Data-Dependent Scaling of CNN's First Layer for Improved Image Manipulation Detection

Ivan Castillo Camacho and Kai Wang(✉)

Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, Grenoble, France
{ivan.castillo-camacho,kai.wang}@gipsa-lab.grenoble-inp.fr

Abstract. Convolutional Neural Networks (CNNs) have become an effective tool to detect image manipulation operations, *e.g.*, noise addition, median filtering and JPEG compression. In this paper, we propose a simple and practical method for adjusting the CNN's first layer, based on a proper scaling of first-layer filters with a data-dependent approach. The key idea is to keep the stability of the variance of data flow in a CNN. We also present studies on the output variance for convolutional filter, which are the basis of our proposed scaling. The proposed method can cope well with different first-layer initialization algorithms and different CNN architectures. The experiments are performed with two challenging forensic problems, *i.e.*, a multi-class classification problem of a group of manipulation operations and a binary detection problem of JPEG compression with high quality factor, both on relatively small image patches. Experimental results show the utility of our method with a noticeable and consistent performance improvement after scaling.

Keywords: Image forensics · Convolutional Neural Network · First-layer convolutional filter · Image manipulation detection · Stability of variance

1 Introduction

Rapid technology development of cameras to capture digital images comes together with a big suite of software to modify an image. Now changes on an image can be so subtle that noticing them for the naked eye is a difficult task. At the same time, the development of techniques that analyze intrinsic fingerprints in image data, *i.e.*, the image forensics research, is one of the most effective ways to solve challenges related to the authentication of digital images.

Basic image manipulation operations such as median filtering, resampling and noise addition are commonly used during the creation of tampered images. Our objective is to detect traces left by these manipulation operations in an image. It is not a surprise that the current trend is to use Convolutional Neural Network (CNN) to detect image manipulations because of its very good forensic performance. In the meanwhile, image forensics researchers in general agree that

X. Zhao et al. (Eds.): IWDW 2020, LNCS 12617, pp. 208–223, 2021.
https://doi.org/10.1007/978-3-030-69449-4_16

specific design is required in order to build successful CNNs for forensic tasks. In particular, the CNN's first layer needs to be carefully designed so as to extract useful information relevant to the forensic task at hand [1,4,9]. Such relevant information is often believed to be in the so-called image residuals, roughly speaking, in the high-frequency components of the image.

Accordingly, for image manipulation detection, a common choice for the CNN's first layer is high-pass filters. Although satisfying results can be achieved, one important aspect, *i.e.*, the stability of the amplitude of the data flow in CNN, has often been ignored or has not been carefully studied. We have the intuition that after the image data passes through a first layer of high-pass filters, the filters' output becomes a weak signal. This would be detrimental to the training of CNN because the data flow shrinks. In this paper, we show that this signal shrinking indeed exists for first-layer filters generated by several popular initialization algorithms that have been used for detecting image manipulations. In addition, with a proper formulation of the first-layer's convolution operation and based on natural image statistics, we provide an intuitive explanation regarding the signal shrinking and subsequently propose a simple scaling method to enhance the output signal. Experimental results, with different first-layer initializations, CNN architectures and classification problems, show that the proposed scaling method leads to consistent improvement of forensic performance.

The remainder of this paper is organized as follows. In Sect. 2, we briefly review the related work. In Sect. 3 we present an experimental study, as well as an intuitive theoretical explanation, regarding the *variance* of the output signal of CNN's first-layer filters of different initializations. Our proposed data-dependent scaling method is described in Sect. 4. Experimental results are presented in Sect. 5. Finally, conclusions are drawn in Sect. 6.

2 Related Work

Early methods for detecting basic image manipulation operations were based on feature extraction and classifier training. Different handcrafted features were proposed to detect specific and *targeted* manipulation operation, *e.g.*, median filtering, resampling and JPEG compression. Afterwards, researchers focus on the more challenging problem of developing a *universal* method for image manipulation detection. Various methods have been proposed, based on steganalysis features, image statistical model and more recently deep learning.

During the last decade, deep learning methods, including CNNs, have gained outstanding success in a wide range of research problems in the computer vision field. CNNs can learn themselves useful features from given data, effectively replacing the difficult task of handcrafted feature design for human experts. In the recent years, CNNs have also been used to solve image forensic problems. Researchers have noticed a fundamental difference between computer vision tasks and image forensics tasks. The former focuses on the semantic content of images, while the latter often looks for a weak signal representing the difference between authentic and manipulated images. Accordingly, image forensics

researchers found that directly applying CNN initialization borrowed from the computer vision field, *e.g.*, the popular Xavier initialization [7], results in rather limited performance in detecting image manipulation operations [1,4]. Different customized CNN initialization algorithms have been proposed to cope better with forensic tasks. The basic idea of these methods are more or less similar, *i.e.*, generating or using a kind of high-pass filters at the CNN's first layer.

SRM (Spatial Rich Model) filters are one popular and effective way to initialize the first layer of CNNs that are used to solve image forensic problems, *e.g.*, the detection of manipulation operations [3], of splicing and copy-move forgeries [10], and of inpainted images [9]. SRM filters, a group of handcrafted high-pass filters originally designed for steganalysis [5], are put at CNN's first layer as initialization and this often leads to very good forensic performances. Indeed, as shown later in this paper, in many cases that we tested, SRM filters outperform other kinds of first-layer filters, especially after the proposed scaling.

