-10)	72 (-20)
$\times 0.76$	78 (-13)	81 (-5)	81 (-16)	73 (-25)	84 (-5)	79 (-13)
$\times 1.15$	55 (-36)	80 (-6)	87 (-10)	85 (-13)	79 (-10)	77 (-15)
$\times 1.25$	51 (-40)	75 (-11)	74 (-23)	81 (-17)	67 (-22)	70 (-22)

an image. In the following, "source" indicates training samples that have not undergone re-sizing. "Target" means the testing samples which have undergone re-sizing before applying a possible manipulation that we want to detect. This study is interested in the impact of image re-sizing as *pre-processing* operations on detection of image manipulations. Bi-cubic interpolation is used to re-size testing images as, in general, it is the hardest case for carrying out successful adaptation. Drops of detection accuracy of the GMM-based method, when there is no adaptation, can be observed in Table 2. From this table, we can also see that the method works quite well on 8×8 patches of images of original size, with an average accuracy of about 92%. However, the performance drop, when there is re-sizing pre-processing, is sometimes quite significant.[1] This has motivated our work of detector adaptation presented in the next section.

3 Proposed Approach

3.1 Classification Pipeline

Our pipeline is largely inspired by the method of Fan *et al.* [5] as it is one of the state-of-the-art methods for detecting manipulations on small patches. Firstly

[1] As shown in Section 4, performance decrease also exists for CNN-based method [1].

Gaussian Mixture Models (GMMs) are trained, one for each set of patches (original, Gaussian filtered, median filtered, *etc.*), six models in total (see Table 1). Models are trained to maximize likelihood on patches with the Expectation-Maximization (EM) algorithm. Log-likelihood for sample $\boldsymbol{x}_l$ under a mixture of N Gaussian components, parameterized by $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{,k=1,2,\ldots,N}$, is:

$$\mathcal{L}(\boldsymbol{x}_l|\boldsymbol{\theta}) = \log\left(\sum_{k=1}^{N} \pi_k \mathcal{N}(\boldsymbol{x}_l|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right), \tag{1}$$

with π_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ respectively the weight, mean and multivariate (full) covariance matrix for k^{th} component in the mixture. Here DC component of each patch is removed so patch mean is 0 thus $\boldsymbol{\mu}_k$ are all zeros. After the GMMs are trained, a very quick and efficient technique to produce a decision for a testing sample is to compute log-likelihood for each GMM and compare these values. In the case of binary classification, it means calculating the log-likelihood ratio between the GMM of manipulated patches and that of original patches [5], as:

$$r(\boldsymbol{x}_l) = \frac{\mathcal{L}_{GMM_{manip}}(\boldsymbol{x}_l)}{\mathcal{L}_{GMM_{ori}}(\boldsymbol{x}_l)}. \tag{2}$$

If the ratio $r(\boldsymbol{x}_l) > 1$ then the decision should be that sample patch $\boldsymbol{x}_l$ is a manipulated one, otherwise it is original.

In the first step of EM (E step), we need to compute component scores which are likelihood values with regard to each Gaussian component in the GMM:

$$r_k^{(l)} = \pi_k \mathcal{N}(\boldsymbol{x}_l|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{3}$$

We notice that these component scores form a more detailed descriptor than the log-likelihood value for patch $\boldsymbol{x}_l$. Therefore in this paper we propose to use them as features to feed a classifier. For binary classification, the $2N$-dimensional feature vector $(r_{1,ori}^{(l)}, r_{2,ori}^{(l)}, \ldots, r_{N,ori}^{(l)}, r_{1,manip}^{(l)}, r_{2,manip}^{(l)}, \ldots, r_{N,manip}^{(l)})$ of patch sample $\boldsymbol{x}_l$ is a concatenation of component scores of the two trained GMMs under comparison, each having N components. We use a small Dense Neural Network (DNN) as classifier whose architecture is described in Section 4. As expected, experiments show that performances of baseline scenario (*i.e.*, when testing samples are not re-sized) are almost identical to the detector based on log-likelihood ratio. In the following we propose a weakly supervised adaptation method of our classification pipeline composed of GMMs and DNN.

