Convolutional Neural Network Initialization Approaches for Image Manipulation Detection

Ivan Castillo Camacho[a], Kai Wang[a,*]

[a] *Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France*

Abstract

Nowadays it is common to see image forgeries on almost every media in both professional and personal contexts. From visual retouches to deliberate fake scenes, the technology used to create image forgeries gets easier to use for all users. At the same time, different techniques to assess the authenticity of the content of an image have appeared by taking advantage of the deep learning paradigm. In this paper we propose two initialization approaches for Convolutional Neural Networks (CNNs) used for the detection of image manipulation operations. We focus on the variance stability for the output of a convolutional filter in CNN. Our first proposal is a scaling approach for first-layer convolutional kernels which can cope well with filters generated by different algorithms. Our second proposal is a random high-pass filter initialization approach for CNN's first convolutional layer. The first proposal explicitly computes simple statistical properties of the input signal, while the second approach incorporates the consideration of input statistics in the filter derivation without the need of carrying out explicit computation on the input. Experimental results show the utility of both approaches with improved performance in different image manipulation detection problems and on different CNN architectures.

Keywords: Image forensics, Neural network, Image manipulation detection, Convolutional filter, Variance stability

1. Introduction

Manipulating an image is no longer a task that requires high-level skills. From easy-to-use software tools to popular smartphone applications, image forgeries are within reach with fast and realistic results. One main objective of image forensics research is to develop a collection of techniques for the analysis of traces left by image forgeries in an attempt to verify the authenticity of image [1, 2]. The availability of these techniques is of paramount importance because fake

*Corresponding author

Email addresses: `ivan.castillo-camacho@gipsa-lab.grenoble-inp.fr` (Ivan Castillo Camacho), `kai.wang@gipsa-lab.grenoble-inp.fr` (Kai Wang)

images can have serious consequences if used to mislead public opinion when an important decision is at stake, *e.g.*, biasing political campaigns [3].

With the help of deep learning and more specifically CNN (Convolutional Neural Network), we can create models for detecting basic image manipulation operations which are typically used during the creation of a real image forgery. Some examples of these manipulation operations are JPEG (Joint Photographic Experts Group) compression, median filtering, noise addition, *etc.* Regarding CNN architectures in the image forensics field, the first layer has significant importance because this is the layer receiving the input data [4, 5, 6], and accordingly it has a big influence on the final performance of the model. Majority of image manipulation detection works rely on the usage of high-pass filters which allow us to study the subtle traces left by manipulation operations. Nevertheless, as shown later in this paper, the output variance of this kind of high-pass filters shrinks in a noticeable manner, which in practice limits the final performance of the network. Using natural image statistics, in this work we provide an intuitive explanation on the signal shrinkage.

The main contribution of our work is the design and implementation of two effective approaches for initializing the first layer of a CNN for the detection of image manipulation operations. The first data-dependent approach takes into account the statistical properties of the training data to properly scale a given convolutional filter. We also propose a second, so-called data-independent approach of random high-pass filter initialization. Different from the first approach, the second approach does not require explicit computation on the input signal. We show on several classification problems and different CNN architectures that both approaches lead to an overall better forensic performance when compared with existing algorithms.

The remainder of this paper is organized as follows. In Section 2, a brief review of related works is provided. In Section 3 we present theoretical and experimental studies regarding the variance stability of input and output signals of first-layer convolutional filters in a CNN, which provide useful insights for the work described in the remaining sections. Our first proposal of data-dependent filter scaling approach is presented in Section 4, followed by the second proposal of a random high-pass initialization approach in Section 5. Experimental results of both proposals are given in Section 6. At last, we conclude and suggest some future working directions in Section 7. The first data-dependent scaling approach was previously presented in a conference paper [7]. Compared with the conference version, a new data-independent approach (Section 5) is proposed in this paper and more experimental results are presented (Section 6). Another new content is a theoretical explanation (in Section 3.3 of this paper) about the somewhat surprising results of a popular CNN initialization algorithm.

2. Related Work

As mentioned in the last section, in this paper we focus on the detection of basic image manipulation operations which are typically involved in the creation of a real image forgery, including for example median filtering, resampling, noise addition, and so on. These image manipulation operations slightly change some statistics of an image and in general leave subtle traces in the image's high-frequency components. At the early stage of the research on image manipulation detection, researchers were interested in designing handcrafted features for the detection of each kind of manipulation, *e.g.*, median filtering [8, 9], JPEG compression [10, 11] and resampling [12]. Research interests were then shifted from the aforementioned *targeted* methods to the *general-purpose* methods which can allow us to detect different kinds of manipulations with a same feature or a same model. Existing general-purpose detection methods make use of steganalysis features [13] or models of local image statistics [14].

Recent works have taken advantage of the deep learning paradigm in the form of CNNs. The big advantage is their useful feature learning capability in a data-driven fashion, without the need of creating handcrafted features which are in general difficult to design and sometimes suboptimal. Although initially CNNs were mainly used in the computer vision field, it did not take long for image forensics researchers to test their performances. The direct use of CNNs in image forensic problems did not give results as good as expected. One fundamental point is that in forensic problems we focus on the subtle differences usually in image's high-frequency components but not on the image's semantic content. Specifically, traditional CNN initializations such as the widely used Xavier initialization [15] have limited performance on image forensic tasks, and special initializations are required to cope better with forensic problems [5, 6].

As mentioned earlier, in image forensics we are usually interested in exposing traces left in the high-frequency components of an image. Therefore, it is in practice beneficial to use high-pass filters at the first layer of CNN, so as to extract the relevant high-frequency information for further analysis. One effective way is to initialize the first-layer convolutional filters with the well-known SRM (Spatial Rich Model) filters. SRM filters were initially designed to extract discriminative steganalysis features in the high-pass filtered domain [16]. Recently they have been successfully used for first-layer initialization of CNNs designed for solving image forensic problems, including the detection of image manipulation operations [17] and other tasks [18, 19].

A new constrained CNN was proposed by Bayar and Stamm in [6], for the detection of image manipulations. A constraint is enforced on the first-layer convolutional filters, to ensure that the network learns high-pass filters at the first layer. This constraint is realized by an additional normalization procedure of filter coefficients applied at each step of network training. After the normalization the filter's center coefficient is set to be -1, while the sum of all other coefficients is equal to 1. This forms a high-pass filter for the extraction of discriminative information for manipulation detection.

Recently, an extension of the well-known Xavier initialization [15] was proposed by Castillo Camacho and Wang [20]. The Xavier initialization is broadly used in the computer vision field to generate random filters for CNN initialization. The authors of [20] extended this algorithm so that the new algorithm can generate high-pass initialized convolutional filters which are more suitable for the image manipulation detection task than the conventional Xavier filters.

In this paper, we consider and compare our proposed approaches with four existing algorithms for first-layer initialization of CNNs used to detect image manipulations, all briefly mentioned above and hereafter noted by Xavier [15], SRM [16], Bayar [6], and Castillo [20]. The first one [15] is a classical algorithm from the computer vision community, while the other three algorithms [16, 6, 20] all use high-pass filters for the initialization of CNN's first convolution layer. In the following, we first point out the potential shortcomings of the four algorithms and show that they in general result in output signal shrinkage after convolution. We demonstrate that our first proposal, *i.e.*, the data-dependent scaling approach, can effectively solve this signal shrinkage problem by carrying out a proper scaling of filters generated by any of the four algorithms. Additionally, our second proposal of random high-pass initialization shows better performance than all these four algorithms in their original non-scaled version.

3. Studies on the Output Variance of a Convolutional Filter

Keeping a stable data flow in a CNN is helpful for the training of network, and the data flow stability is usually measured by the stability of the *variance* of the signal in CNN [15, 21]. This means that in the ideal case, after passing through a convolutional filter (also called a convolutional *kernel*), the variance of the output signal should be the same as the variance of the filter input. In this section we present our derivation of the variance computation for the output signal of a convolutional filter. The fundamental difference between our derivation and existing algorithms (*e.g.*, Xavier [15] and Castillo [20]) is that we assume realistic statistical properties for the network input of natural image pixels, in particular we do not treat the input pixels as mutually independent as what is assumed in Xavier [15] and Castillo [20]. As shown later in this section, our theoretical derivation establishes a mathematical relationship between the variance of the output of a convolutional filter and the statistical properties of the filter input, *i.e.*, image pixel values. Accordingly, we can then make use of the derived result to explain the output signal shrinkage, *i.e.*, the variance of filter output is (much) smaller than that of filter input, which can be observed for all the four existing initialization algorithms [15, 16, 6, 20]. To our knowledge, the assumption of realistic input statistics and the explanation are new in the literature of deep-learning-based image manipulation detection. For the sake of readability and easy understanding of the mathematical derivation presented in this paper, we provide in Table 1 the main symbols and operators/expressions used in our derivation.

3.1. Motivation and formulation

We notice that the four existing algorithms may have limitations and that they may not be able to ensure the data flow stability before and after the processing of a convolutional filter. In consequence, the variance of convolutional filter output may be very different from the variance of filter input. More precisely, in SRM [16] and Bayar [6] algorithms, there is no explicit modelling of the

Table 1: List of main symbols and operators/expressions used in this paper.

x_i	Representing an input image pixel, a scalar value
w_i	Representing a coefficient of convolutional filter, a scalar value
y	Local output of convolutional filter, a scalar value
X	A group of input pixels, vector of scalar values
W	A group of filter coefficients, vector of scalar values
S	Scaling factor of filter, used and deduced in first approach
C	Filter center coefficient, used and deduced in second approach
$\langle ., . \rangle$	Inner product of two vectors
E(.)	Mathematical expectation of a random variable
Var(.)	Variance of a random variable
Cov(., .)	Covariance of two random variables
Skewness(.)	Skewness of a random variable
$U(p, q)$	Uniform distribution between p and q
C_r^b	Combination notation of "r choose b"

relationship between the input and output of the filter: they simply use hand-crafted SRM filters or ad-hoc normalized filters at the first convolutional layer. Xavier [15] and Castillo [20] take into account the relationship between input and output variances in the mathematical modelling for the filter derivation. However, both algorithms assume that network input is composed of random variables of independent and identical distribution (iid), but it is clear that neighboring pixels in natural images have *strong local correlations* [22, 23]. In consequence, this unrealistic yet popular assumption used by Xavier [15] and Castillo [20] may lead to unexpected or biased results for initialized filters.

In the following, we present our mathematical derivation for the computation of the output variance of first-layer convolutional filter that can be generated by any underlying initialization algorithm. The derived result allows us to accurately predict the *actual variance value* of the output signal. For this purpose, we start from the basic equation of the computation of the local output of a convolutional filter, as given below:

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

where $\mathrm{W} = (w_1, w_2, ..., w_N)$ are coefficients of first-layer convolutional filter, $\mathrm{X} = (x_1, x_2, ..., x_N)$ represent the local input (*i.e.*, a group of pixel values in input image) "seen" by the filter, and y is the local output.

For any given filter generated by an arbitrary algorithm, we consider the filter coefficients w_i as known constants. Moreover, different from Xavier [15] and Castillo [20] which assume that the input x_i's are mutually independent, here we consider x_i's as *mutually dependent* random variables following the classical observation of natural image statistics [22, 23] (*i.e.*, neighboring pixels are highly correlated). The output y is a weighted sum of the filter coefficients and the local input, so by using properties of the variance calculation, we obtain the following equation for the computation of the output variance:

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

It can be seen that the output variance is related to the variance and covariance terms of the components of the input $\mathrm{X} = (x_1, x_2, ..., x_N)$.

We then make some simplifications and approximations for the above Eq. (2), by using one classical and well-known property of natural image statistics [22, 23]: the property of approximate translation invariance which in our case means that the variances of pixels at different positions are almost the same, *i.e.*, $\mathrm{Var}(x_1) \approx \mathrm{Var}(x_2) \approx ... \approx \mathrm{Var}(x_N) \approx \mathrm{Var}(x)$ with $\mathrm{Var}(x)$ the overall variance of the input. After applying this property we obtain the following equation:

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

with $Z_{ij} = \mathrm{Cov}(x_i, x_j)/\mathrm{Var}(x)$. Later in this paper, we will make use of another well-known property of natural image statistics [22, 23]: the property of high correlation between neighboring pixels, which means that within small image patches $\mathrm{Cov}(x_i, x_j)$ and $\mathrm{Var}(x_i)$ have very similar values. Together with the above translation invariance property, we can see that the values of $Z_{ij} = \mathrm{Cov}(x_i, x_j)/\mathrm{Var}(x)$ are very close to 1 for pixels in small patches of natural images. In practice, the above Eq. (3) constitutes an accurate way to predict the output variance for any given convolutional filter, by using simple statistics of the input. Moreover, as presented in the remaining of this section, with this equation and some relevant derivations, we are able to understand the output signal shrinkage which is observed for all the four considered initialization algorithms mentioned earlier [16, 6, 20, 15].