Bayar and Stamm [1] proposed a new type of *constrained* filters for the first layer of a CNN designed to detect image manipulation operations. The idea is to constrain the network's first layer to learn a group of high-pass filters. This is realized by normalizing the filters before each forward pass of the CNN training. The normalization consists of two steps: firstly, the center element of filter is reset as -1; secondly, all the non-center elements are scaled so that they sum up to 1. In this way the sum of all filter elements is 0, and the constrained first-layer filter behaves like a high-pass one which is effective in suppressing image content. Recently, Castillo Camacho and Wang [2] proposed an alternative way of initializing CNN's first layer for image manipulation detection. This is essentially an adaptation of the conventional Xavier initialization [7] to the situations where it is required to generate high-pass filters after initialization. This method can generate a set of random high-pass filters to be put at CNN's first layer.

In this paper, we consider four different algorithms for first-layer initialization of CNNs with the application to image manipulation detection: the conventional Xavier initialization from the computer vision community [7], the initialization with SRM filters [5], Bayar and Stamm's constrained filters [1], and Castillo Camacho and Wang's high-pass filter initialization [2]. We show that all the four methods can produce filters which shrink the input signal at their output, and that our proposed data-dependent scaling can noticeably and consistently improve the forensic performance for all the four initialization algorithms when tested on different CNN architectures and forensic problems.

3 Variance of Output of Convolutional Filter

It is demonstrated that the stability of the data flow in CNN, as reflected by the *variance* of the signal in and out a layer, is beneficial for the training of CNN [7,8]. Ideally, the variance of input and output of a layer should be *equal* to each other. In this section, we first show that we can predict the variance of the output of a convolutional filter by using statistics of input signal and elements of the filter. Then, we present observations and understandings regarding the output

variance for the four initialization algorithms of convolutional filter which are mentioned in the last section. For the sake of brevity, the four algorithms are hereafter called Xavier [7], SRM [5], Bayar [1], and Castillo [2].

3.1 Formulation

Motivation. We observe potential limitations of the four initialization algorithms and have the intuition that the variance of output of a convolutional filter initialized by them may change substantially compared to the input. SRM [5] and Bayar [1] do not take into account the output variance during the initialization, because the two algorithms put directly third-party SRM filters or normalized high-pass filters at first layer without modelling the relation between input and output. Xavier [7] and Castillo [2] consider the input-output relation and generate pseudo-random filters. These two initialization algorithms are based on a statistical point of view and realized by drawing pseudo-random samples, so in practice properties of initialized filters may differ for different realizations. Therefore, it is interesting and important to experimentally and theoretically study the *actual output variance* for each realization of initialized filter.

Formulation for Computing Output Variance. A convolutional layer used in CNN contains a set of learnable filters (also called *kernels*) [8]. During the forward pass, the kernel moves in a sliding-window manner across the input and computes a weighted sum of the local input data and the kernel. This procedure results in a so-called activation map comprising all the local results computed at every sliding movement of the kernel. Now assume that the kernel contains N scalars denoted by $\mathrm{W} = (w_1, w_2, ..., w_N)$, then the local input data involved in the computation also contains N scalars, denoted by $\mathrm{X} = (x_1, x_2, ..., x_N)$. It is easy to see that the local output y is simply the dot product of W and X, as:

$$y = \langle \mathrm{W}, \mathrm{X} \rangle = \sum_{i=1}^{N} w_i.x_i. \tag{1}$$

In Xavier [7] and Castillo [2], both w_i and x_i are assumed as independent random variables. In this paper, we take a *new and more practical* point of view. Since we focus on a proper scaling of a given kernel, we assume that the kernel elements w_i are *known constants*, which can be generated by any initialization algorithm. In addition, we do not consider x_i as independent; instead, we consider them as *mutually correlated* random variables reflecting the natural image statistics [11]. With these assumptions and based on the property of variance of weighted sum of variables, we can compute the variance of the output y as

$$\begin{aligned} \mathrm{Var}(y) &= \mathrm{Var}\left(\sum_{i=1}^{N} w_i.x_i\right) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \mathrm{Cov}(x_i, x_j) \\ &= \sum_{i=1}^{N} w_i^2 \mathrm{Var}(x_i) + 2\sum_{1\le i} \sum_{<j\le N} w_i w_j \mathrm{Cov}(x_i, x_j). \end{aligned} \tag{2}$$

Table 1. Considered manipulation operations and their parameters.

Median filtering	$FilterSize = 3$
Gaussian blurring	$StandardDeviation = 0.5$, $FilterSize = 3$
Additive Gaussian noise	$StandardDeviation = 1.1$
Resampling	$ScalingFactor \in \{0.9, 1.1\}$
JPEG compression	$QualityFactor \in \{90, 91, ..., 100\}$

The last expression just divides all the relevant terms into two groups: variance terms and covariance terms of the input signal components $(x_1, x_2, ..., x_N)$.