3.2 Weakly Supervised Adaptation

The proposed adaptation method comprises two sub-steps. First, GMMs are adapted so that they fit better to the testing samples which have undergone re-sizing. Then the DNN classifier is adapted by fine-tuning the network. Both steps are accomplished in a weakly supervised manner, *i.e.*, by using a very limited number of labeled testing samples of 8×8 patches.

In the following, we first show that if DC components of patches are removed (*i.e.*, Gaussian component's means $\boldsymbol{\mu}_k = \mathbf{0},\ \forall k = 1, 2, \ldots, N$), the weighted sum of covariance matrices of a GMM is equal to the covariance matrix of the data. We have $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$ a multi-dimensional random variable. Here $p = 64$ as patches are 8×8. Let f be the Probability Density Function (PDF) of random variable $\boldsymbol{X}$ with DC component removed and f_k the PDFs of each component of the related Gaussian mixture, then we have:

$$f(\boldsymbol{x}_l) = \sum_{k=1}^{N} \pi_k \ f_k(\boldsymbol{x}_l) = \sum_{i=1}^{N} \pi_k \ \mathcal{N}(\boldsymbol{x}_l | \mathbf{0}, \boldsymbol{\Sigma}_k), \tag{4}$$

with $\boldsymbol{x}_l$ a p-dimensional sample of the random variable $\boldsymbol{X}$, $\boldsymbol{\Sigma}_k$ and π_k respectively the covariance and the weight for the k^{th} component in the mixture. Now let us compute elements of the covariance matrix of $\boldsymbol{X}$:

$$\begin{aligned} &\operatorname{cov}(X_i, X_j) = \mathbb{E}_f[X_i X_j] - \mathbb{E}_f[X_i]\,\mathbb{E}_f[X_j] = \mathbb{E}_f[X_i X_j] \\ &= \sum_{k=1}^{N} \pi_k \,\mathbb{E}_{f_k}[X_i X_j] = \sum_{k=1}^{N} \pi_k (\boldsymbol{\Sigma}_k^{(i,j)} + \boldsymbol{\mu}_k^{(i)} \boldsymbol{\mu}_k^{(j)}) = \sum_{k=1}^{N} \pi_k \boldsymbol{\Sigma}_k^{(i,j)}, \end{aligned} \tag{5}$$

where superscripts $(i), (j)$ and (i, j) are element index within the corresponding vector and matrix. Considering that variance is a special case of covariance, from Eq. (5) we can see that covariance matrix of the data is equal to the weighted sum of covariance matrices of the Gaussian mixture.

We assume that we have only a few labeled samples on target domain, not enough to train a model from scratch (this needs around 200000 samples of each class) but enough to compute empirical covariance matrix per class on target (around 1000 samples for each class). GMMs' parameters should be slightly adjusted so as to enhance the descriptive capability of the model on target data. GMMs can be adjusted in two ways: the weights or the covariance matrices (the means are zeros). Beside that, our aim is to have a quick adaptation solution. Therefore, we choose to adapt the GMM weights. The weights contain less parameters (only a vector and not matrices). Adaptation of GMMs' weights can be formulated as an optimization problem:

$$\begin{aligned} &\underset{w_k}{\text{minimize}} \left\| \left(\sum_{k=1}^{N} w_k \times \boldsymbol{\Sigma}_k \right) - \boldsymbol{\Sigma}_{data} \right\|_F \\ &\text{subject to } \sum_{k=1}^{N} w_k = 1, \text{ and } \ 0 < w_k < 1, \ \forall k = 1, 2, \ldots, N. \end{aligned} \tag{6}$$

In Eq. (6), w_k are adapted GMM weights to be deduced, $\boldsymbol{\Sigma}_{data}$ is the empirical covariance matrix on target domain, and F stands for Frobenius norm. We do acknowledge that semi-definite positive matrices lie on a Riemannian manifold, thus with a curvature. So a geodesic distance would be more adapted; however we do not notice any differences in classification performances or results

of optimization using Euclidean distance instead of geodesic distance, although geodesic distance is way more expensive and slower to compute. Therefore in practice Frobenius norm is used.