3.2. Convolutional filter initialized with high-pass filters

We first study the output variance of convolutional kernels initialized with high-pass filters generated by the three existing algorithms SRM[1] [16], Bayar [6] and Castillo [20]. The size of convolutional filters is 5×5 (so we have $N = 25$). The input is grayscale patches of size 64×64 from the Dresden database [24]. The considered image manipulations and their parameters are

[1] Please refer to `https://github.com/tansq/WISERNet/blob/master/filler.hpp`, from line 347 for the values of the coefficients of 30 SRM filters.

Table 2: List of image manipulation operations to be detected. For resampling and JPEG compression, manipulation parameter for each sample is randomly selected from the given set. "*std*" means standard deviation and "QF" means quality factor.

Median filtering	$WindowSize = 3$
Gaussian blurring	$std = 0.5$, $WindowSize = 3$
Additive Gaussian noise	$std = 1.1$
Resampling	$Factor \in \{0.9, 1.1\}$
JPEG compression	$QF \in \{90, 91, ..., 100\}$

presented in Table 2. The output variance is computed in two different ways: 1) by using Eq. (3) with our mathematical derivation; 2) by carrying out actual convolution computation on input. We find that these two options result in almost identical results of output variance. The relative difference is typically less than 5%. This proves the validity of the derived Eq. (3) which provides a practical and reliable way to calculate output variance without the need of performing convolution operation.

We compute the ratio of output and input variances, *i.e.*, $\mathrm{Var}(y)/\mathrm{Var}(x)$, for the three high-pass filter initialization algorithms and find that they all result in very small values of this variance ratio, meaning that the output signal significantly shrinks. For the 30 SRM filters [16], the variance ratio values lie within the interval of 0 to 0.02, and the mean value of $\mathrm{Var}(y)/\mathrm{Var}(x)$ is around 0.005 with most of the variance ratio values being less than 1%. For Bayar [6] and Castillo [20], we generate for each algorithm $10,000$ simulated filters. The mean value of $\mathrm{Var}(y)/\mathrm{Var}(x)$ among the $10,000$ simulations is about 0.005 and 0.010, respectively for Bayar [6] and Castillo [20]. Therefore, all the three high-pass filter initialization algorithms have very small variance ratio values, reflecting the drastic signal shrinkage at the output of convolutional kernel.

In the following, we provide an explanation for the observed signal shrinkage mentioned above. This starts with a common characteristic of high-pass filters, *i.e.*, we have $\sum_{i=1}^{N} w_i = 0$ which means that all the coefficients in a high-pass filter sum up to 0. Commonly used high-pass filters, such as gradient-based edge detection filter and Laplacian filter, all have this characteristic. From the property $\sum_{i=1}^{N} w_i = 0$, we can easily 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 reorganizing the $w_i . w_j$ terms into groups of same or different indexes, we obtain the equation below:

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

For SRM filters [16] and the constrained filter of Bayar [6], the filter coefficients sum up exactly to 0. For Castillo [20], the sum of filter coefficients is approximately equal to 0 because theoretically the mathematical expectation of this sum is 0 (details in [20]). The expression on the left side of Eq. (4) above looks very familiar. In fact by replacing Z_{ij} by 1 for the term in parentheses of

Eq. (3), we obtain this same expression. Furthermore, for pixel values within small patches of natural images, the values of $Z_{ij} = \mathrm{Cov}(x_i, x_j)/\mathrm{Var}(x)$, *i.e.*, the covariance of two pixels divided by the overall pixel variance, are close to 1. We verified on Dresden database that indeed Z_{ij} takes values that are very close to 1 in patches of size 5×5 (same size as the convolutional filter), with a minimum value of 0.9573 achieved for a pair of pixels that have the maximum distance within 5×5 patch. In all, we can see that when a high-pass filter (with the property shown in Eq. (4)) takes natural image pixels (with Z_{ij} values close to 1) as input, the expression in the parentheses of Eq. (3) would take a very small value. This expression is a multiplicative factor relating the input and output variances of a convolutional filter as shown in Eq. (3), and this explains why high-pass filters have a small output variance $\mathrm{Var}(y)$ when compared to the input variance $\mathrm{Var}(x)$.

The argument presented in the previous paragraph helps us understand the reason of having a very small output variance for all the three high-pass filter initialization algorithms of SRM [16], Bayar [6] and Castillo [20]. For SRM [16] and Bayar [6] this is mainly caused by the fact that no consideration has been taken on the data flow stability, while for Castillo [20] this is in part caused by the unrealistic iid assumption of input signal components $(x_1, x_2, ..., x_N)$.

3.3. Convolutional filter with Xavier initialization

It is also interesting to study the distribution of the ratio of output and input variances for the conventional Xavier initialization [15] from the computer vision community. We use PyTorch to generate $10,000$ random Xavier filters of size 5×5, and the obtained empirical histogram of the variance ratio $\mathrm{Var}(y)/\mathrm{Var}(x)$ is shown in Figure 1. It can be observed that the histogram has very high occurrences at small values within the range from 0 to 0.3. It is rather surprising to see such a big mass concentration at small values, because the Xavier initialization was designed to maintain the variance stability. Ideally the histogram should have high occurrences around the value 1 at which the output variance is equal to the input variance. However, in practice the output-input variance ratio $\mathrm{Var}(y)/\mathrm{Var}(x)$ has big chance to take small values. In fact, the empirical probability to have $\mathrm{Var}(y) < \frac{1}{2}\mathrm{Var}(x)$ (*i.e.*, output variance is smaller than half of input variance) is about 52%. These simulation results demonstrate that the output signal shrinkage problem also exists for Xavier initialization.

Theoretical explanation for the histogram shape of Xavier simulations

We give further explanation for the shape of the histogram in Figure 1. The objective is to theoretically compute the key statistical properties, *i.e.*, mean, variance and skewness, of the output-input variance ratio $\mathrm{Var}(y)/\mathrm{Var}(x)$, based on a proper formulation and a realistic assumption of natural image statistics for filter input. If the theoretical prediction coincides well with the empirical observation as given by the simulations and the histogram, this will be a strong evidence for the appropriateness of our theoretical formulation and of the assumption of input statistics. This would also help us well understand the rather unexpected empirical behavior of the Xavier algorithm.

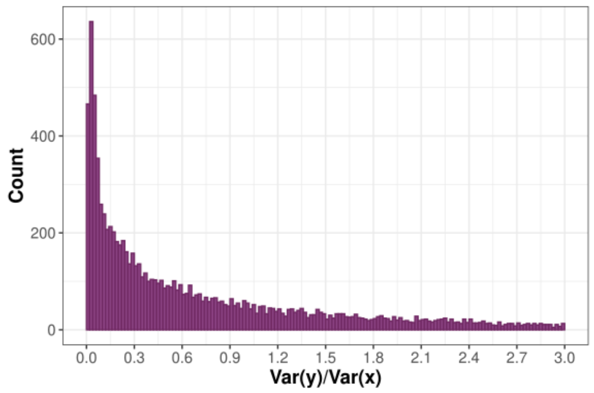

Figure 1: Histogram of the ratio of output and input variances $\text{Var}(y)/\text{Var}(x)$ for $10,000$ simulated filters generated by the Xavier initialization algorithm [15].

We notice that the histogram in Figure 1 for the output-input variance ratio of the popular Xavier initialization [15] is somewhat surprising because we would rather expect a high peak at 1 for which Xavier keeps the stability of output and input variances. However, in practice we observe very high occurrences at small values; together with a long tail on the right (probably due to numerical sampling used in the initialization and we do not completely show the tail in the histogram), the value of empirical mean is around 1. In addition, we can seen that the ratio $\text{Var}(y)/\text{Var}(x)$ has a certain variance because it can take both rather small and relatively big values. More importantly, the histogram is featured with a big *positive skewness*, which means that the tail on the right side is longer and the mass is concentrated on the left side of the histogram. In the following, we show that with the correct assumption on the input statistics and a proper theoretical formulation we can predict the values of the mean, variance and skewness of the empirical histogram in Figure 1, which are important statistical properties reflecting the shape of the histogram.

As mentioned earlier, in our simulations we generated $10,000$ Xavier filters of shape 5×5. These filters were created using the PyTorch function `torch.nn.init.xavier_uniform_`. The uniform distribution used for drawing pseudo-random samples is $U(-\sqrt{3/25}, \sqrt{3/25})$, based on the general form of uniform distribution $U(-\sqrt{3/N}, \sqrt{3/N})$ for Xavier initialization of a filter comprising N scalar coefficients (here we have $N = 5 \times 5 = 25$). With these characteristics we can study the theoretical properties of the output-input variance ratio and compare with the results of experimental simulations.

For this scenario of Xavier filter simulations, we consider the statistical properties of $\text{X} = (x_1, x_2, ..., x_N)$ as fixed while $\text{W} = (w_1, w_2, ..., w_N)$ are sampled in a pseudo-random way to create each of the $10,000$ Xavier filters. We know that in each simulation the output variance $\text{Var}(y)$ is given by Eq. (2) and approximated by Eq. (3). The computation of $\text{Var}(y)$ can be further approximated by replacing $\text{Cov}(x_i, x_j)$ by $\text{Var}(x)$, or equivalently by replacing the Z_{ij} terms in

Eq. (3) by 1. This gives the following Eq. (5) where we can notice that Var(y) is a random variable dependent on $w_{i,i=1,2,...,N}$ while Var(x) is the fixed overall variance of the input.

$$\begin{aligned}\mathrm{Var}(y) &= \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \mathrm{Cov}(x_i, x_j) \\ &\approx \mathrm{Var}(x)\sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \\ &= \mathrm{Var}(x)(w_1w_1 + \cdots + w_1w_N + w_2w_1 + \cdots + w_Nw_N).\end{aligned} \tag{5}$$

For the sake of clarity and easy understanding of the subsequent derivations, we develop all the N^2 terms of w_iw_j in the parentheses above.

1) Mean of Var(y)/Var(x)

We begin by calculating the mean of Var(y) as detailed in Eq. (6) below. The last equation is obtained with help of the property of Xavier initialization [15] for which we have $\mathrm{E}(w_i^2) = \mathrm{Var}(w_i) = \frac{1}{N}$ (for Xavier w_i has zero mean). Additionally, many of the N^2 terms of $\mathrm{E}(w_iw_j)$ in the summation are equal to zero when $i \neq j$ (because $w_1, w_2, ..., w_N$ follow zero-mean iid). In the end, only the N terms of $\mathrm{E}(w_iw_i)$ contribute to the sum.

$$\begin{aligned}\mathrm{E}(\mathrm{Var}(y)) &\approx \mathrm{Var}(x)\mathrm{E}(w_1w_1 + \cdots + w_1w_N + w_2w_1 + \cdots + w_Nw_N) \\ &= \mathrm{Var}(x)\sum_{i=1}^{N}\sum_{j=1}^{N}\mathrm{E}(w_iw_j) = \mathrm{Var}(x)\sum_{i=1}^{N}\mathrm{E}(w_iw_i) \\ &= \mathrm{Var}(x)N\mathrm{E}(w_i^2) = \mathrm{Var}(x).\end{aligned} \tag{6}$$

The derived theoretical mean of output-input variance ratio, *i.e.*, E(Var(y)/Var(x)), is actually the desired value 1 of Xavier initialization. However as we showed previously, the histogram of Var(y)/Var(x) does not have a high peak at 1; in contrast, it has a big mass concentration on the left and a long tail on the right, as well as a certain range of variation. In the following we will theoretically derive the variance and skewness of the ratio of output-input variances to well understand the shape of the empirical histogram, in particular the relatively big variation range and the asymmetric distribution of the variance ratio.

2) Variance of Var(y)/Var(x)

Ideally the variance of Var(y) should be very small, reflecting the stability of output variance in a large number of simulations. Nevertheless, we can see from the simulations that Var(Var(y)) would not be very small because the variance ratio Var(y)/Var(x) can be a relatively small or a relatively big value. In order to calculate Var(Var(y)), we use the standard formula of the variance of the sum of (potentially) correlated random variables as shown in the third row of

Eq. (7) below.

$$
\begin{aligned}
\mathrm{Var}(\mathrm{Var}(y)) &\approx \mathrm{Var}\left(\mathrm{Var}(x)\sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j\right) \\
&= \mathrm{Var}^2(x)\mathrm{Var}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j\right) \\
&= \mathrm{Var}^2(x)\left[\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\mathrm{Cov}(w_i w_j, w_k w_l)\right] \\
&= \mathrm{Var}^2(x)\left[\frac{4}{5N}+\frac{2(N-1)}{N}\right].
\end{aligned} \tag{7}
$$

In order to obtain the final result in the last row above, we make use of the fact that among N^4 terms of $\mathrm{Cov}(w_i w_j, w_k w_l)$ in the summation on the third row of Eq. (7), many of them are equal to zero. For example, this is the case for terms like $\mathrm{Cov}(w_i^2, w_k w_l) = 0$ (when $i = j$ and $i \neq k \neq l$) or $\mathrm{Cov}(w_i w_j, w_k w_l) = 0$ (when the four indexes are all different). In fact, only the N terms of $\mathrm{Cov}(w_i^2, w_i^2) = \frac{4}{5N^2}$ and the $2N(N-1)$ terms of $\mathrm{Cov}(w_i w_j, w_i w_j) = \frac{1}{N^2}$ contribute to the sum of the covariance terms, which leads to the final equation. The detailed derivation is omitted here for the sake of brevity but should be easy to be reproduced. One hint is that the different covariance terms $\mathrm{Cov}(w_i w_j, w_k w_l)$ can be easily computed by using properties of $w_{i,i=1,2,\ldots,N}$, *i.e.*, they follow iid of uniform distribution $U(-\sqrt{3/N}, \sqrt{3/N})$. Another hint is that we can group the N^4 covariance terms into different categories and count the number of terms in each category with basic knowledge of combinatorics.