Furthermore, it is well-known that natural images have approximate *translation invariance* [11], implying that $\mathrm{Var}(x_i), i = 1, 2, ..., N$ are almost identical. In addition, the neighboring pixels are usually highly-correlated [11], which means that $\mathrm{Cov}(x_i, x_j)$ is close to $\mathrm{Var}(x_i)$. We approximate $\mathrm{Var}(x_i)$ by $\mathrm{Var}(x)$, the overall variance of input. Then we have the following approximation of Eq. (2):

$$\mathrm{Var}(y) \approx \mathrm{Var}(x) \left(\sum_{i=1}^{N} w_i^2 + 2 \sum_{1 \le i} \sum_{< j \le N} w_i w_j C_{ij} \right), \tag{3}$$

with $C_{ij} = \mathrm{Cov}(x_i, x_j)/\mathrm{Var}(x)$ which are in practice smaller than but very close to 1 for small natural image patches due to high correlation of neighboring pixels. Experimentally the above equation approximates very well the output variance. It also helps us to understand the output variance of popular initialized filters used for manipulation detection, as presented in the remaining of this section.

3.2 Convolutional Filter Initialized with High-Pass Filter

We first consider convolutional filter initialized as each of the 30 SRM filters[1] of shape 5×5 (so here $N = 25$). In order to test on real data for convolutional filter, we take as input 64×64 grayscale image patches generated from the Dresden database [6]. The image manipulation operations that we want to detect are listed in Table 1. We then compute the variance of output of each SRM filter by two different methods: the first one with Eq. (3) and the second one with actual convolution between the input and the filter. Hereafter, we call the first as *covariance-based method* because Eq. (3) is mainly based on the covariance terms $\mathrm{Cov}(x_i, x_j)$ of the input signal components $(x_1, x_2, ..., x_N)$, and we call the second one as *convolution-based method*. For the first method, the covariance terms are estimated from 5×5 small patches (same size as SRM filters) which are randomly extracted from the aforementioned 64×64 Dresden image patches.

The results of $\mathrm{Var}(y)/\mathrm{Var}(x)$, *i.e.*, the ratio of output and input variance, are shown in Fig. 1. We can see that the amplitude of $\mathrm{Var}(y)/\mathrm{Var}(x)$ is very

[1] The 30 SRM filters can be found in the `class` of `SrmFiller`, starting from line 347 of this webpage https://github.com/tansq/WISERNet/blob/master/filler.hpp.

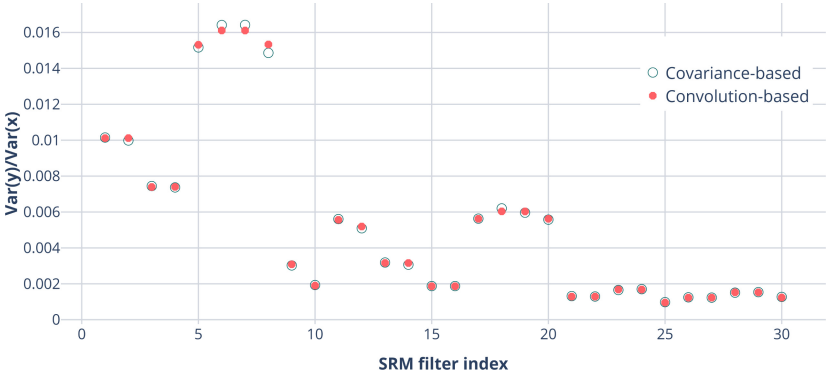

Fig. 1. The value of $\mathrm{Var}(y)/\mathrm{Var}(x)$ for each of the 30 SRM filters obtained by using the covariance-based method of Eq. (3) and the convolution-based method.

small for all 30 SRM filters, which lies basically in a range from 0 to 0.016 with a mean of about 0.005. The output of majority of SRM filters has a variance smaller than 1% of input variance, reflecting the signal shrinkage. It can also be observed that the two methods to obtain output variance give very close results of $\mathrm{Var}(y)/\mathrm{Var}(x)$, implying the coherence of the prediction by Eq. (3) with the practical convolution results.

In order to understand the small output variance for SRM filters, we start from one important property of these high-pass filters. Like many high-pass filters, *e.g.*, Laplacian filter, the sum of all filter elements is equal to 0 for all 30 SRM filters of shape 5×5 (*i.e.*, $N = 25$), which means that we have $\sum_{i=1}^{N} w_i = 0$ (*cf.*, link in footnote (See footnote 1)). It is then easy to deduce that $\sum_{j=1}^{N} w_j . \sum_{i=1}^{N} w_i = \sum_{i=1}^{N} \sum_{j=1}^{N} w_i . w_j = 0$. By dividing the $w_i . w_j$ terms into two groups, we obtain

$$\sum_{i=1}^{N} w_i^2 + 2 \sum_{1 \le i} \sum_{< j \le N} w_i w_j = 0. \tag{4}$$

The left-hand side of the above equation is almost same as the term in the bracket of Eq. (3), except that in the above Eq. (4) we replace C_{ij} by 1. As mentioned earlier, for small natural image patches, we have the property that C_{ij} are usually smaller than but very close to 1. This is verified by experiments on Dresden database where the minimum C_{ij} value is 0.9573 for 5×5 small patches. Not surprisingly, this minimum C_{ij} value is attained between two pixel positions which are the farthest from each other within the 5×5 small patch. From the above analysis we can see that the term in the bracket of Eq. (3) is close to 0, which results in a small value of output variance $\mathrm{Var}(y)$ for 30 SRM filters. This intuitively explains the small $\mathrm{Var}(y)/\mathrm{Var}(x)$ values shown in Fig. 1.