These adjustments of weights for each GMM can be seen as a fine-tuning for feature extraction. It is a means of reducing discrepancy between features of source and target domains. In the second step of our method, *classifier adaptation* by fine-tuning the DNN is carried out to cope with drifts in features and therefore enhance discriminative capability of the classifier.

4 Experiments

In this section, we present some experimental results to show the feasibility of the proposed weakly supervised adaptation method.

Dataset and implementation. For experiments, we looked for a large database (around 1000 images), in high-resolution (more realistic), in RAW format to be able to control the image development process and with as many image sources as possible. Dresden database [6] is the best match with these expectations with 1200 images, in RAW format from various source cameras, different scenes and exposures and of around 2000×4000 resolution. We select randomly 30% of images as testing images. Re-sizing is performed on the full-sized images (not patches) and before applying the potential manipulations. This is indeed a pre-processing operation which does not alter fingerprints of manipulations but only statistics of images. The remaining 70% are used to train the GMMs and the DNN classifier. This training set is never re-sized. For training, we use 200000 patches of 8×8 pixels for each class. These patches are extracted randomly from images in the training set, with same number of patches coming from each image. This makes 400000 patches for each binary classification problem.

GMMs have $N = 75$ components each. This number has been chosen via cross-validation to reach a good trade-off between classification accuracy and model complexity (as well as training time). For GMM training, we used Scikit-Learn implementation with 5 initializations for π_k and $\boldsymbol{\Sigma}_k$ with k-means. $\boldsymbol{\mu}_k$ are initialized to be zeros. Initialization that obtains best likelihood on data is selected. For the classification part, we perform binary classification of original patches *vs.* manipulated patches. We used Keras (with Tensorflow backend) for the DNN implementation and training. It is a very simple network with two hidden layers of respectively 256 and 128 neurons, ReLU activation, dropout of 0.5 and Adam optimizer with default parameters for minimizing *cross-entropy* loss. This architecture has not been optimized as it is not a crucial part given that classification is quite easy. For fine-tuning, learning rate is reduced to 10^{-4}. Batch size is 128. 1000 samples of each class are used to compute empirical covariance matrices in order to be able to adapt GMM weights and fine-tune the DNN. Code will be soon available on-line.

With DNN fine-tuning only. DNN fine-tuning helps to improve the detection accuracy, but sometimes the improvement is rather limited (Table 3, rows of

Table 3: Testing accuracy (in %) of DNN fine-tuning, combined without or with weights adaptation of GMMs. The improved accuracy of weakly supervised adaptations, compared to the case of "without adaptation", is given in parentheses. Testing accuracy without re-sizing is also given in the second row for reference. The last column of "AVG" presents the average accuracy (and average accuracy improvement in parentheses, if any) of the 5 classification problems.

	GF	MF	USM	WGN	JPEG	AVG
without re-sizing	91	86	97	98	89	92
re-sizing ×0.51 (without adaptation)	64	75	73	69	79	72
re-sizing ×0.51 (DNN fine-tuning only)	72 (+8)	75 (+0)	91 (+18)	71 (+2)	82 (+3)	78 (+6)
re-sizing ×0.51 (GMM adaptation + DNN fine-tuning)	78 (+14)	76 (+1)	92 (+19)	70 (+1)	86 (+7)	80 (+8)
re-sizing ×0.76 (without adaptation)	78	81	81	73	84	79
re-sizing ×0.76 (DNN fine-tuning only)	78 (+0)	77 (-4)	92 (+11)	81 (+8)	84 (+0)	82 (+3)
re-sizing ×0.76 (GMM adaptation + DNN fine-tuning)	83 (+5)	82 (+1)	94 (+13)	85 (+12)	84 (+0)	86 (+7)
re-sizing ×1.15 (without adaptation)	55	80	87	85	79	77
re-sizing ×1.15 (DNN fine-tuning only)	66 (+11)	82 (+2)	95 (+8)	96 (+11)	79 (+0)	84 (+7)
re-sizing ×1.15 (GMM adaptation + DNN fine-tuning)	70 (+15)	85 (+5)	95 (+8)	96 (+11)	82 (+3)	86 (+9)
re-sizing ×1.25 (without adaptation)	51	75	74	81	67	70
re-sizing ×1.25 (DNN fine-tuning only)	63 (+12)	78 (+3)	95 (+21)	90 (+9)	70 (+3)	79 (+9)
re-sizing ×1.25 (GMM adaptation + DNN fine-tuning)	66 (+15)	80 (+5)	95 (+21)	95 (+14)	78 (+11)	83 (+13)