In our case we have $N = 25$, so according to Eq. (7) it can be deduced that $\mathrm{Var}(\mathrm{Var}(y)) \approx 1.952\mathrm{Var}^2(x)$, *i.e.*, we have $\mathrm{Var}(\mathrm{Var}(y)/\mathrm{Var}(x)) \approx 1.952$. The $10,000$ simulations show that the histogram of $\mathrm{Var}(y)/\mathrm{Var}(x)$ has an empirical variance of about 1.858. The theoretical and simulated results are close to each other and the small difference is mainly due to the (reasonable) approximation made in the derivation, *i.e.*, we approximate all the $\mathrm{Cov}(x_i, x_j)$ terms by $\mathrm{Var}(x)$ in Eq. (5). We can also observe that the variance of the output variance is not small, being about 1.9 times of the square of the input variance.

3) Skewness of $\mathrm{Var}(y)/\mathrm{Var}(x)$

The skewness is probably the most interesting and important statistical property for understanding the shape of the histogram in Figure 1 which has a big mass concentration on the left and a long tail on the right. In order to compute the skewness of $\mathrm{Var}(y)$, we start with the general formula of skewness computation as given in Eq. (8) below:

$$
\mathrm{Skewness}(\mathrm{Var}(y)) = \frac{\mathrm{E}[\mathrm{Var}^3(y)] - 3\mathrm{E}[\mathrm{Var}(y)]\mathrm{Var}(\mathrm{Var}(y)) - \mathrm{E}^3[\mathrm{Var}(y)]}{(\mathrm{Var}(\mathrm{Var}(y)))^{\frac{3}{2}}}. \tag{8}
$$

We have the formula of $\mathrm{E}(\mathrm{Var}(y))$ and $\mathrm{Var}(\mathrm{Var}(y))$ respectively in Eq. (6) and Eq. (7). Therefore, in order to calculate the skewness we now only need to compute the term $\mathrm{E}[\mathrm{Var}^3(y)]$. By using Eq. (5) we can write this term as:

$$\begin{aligned}\mathrm{E}[\mathrm{Var}^3(y)] &\approx \mathrm{Var}^3(x)\mathrm{E}\left[\left(\sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j\right)^3\right] \\ &= \mathrm{Var}^3(x)\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{m=1}^{N}\sum_{n=1}^{N}\mathrm{E}\left[w_i w_j w_k w_l w_m w_n\right].\end{aligned} \tag{9}$$

There are N^6 terms in the summation above, but similar to the derivation of the variance of $\mathrm{Var}(y)$ earlier in this subsection, only several kinds of specific terms are not zero and contribute to the sum. There are three kinds of non-zero terms respectively with the form of $\mathrm{E}(w_i^6)$, $\mathrm{E}(w_i^2 w_j^4)$ and $\mathrm{E}(w_i^2 w_j^2 w_k^2)$. In fact, still using the hints mentioned earlier for the computation of $\mathrm{Var}(\mathrm{Var}(y))$, we can show that after development the other kinds of terms all have at least a multiplicative term like $\mathrm{E}(w_i)$, $\mathrm{E}(w_i^3)$ or $\mathrm{E}(w_i^5)$ which is equal to 0 due to the fact that in Xavier w_i follows a zero-mean uniform distribution.

Regarding the non-zero terms, after some simple calculations we can deduce that there are N terms like $\mathrm{E}(w_i^6) = 27/(7N^3)$, $N(N-1)\mathrm{C}_6^2 = 15N(N-1)$ terms like $\mathrm{E}(w_i^2 w_j^4) = 27/(15N^3)$, and $\mathrm{C}_N^3\mathrm{C}_6^2\mathrm{C}_4^2 = 15N(N-1)(N-2)$ terms like $\mathrm{E}(w_i^2 w_j^2 w_k^2) = 1/N^3$, where C_r^b is the combination notation of "r choose b". With these results we can compute the expectation of $\mathrm{Var}^3(y)$ as shown in Eq. (10) below.

$$\begin{aligned}\mathrm{E}[\mathrm{Var}^3(y)] &\approx \mathrm{Var}^3(x)\left[\frac{N.27}{7N^3} + \frac{15N(N-1).27}{15N^3} + \frac{15N(N-1)(N-2)}{N^3}\right] \\ &= \mathrm{Var}^3(x)\left[\frac{105N^2 - 126N + 48}{7N^2}\right].\end{aligned} \tag{10}$$

Here with $N = 25$, we have $\mathrm{E}[\mathrm{Var}^3(y)] \approx 14.29\mathrm{Var}^3(x)$. With Eq. (10) for $\mathrm{E}[\mathrm{Var}^3(y)]$, Eq. (6) for $\mathrm{E}(\mathrm{Var}(y))$ and Eq. (7) for $\mathrm{Var}(\mathrm{Var}(y))$, we can calculate the skewness of $\mathrm{Var}(y)$ by using Eq. (8). This results in a theoretical value of about 2.726 for the skewness, *i.e.*, $\mathrm{Skewness}(\mathrm{Var}(y)/\mathrm{Var}(x)) \approx 2.726$. From the $10,000$ simulations we obtain an empirical skewness value of about 2.811 for the output-input variance ratio. These results confirm the validity of the theoretical prediction as well as a relatively big positive skewness value for the histogram with a big tail on the right and a dense mass concentration on the left.

From this theoretical analysis of the mean, variance and skewness of the quantity $\mathrm{Var}(y)/\mathrm{Var}(x)$, we demonstrate that with the new and correct assumption of strong correlation between neighboring pixels in natural images, we can accurately predict and explain the statistical properties of the output-input variance ratio histogram for Xavier initialization. By contrary, this would not have been possible with the previous assumption of independently and identically distributed pixel values. Indeed, the shape of the histogram is somewhat surprising and unexpected because in most cases the variance of the output is smaller than the input, implying that the variance stability is not maintained as what is expected for Xavier initialization.

In all, the theoretical and experimental studies presented in this section have deepened our understanding about existing CNN initialization algorithms and motivated our work on the design and implementation of new approaches. We also hope that these studies may provide insights and inspirations for subsequent works among the research community.

4. First Approach: Data-Dependent Scaling of Convolutional Filter

As analyzed and discussed in the previous section, the output signal shrinkage occurs for all the four existing initialization algorithms of convolutional filters. In particular, for algorithms generating high-pass filters [16, 6, 20], the variance of output signal is in most cases less than 1% of the variance of input. Additionally and surprisingly, even for the conventional Xavier initialization [15], the output variance has quite high probability to be considerably smaller than the input variance. This signal shrinkage is in general not favorable for the network training, and we show in Section 6 with example that occasionally this data flow shrinkage results in training failure of CNN.

The idea of our first approach to solving the signal shrinkage problem is very simple. For a given filter $\mathrm{W} = (w_1, w_2, ..., w_N)$ generated by any underlying algorithm, we derive a proper scaling factor S and use the scaled filter $\widetilde{\mathrm{W}} = S.\mathrm{W}$ as the initialization of first-layer convolutional kernel. The scaling factor S ensures that the scaled filter $\widetilde{\mathrm{W}}$ leads to almost identical values for the input and output variances. We call it a data-dependent approach because the derivation of scaling factor is dependent on explicit computation on the input data. According to how the scaling factor is computed, we propose two different versions of this first approach, as briefly described below.

Covariance-based method. We call the first version as a covariance-based method, because this method computes the variance and covariance terms of the input signal components for the calculation of the scaling factor. More precisely, it can be observed from Eq. (3) that the input variance $\mathrm{Var}(x)$ and the output variance $\mathrm{Var}(y)$ is related by a multiplicative factor dependent on simple statistical properties of the input data, *i.e.*, the Z_{ij} terms which is equal to $\mathrm{Cov}(x_i, x_j)/\mathrm{Var}(x)$. It is easy to see that if we want to have $\mathrm{Var}(y) \approx \mathrm{Var}(x)$, we need to reverse the effect of this multiplicative factor *i.e.*, the term in the parentheses in Eq. (3). Therefore, the scaling factor S for a given filter $\mathrm{W} = (w_1, w_2, ..., w_N)$ is simply computed as:

$$S = \sqrt{1 \Big/ \left(\sum_{i=1}^{N} w_i^2 + 2 \sum_{1 \leq i} \sum_{< j \leq N} w_i w_j Z_{ij} \right)}. \tag{11}$$

In practical implementation, it is sufficient to compute the Z_{ij} terms by using a small amount of training data, *e.g.*, selecting 10% of training samples and extracting 10 random patches of 5×5 pixels from each selected training sample. Experimentally this gives very reliable and stable result of scaling factor, even when compared to the computation on the whole training dataset.

Convolution-based method. We call the second version as a convolution-based method because it computes the scaling factor in a straightforward way by carrying out convolution operation. Similar to the covariance-based method, it is safe to perform convolution on a small amount of input data $\widehat{x}$ (*e.g.*, randomly selecting 10% of training samples), so as to obtain the corresponding small amount of output data $\widehat{y}$. The scaling factor S is simply obtained as:

$$S = \sqrt{\mathrm{Var}(\widehat{x})/\mathrm{Var}(\widehat{y})}. \tag{12}$$

Technically, among the two methods we have a slight preference for the covariance-based method. The reason is that it is sufficient to perform *only once* the main part of the computation of the covariance-based method (*i.e.*, the computation of input's variance and covariance terms), and later these terms can be reused for any new filter to be scaled. Contrarily, for the convolution-based method, the main computation of convolution operation needs to be done for every given filter. Nevertheless, the computational cost of both methods is very low. The calculation of scaling factor for a given filter takes less than 3 seconds for both methods, on a computer equipped with a 2.50GHz CPU and an Nvidia® GeForce 1080 Ti GPU. Such computation times are orders of magnitude lower than the time required to train a CNN.

5. Second Approach: Random High-Pass Initialization

5.1. Motivation

In the previous section we described a data-dependent scaling approach for the initialization of the first layer of a CNN for the image manipulation detection problem. Nevertheless, there are practical scenarios in which it is not easy or not possible to reliably calculate the necessary statistical properties of the training data due to different constraints in the deployment. For example, the training data may come in a sequential and dynamic way denying the option to reliably calculate beforehand the covariance terms and the scaling factor from the beginning. A scarce-data scenario where the available data would not be enough to have a confident statistical estimation may also be in place. Indeed, in practice when we have very few data for the estimation of input statistics, we may obtain a very small scaling factor; this inappropriate scaling factor will even further shrink the output signal and thus decrease the performance after scaling. Additionally, although the computation complexity of our scaling solution is negligible in the deep learning field, one main trend is the usage of end-to-end methods without pre-processing steps for better technical flexibility. For these reasons we believe that it is necessary and beneficial to propose a second, random

w_1	w_2	w_3	w_4	w_5
w_6	w_7	w_8	w_9	w_{10}
w_{11}	w_{12}	C	w_{13}	w_{14}
w_{15}	w_{16}	w_{17}	w_{18}	w_{19}
w_{20}	w_{21}	w_{22}	w_{23}	w_{24}

x_1	x_2	x_3	x_4	x_5
x_6	x_7	x_8	x_9	x_{10}
x_{11}	x_{12}	x_{25}	x_{13}	x_{14}
x_{15}	x_{16}	x_{17}	x_{18}	x_{19}
x_{20}	x_{21}	x_{22}	x_{23}	x_{24}

Figure 2: Notations of the filter of our RHP initialization approach (left) and of the input (right). We show as an example the case of a 5×5 filter.

high-pass initialization approach which does not explicitly utilize the input data, *i.e.*, it does not need to calculate explicitly the variance and covariance statistics of input or to carry out convolution computation on the input data. Such an approach follows the mainstream of data-independent random initialization approach for CNNs in the research community. It is worth mentioning that "data-independent" is perhaps not a very rigorous term to describe our second approach because it indeed considers the statistics (with approximations) of the input data; nevertheless this second approach does not explicit use the input data for filter initialization. With a little abuse in using this term, we still from time to time call this second proposal as a data-independent approach.