Regarding the high-pass filter initialized by Bayar [1] and Castillo [2], we simulated 10,000 filters with both algorithms and calculated the variance of

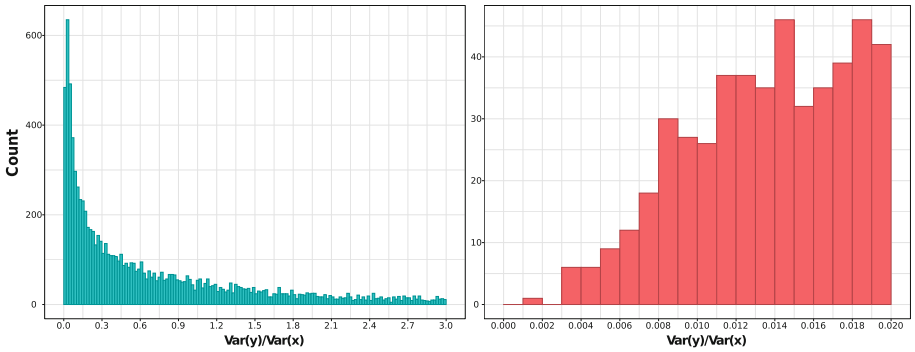

Fig. 2. Two histograms of occurrences of output-input variance ratio $\mathrm{Var}(y)/\mathrm{Var}(x)$ for 10,000 Xavier filters. Please refer to main text for detailed explanation.

output of simulated filters. We also observe very small values of the ratio of output-input variance, with 0.005 and 0.006 as mean value of $\mathrm{Var}(y)/\mathrm{Var}(x)$, respectively for Bayar and Castillo. Due to space limit, we do not show detailed results, but these small mean values further confirm the behavior that high-pass filters result in small output variance with significant signal shrinkage.

3.3 Convolutional Filter with Xavier Initialization

With curiosity, we also carry out studies for Xavier [7] which generates 5×5 filters filled with pseudo-random samples drawn from a zero-mean uniform distribution. We created 10,000 Xavier filters using PyTorch and Fig. 2 shows two histograms of occurrences of output-input variance ratio, *i.e.*, $\mathrm{Var}(y)/\mathrm{Var}(x)$: the left one is for the range of 0 to 3 with a bin width of 0.02, while the right one shows detailed occurrences for the first bin of the left histogram for the range of 0 to 0.02 with a bin width of 0.001. We computed the mean value of $\mathrm{Var}(y)/\mathrm{Var}(x)$ for the 10,000 simulations of Xavier and found that the mean is close to 1 (desired value of Xavier); this is because of a long tail of big values that we do not completely show in Fig. 2. The left histogram of Fig. 2 does not have a peak around 1; instead, the highest occurrences occur near 0. This is a little surprising yet understandable according to Eq. (3). In fact, the elements of Xavier filter are drawn from a zero-mean distribution, so the bracket term in Eq. (3) tends to have a small value. However, due to numerical sampling and in particular considering the relatively low number of 25 pseudo-random samples (for 5×5 filter), it is possible for the bracket term to take big values in certain simulations. Experimentally this bracket term of Eq. (3) can be as big as 13 for some Xavier filters. In addition, from the right histogram of Fig. 2, for Xavier the occurrences of $\mathrm{Var}(y)/\mathrm{Var}(x)$ being very small values, *i.e.*, less than 0.01, is very low: 109 out of 10,000 simulations (*i.e.*, around 1% probability). In contrast, the majority of this variance ratio is less than 0.01 for high-pass filters as presented in last subsection. We guess that it is still related to the numerical sampling of Xavier

because it can be rare to have 25 pseudo-random samples which sum up to a value extremely close to 0.

4 Scaling of Convolutional Filter

From results and analysis in Sect. 3, we can see that the variance of output of convolutional filter initialized by popular algorithms can be significantly smaller than the variance of input. This is particularly true for high-pass filter: the ratio of output-input variance $\mathrm{Var}(y)/\mathrm{Var}(x)$ is usually smaller than 0.01. The output signal after convolution operation substantially shrinks. This can be detrimental to the training of CNN, and as shown later in Sect. 5 the CNN training sometimes fails in such situations.

Using a data-dependent approach (*i.e.*, dependent on input data), we propose a simple yet effective scaling of the first-layer convolutional filter. The idea is to keep the variance stable after scaling for the input and output of any given filter generated by popular initialization algorithms. Corresponding to the two methods to compute output variance in Sect. 3, we propose two different ways to calculate the scaling factor s, as presented below. After obtaining the scaling factor, the elements of the given filter $\mathrm{W} = (w_1, w_2, ..., w_N)$ are properly scaled as $\widetilde{\mathrm{W}} = s.\mathrm{W}$. We then initialize the first-layer filter with the scaled version $\widetilde{\mathrm{W}}$.