"DNN fine-tuning only"). By fine-tuning, the classifier's decision boundary is slightly adjusted, somehow similar to the case of selecting a new threshold for the comparison of likelihood (instead of 1 initially). In order to further enhance the discriminative power of the whole forensic pipeline, it is necessary to also adapt GMMs, the underlying feature extractor, which are until now trained solely on the source data while being "blind" to the target domain. Therefore, GMMs should be tweaked, more precisely their weights, in order to better fit the target data (as described in Section 3.2).

Table 4: Testing accuracy (in %) of DNN fine-tuning, combined without or with weights adaptation of GMMs, for the case of mixed re-sizing factors. Re-sizing factor is drawn following uniform law within the specified interval.

	GF	MF	USM	WGN	JPEG	AVG
re-sizing $\times[0.48, 0.72]$ (without adaptation)	71	81	76	72	87	77
re-sizing $\times[0.48, 0.72]$ (DNN fine-tuning only)	78 (+7)	81 (+0)	91 (+15)	76 (+4)	87 (+0)	83 (+6)
re-sizing $\times[0.48, 0.72]$ (GMM adaptation + DNN fine-tuning)	83 (+12)	80 (-1)	92 (+16)	80 (+8)	88 (+1)	85 (+8)
re-sizing $\times[1.12, 1.27]$ (without adaptation)	53	78	81	83	74	74
re-sizing $\times[1.12, 1.27]$ (DNN fine-tuning only)	63 (+10)	78 (+0)	95 (+14)	91 (+8)	75 (+1)	80 (+6)
re-sizing $\times[1.12, 1.27]$ (GMM adaptation + DNN fine-tuning)	64 (+11)	83 (+5)	98 (+17)	95 (+12)	78 (+4)	84 (+10)

With GMM weights adaptation. We observe in Table 3 some clear improvements (*e.g.*, for WGN and JPEG under upsampling of $\times 1.25$) when fine-tuning of DNN is conducted jointly with GMM weights adaptation. In addition, there is consistent average accuracy improvement under all the considered re-sizing factors (last column of Table 3) for adaptation of both GMMs and DNN, when compared to DNN fine-tuning only. The standard deviation of results is under 10^{-1}. Accuracy increase offered by adaptation of GMMs and DNN depends on manipulations. For example our method is able to recover up to +19% for sharpening (USM) and re-sizing of $\times 0.51$, but there are not such improvements for median filtering (MF). Median filtering is the manipulation with the smallest score (86% in Table 2, row of $\times 1$) on baseline (without re-sizing of testing set) and is also across re-sizing factor one of the hardest to deal with. Beside that, our method works better with upsampling (for example +13% for average accuracy improvement with a factor of $\times 1.25$ and less for factors $\times 0.51$ and $\times 0.76$). Our conjecture is that with downsampling, some striking local dependencies in patches are partially removed so it needs more complex transformation than a simple weights adjustment to allow GMM to well describe them. With upsampling, the dependencies are somehow mildly smoothed so it is easier to adapt.

Mixed re-sizing factors. Our method also performs well with a mix of re-sizing factors. As shown in Table 4, our method still obtains good results when factors are randomly (following uniform distribution) drawn within an interval. This is not a surprise as our method only intends to adapt the GMM-based feature extractor to the new covariance of the data and the DNN classifier to these new features, without taking into account the specific factor value and algorithm of the re-sizing pre-processing.

Table 5: Testing accuracy (in %) of an improved version of Bayar and Stamm's CNN-based method [1], for cases of with and without fine-tuning. The improved accuracy of weakly supervised fine-tuning, compared to the case of "without fine-tuning", is given in parentheses. The baseline testing accuracy without any re-sizing pre-processing is given in the second row. Results in this table are to be compared with those given in Table 3.