5.2. Formulation and derivation

Our objective in this section is to propose a new RHP (Random High-Pass) initialization approach of convolutional filters for image manipulation detection problems. The generated random high-pass filters will be used as initialization for the first layer of a CNN. To accomplish this, we use as filter template such as the one shown in Figure 2. The symbol C represents an unknown constant in the filter, $\mathrm{W} = (w_1, w_2, ..., w_N)$ are independent scalar random variables following an appropriate distribution, and $\mathrm{X} = (x_1, x_2, ..., x_N, x_{N+1})$ contains a group of *mutually correlated* random variables representing the input pixel values. We would like to mention again that this realistic formulation of mutually correlated input pixel values (*i.e.*, x_i's) is the fundamental difference between our approach and the formulation of Xavier [15] and Castillo [20] where x_i's are considered as independent random variables. Despite the fact that the template shown in Figure 2 has a shape of 5×5, our proposal can be applied to different filter shapes where there are two groups of coefficients: the unknown constant C and the remaining N random coefficients $w_{i,i=1,2,...,N}$. For the template shown in Figure 2 we have $N = 24$. The actual values of $w_{i,i=1,2,...,N}$ are sampled from an adequate distribution which plays an important role as described later.

With the objective of ensuring the high-pass behavior (*i.e.*, filter coefficients sum up to 0) for the designed filter, the mathematical expectation of the random variable w_i should be equal to $-\frac{C}{N}$ in order to compensate for the unknown constant C. By having this characteristic, the expectation of the sum of all $w_{i,i=1,2,...,N}$ is equal to $-C$, together with C making the initialized kernel resemble a high-pass filter. As mentioned before, in Xavier [15] and Castillo [20], both w_i and x_i were assumed as independent random variables. In our first proposal described in Section 4, we considered w_i as known constants (*i.e.*, coefficients of a given filter to be scaled) while x_i as correlated random variables. Here for our second proposal of RHP approach, $w_{i,i=1,2,\ldots,N}$ are independent random variables, and $x_{i,i=1,2,\ldots,N,N+1}$ are mutually correlated random variables following natural image statistics.

We know that the local operation of an input and a convolutional filter can be expressed as an inner product. With a filter of the characteristics described above and with the notations of filter and input as illustrated in Figure 2 (with 5×5 filter as example), the output can be calculated as

$$y = w_1x_1 + w_2x_2 + \ldots + w_Nx_N + Cx_{N+1}. \tag{13}$$

Similar to the derivation in the last section and still using the general formula of the variance of a sum of (potentially) correlated random variables, we can compute the variance of the output y by dividing the possible terms into four sets, as shown below:

$$\begin{aligned}\mathrm{Var}(y) = &\sum_{i=1}^{N} \mathrm{Var}(w_ix_i) + \mathrm{Var}(Cx_{N+1}) \\ &+ 2\sum_{1\le i}\sum_{<j\le N} \mathrm{Cov}(w_ix_i, w_jx_j) + 2\sum_{i=1}^{N} \mathrm{Cov}(w_ix_i, Cx_{N+1}).\end{aligned} \tag{14}$$

In the above Eq. (14), there are two sets of variance terms and two sets of covariance terms. These four sets of terms are obtained by considering all the possible combinations between the two groups of coefficients in the designed filter: the first group of unknown constant C and the second group of N random variables w_i. By using the variance property of product of random variables [25] and the expectation of w_i (*i.e.*, $\mathrm{E}(w_i) = -\frac{C}{N}$), we obtain the following Eq. (15) for the variance of y:

$$\begin{aligned}\mathrm{Var}(y) = &\sum_{i=1}^{N} \mathrm{Var}(x_i)\left[\mathrm{Var}(w_i) + \frac{C^2}{N^2}\right] + C^2\mathrm{Var}(x_{N+1}) \\ &+ 2\sum_{1\le i}\sum_{<j\le N} \frac{C^2}{N^2}\mathrm{Cov}(x_i, x_j) - 2\sum_{i=1}^{N} \frac{C^2}{N}\mathrm{Cov}(x_i, x_{N+1}).\end{aligned} \tag{15}$$

Following the realistic and classical assumption of almost identical variance for all input pixels $x_{i,i=1,2,\ldots,N,N+1}$ (*i.e.*, the approximate translation invariance in natural images [22, 23] which was also utilized in the formulation for computing output variance in Section 3.1), we can make some adjustments to Eq. (15) by substituting $\mathrm{Var}(x_i)$ and $\mathrm{Var}(x_{N+1})$ by $\mathrm{Var}(x)$ (*i.e.*, the total variance of input). In addition, to simplify the notations we replace $\mathrm{Cov}(x_i, x_j)/\mathrm{Var}(x)$ by Z_{ij} and $\mathrm{Cov}(x_i, x_{N+1})/\mathrm{Var}(x)$ by Z_{iN+1}. The above approximation and simplification give the following Eq. (16):

$$\mathrm{Var}(y) \approx \mathrm{Var}(x) \left(\sum_{i=1}^{N} \left[\mathrm{Var}(w_i) + \frac{C^2}{N^2} \right] + C^2 + 2 \sum_{1 \le i} \sum_{< j \le N} \frac{C^2}{N^2} Z_{ij} - 2 \sum_{i=1}^{N} \frac{C^2}{N} Z_{iN+1} \right). \tag{16}$$

Next, following the reasonable assumption of highly-correlated neighboring pixels in natural images [22, 23], we can substitute both Z_{ij} and Z_{iN+1} by 1 because theoretically and experimentally these terms are smaller than but very close to 1 (*cf.*, the discussion below Eq. (4) in Section 3.2). With this approximation applied to Eq. (16), we obtain the following Eq. (17):

$$\begin{aligned} \mathrm{Var}(y) &\approx \mathrm{Var}(x) \left(N\mathrm{Var}(w_i) + N\frac{C^2}{N^2} + C^2 + 2\frac{N(N-1)}{2}\frac{C^2}{N^2} - 2N\frac{C^2}{N} \right) \\ &= \mathrm{Var}(x) \left[N\mathrm{Var}(w_i) \right]. \end{aligned} \tag{17}$$

Interestingly, all the terms where C is involved nicely disappear after applying the considered approximations, which makes our derived result simple and concise. It is now clear from the above Eq. (17) that in order to make $\mathrm{Var}(x)$ and $\mathrm{Var}(y)$ close to each other, we should have the following property for filter coefficients w_i:

$$\mathrm{Var}(w_i) = \frac{1}{N}. \tag{18}$$

At first glance, the above Eq. (18) happens to be the same as the one deduced for the Xavier initialization [15], but there are fundamental differences. First, the filter template for our case will generate a high-pass filter suitable for image forensics problems, while the Xavier initialization in general does not have this property. More precisely, in our filter template we have a new unknown constant C together with w_i forming a high-pass filter, while Xavier filter does not have this unknown constant. Second, the Xavier initialization assumes that both input X and filter coefficients W are random variables following zero-mean iid, while in our RHP initialization approach x_i's are mutually correlated random variables and w_i's are random variables with a non-zero mathematical expectation as $\mathrm{E}(w_i) = -\frac{C}{N}$. We can see that the statistical property of w_i is related to the unknown constant C. This is explained with further derivations in the next paragraph.

For the sampling of $w_{i,i=1,2,\ldots,N}$, we use a simple uniform distribution with the previously mentioned expectation of $-\frac{C}{N}$, *i.e.*, $U\left(-\frac{2C}{N}, 0\right)$ or $U\left(0, -\frac{2C}{N}\right)$ depending on the sign of C. We choose to use this distribution because it is probably the simplest distribution with the prescribed mathematical expectation. From the chosen distribution we can easily calculate the variance of w_i as $\mathrm{Var}(w_i) = \frac{C^2}{3N^2}$. In the meanwhile, from Eq. (18) we know that in order to make the input and output variances comparable we should have $\mathrm{Var}(w_i) = \frac{1}{N}$. After combining these two equations we obtain the following Eq. (19) to derive the value of C:

$$\mathrm{Var}(w_i) = \frac{C^2}{3N^2} = \frac{1}{N} \quad \Longrightarrow \quad C = \pm\sqrt{3N}. \tag{19}$$

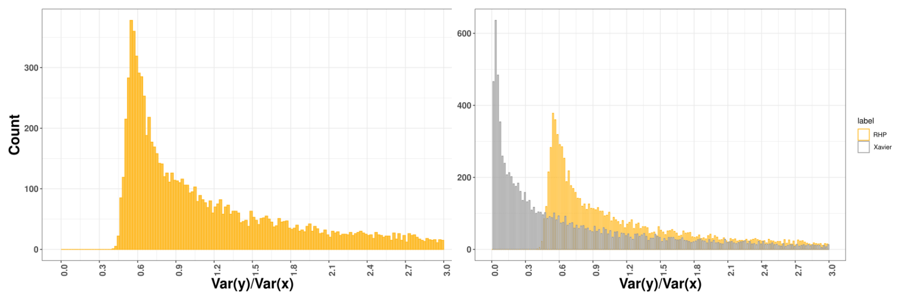

Figure 3: Histogram of occurrences of output-input variance ratio for our RHP approach and the Xavier initialization algorithm [15]. Figure on the left shows the histogram of occurrences of the variance ratio $\mathrm{Var}(y)/\mathrm{Var}(x)$ for $10,000$ simulations of our RHP filters. On the right, we show the comparison between the histogram of our RHP approach and the histogram of $10,000$ Xavier filters.

Accordingly, we also obtain the interval of the uniform distribution for the sampling of w_i as $U\left(-\frac{2\sqrt{3N}}{N}, 0\right)$ or $U\left(0, \frac{2\sqrt{3N}}{N}\right)$. With a derived filter such as the one with the template of 5×5 shown in Figure 2 left ($N = 24$), we have the value of C as $C = \pm\sqrt{72} \approx \pm 8.485$.

5.3. Output variance

Figure 3 left shows the histogram of the output-input variance ratio, *i.e.*, $\mathrm{Var}(y)/\mathrm{Var}(x)$, for $10,000$ simulated filters using our proposed RHP initialization, while Figure 3 right shows a comparison between our RHP approach and the well-known Xavier initialization [15]. As we can see, for our RHP approach the high occurrences no longer happen in an interval of small values as what happens with Xavier algorithm. Practically there is no occurrence for our approach within the interval where Xavier algorithm has the majority of occurrences. Indeed, the minimum value of output-input variance ratio for our RHP approach in the $10,000$ simulations is 0.41 with a median of 1.01, so there is no occurrence at all from 0 to 0.40 for our approach. By contrast, the Xavier algorithm has a minimum value of 0.002, a median of 0.46, and a big mass concentration of occurrences between 0 and 0.3. We can also observe that for our RHP initialization approach, the majority of occurrences occur in the range of 0.5 to 0.9, which is closer to an ideal scenario of 1.0 but not centered at 1.0 (though the median 1.01 as reported above is very close to 1). Our explanation is that this shift occurs due to the approximations that we make in our derivations, in particular we approximate all the $Z_{ij} = \mathrm{Cov}(x_i, x_j)/\mathrm{Var}(x)$ terms by 1. Further theoretical studies in an attempt to design a random high-pass initialization with a histogram ideally centered at 1 are one part of our future work.

The preliminary results in Figure 3 show the effectiveness of our RHP initialization approach. In the next section we present experimental performances obtained for different image manipulation detection problems and on different

CNN architectures for our two approaches, *i.e.*, the first proposal of the data-dependent filter scaling approach (Section 4) and the second proposal of the random high-pass initialization approach (this Section 5).

6. Experimental Results

In this section we conduct a series of experiments for the test and validation of our initialization approaches in different scenarios. In the experiments, we consider and compare with the four existing CNN initialization algorithms mentioned earlier (Xavier [15], SRM [16], Bayar [6], and Castillo [20]). In order to test on different CNN architectures, besides the well-known network of Bayar and Stamm [6], we carry out experiments on a smaller network built by ourselves. Two different image manipulation detection problems are considered: a multi-class classification problem of a set of manipulations and a challenging binary detection problem of high-quality JPEG compression.

In one set of experiment, we also perform test on a variant of the network of [6] with more kernels at first layer. This is still for the purpose of considering more experimental settings for the validation of the proposed approaches. Moreover, we conduct a comparative experiment with the well-known batch normalization technique [26] under the multi-class classification scenario. We also illustrate the potential problem of randomly choosing a few handcrafted filters from a pool of such filters when used for CNN initialization.

Our experimental studies are based on PyTorch and Nvidia® GeForce 1080 Ti GPU. The experimental data are generated from the Dresden database [24] which comprises 1491 unprocessed raw images (in `.NEF` and `.DNG` format) with resolutions ranging from 1828×1372 to 3872×2592. For the preparation of experimental data, we first divide the Dresden images into three groups respectively for training, validation and testing with the fraction of 3:1:1. The raw Dresden images are read and then converted to grayscale, and the converted grayscale images are saved in lossless `.TIF` format from which 64×64 random patches are extracted. We follow [6] to have the same number of extracted patches, and this will be detailed later in this section. The size of patches (*i.e.*, 64×64) is rather small and this makes it challenging to detect manipulation operations on them.