Covariance-Based Method. From Eq. (3), it can be seen that in order to make $\mathrm{Var}(y)$ and $\mathrm{Var}(x)$ approximately equal to each other, we need to compensate for the effect of the term in the bracket. So the scaling factor is computed as:

$$s = \sqrt{1 \Big/ \left(\sum_{i=1}^{N} w_i^2 + 2 \sum_{1 \le i} \sum_{< j \le N} w_i w_j C_{ij} \right)}. \tag{5}$$

In practice, we take random small patches of the same shape of the convolutional filter to be scaled (*e.g.*, 5×5) from a small portion of the training data. We then estimate the variance and covariance terms on these small patches to obtain the values of $C_{ij} = \mathrm{Cov}(x_i, x_j)/\mathrm{Var}(x)$. Afterwards the scaling factor s is computed by using Eq. (5), and at last we obtain the scaled version $\widetilde{\mathrm{W}}$ of any given filter W from the considered four initialization algorithms.

Convolution-Based Method. This is a straightforward approach. The output $\widehat{y}$ is computed, for a small portion of the training data $\widehat{x}$ as input, by carrying out the convolution operation. The scaling factor is simply calculated as

$$s = \sqrt{\mathrm{Var}(\widehat{x})/\mathrm{Var}(\widehat{y})}. \tag{6}$$

From a practical point of view, the covariance-based method might be a slightly better option than the convolution-based method mainly because of its higher flexibility. In fact, for the covariance-based method, the computation of the variance and covariance terms of the input can be performed *only once for*

any number of filters for which we want to scale. By contrast, the convolution-based method has to be rerun every time we have a new filter to analyze. Nevertheless, it is worth mentioning that both methods are experimentally quick enough to be used in CNN initialization. The running time is about several seconds, as presented in the next paragraph.

According to our experiments, for a training set of about 100,000 images of 64×64 pixels from all classes, taking 10% of the training data for the convolution-base method and 10 small patches (*e.g.*, of 5×5 pixels) per image of the 10% training data for the covariance-based method, we achieve a good trade-off between computation time and stability of the result. Using more training data has very small impact on the obtained scaling factor. Even using 100% of the training set results in a change smaller than 0.1%. The amount of time to calculate the scaling factor is less than 3 s per first-layer filter for both methods, on a desktop with Intel® Xeon E5-2640 CPU and Nvidia® 1080 Ti GPU (covariance-based method on CPU and convolution-based method on GPU). This is run for one time before the CNN training. The computation time increases very slowly when having more filters for the covariance-based method, because as mentioned above the variance and covariance terms can be reused. We believe that the computation time of scaling factor is negligible when compared to the typical time required to train a CNN model.

5 Experimental Results

Several experiments are performed in order to test and show the efficiency of our proposed scaling. These experiments consider the four filter initialization algorithms mentioned earlier, two CNN architectures (CNN of Bayar and Stamm [1] and a smaller CNN without fully-connected layer designed by ourselves), and two forensic problems (a multi-class problem of detecting a group of manipulation operations and a binary problem of detecting JPEG compression of high quality factor). For the multi-class problem, we also consider a different number of filters used in the first layer of the CNN of Bayar and Stamm [1]. The implementation and experiments were conducted using PyTorch v1.4.0 with Nvidia® 1080 Ti GPU. The experimental data was created from the Dresden database [6]. Full-resolution Dresden images are split for training, validation and testing with ratio of 3:1:1 and converted to grayscale. Patches of 64×64 pixels were randomly extracted from full-resolution grayscale Dresden images. This relatively small size of image patches makes the forensic problems more challenging.

5.1 Multi-class Problem with CNN of Bayar and Stamm [1]

We first consider the multi-class problem of classifying six different kinds of image patches: the original patches and the five classes of manipulated patches as explained in Table 1. The parameters for the resampling and JPEG compression manipulations are taken randomly from the specified sets in Table 1. The total number of patches in training set is 100,000 (≈16,667 patches per class), while

the number of patches in testing set is 32,000 (≈5,333 patches per class). The number of training and testing samples is same as in [1]. It is worth mentioning that the manipulations and their parameters in this paper are borrowed from [2] and more challenging than those in [1]. The patch size is also smaller than [1]: our patches are of 64×64 pixels, while [1] mainly considers 256×256 patches.

Table 2. Test accuracy for the multi-class forensic problem (in %, average of 5 runs). The experiments were performed with four initialization algorithms and their scaled versions for first-layer filters of the CNN of Bayar and Stamm [1]. In parentheses is the improvement of scaled version compared to the corresponding original version.

Initialization	Original version	Convolution-based scaling	Covariance-based scaling
Bayar [1] A.	94.19	96.04 (+1.85)	96.02 (+1.83)
Bayar [1] B.		96.15 (+1.96)	96.22 (+2.03)
Castillo [2]	93.71	96.45 (+2.74)	96.42 (+2.71)
SRM [5]	94.39	96.54 (+2.15)	96.55 (+2.16)
Xavier [7]	93.48	94.61 (+1.13)	94.71 (+1.23)

We use the successful CNN architecture of Bayar and Stamm [1] in this set of experiments and initialize the three filters in the CNN's first layer with four different algorithms: Bayar [1], SRM [5], Castillo [2], and Xavier [7]. We carry out 5 runs for each algorithm and the corresponding two scaled versions. For SRM, for each run we randomly select 3 filters from the pool of 30 SRM filters. We compare each original initialization algorithm with their scaled versions obtained with the covariance-based method and the convolution-based method presented in Sect. 4. For fair comparisons, we make sure that for each run the scaled versions share the same "base filters" of the original version before performing scaling. We follow exactly the same training procedure described in [1], including number of epochs, optimization algorithm, learning rate schedule, *etc.*