	GF	MF	USM	WGN	JPEG	AVG
without re-sizing	79	85	91	86	79	84
re-sizing $\times 0.51$ (without fine-tuning)	68	82	80	66	76	74
re-sizing $\times 0.51$ (fine-tuning)	73 (+5)	82 (+0)	85 (+5)	69 (+3)	77 (+1)	77 (+3)
re-sizing $\times 0.76$ (without fine-tuning)	73	83	89	80	79	81
re-sizing $\times 0.76$ (fine-tuning)	74 (+1)	83 (+0)	89 (+0)	80 (+0)	79 (+0)	81 (+0)
re-sizing $\times 1.15$ (without fine-tuning)	59	76	79	86	61	72
re-sizing $\times 1.15$ (fine-tuning)	67 (+8)	80 (+4)	90 (+11)	86 (+0)	66 (+5)	78 (+6)
re-sizing $\times 1.25$ (without fine-tuning)	55	72	74	82	55	68
re-sizing $\times 1.25$ (fine-tuning)	65 (+10)	77 (+5)	91 (+15)	86 (+4)	60 (+5)	76 (+8)

Comparisons with Bayar and Stamm's CNN-based method. We compare with the state-of-the-art deep-learning-based method in [1]. The approach is different from ours as feature extraction and classification are carried out in an end-to-end way in the CNN. As shown in the following, the CNN-based method also experiences some performance drops due to re-sizing pre-processing. In order to make the CNN work with 8×8 patches, one or some of the four *pooling* layers of the network in [1] have to be removed. Otherwise outputs of these layers drop to 1×1 and following 2D convolution is not possible anymore. We have tried different configurations and numbers of retained pooling layers (0, 1 or 2 retained layers are technically possible) and found that we obtain better performance without any pooling. This is understandable as pooling layers would cause loss of information. Performances are also better with a learning rate of 10^{-4} instead of 10^{-3} (value suggested in the original paper [1]). Results reported in Table 5 have been obtained with these improved settings, best that we can get after many experiments. We can notice that the baseline scores (without re-sizing) of CNN-based method are lower than our GMM-based method, with an average accuracy of 84% for CNN *vs.* 92% for GMM (see Tables 5 and 3, row of without re-sizing). There is also performance drop for CNN-based method under re-sizing pre-processing, especially for upsampling. The performance de-

crease can be as big as -24% for detection of JPEG compression with re-sizing of $\times 1.25$ (see Table 5, with a decrease from 79% to 55%). We observe comparable performance drop under re-sizing pre-processing for different settings of CNN that we tried during our experiments. In general, although CNN has smaller amount of accuracy decrease (this point deserves further studies), but the final decreased accuracy without adaptation is comparable for CNN-based and GMM-based methods (*cf.* the corresponding rows in Tables 5 and 3), with a same trend of more accuracy decrease under upsampling for both methods.
For CNN-based method, we used Caffe implementation from authors available at `https://gitlab.com/MISLgit/constrained-conv-TIFS2018`. Images selected for training and testing, patches generated and number of patches generated are exactly the same as with the GMM-based method. The weakly supervised adaptation is realized by the conventional way of fine-tuning the CNN. Again, we made efforts to try different strategies, *i.e.*, fine-tuning the first few, the last few, and all layers of the network. We find that only fine-tuning the dense layers at the end of CNN gives slightly better performance than other strategies. Learning rate has been reduced to 10^{-5}. This learning rate and fine-tuning setting gave us the best performances of adaptation in our experiments. Fine-tuning of CNN is performed on the same number of re-sized samples (2000 patches per class). Adaptation of CNN helps to improve the accuracy under re-sizing pre-processing, especially for upsampling, as shown in Table 5. In general, our adaptation of GMM-based method gives more improvement and higher improved accuracy than CNN-based method. The final accuracy of GMM-based method is 80%, 86%, 86% and 83% for the four re-sizing factors (Table 3), against respectively 77%, 81%, 78% and 76% for the CNN-based method (Table 5).