6.1. Multi-class problem with the CNN of Bayar and Stamm

The objective for the multi-class forensic problem is to correctly classify six classes of original and manipulated patches. For the creation of manipulated patches, we make use of the five manipulation operations listed in Table 2. As mentioned earlier, we follow [6] to have exactly the same number of training and testing patches. More precisely, the training set comprises $100,000$ patches ($\approx 16,667$ per class) and the testing set includes $32,000$ patches ($\approx 5,333$ per class). The manipulation operations and their parameters in our experiments are the same as those in [20, 7] and are more challenging when compared with [6]. Moreover, we use 64×64 patches in our experiments while authors of [6]

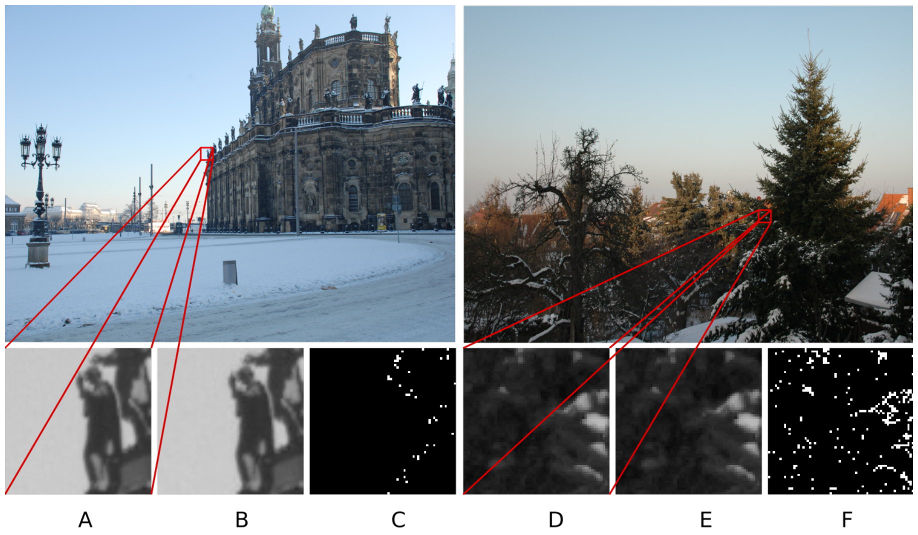

Figure 4: Examples of images from Dresden database [24] and the generated patches used in our experiments. Bottom images on columns A and D show original patches while columns B and E show the JPEG compressed and median filtered patch respectively. Columns C and F show the thresholded absolute differences for the two pairs of original and manipulated patches. The threshold is set as 20.

mainly use 256×256 patches. This also makes it more difficult to detect manipulations in experiments presented in this paper. Figure 4 shows on the first row two images of Dresden database. The bottom row shows both original and manipulated patches along with a binary representation of thresholded absolute difference between pair of original and manipulated patches. We can notice that the difference between original and manipulated patches, both visually and computationally, is quite small, reflecting that we indeed deal with a challenging forensic problem.

Experiments in this subsection are performed on the well-known network designed by Bayar and Stamm [6] which has 3 filters at the first layer. The first-layer convolutional filters are initialized in five different ways: Bayar [6], SRM [16], Castillo [20], Xavier [15] and our RHP approach. In order to enhance the statistical significance of our experiments, 5 different runs are performed for each algorithm, as well as for the scaled versions obtained by applying our data-dependent scaling approach described in Section 4 (both covariance-based and convolution-based methods are tested). Regarding the initialization with SRM, in each run we randomly choose 3 filters from the set of 30 available SRM filters (*cf.*, link in footnote 1). In order to achieve fair and meaningful comparisons, for each run it is ensured that the original and scaled versions have the same "base filters" and that the only difference is the scale of filters. The network training procedure follows the original paper of Bayar and Stamm [6]. CNN training parameters are the followings: the batch size is 64, the total number of epochs is 60, the optimizer is based on stochastic gradient descent with the momentum equal to 0.95 and the weight decay equal to $5e{-}4$, the initial learning rate is $1e{-}3$, and the learning rate follows a step decay schedule with a multiplicative decay factor of 0.5 applied every 6 epochs.

For the constrained algorithm of Bayar [6], two variants are designed and tested for the filter scaling. In the first variant denoted by "Bayar A.", the filter scaling is carried out at each step of network training just after Bayar's constraint enforcement via filter normalization (*cf.*, Section 2). In the second variant denoted by "Bayar B.", we only apply scaling on the initialized constrained filter, and later the network is trained freely without any constraint or scaling. With this second variant "Bayar B." which has also much lower computational cost than the first variant, we would like to test the idea that after performing a good initialization with an appropriate filter scaling, it would not be necessary to enforce the constrained training proposed in [6].

Following [6, 20], in this paper we measure the forensic performance by using the classification accuracy on testing set (*i.e.*, the test accuracy), which is computed as the percentage of correctly classified patches among all the test patches of different classes. The computation of test accuracy is given in Eq. (20), where M is the number of predictions (*i.e.*, the number of samples in testing set), $\tilde{l}_i$ is the ground-truth label of i-th sample, l_i is the predicted label of i-th sample, and $\mathbb{1}(.)$ is the indicator function.

$$\text{Accuracy} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{1}(\tilde{l}_i = l_i). \tag{20}$$

Table 3 presents the test accuracies of different algorithms under the multi-class classification scenario. As mentioned earlier, the accuracy values are average of 5 different runs, and we make sure that for each group of run same "base filters" are shared between original and scaled versions. Regarding our RHP proposal, we do not consider a scaled version given the fact that a random initialization without any pre-processing (*e.g.*, scaling) is the objective for this proposal. Furthermore, as described later, filter values obtained from the RHP approach are in fact similar to the scaled ones.

In Table 3 the best result of unscaled version (*i.e.*, in the column of "Original version") is in bold, and we show in parentheses the performance improve of scaled version compared to the corresponding unscaled version in the same row. We can see from Table 3 that in comparison to the original unscaled version of all the other algorithms, our RHP approach has the highest test accuracy with an improvement of at least 1.96% (96.35% vs. 94.39%, compared to the second best SRM algorithm). We can also observe that the accuracy of RHP is similar to but slightly lower than the results of scaled Castillo and scaled SRM. This is probably because the RHP approach does not ensure perfect equality of input and output variances, as illustrated in Figure 3. Nevertheless, the proposed RHP initialization is computationally more efficient than initializations with scaled filters, and it is also technically more flexible by avoiding a pre-processing step of statistics or convolution computation which explicitly uses the input data.

Table 3: Test accuracy (in %, average of 5 runs) of the multi-class classification problem, on the network of Bayar and Stamm [6]. We report in parentheses the accuracy improvement of scaled version in comparison to the original version in the same row.

Initialization	**Original version**	**Convolution-based scaling (our)**	**Covariance-based scaling (our)**
Bayar [6] A.	94.19	96.04 (+1.85)	96.02 (+1.83)
Bayar [6] B.		96.15 (+1.96)	96.22 (+2.03)
Castillo [20]	93.71	96.45 (+2.74)	96.42 (+2.71)
SRM [16]	94.39	96.54 (+2.15)	96.55 (+2.16)
Xavier [15]	93.48	94.61 (+1.13)	94.71 (+1.23)
RHP init. (our)	**96.35**	-	-

Additionally, our proposed filter scaling approach is able to consistently improve the forensic performance, as reflected by values in parentheses in Table 3. After scaling, the increase in terms of test accuracy goes from 1.13% to 2.74%. Another observation is that the two versions of our scaling approach give almost identical performance improvement. Experimentally, the scaling factors derived by the two versions (covariance-based or convolution-based) have almost same value. It is also interesting to notice that among the two variants of scaled Bayar, "Bayar B." achieves slightly higher test accuracy than "Bayar A." and in the meanwhile is more efficient in terms of computation cost. This probably means that our intuition mentioned earlier may be correct, that is, by using a good initialization with correct scale we would not be forced to carry out constrained training. We would like to clarify that the forensic performance of Bayar presented in Table 3 is generally lower than the results in the original paper [6]. The main reason is that as mentioned at the beginning of this section, the detection problem in our experiments is more difficult by considering manipulations with more challenging parameters and patches of smaller size. In the meanwhile, the performance of Castillo presented in Table 3 is higher than in the original paper [20]. The reason would be the difference in the network training options. In our experiments, we use the same training procedure of Bayar and Stamm [6], in particular the training comprises more epochs when compared to [20]. The last observation is that the conventional Xavier initialization indeed performs less well than all the other algorithms which generate high-pass filters. In accordance with the discussions in Sections 1 and 2, this observation implies that due to the particularity of the manipulation detection problem, new initialization algorithms are required for this specific forensic task.

Figure 5 shows some curves of evolution of test accuracy in the course of the 60 epochs of the network training. It can be seen that our proposed scaling and RHP approaches allow us to have quicker increase of test accuracy. In particular, for SRM the accuracy increase at the beginning of training is significantly faster after applying scaling. Throughout the training procedure, the evolution curve of scaled SRM and Xavier is always located up above the curve of the corresponding unscaled version. At last, the curve of our RHP approach has a

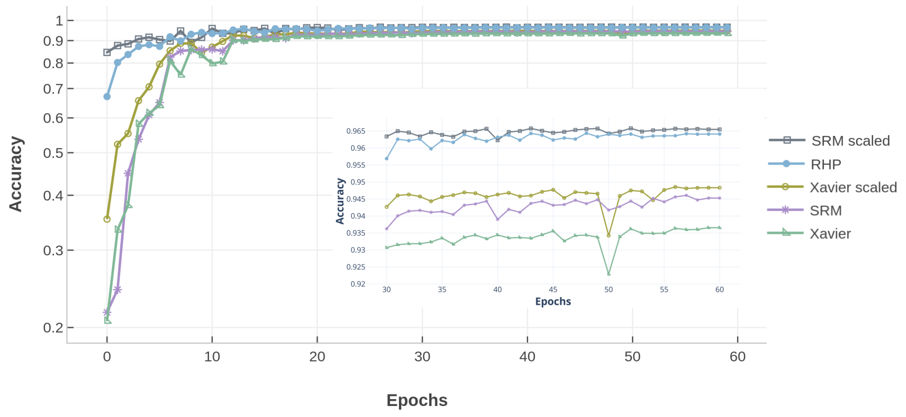

Figure 5: Evolution curves of test accuracy (average of 5 runs) for the multi-class classification problem, with the network of Bayar and Stamm [6].

Table 4: Values obtained in an example 5×5 filter with our RHP initialization approach. The value of center coefficient C is 8.4853 and a uniform distribution of $U(-0.7071, 0)$ is used for drawing pseudo-random samples for non-center coefficients.

-0.2313	-0.0666	-0.1706	-0.5250	-0.5772
-0.5182	-0.0018	-0.5467	-0.2407	-0.0559
-0.4711	-0.5040	8.4853	-0.5513	-0.3093
-0.4237	-0.1851	-0.6255	-0.5770	-0.5398
-0.2109	-0.6927	-0.6765	-0.1684	-0.0787

good behavior of performance increase, being close to the curve of scaled SRM.

It is interesting to take a look at the values in filters generated by our RHP initialization and compare with scaled high-pass filters. We carry out comparison with the initial and final values of the scaled version of a simple SRM high-pass filter with $+1$ and -1 as the only two non-zero values. This filter is scaled with a factor of 10.174 and at the end of 60 training epochs for this multi-class problem, these two non-zero coefficients have absolute values close to 8.0. Meanwhile, the center coefficient in 5×5 filter of our RHP approach has a value of 8.4853, suggesting a good amplitude of filter coefficients for RHP. The details of the initial values in one simulation of RHP are shown in Table 4.

Experiments on a variant of Bayar and Stamm's CNN with 30 filters at first layer. In this set of experiments, we change the number of first-layer filters of the network of Bayar and Stamm [6] from 3 to 30 and conduct tests still under the multi-class classification scenario. The main purpose is to perform experiments with a slightly different CNN. A minor side benefit is that with this network variant we can make use of all the available 30 SRM filters (*cf.*, link in footnote 1). Table 5 shows the obtained results. Here for Bayar, for the sake of brevity we only present results of the scaled option "Bayar B."

Table 5: Test accuracy (in %, average of 5 runs) of the multi-class classification problem, on a variant of the network of Bayar and Stamm [6] with 30 filters at first layer.

Initialization	**Original version**	**Convolution-based scaling (our)**	**Covariance-based scaling (our)**
Bayar [6] B.	94.91	96.11 (+1.20)	96.04 (+1.13)
Castillo [20]	94.11	96.31 (+2.20)	96.32 (+2.21)
SRM [16]	94.37	96.51 (+2.14)	96.49 (+2.12)
Xavier [15]	91.80	96.03 (+4.23)	96.02 (+4.22)

which has lower computational cost and achieves comparable test accuracy when compared to the other option. The results in Table 5 demonstrate again that our filter scaling approach is able to consistently improve the test accuracy. In addition, if we compare the results in Table 3 (3 first-layer filters) and Table 5 (30 first-layer filters), one interesting observation is that with 30 filters the performance of scaled Xavier is noticeably improved (94.71% vs. 96.02% with covariance-based scaling). One possible explanation is that with more filters, Xavier would have higher probability to generate an effective filter suited for manipulation detection and that the impact of such effective filters is enhanced after scaling. This is just an intuitive explanation and future work is planed for the understanding of the relationship between number of filters and the detection performance, which nevertheless is beyond the scope of this paper.