For Bayar algorithm [1], we have tested two variants of the scaling of the first-layer filters. The first one ("Bayar A.") follows closely the idea of Bayar's original constrained training strategy: we carry out scaling of the normalized high-pass filter at the beginning of each forward pass (please refer to the second last paragraph of Sect. 2 for detail of the normalization procedure proposed in [1]). The second variant ("Bayar B.") is computationally cheaper and less complex: the scaling of normalized high-pass filter is only performed in the initialization, and we no longer impose normalization constraint during training. Our intuition behind the second variant is that with a proper scaling of initialized filters even a free training without the constraint of [1] may provide satisfying performance.

The detection performances in terms of test accuracy (*i.e.*, classification accuracy on testing set) for this multi-class problem are presented in Table 2. The reported results are average of 5 runs with randomness, *e.g.*, different first-layer

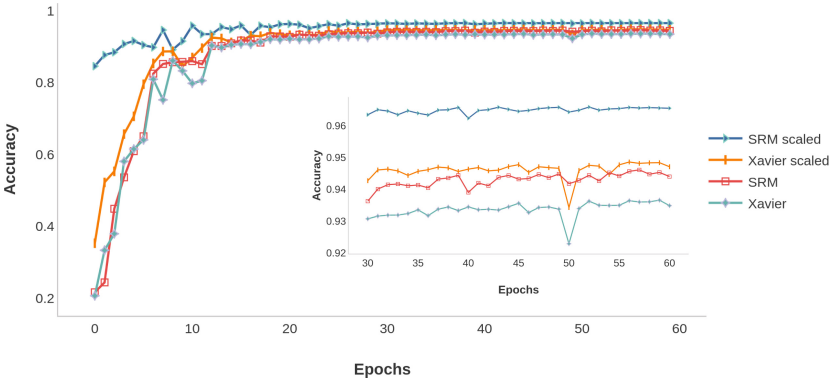

Fig. 3. Curves of test accuracy (average of 5 runs) for the multi-class forensic problem, during the whole 60 training epochs of the CNN of [1]. The curves are for SRM and Xavier, original version and scaled version by the covariance-based method.

"base filters". However, for each run, the "base filters" are the same for the original and scaled versions: original version direct uses these filters, while scaled versions apply proper scaling on the "base filters" and then use the scaled ones.

We can see from Table 2 that the test accuracy of all the initialization algorithms is consistently and noticeably improved after scaling. Improvement of at least 1.13% and as high as 2.74% is obtained. We also observe that the results of the two scaling methods are very close to each other. We checked the computed scaling factors and found that they are indeed almost identical for the two methods. Furthermore, for the two scaling variants of Bayar [1], variant B gives slightly better results, which is also computationally cheaper as it only performs scaling in initialization without enforcing any constraint during CNN training. This implies that with a good initialization after proper scaling, it might not be necessary to impose training constraint. It is worth mentioning that the results of Bayar in Table 2 are in general lower than those reported in [1] because we now consider a more challenging forensic problem with more difficult manipulations and on smaller patches. The results of Castillo in Table 2 are better than those presented in [2]. This may be due to the differences in the number of training epochs (we follow [1] and train with more epochs) and in the adopted optimization algorithm and learning schedule. In addition, the three kinds of high-pass filters (especially SRM) indeed outperform Xavier, before and after scaling. This demonstrates the difference between forensics and computer vision tasks. Nevertheless, the performance of Xavier is also improved after scaling because as analyzed in Sect. 3.3 Xavier can also result in small variance of output. In fact, for Xavier the probability to have $\mathrm{Var}(y)$ smaller than half of $\mathrm{Var}(x)$ is about 52.20% in our 10,000 simulations.

We also observe that the proposed scaling helps to have quicker increase of forensic performance during CNN training. We show in Fig. 3 curves of test accuracy of SRM and Xavier (average of 5 runs), before and after the covariance-based scaling. It can be observed that the convergence speed is considerably

Table 3. Test accuracy for the multi-class forensic problem (in %, average of 5 runs). We still use the CNN of [1] but change the number of first-layer filters from 3 to 30.

Initialization	Original version	Convolution-based scaling	Covariance-based scaling
Bayar [1] B	94.91	96.11 (+1.20)	96.04 (+1.13)
Castillo [2]	94.11	96.31 (+2.20)	96.32 (+2.21)
SRM [5]	94.37	96.51 (+2.14)	96.49 (+2.12)
Xavier [7]	91.80	96.03 (+4.23)	96.02 (+4.22)

improved for SRM. For both algorithms, the curve of scaled version is always above that of original version during the whole 60 epochs. It is also interesting to notice that the scaled Xavier performs slightly better than the original SRM.