In all, it is interesting to see that GMM-based method (using explicit image statistical models) has better performance than CNN for this forensic problem on small patches, in terms of both baseline accuracy without re-sizing and improved accuracy after adaptation to re-sizing pre-processing. It is clear that more theoretical and experimental investigations are needed to better understand these results. Nevertheless, the results are encouraging for us to continue on this image-model-based approach when nowadays CNN becomes dominating in many image forensic research problems.

Example of image forgery localization. Evaluating and analyzing the use of the proposed image manipulation detection method, probably jointly with other forensic methods, for the localization of image forgery is out of the scope of this paper and constitutes one part of our future work. In the following, we only present an example of forgery localization to show the feasibility of our method for this task. As highlighted in red circle in Fig. 1.(a), a small part (89×187 pixels) has been taken from a *source* image of the testing dataset and inserted into a *host* image also in the testing set. This inserted part (thus the source image) has previously been JPEG compressed with quality factor $Q = 90$, while the host image is an original one. We try to detect this JPEG compression and therefore localize the splicing forgery indirectly.
Output map of our GMM-based detector on image of original size is shown in

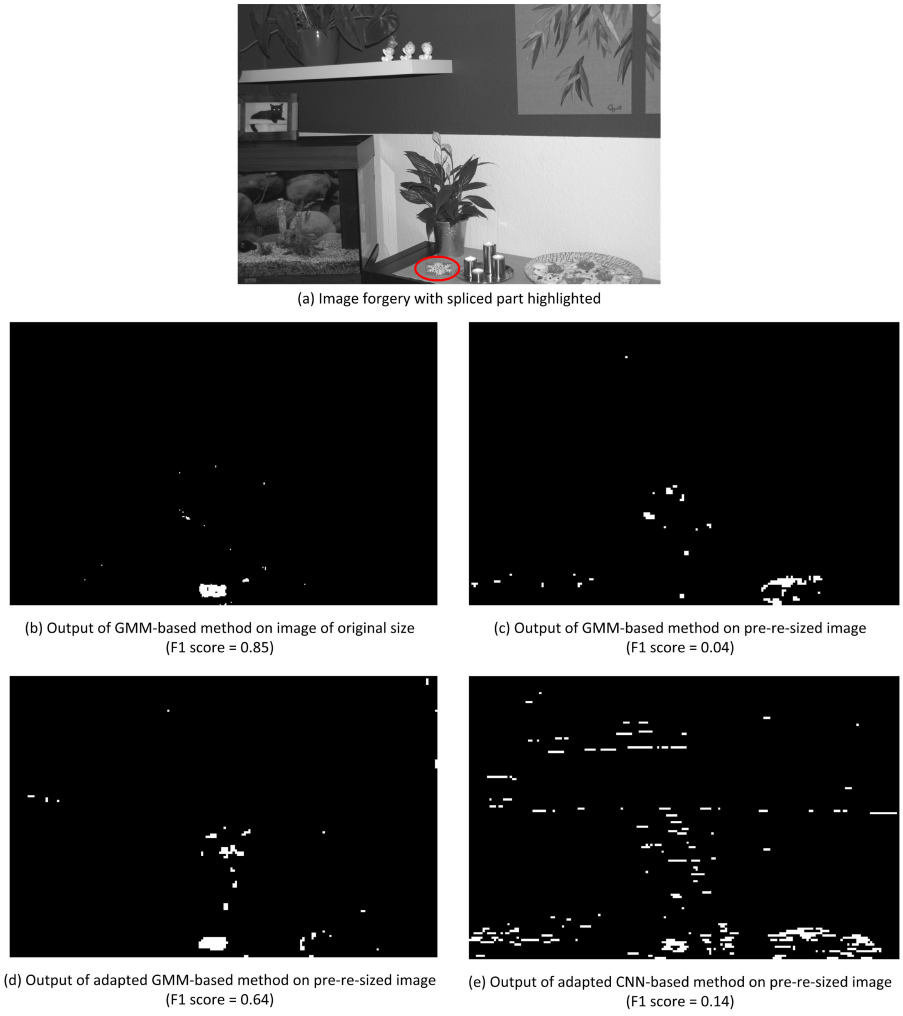

Fig. 1: Example of image forgery localization.