6.2. Comparison with batch normalization

The stability of the data flow in the learning process of a CNN can be achieved in different manners with the so-called proactive and reactive measures. If we want to start the learning process with a stable data flow, as a proactive measure, the initialization used for the kernel weights has to be taken care of. Differently, a reactive measure would be the usage of functions such as batch normalization [26] to re-center and re-scale our training batch during the learning stage.

For this set of experiment, we show the comparison of batch normalization, our scaling-based initialization (for Xavier [15] and SRM [16]), and our RHP initialization, in the multi-class forensic scenario with 3 filters in the first layer of the CNN of Bayar and Stamm. The network training and all other experimental settings remain unchanged as those described in Section 6.1.

Table 6 shows firstly three scenarios for Xavier and SRM initializations considering the different combinations of our scaling approach and the batch normalization technique on the first layer. Then, for our RHP initialization we show the cases with and without batch normalization for the first layer. We can see that for Xavier and SRM, our scaling approach alone works better than only using batch normalization, with an improvement of 0.25% and 1.11% respectively for Xavier (*i.e.*, 94.61% vs. 94.36%) and SRM (*i.e.*, 96.54% vs. 95.43%). It is also interesting to notice that in the case of Xavier, jointly using both techniques results in a higher accuracy. Our RHP approach also gets a small

Table 6: Test accuracy (in %, average of 5 runs) for the multi-class classification problem. The two blocks of rows named "Xavier" and "SRM" present results for our scaling-based approach and/or the batch normalization technique, when combined together or applied separately for the first layer. The block of rows "RHP init. (our)" compares our RHP approach for two cases of with and without batch normalization at the first layer.

Initialization	**Covariance-based scaling (our)**	**Batch normalization**	**Test accuracy**
Xavier [15]	Yes	Yes	**95.04**
	No	Yes	94.36
	Yes	No	94.61
SRM [16]	Yes	Yes	96.53
	No	Yes	95.43
	Yes	No	**96.54**
RHP init. (our)	-	Yes	**96.51**
	-	No	96.35

improvement when combined with batch normalization. For SRM filters, the performance is comparable when using both techniques or using only our scaling approach (*i.e.*, 96.53% vs. 96.54%). Our scaling and RHP initialization approaches are also computationally much cheaper than batch normalization, because our approaches are applied only once during initialization while batch normalization (with additional learnable parameters) should be applied on each training batch and subsequently optimized for each step during training.

These results suggest that although batch normalization is an effective technique to maintain a stable data flow within a CNN, a well-designed initialization algorithm such as our scaling proposal or our RHP approach is nevertheless of greater importance, because experimentally our initializations work better than batch normalization when applied separately. Moreover, the combination of an adequate initialization algorithm and the batch normalization technique in general can lead to a slightly higher accuracy.

6.3. Binary problem of JPEG detection with CNN of Bayar and Stamm

We observe experimentally that among all the five manipulations considered in the multi-class classification problem, JPEG compression (with very high quality factor as shown in the last row of Table 2, *i.e.*, ranging from 90 to 100) is the hardest one to be correctly detected. In this subsection, we carry out experiments on a binary classification problem of original and JPEG compressed patches with quality factor given in Table 2. Our purpose is to consider a new yet challenging manipulation detection problem for the validation of our proposed approaches. We use the network (with same training hyper-parameters and an adaptation to binary detection) and experimental data (only of original and JPEG compressed patches) specified in Section 6.1 for the experiments of this binary detection problem.

Table 7: Test accuracy (in %, average of 5 runs) for the binary JPEG compression detection problem, on the network of Bayar and Stamm [6].

Initialization	**Original version**	**Convolution-based scaling (our)**	**Covariance-based scaling (our)**
Bayar [6] B.	88.27	90.80 (+2.53)	90.80 (+2.53)
SRM [16]	78.24	92.33 (+14.09)	92.44 (+14.20)

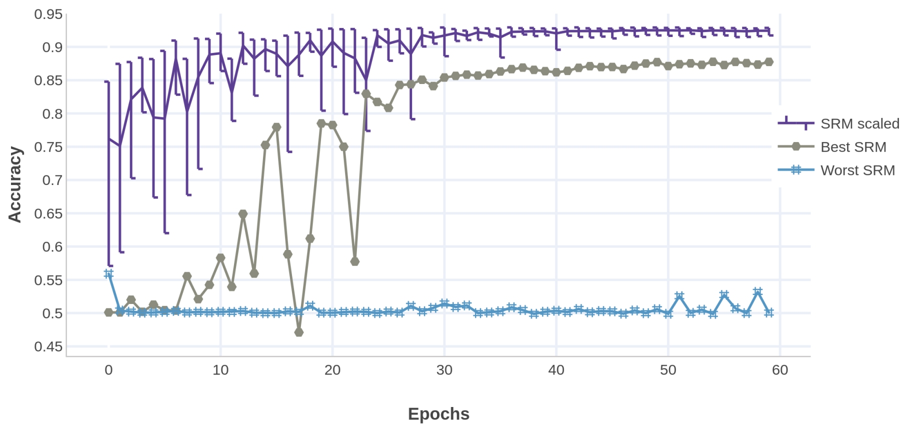

Figure 6: Evolution curves of test accuracy for the binary JPEG compression detection problem. The curve of "SRM scaled" shows minimum, average and maximum values of test accuracy among 5 runs for each epoch. The curves of "Best SRM" and "Worst SRM" show the test accuracy for respectively the best and the worst run among 5 runs of original SRM.

Our experiments are performed for Bayar [6] and SRM [16] with "Bayar B." for scaled version of Bayar. The experimental results are shown in Table 7. We can see that the original Bayar and SRM have quite limited performance for this challenging JPEG compression detection problem. In particular, the CNN training with original SRM occasionally fails (some details will be given in the next paragraph). However, after applying our filter scaling approach, the test accuracy of scaled SRM has a significant increase (of about 14%) when compared with the unscaled original version.

Figure 6 shows several evolution curves of test accuracy throughout the network training procedure, for original SRM and scaled SRM (obtained with the covariance-based method). The network training with original unscaled SRM can sometimes fail, as reflected by the curve of "Worst SRM" in Figure 6. The performance varies a lot among the 5 different runs of original SRM probably because of the randomly selected 3 SRM filters in each run. We have the empirical observation that some SRM filters tend to result in lower performance. In the meanwhile, from Figure 6 we can observe that after applying filter scaling, both the final performance and the increase speed of test accuracy are largely improved. We would like to mention once again that for a pair of original and scaled SRM, they share the same "base filters" and the only difference is the scale of the filter coefficients. This means that even for weak SRM filters which lead to CNN training failure, after applying the scaling approach on these filters we can still achieve very satisfactory forensic performance.

6.4. On the selection of SRM filters

Choosing randomly a few filters from a finite pool can sometimes result in a weaker performance than expected. As we mentioned before, there are 30 handcrafted filters available in the pool of SRM [16] filters (*cf.*, link in footnote 1). From the results in the previous subsection, we have observed that certain SRM filters perform less well than others for the image manipulation detection problem. This is the case of the results of the JPEG binary problem as shown in Table 7 and Figure 6. Among the 5 runs, one of them contained 1 *diagonal filter* which resulted in a bad forensic performance with almost random guess for test accuracy of the original unscaled version. Here diagonal filter means a filter only with non-zero coefficients in the diagonal or anti-diagonal of the filter matrix. In total, there are 10 diagonal filters in the pool of 30 SRM filters. Experimentally such diagonal SRM filters in general have a lower performance than non-diagonal ones in the considered forensic problems. In this subsection, we would like to test the "unlucky" scenario where 3 diagonal SRM filters are chosen and compare its performance with our RHP initialization approach.

For this experiment we carry out tests with the same JPEG binary classification problem as described in Section 6.3. The CNN architecture is the one designed by Bayar and Stamm [6] with 3 filters at first layer. We compare the performance of 3 diagonal SRM filters, the corresponding scaled version (convolution-based method), and our RHP proposal. We show in the second column of Table 8 the average test accuracy of 5 runs for the three different techniques at the end of the 60 epochs of training. In each run, 3 SRM diagonal filters are randomly selected as the initialization of first-layer filters. As we can observe, after scaling the selected diagonal filters, we can only obtain a rather limited average test accuracy of 87.14%. This is probably due to the limited forensic capability of these diagonal filters. By contrast, our RHP approach achieves higher average accuracy of 91.37%. We also show in the last column of Table 8 the worst test accuracy among the 5 runs for each method. The difference between the average and the worst values is the smallest for our RHP approach, which demonstrates the stability of its performance.

It is interesting to mention that for the curve of "Worst SRM" shown in Figure 6 in the experiment of the last subsection for the unscaled SRM filters, we had 1 diagonal filter among the 3 selected ones. In the experiment of this subsection, we always choose 3 diagonal SRM filters to be put at the CNN's first layer. In the former case (1 diagonal filter among 3), sometimes the training fails but after scaling the performance is much better (*e.g.*, the worst run in Figure 6). In the latter case (3 diagonal filters), we did not encounter training failure but after scaling there is less improvement. We have two possible explanations for these observations: first, the performance after scaling may be related to the overall forensic capability of all the 3 selected filters (non-diagonal SRM filters

Table 8: Comparison of test accuracy (in %) for 3 diagonal SRM filters at first layer, the corresponding scaled version and our RHP approach, for the binary JPEG compression detection problem on the network of Bayar and Stamm [6]. The second column shows the average test accuracy for 5 runs while the third column presents the worst test accuracy among the 5 runs.

Method	**Average**	**Worst run**
SRM [16] diagonal filters	84.50	82.98
SRM [16] diagonal filters with convolution-based scaling	87.14	86.49
RHP init. (our)	**91.37**	**91.25**

are stronger); second, the combined use of diagonal and non-diagonal SRM filters may cause training failure perhaps due to difficulties in achieving synergy between different kinds of filters during training. We are aware that these are only intuitive explanations, further studies are required to understand these interesting observations.

In all, from the experiments presented in this subsection, we can see that choosing randomly a few filters from a pool of handcrafted filters may present some risks, especially when the selected filters have limited forensic capability. Favorably, this uncertainty does not occur for our RHP approach.

6.5. Multi-class and binary classification problems on a different CNN

In this subsection, we carry out experiments of both manipulation detection problems (*i.e.*, the multi-class and the JPEG binary problems) on a different CNN built by ourselves. This new network does not have any fully-connected layers and has less learnable parameters than the network of Bayar and Stamm [6]. More precisely, the network in [6] has about 337K parameters while our new smaller CNN has only about 41K parameters. The fewer parameters of our new CNN are mainly due to the removal of the fully-connected layers which contain many parameters and which exist in the network of Bayar and Stamm. Here our purpose is to test the proposed initialization approaches on a different and lighter network of rather a "modern flavor" without the somehow "old fashioned" fully-connected layers. The design of better or even optimum CNN architecture for a forensic problem at hand is a very interesting working direction that we would like to explore in the future. Figure 7 shows a visual comparison of the architectures of our CNN and the CNN of Bayar and Stamm [6].

We consider both the multi-class classification problem and the binary JPEG compression detection problem in this set of experiments, with the same experimental data and configurations presented in the previous sections. We carry out comparisons of the original Bayar [6] and SRM [16], their corresponding scaled version ("Bayar B." for scaled Bayar), and our RHP approach. The obtained experimental results are shown in Table 9. First, we can observe once again that our scaling approach is able to consistently improve the forensic performance after scaling, as shown by the improved test accuracy values in parentheses in the last two columns with a maximum increase of 5.27%. Additionally, our proposed RHP approach has higher test accuracy than the original unscaled SRM

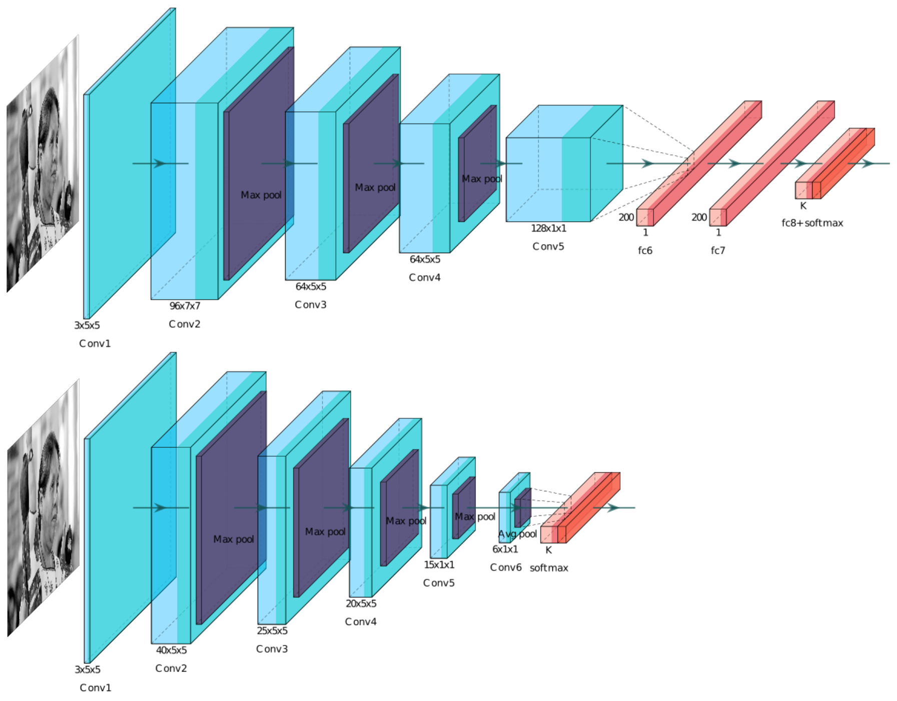

Figure 7: Image on the top shows the CNN architecture of Bayar and Stamm [6], while the architecture of our smaller CNN is depicted in the image on the bottom.

and Bayar, especially for the JPEG binary problem. When compared with the scaled versions, the RHP approach performs better than scaled Bayar in both multi-class and JPEG binary problems. Scaled SRM achieves higher accuracy for both problems than the RHP approach. The latter has nevertheless its own advantages: different from the data-dependent scaling approach, the RHP initialization does not explicitly use input data and thus has higher flexibility, lower computational cost and broader application range. In addition, as shown in the previous subsection, it may present some risks when randomly choosing handcrafted SRM filters for initializing convolutional kernels in a CNN.