With 30 Filters at First Layer. Next, we present results for the same multi-class problem while changing the number of filters in the first layer of the CNN of [1] to 30. We make this change for two reasons: first, to test our approach with a different number of filters in the first layer; and second, to use all the 30 SRM filters which is a common practice in image forensics, *e.g.*, for detecting splicing and copy-move forgeries [10]. For this scenario we still test the four initialization algorithms but only use variant B for scaled Bayar as it proved to obtain slightly better results while being computationally cheaper. The results are presented in Table 3. Again, we observe that scaling the filters with any of the two methods leads to consistently better test accuracy, with an improvement ranging from 1.13% to 4.23%. We notice from Tables 2 and 3 that after increasing the number of first-layer filters, 1) the original version of high-pass initialization (Bayar, Castillo and SRM) has slightly improved or comparable performance while the accuracy of Xavier decreases; and 2) the scaled version of Bayar, Castillo and SRM has comparable performance with the case of 3 filters while Xavier has noticeable improvement. We guess the reason for the good performance of scaled Xavier may be that with 30 filters there is more chance to have a very good filter which after scaling can improve the result. Understanding these observations is not the focus of our paper, and we plan to conduct further analysis in the future.

5.2 JPEG Binary Problem with CNN of Bayar and Stamm [1]

We notice in the multi-class problem that JPEG compression is the most difficult manipulation to detect. In this section we consider the binary classification between original patches and JPEG compressed patches with parameters in Table 1 (*i.e.*, very high quality factor between 90 and 100). This allows us to test the proposed scaling on a different challenging forensic problem. We use the CNN of Bayar and Stamm [1]. The number of training and testing patches per class is the same as in last subsection. All CNN training settings are kept unchanged.

For this binary problem, we consider two initialization algorithms of Bayar [1] and SRM [5], original and scaled versions (variant B for scaled Bayar). Table 4 presents the obtained results (average of 5 runs). This challenging problem makes the original version of both Bayar and SRM struggle to achieve a good performance. Especially, training of SRM can occasionally fail, leading to accuracy close to random guess. Much better average test accuracy is achieved by scaled versions. For SRM [5], a boost of more than 14% is obtained with scaling.

Table 4. Test accuracy for the binary JPEG forensic problem (in %, average of 5 runs). The experiments were performed with Bayar and SRM, original and scaled versions (variant B for scaled Bayar), on the CNN of [1].

Initialization	Original version	Convolution-based scaling	Covariance-based scaling
Bayar [1] B.	88.27	90.80 (+2.53)	90.80 (+2.53)
SRM [5]	78.24	92.33 (+14.09)	92.44 (+14.20)

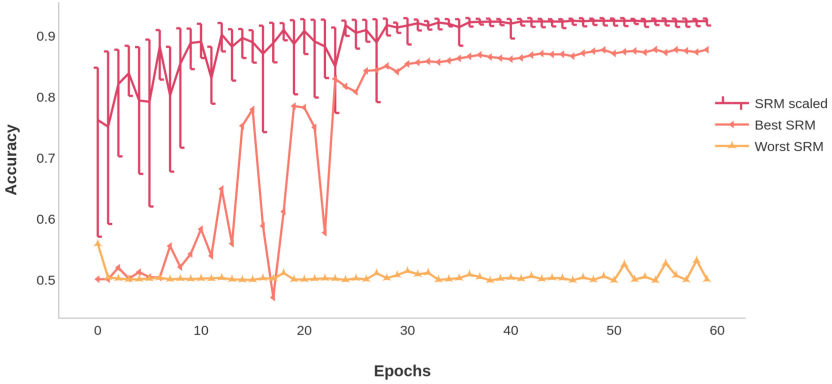

Fig. 4. Curves of test accuracy for the JPEG binary forensic problem: scaled version of SRM (average of 5 runs) with bars of maximum and minimum accuracy at each epoch among 5 runs; and the best run and the worst run of original version of SRM.

We show in Fig. 4 some curves of test accuracy for scaled (covariance-based) and original SRM [5]. The curve of scaled version shows the maximum and minimum test accuracy together with the average at each epoch among the 5 runs. For the original version of SRM we can have very different results. Therefore, we show the best and the worst curves of test accuracy among all the 5 runs. As we can see the worst case does not improve during the whole procedure and the test accuracy remains close to 50%. The difference may come from the randomly selected three first-layer SRM filters in each run (certain SRM filters perform worse than others according to our observation). We would like to mention that for each run, although we select randomly different SRM filters, the

Table 5. Test accuracy for the multi-class and binary problems with our proposed smaller CNN without fully-connected layer (in %, average of 5 runs). The columns of "Bayar" and "SRM" present results of original version. "Scaling-conv" and "Scaling-cov" represent respectively convolution-based and covariance-based scaling method.

Problem	Bayar	Scaling-conv	Scaling-cov	SRM	Scaling-conv	Scaling-cov
Multi-class	95.24	96.17(+0.93)	96.18(+0.94)	95.72	97.06(+1.34)	97.09(+1.37)
Binary	90.56	93.72(+3.16)	93.74(+3.18)	89.85	95.11(+5.26)	95.12(+5.27)

same selected filters are used to carry out comparisons between the original and scaled versions. Therefore, even for filters that result in bad performance for the original version, we can obtain a much better performance after scaling them.