Fig. 1.(b). Just next to it, in (c) we show the output of the same detector but for the situation where source image and host image have been pre-re-sized with factor of $\times 0.51$ before JPEG compression and spliced part insertion, respectively. The map in (d) is the result of our detector after adaptation to pre-re-sizing. In (e) it is the output map with re-sizing pre-processing for the adapted CNN-based detector of Bayar and Stamm [1] (see previous part of this section for details of CNN improvement and adaptation). As we deal with small patches, output map could be noisy, so for better visualization each output map in (b)-(e) has undergone a 3×3 median filtering to smooth the output. The F1 scores given in Fig. 1 are computed on maps before smoothing. As can be expected from the

performances on 8×8 patches, manipulation is quite well localized on image of original size in (b) but not well on the pre-re-sized image in (c). In particular, here we show an example where our detector without adaptation has dramatic performance drop on JPEG compressed patches, *i.e.*, in the spliced part which has rich texture, and where there is noticeable improvement after adaptation. This performance drop is however understandable because with downsampling pre-processing (here of factor $\times 0.51$), in general both original and compressed patches can have more mid- and high-frequency components. JPEG compression introduces JPEG grid and also removes mainly high-frequency components within each block. The grid artifact is quite similar with or without re-sizing because it is a pre-processing operation. Therefore, a downsampled JPEG compressed patch with rich texture can be very "similar", in a forensic sens, to an uncompressed patch without re-sizing. This may explain the low accuracy in the JPEG compressed spliced part as illustrated in Fig. 1.(c), which is below the accuracy on whole re-sized testing data reported in Table 3.

Our adaptation method improves the localization accuracy, as reflected by the map and F1 score in Fig. 1.(d). It is worth mentioning that the localization on pre-re-sized image is harder because spliced part becomes smaller after downsampling pre-processing. The adapted CNN-based method works not as well as our method on this example, in particular with a number of false alarms in the pristine part of the spliced image as shown in (e). As mentioned above, a thorough evaluation and analysis is scheduled as future work.

Computational cost. Our adaptation method is very quick. Solving the optimization problem in Eq. (6) takes around 1 minute, and DNN fine-tuning 2 minutes. Training six GMMs from scratch each with 200000 samples takes around 12 hours in total on CPU, and DNN training about 20 minutes. Training CNN-based method of Bayar and Stamm lasts around 2 hours on GPU for every binary classification problem with 200000 samples for each class. This is quite fast because of small size of patches. Fine-tuning CNN takes about 10 minutes. We use a standalone computer with Intel Xeon CPU E5-2630, 64GB RAM and Nvidia 1080 Ti GPU. In all, the adaptation of our method is slightly faster then CNN-base method (about 4 minutes *vs.* 10 minutes). The GMM training would be faster than CNN training if we could use a GPU implementation.

5 Conclusion

This work outlined how re-sizing as pre-processing could alter performances of a manipulation detector based on local image statistics and a state-of-the-art CNN-based detector. We propose a method to adapt both GMM-based feature extractor, by adjusting weights, and DNN classifier, by fine-tuning. Experimental results show the feasibility of the proposed weakly supervised adaptation method which tries to better fit covariance of target domain data for the GMM feature extractor. In some cases with our method using 2000 labeled target samples (1000 per class), we obtain almost same results as re-training from scratch

with 400000 labeled samples, *e.g.*, detection of Gaussian noise addition with re-sizing of $\times 1.25$. Our adaptation takes a few minutes instead of several hours for re-training from scratch. This provides a shortcut in terms of flexibility and computing power. Our image-statistics-based detector outperforms an improved version of the state-of-the-art CNN-based detector [1], in terms of both baseline and adapted accuracy for the situations without and with re-sizing pre-processing, respectively. However, we are aware that this CNN-based method has been originally designed for bigger patches but not for small ones; in the meanwhile, to the best of our knowledge, there are until now no published results of manipulation detection on 8×8 patches for CNN-based and other methods. Therefore, in the future more appropriate CNNs have to be designed for this specific case and compared with image-statistics-based method.