One interesting observation, when comparing the results on the CNN of Bayar and Stamm [6] (Table 3 and Table 7) and on our smaller network (Table 9), is that the latter in general provides better performance. It is interesting to investigate on the potential connections between the network architecture and the forensic performance. Nevertheless such investigation is beyond the focus of this paper because our main purpose in this subsection is to validate the effectiveness of our scaling and RHP approaches on a different CNN.

6.6. Experiments on another image database with additional settings

In order to further prove the effectiveness of the proposed approaches, we carry out additional experiments on another image database (the RAISE database

Table 9: Test accuracy (in %, average of 5 runs) for the multi-class and JPEG binary classification problems, on a smaller CNN built by ourselves.

Problem	**Original version**	**Convolution-based scaling (our)**	**Covariance-based scaling (our)**
Bayar [6] B.			
Multi-class	95.24	96.17(+0.93)	96.18(+0.94)
JPEG binary	90.56	93.72(+3.16)	93.74(+3.18)
SRM [16]			
Multi-class	95.72	97.06(+1.34)	97.09(+1.37)
JPEG binary	89.85	95.11(+5.26)	95.12(+5.27)
RHP init. (our)			
Multi-class	**96.76**	-	-
JPEG binary	**94.45**	-	-

[27]) with additional settings of the size of input patches and the size of first-layer filters. We consider the binary JPEG classification problem for the experiments on RAISE database. For the data preparation, we first randomly select 1491 RAISE images (same number as the Dresden images used in previous experiments), which are saved in lossless `.TIF` format and never processed. RAISE images have high resolutions mostly with typical sizes of 4288×2848 or 4928×3264. As before, we randomly split these 1491 unprocessed RAISE images into training, validation and testing sets with ratio of 3:1:1 and then convert the images to grayscale. Afterwards, the preparation of patch samples is the same as for the Dresden database, with the same number of samples and the same high-quality JPEG compression applied as manipulation (with random JPEG quality factor between 90 and 100). The difference is that here we extract patches of different sizes and carry out classification on these different sets of patches. We use the network of Bayar and Stamm [6], and the training procedure and parameters are the same as those specified in subsection 6.1.

In a first group of experiments, we check the forensic performance on input patches of different sizes (256×256, 64×64 and 32×32 pixels), for SRM [16], scaled SRM (our first approach, convolution-based), and the RHP initialization (our second approach). The striding and pooling of the one or two convolutional layers after the first-layer filters are slightly adjusted as so to accommodate the network to input patches of 32×32 or 256×256 pixels. The obtained results of test accuracies are presented in Table 10. We can observe that the test accuracy becomes higher as the patch size increases. This is understandable because larger patches contain more information, which is more favorable for forensic classification. Probably for the same reason, we find that the training of SRM [16] has more chance to fail on small patches of 32×32 pixels. In this sense, the improvement after applying our first scaling approach on SRM is larger for input patches of smaller sizes (please compare the rows of "SRM [16] (all 5 runs)" and "SRM scaled (our)" in Table 10). Furthermore, in order to have a fair analysis

Table 10: Test accuracy (in %) on the RAISE database [27] with different sizes of input patches of 256×256, 64×64 and 32×32 pixels, for the JPEG binary classification problem by using the network of Bayar and Stamm [6]. The results are average of 5 runs, except for the row of "SRM [16] (three best)" where we show the average of the three highest test accuracy values (with successful convergence) among the 5 runs of the unscaled SRM algorithm [16]. For the approach of "SRM scaled (our)" we present in the parentheses the accuracy improvement when compared to the corresponding result in row of "SRM [16] (three best)".

Patch size	256×256	64×64	32×32
SRM [16] (all 5 runs)	76.24	75.30	70.17
SRM [16] (three best)	89.54	86.50	83.34
SRM scaled (our)	92.74(+3.20)	89.72(+3.22)	86.55(+3.21)
RHP init. (our)	90.84	89.18	84.33

of the performance improvement, we show the average test accuracy of three best runs (all with successful convergence) of the original unscaled SRM in the row of "SRM [16] (three best)". Then, considering the average of three best runs as baseline, we report the test accuracy improvement of our first scaling approach in parentheses in the row of "SRM scaled (our)". We can observe that the increase of test accuracy obtained by scaling SRM filters is quite stable for different patch sizes, and this demonstrates the stability and effectiveness of our first approach. It can be noticed that the performance on 64×64 patches is rather comparable between RAISE database and Dresden database, with a slightly lower performance on RAISE database. Our explanation is that in general RAISE images are of higher resolution than Dresden images. The higher resolution of RAISE images makes the modification introduced by manipulation smaller and more difficult to detect. At last, similar to the results of the experiments on Dresden database, the performance of our second RHP approach (the last row of Table 10) is quite satisfying and slightly lower than the scaled SRM obtained by our first approach.

In a second group of experiments, we compare the forensic performance for different sizes of first-layer filters of 3×3, 5×5 and 7×7. The padding of the convolutional layer right after the first-layer filters is slightly changed to accommodate the network architecture to filter size of 3×3 and 7×7. We still consider SRM [16], scaled SRM and RHP initialization for the experiments. For SRM, we only consider the 17 filters of size smaller than or equal to 3×3, among the pool of 30 SRM filters, for the experiments with 3×3 first-layer filters. For 7×7 first-layer filters we carry out zero-padding to change the size of the SRM filters to 7×7. The obtained results are shown in Table 11. We can observe that the scaled SRM (our first approach) and the RHP initialization (our second approach) behave consistently well with stable performances under different sizes of first-layer filters. For all the three cases our two approaches perform better than unscaled SRM as shown in Table 11, and the test accuracy can be consistently improved after applying our first scaling approach on original SRM. Another observation is that the original unscaled SRM algorithm performs poorly for 7×7 filters (all 5 runs fail to converge), while it has stable and

Table 11: Test accuracy (in %, average of 5 runs) on the RAISE database [27] with different sizes of first-layer convolutional filters of 3×3, 5×5 and 7×7, for the JPEG binary classification problem by using the network of Bayar and Stamm [6], on 64×64 patches.

Filter size	3×3	5×5	7×7
SRM [16]	87.94	75.30	50.12
SRM scaled (our)	90.03	89.72	89.77
RHP init. (our)	89.31	89.18	89.07

satisfying performance with 3×3 filters (all 5 runs successfully converge). In all, we see some very interesting observations through the additional experiments on RAISE database as presented in this subsection. We leave the understanding of these interesting observations as future work.

6.7. Detection of images created by Generative Adversarial Networks

Finally we consider a new forensic problem and carry out tests with another deep and popular network. GAN (Generative Adversarial Network) models have recently been on the news for their capability of creating fake yet visually very realistic images and videos commonly known as deepfakes. This problem has been on the rise in part due to the realistic GAN-generated videos impersonating political or other popular figures. These deepfakes could bring distress to the society if created with malicious purposes. Conventional image tampering operations (*e.g.*, splicing, copy-move and inpainting) are mostly carried out by using advanced image processing and editing tools (*e.g.*, Photoshop and GIMP), while recently GANs have provided a new machine-learning-based way to generate fake images (*i.e.*, the so-called deepfakes). Both are effective approaches to creating fake images with altered or fictive semantic meaning, but they may leave different traces in the created fake images. In this subsection, we are interested in detecting deepfake images whose contents are generated by GANs which aim to mimic the properties of real images used during the network training. We choose to carry out experiments with this deepfake detection problem to show a different topic in the image forensics research community and demonstrate the efficiency of our proposal when tested on a well-known CNN architecture.

In this experiment we conduct tests for a binary classification scenario to detect between GAN-generated and natural images by considering different groups of deepfake images generated by different GAN tools. For this setting we use as reference the recent work of Wang *et al.* [28], where the popular ResNet50 is used to analyze the artifacts left by GANs. We use the same dataset and code of [28][2] in order to compare in a fair manner, while only affecting the initialization method on the first layer of the network.

The main dataset of deepfake images for this experiment is generated with the ProGAN network [29]. Figure 8 shows examples of some image categories

[2]Code and dataset are available at `https://github.com/peterwang512/CNNDetection`.

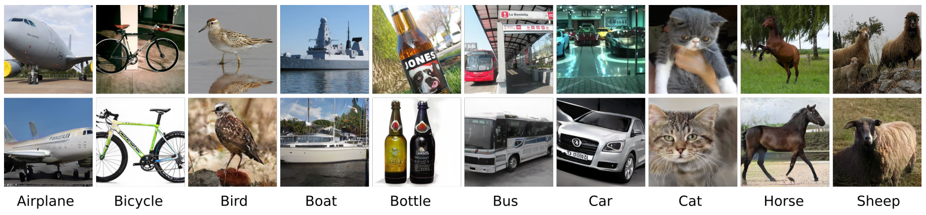

Figure 8: Examples for real and GAN-generated images of different categories in a binary classification scenario. Top row shows real images while deepfake ones are shown below. All images come directly from the shared dataset of [28] where the GAN-generated ones were created with ProGAN network [29].

with the top row for natural examples and the bottom row for the GAN-generated ones. There are in total 20 categories of images (Airplane, Boat, Cat, Horse, *etc.*), and each category comprises 18,003 images of 256×256 pixels for each class of natural or deepfake images. This makes a total of 720,120 images for the training set. The validation and testing sets corresponding to the ProGAN network contain respectively 8,000 images, with 200 images from each class in each category.

Following the strategy proposed by Wang *et al.* [28], we use the popular ResNet50 as CNN architecture. In their original proposal, the network is pre-trained with ImageNet [30]. Then the authors conduct further training of the pre-trained ResNet50 on the training set of the deepfake binary classification problem in which the fake images are created by the ProGAN network. In order to test our method we initialize the first layer of the pre-trained ResNet50 with our RHP initialization. This first layer contains 64 filters of shape 7×7. All other layers (initialized with pre-trained ImageNet weights for both Wang *et al.*'s method [28] and ours) and all experimental settings are the same as those used in [28]. Therefore the only difference is the initialization of the first-layer filters, *i.e.*, ImageNet pre-trained weights for the state-of-the-art method of Wang *et al.* [28] and RHP initialization for our method.

Experimentally we use Adam optimizer with a starting learning rate of $1e{-}4$ as proposed by the authors of [28]. The code used for this experiment is the one shared on-line by the same authors. We use the provided training scripts where Gaussian blur and JPEG compression are considered as data augmentation techniques during the training stage. JPEG compression is performed with a quality factor taken from a uniform distribution on the set of $\{30, 31, ..., 100\}$. Blurring operation is performed with standard deviation parameter σ taken from a uniform distribution within the interval $[0, 3]$. Both of these techniques are applied with a pre-defined percentage of probability. Original results by Wang *et al.* [28] proved that by using these data augmentation techniques good *generalization* performance can be obtained on testing sets of deepfake images created by tools other than ProGAN. In fact as presented above, fake images in the training and validation sets are all generated by the ProGAN network. Here the generalization capability of a trained CNN model means the detection

Table 12: Testing sets used for the evaluation of the generalization capability of detecting GAN-generated images.

Model	Number of images
ProGAN [29]	8,000
StyleGAN [31]	12,000
BigGAN [32]	4,000
CycleGAN [33]	2,600
StarGAN [34]	4,000
GauGAN [35]	10,000
CRN [36]	12,800
IMLE [37]	12,800
SITD [38]	360
SAN [39]	440
Deepfake [40]	5,400

performance of the model on "unseen" testing data of deepfake images generated by GAN tools that remain unknown during the training phase.