5.3 Multi-class and Binary Problems on a Different Smaller CNN

We then test both the multi-class and JPEG binary problems on a different CNN designed by ourselves. We first describe the architecture of this smaller network. Let Ck(M or A) denote a Convolutional-BatchNorm-Tanh(-MaxPool or -AveragePool) layer with k filters. For the first layer we use Hk which denotes a Convolutional layer with k filters. The architecture of our smaller CNN is H3-C40M-C25M-C20M-C15M-C6A. The first four layers have a kernel size of 5×5 while for the last two layers the kernel size is 1×1. All convolutional stride size is 1. The first layer and the last two layers do not have zero-padding, for the other layers the padding size is 2. This is a network without fully-connected layer. To compare with, the architecture proposed by Bayar and Stamm [1] is H3-C96M-C64M-C64M-C128A-F200-F200-F6, where Fk denotes a fully-connected layer with k neurons and Tanh. The number of learnable parameters of the CNN of [1] is about 337K, while our smaller CNN has about 41K parameters.

Using our smaller CNN, we test both the multi-class and the binary problems on a different CNN architecture. All the data preparation and experimental setting are the same as those described in Sects. 5.1 and 5.2. For this set of experiments, we compare the original and scaled versions of Bayar [1] and SRM [5] (variant B for scaled Bayar). Table 5 presents the obtained results. We can see that in all cases the scaled version leads to improved performance compared to the original version. The improvement of test accuracy goes from 0.93% for multi-class problem with Bayar, to 5.27% for binary problem with SRM.

Our objective in this subsection is to show that with a different CNN, our proposed scaling can still reliably improve the performance for different initialization algorithms and forensic problems. Meanwhile, it can be noticed that performance is better with our smaller CNN when compared to the network of [1]. The understanding of this point is beyond the scope of this paper. Our guess is that the forensic problems and/or the amount of data cope better with the smaller CNN's size (less parameters) and architecture (only comprising convolutional layers without fully-connected layer). The thorough analysis and understanding of the relationships between these factors is one part of our future work.

6 Conclusion

We propose a new and effective scaling approach for adjusting first-layer filters of CNNs used for image manipulation detection. The proposed scaling is computationally efficient and data-dependent (*i.e.*, scaling factor dependent on the input). We also present theoretical and experimental studies which help to understand why the ratio of output-input variance for first-layer convolutional filter can be a (very) small value. Experimental results, with different CNNs, filter initialization algorithms and forensic problems, show that our proposed scaling can consistently improve performance of CNN-based image manipulation detection. Although practically and intuitively we can understand the effectiveness of the proposed filter scaling operation, a rigorous theoretical analysis would be necessary to explain the observed performance improvement. We would like to conduct research on a possible scaling of deeper layers and to extend experiments to more CNNs (*e.g.*, XceptionNet and ResNet), training settings and multimedia security problems. It is also interesting to study the impact of amount of training data and CNN architecture on the forensic performance. One limitation of our work is that we only consider image manipulation detection under ideal laboratory conditions. We plan to carry out relevant studies for more challenging and practical forensic problems, *e.g.*, the detection of malicious and realistic image forgeries and image forensics with mismatch between training and testing data.

Acknowledgments. This work is partially funded by the French National Research Agency (DEFALS ANR-16-DEFA-0003, ANR-15-IDEX-02) and the Mexican National Council of Science and Technology (CONACYT).

References

1. Bayar, B., Stamm, M.C.: Constrained convolutional neural networks: a new approach towards general purpose image manipulation detection. IEEE Trans. Inf. Forensics Secur. **13**(11), 2691–2706 (2018)
2. Castillo Camacho, I., Wang, K.: A simple and effective initialization of CNN for forensics of image processing operations. In: Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, Paris, France, pp. 107–112 (2019)
3. Chen, B., Li, H., Luo, W., Huang, J.: Image processing operations identification via convolutional neural network. Sci. China Inf. Sci. **63**(3) (2020). https://doi.org/10.1007/s11432-018-9492-6
4. Chen, J., Kang, X., Liu, Y., Wang, Z.J.: Median filtering forensics based on convolutional neural networks. IEEE Sig. Process. Lett. **22**(11), 1849–1853 (2015)
5. Fridrich, J., Kodovský, J.: Rich models for steganalysis of digital images. IEEE Trans. Inf. Forensics Secur. **7**(3), 868–882 (2012)
6. Gloe, T., Böhme, R.: The Dresden image database for benchmarking digital image forensics. In: Proceedings of the ACM Symposium on Applied Computing, Sierre, Switzerland, pp. 1584–1590 (2010)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, pp. 249–256 (2010)

8. Li, F.F., Johnson, J., Yeung, S.: Neural networks part 2: setting up the data and the loss (2018). Course notes of Stanford University "CS231n: Convolutional Neural Networks for Visual Recognition". http://cs231n.github.io/neural-networks-2/. Accessed 16 Dec 2020
9. Li, H., Huang, J.: Localization of deep inpainting using high-pass fully convolutional network. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 8301–8310 (2019)
10. Liu, Y., Guan, Q., Zhao, X., Cao, Y.: Image forgery localization based on multi-scale convolutional neural networks. In: Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, Innsbruck, Austria, pp. 85–90 (2018)
11. Simoncelli, E.P., Olshausen, B.A.: Natural image statistics and neural representation. Ann. Rev. Neurosci. **24**(1), 1193–1216 (2001)