The performance of the proposed adaptation under downsampling needs to be improved, probably with a stronger GMM adaptation which also adjusts the components' covariance structure. We would like to mention that in the literature post-processing is commonly studied when testing a forensic detector, however effects of pre-processing on performances have been much less investigated. In this paper we introduce new concerns related to the pre-processing and a new methodology to carry out light-weight adaptation. We plan to extend this framework to cope with other pre-processing operations, the case of unsupervised adaptation, as well as other forensics domain adaptation scenarios which have been receiving more and more attention among the research community [4]. We also intend to conduct studies on methods based on other type of features such as [13,10]. At last, we think that it is possible to improve adaptation of CNN for specific problems of digital image forensics by using approaches other than fine-tuning and we plan to work on it.

Acknowledgement

This work is supported by French National Research Agency (DEFALS ANR-16-DEFA-0003, ANR-15-IDEX-02).

References

1. Bayar, B., Stamm, M.C.: Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection. IEEE Transactions on Information Forensics and Security **13**(11), 2691–2706 (2018)
2. Bondi, L., Baroffio, L., Güera, D., Bestagini, P., Delp, E.J., Tubaro, S.: First steps toward camera model identification with convolutional neural networks. IEEE Signal Processing Letters **24**(3), 259–263 (2017)
3. Cozzolino, D., Poggi, G., Verdoliva, L.: Splicebuster: A new blind image splicing detector. In: Proceedings of the IEEE International Workshop on Information Forensics and Security. pp. 1–6 (2015)
4. Cozzolino, D., Thies, J., Rössler, A., Riess, C., Nießner, M., Verdoliva, L.: Forensictransfer: Weakly-supervised domain adaptation for forgery detection. arXiv CoRR pp. 1–12 (2018), `https://arxiv.org/abs/1812.02510`

5. Fan, W., Wang, K., Cayre, F.: General-purpose image forensics using patch likelihood under image statistical models. In: Proceedings of the IEEE International Workshop on Information Forensics and Security. pp. 1–6 (2015)
6. Gloe, T., Böhme, R.: The Dresden image database for benchmarking digital image forensics. In: Proceedings of the ACM Symposium on Applied Computing. pp. 1585–1591 (2010)
7. Ker, A., Pevný, T.: A mishmash of methods for mitigating the model mismatch mess. In: Proceedings of the SPIE Media Watermarking, Security, and Forensics. vol. 9028, pp. 90280I:1–90280I:15 (2014)
8. Kodovský, J., Fridrich, J.: Effect of image downsampling on steganographic security. IEEE Transactions on Information Forensics and Security **9**(5), 752–762 (2014)
9. Kodovský, J., Sedighi, V., Fridrich, J.: Study of cover source mismatch in steganalysis and ways to mitigate its impact. In: Proceedings of the SPIE Media Watermarking, Security, and Forensics. vol. 9028, pp. 90280J:1–90280J:12 (2014)
10. Li, H., Luo, W., Qiu, X., Huang, J.: Identification of various image operations using residual-based features. IEEE Transactions on Circuits and Systems for Video Technology **28**(1), 31–45 (2018)
11. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Is object localization for free? - Weakly-supervised learning with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 685–694 (2015)
12. Piva, A.: An overview on image forensics. ISRN Signal Processing (496701), 1–22 (2013)
13. Qiu, X., Li, H., Luo, W., Huang, J.: A universal image forensic strategy based on steganalytic model. In: Proceedings of the ACM Workshop on Information Hiding and Multimedia Security. pp. 165–170 (2014)
14. Quan, W., Wang, K., Yan, D.M., Zhang, X.: Distinguishing between natural and computer-generated images using convolutional neural networks. IEEE Transactions on Information Forensics and Security **13**(11), 2772–2787 (2018)
15. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 4077–4087 (2017)
16. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 3630–3638 (2016)
17. Wu, Y., Abd-Almageed, W., Natarajan, P.: Busternet: Detecting copy-move image forgery with source/target localization. In: Proceedings of the European Conference on Computer Vision. pp. 170–186 (2018)