The results obtained by Wang *et al.* [28] show a final accuracy of about 100.0% on the training and validation sets. This is also the case when we initialize the first layer with our RHP approach. There is no room for improvement regarding this point. Nevertheless, we can observe performance differences when we compare the generalization capability of the trained networks. In order to test the generalization results, we consider 11 different testing sets coming from a number of state-of-the-art GANs for creating deepfake images with style transfer applied to a set of source images. The network architectures and training settings of these GANs are all different. The resulting number of images for each testing set is shown in Table 12. These testing sets are shared on-line by Wang *et al.* [28].

To make a fair comparison, we follow the original idea of Wang *et al.* [28] to train the networks with the ProGAN dataset, and then test the generalization capability of trained networks on the different testing sets listed in Table 12. This represents a real-world scenario where diverse and unknown GAN tools with different characteristics are tested after training CNNs on data from a unique tool (here the ProGAN network). As mentioned previously and reported in [28] data augmentation techniques can improve the generalization performance, and we apply these techniques in the experiments. Table 13 presents the generalization results with two different data augmentation options: "Case A." where only blurring is applied with 50% of probability; "Case B." where both blurring and JPEG compression are applied with a probability of 50%. In the table, we show the results obtained by Wang *et al.* [28] where the first layer is initialized with pre-trained ImageNet weights and those obtained by our RHP initialization.

As we can see from the results in Table 13, initializing the first layer with

Table 13: Generalization results for the different testing sets with comparisons between Wang *et al.*'s [28] ImageNet pre-trained weights and our RHP initialization on the first layer. Please refer to main text for the meaning of "Case A." and "Case B." of data augmentation options. We show the better result for each testing set and each case in bold. For all scenarios networks were trained with the ProGAN dataset. Following [28], results are reported in terms of Average Precision (in %) on individual testing set and in terms of mean Average Precision (mAP, in %) for the overall performance on all testing sets.

	Case A.		**Case B.**	
Testing set	**ImageNet weights init.**	**Our RHP init.**	**ImageNet weights init.**	**Our RHP init.**
ProGAN [29]	**100.0**	**100.0**	**100.0**	**100.0**
StyleGAN [31]	**99.0**	98.8	98.5	**98.6**
BigGAN [32]	82.5	**84.7**	88.2	**89.0**
CycleGAN [33]	90.1	**93.5**	96.8	**97.0**
StarGAN [34]	**100.0**	**100.0**	95.4	**96.3**
GauGAN [35]	74.7	**75.7**	**98.1**	**98.1**
CRN [36]	66.6	**73.6**	98.9	**99.4**
IMLE [37]	66.7	**73.9**	**99.5**	**99.5**
SITD [38]	**99.6**	**99.6**	92.7	**95.8**
SAN [39]	53.7	**53.9**	63.9	**66.1**
Deepfake [40]	**95.1**	93.4	66.3	**69.3**
mAP	84.4	**86.1**	90.8	**91.7**

our RHP approach leads to a better performance for the majority of the testing sets. Following the original paper of Wang *et al.* [28], we use the Average Precision and mean Average Precision (mAP) as respectively the performance metric for an individual testing set and the overall performance metric on all testing sets (for both metrics, a higher value means a better performance). In both "Case A." and "Case B." the mean Average Precision is slightly better for our method, as shown in the last row of Table 13. One possible explanation is that our initialization can make the network more oblivious to image content and more sensitive to the traces of GANs which are likely to be in the high-frequency component of images.

6.8. Discussion

In this section, we have demonstrated through experiments the effectiveness of the proposed CNN initialization approaches, on forensic problems of image manipulation detection and deepfakes detection. In the future, we plan to explore the utility of the proposed approaches in other practical application scenarios related to image forensics. First, when we know what original images should be and are very strict about the manipulations (*i.e.*, any image processing operation is considered suspicious and not tolerable, for example in law enforcement), our approaches could be directly applied in a sliding window manner to locate the suspicious patches. Second, even if routine manipulations can be tolerated, it is in general believed that there may still exist special forgery traces within the fake region and near the fake region boundary, in for example splicing and copy-move forgeries [41, 42]. Therefore, still with the application of our approaches on small local patches in a sliding window manner, additionally combined with an efficient clustering algorithm on the extracted features of a manipulation detection CNN, we may be able to locate the fake regions in image forgeries. Third, the manipulation detection approaches can be jointly utilized with existing forgery detectors. For instance, after identifying a pair of copy-moved regions by an existing detector, we can apply manipulation detection on the two regions to determine which one is the so-called source (*i.e.*, original) region and which one is the target (*i.e.*, fake) region. Usually the fake region has been subject to routine manipulations such as scaling and smoothing [43, 4]. We are also interested in investigating the similarities and differences between the detection of fake images created by the conventional tampering operations (splicing, copy-move, inpainting, *etc.*) and the detection of deepfakes generated by GANs. For instance, different high-pass initializations, derived based on either global or local differences between real and fake images, may be beneficial for the detection of fake images fabricated by different tools. Moreover, we would like to go one step further and extend the proposed approaches to other image analysis tasks beyond forensics. For this purpose, the key issue would be the design of appropriate and application-oriented filters which are not necessarily high-pass ones. Proper scaling can be applied on the designed filters (for our first approach), or the filter design can be enhanced by also incorporating the variance stability formulation (for our second approach).

7. Conclusion

In this paper, we first carry out theoretical and experimental studies to analyze the behavior of common CNN initializations in an image manipulation detection scenario and show that commonly used initialization algorithms result in a signal shrinkage at filter output which may affect the overall performance of a network. We show that besides the high-pass initializations, the output signal shrinkage also occurs for the conventional Xavier initialization. We also present theoretical proofs to explain the somewhat surprising results of the Xavier algorithm. We then propose two new initialization approaches for the detection of image manipulation operations. The first proposal is a data-dependent approach which deduces an appropriate factor for the scaling of CNN's first-layer convolutional filters. The second proposal is a random high-pass initialization approach which does not need to carry out explicit computation on input data. Both approaches make use of a corrected and realistic assumption on natural image statistics which existing methods do not take into account. Through a series of experiments with different CNN architectures and different detection problems we show good forensic performances for our two initialization approaches. We believe that each of our proposals has its own benefits depending on the problem and the application scenario at hand. In the future we would like to perform further tests with more CNN architectures such as LSTM (Long Short-Term Memory) or multi-branch models. Another possible working direction is to extend our work to more layers and to study the synergy between different layers. At last, we are interested in enlarging the application range of the proposed approaches as discussed in subsection 6.8.

Acknowledgment

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] A. Piva, An overview on image forensics, ISRN Signal Processing (2013) 496701:1–496701:22.

[2] L. Verdoliva, Media forensics and deepfakes: an overview, IEEE Journal of Selected Topics in Signal Processing 14 (5) (2020) 910–932.

[3] M. Forelle, P. Howard, A. Monroy-Hernández, S. Savage, Political bots and the manipulation of public opinion in Venezuela, arXiv CoRR (2015) 1–8. URL `https://arxiv.org/abs/1507.07109`

[4] I. Castillo Camacho, K. Wang, A comprehensive review of deep-learning-based methods for image forensics, Journal of Imaging 7 (4) (2021) 69:1–69:39.

[5] J. Chen, X. Kang, Y. Liu, Z. J. Wang, Median filtering forensics based on convolutional neural networks, IEEE Signal Processing Letters 22 (11) (2015) 1849–1853.

[6] B. Bayar, M. C. Stamm, Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection, IEEE Transactions on Information Forensics and Security 13 (11) (2018) 2691–2706.

[7] I. Castillo Camacho, K. Wang, Data-dependent scaling of CNN's first layer for improved image manipulation detection, in: Proceedings of the International Workshop on Digital-forensics and Watermarking, Melbourne, Australia, 2020, pp. 208–223.

[8] M. Kirchner, J. Fridrich, On detection of median filtering in digital images, in: Proceedings of the SPIE Media Forensics and Security II, Vol. 7541, San Jose, CA, USA, 2010, pp. 754110:1–754110:12.

[9] X. Kang, M. C. Stamm, A. Peng, K. J. R. Liu, Robust median filtering forensics using an autoregressive model, IEEE Transactions on Information Forensics and Security 8 (9) (2013) 1456–1468.

[10] Z. Fan, R. L. de Queiroz, Identification of bitmap compression history: JPEG detection and quantizer estimation, IEEE Transactions on Image Processing 12 (2) (2003) 230–235.

[11] W. Luo, J. Huang, G. Qiu, JPEG error analysis and its applications to digital image forensics, IEEE Transactions on Information Forensics and Security 5 (3) (2010) 480–491.

[12] A. C. Popescu, H. Farid, Exposing digital forgeries by detecting traces of resampling, IEEE Transactions on Signal Processing 53 (2) (2005) 758–767.

[13] H. Li, W. Luo, X. Qiu, J. Huang, Identification of various image operations using residual-based features, IEEE Transactions on Circuits and Systems for Video Technology 28 (1) (2018) 31–45.

[14] W. Fan, K. Wang, F. Cayre, General-purpose image forensics using patch likelihood under image statistical models, in: Proceedings of the IEEE International Workshop on Information Forensics and Security, Roma, Italy, 2015, pp. 1–6.

[15] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feed-forward neural networks, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 2010, pp. 249–256.

[16] J. Fridrich, J. Kodovský, Rich models for steganalysis of digital images, IEEE Transactions on Information Forensics and Security 7 (3) (2012) 868–882.

[17] B. Chen, H. Li, W. Luo, Image processing operations identification via convolutional neural network, arXiv CoRR (2017) 1–8.
URL `https://arxiv.org/abs/1709.02908`

[18] Y. Liu, Q. Guan, X. Zhao, Y. Cao, Image forgery localization based on multi-scale convolutional neural networks, in: Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, Innsbruck, Austria, 2018, pp. 85–90.

[19] H. Li, J. Huang, Localization of deep inpainting using high-pass fully convolutional network, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 8301–8310.

[20] I. Castillo Camacho, K. Wang, 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, 2019, pp. 107–112.

[21] F.-F. Li, J. Johnson, S. Yeung, Neural networks part 2: Setting up the data and the loss, (Course notes of Stanford University "CS231n: Convolutional Neural Networks for Visual Recognition", visited on 2021-05-19) (2018).
URL `http://cs231n.github.io/neural-networks-2/`

[22] E. P. Simoncelli, B. A. Olshausen, Natural image statistics and neural representation, Annual Review of Neuroscience 24 (1) (2001) 1193–1216.

[23] A. Hyvärinen, J. Hurri, P. Hoyer, Natural Image Statistics: A Probabilistic Approach to Early Computational Vision, Springer-Verlag, London, UK, 2009.

[24] T. Gloe, R. Böhme, The Dresden image database for benchmarking digital image forensics, in: Proceedings of the ACM Symposium on Applied Computing, Sierre, Switzerland, 2010, pp. 1584–1590.

[25] L. A. Goodman, On the exact variance of products, Journal of the American Statistical Association 55 (292) (1960) 708–713.

[26] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the International Conference on Machine Learning, Lille, France, 2015, pp. 448–456.

[27] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, G. Boato, RAISE: A raw images dataset for digital image forensics, in: Proceedings of the ACM Multimedia Systems Conference, 2015, pp. 219–224.

[28] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, A. A. Efros, CNN-generated images are surprisingly easy to spot... for now, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8695–8704.

[29] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of GANs for improved quality, stability, and variation, in: Proceedings of the International Conference on Learning Representations, 2018, pp. 1–26.

[30] J. Deng, W. Dong, R. Socher, L. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[31] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.

[32] A. Brock, J. Donahue, K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, in: Proceedings of the International Conference on Learning Representations, 2019, pp. 1–35.

[33] J.-Y. Zhu, T. Park, P. Isola, A. A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2223–2232.

[34] Y. Choi, M. Choi, M. Kim, J. Ha, S. Kim, J. Choo, StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 8789–8797.

[35] T. Park, M. Liu, T. Wang, J. Zhu, Semantic image synthesis with spatially-adaptive normalization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2337–2346.

[36] Q. Chen, V. Koltun, Photographic image synthesis with cascaded refinement networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1511–1520.

[37] K. Li, T. Zhang, J. Malik, Diverse image synthesis from semantic layouts via conditional IMLE, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 4220–4229.

[38] C. Chen, Q. Chen, J. Xu, V. Koltun, Learning to see in the dark, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 3291–3300.

[39] T. Dai, J. Cai, Y. Zhang, S. Xia, L. Zhang, Second-order attention network for single image super-resolution, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11065–11074.

[40] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, M. Nießner, Faceforensics++: Learning to detect manipulated facial images, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1–11.

[41] R. Salloum, Y. Ren, C.-C. K. Kuo, Image splicing localization using a multi-task fully convolutional network (MFCN), Journal of Visual Communication and Image Representation 51 (2018) 201–209.

[42] P. Zhou, B. Chen, X. Han, M. Najibi, A. Shrivastava, S. Lim, L. Davis, Generate, segment, and refine: Towards generic manipulation segmentation, in: Proceedings of the Association for the Advancement of Artificial Intelligence Conference, 2020, pp. 13058–13065.

[43] D. Cozzolino, G. Poggi, L. Verdoliva, Efficient dense-field copy-move forgery detection, IEEE Transactions on Information Forensics and Security 10 (11) (2015) 2284–2297.