A Simple and Effective CNN Initialization for Forensic Detection of Image Processing Operations

Ivan CASTILLO CAMACHO, Kai WANG

Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab
11 rue des Mathématiques, BP 46, 38402 Saint Martin d'Hères Cedex, France
ivan.castillo-camacho@gipsa-lab.grenoble-inp.fr, kai.wang@gipsa-lab.grenoble-inp.fr

Résumé – Dans cet article, nous présentons une nouvelle méthode d'initialisation pour la première couche de réseaux de neurones convolutionnels (CNNs). Avec la nouvelle initialisation, nous obtenons un ensemble de filtres passe-haut tout en conservant la stabilité statistique de l'entrée et de la sortie. Expérimentalement, la méthode nous permet de mieux identifier les opérations de traitement d'image couramment utilisées pour altérer une image dans un scénario réel.

Abstract – In this paper we present a simple yet effective initialization method for the first layer of convolutional neural networks (CNNs). The proposed initialization gives a set of high-pass filters while keeping the statistical stability of the input and output. Experimentally, this allows us to better identify image processing operations that would normally be used in order to tamper an image in a real scenario.

1 Introduction

Image forensics [1] is a technique that analyzes traces in image data in an attempt to assess the image's authenticity. Identifying whether an image has been modified or not is important because tampered images can have serious consequences on society, *e.g.*, misleading the public opinion or even hindering law enforcement if a doctored picture is taken as a "convincing" proof in court.

In a real tampering scenario, there are several basic operations that can be performed on an image in order to modify it, such as median filtering, resampling and noise addition. Following several recent studies [2, 3, 4, 5], we focus on the detection of these image processing operations which would provide useful hints on exposing image forgeries. To solve this forensic problem, majority of existing methods follow a common strategy. First, local pixel difference or residuals are computed through an adequate filtering to remove or suppress the image content. Then the pixel difference or residuals are taken as the input to a feature extractor followed by classifier training, or more recently to a convolutional neural network (CNN) for joint feature learning and classification. In the recent and effective CNN-based methods [3, 5], the authors put at the beginning of CNN either a *fixed* layer of residual extraction or a *constrained* layer that is forced to learn a group of Laplacian-like filters. We think that such a fixed or constrained first layer is rather restrictive and therefore may limit the learning capacity of CNN, leading to non-optimal forensic detection accuracy. Accordingly, we suggest that it would be a better solution to set up a good initialization for the first layer and then to freely train the CNN. The proposed very simple initialization is able to give better forensic performance when compared with the existing methods.

The rest of our paper is organized as follows. In Section 2 we describe the proposed initialization scheme for the first layer of CNN. The experimental results, comparisons and analysis are presented in Section 3. Finally, the conclusion remarks are drawn in Section 4.

2 The Proposed Initialization

Our method was inspired by the milestone work of Glorot and Bengio [6], also well known as the *Xavier initialization*. More precisely, for initializing filter weights in a CNN, a mathematically rigorous study has been performed which takes into account the stability of the *variance* of the signal of CNN data flow. This helps the signal to reach deep into the network. Indeed, if the filter weights in a network start too small (resp. too large), then the signal shrinks (resp. grows) as it passes through each layer until it becomes too tiny (resp. too massive) to be useful for the network training [6].

In a CNN, each filter (also called a kernel) performs locally a weighted sum of the input data "seen" by the kernel, using the kernel values as weights [7, 8]. Let us assume that the filter comprises n scalars denoted by $\mathrm{W} = (w_1, w_2, ..., w_n)$ (here considered as a group of random variables), then accordingly the input data "seen" locally by this filter also comprises n scalar random variables, denoted by $\mathrm{X} = (x_1, x_2, ..., x_n)$. Now it can be seen that the output y of a filter is simply the inner product between W and X. We can therefore easily derive the variance of y as given in Eq. (1), where the second equality holds under mild conditions that the n random variables

w_1	w_2	w_3
w_4	C	w_5
w_6	w_7	w_8

x_1	x_2	x_3
x_4	x_9	x_5
x_6	x_7	x_8

FIGURE 1 – Shape and notations of the initialized filter (left) and the corresponding input (right), with a 3×3 filter as example.

$w_{i,i=1,2,...,n}$ (resp. $x_{i,i=1,2,...,n}$) follow zero-mean independent and identical distribution (iid) and that w_i and x_i are mutually independent [7, 8].

$$\begin{aligned} \mathrm{Var}(y) &= \mathrm{Var}(w_1x_1 + w_2x_2 + ... + w_nx_n) \\ &= n\mathrm{Var}(w_i)\mathrm{Var}(x_i). \end{aligned} \tag{1}$$

The basic idea of Xavier initialization is to keep the variance of the filter output the same as that of the filter input, so that the data flow keeps stable throughout the CNN, and this helps to facilitate the network training. From the above analysis, we can see that this requirement means that the term $n\mathrm{Var}(w_i)$ in Eq. (1) should be equal to 1, *i.e.*, we have $\mathrm{Var}(w_i) = \frac{1}{n}$. Once we know the variance of w_i, we can draw values from a simple and appropriate distribution (*i.e.*, a uniform or a Gaussian distribution) to be the initial weights of the filter.

Recall that in this paper we want to detect the traces of image processing operations. Although CNNs give a promising methodology towards automatic feature learning, in their current form (mostly from the computer vision community), they are not fully suitable for forensic problems. This is on the grounds that current neural networks tend to learn features representative of an image's content, contrary to forensic traces which are rather content-free and reside towards the high-frequency part of the images [3, 5]. The Xavier initialization is one of the possible reasons to explain this observation, as in general the initialized filters in the first layer are not high-pass and not sensitive to forensic traces. Therefore, our objective here is to design a new initialization method which is able to keep the variance stability of filter's input and output, and at the same time capable of generating high-pass initialized filters. To this end, we make adaptation to the original Xavier initialization as explained in the following.

We first of all choose a simple "template" for the high-pass filters to be initialized, which is shown in Fig. 1 left. The filters in fact can have different sizes in different applications. In the case of a 3×3 filter, we divide the 9 elements into two groups: The first group is the center element with an unknown constant value C, while the remaining $\tilde{n} = 8$ elements are all scalar random variables following a tractable and adequate distribution. Both the value of C and the statistical properties of the random variable distribution are to be derived.

The derivation is based on the variance stability of input and output of our filter, as detailed below. First, in order to have a high-pass filter after initialization, we require that the mathematical expectation of $w_{i,i=1,2,...,\tilde{n}}$ should be equal to $-\frac{C}{\tilde{n}}$ (*i.e.*, $\mathrm{E}(w_i) = -\frac{C}{\tilde{n}}$), so that the expectation of the sum of all the $\tilde{n}$ elements is equal to $-C$, together with the center element forming a high-pass filter (see Fig. 1 left). For the filter input, we set the notations as shown in Fig. 1 right. Afterwards, by applying the variance properties of the sum and the product of random variables, we can compute the variance of the filter output y, for the general case where the number of non-constant filter elements is $\tilde{n}$, as:

$$\begin{aligned} \mathrm{Var}(y) &= \mathrm{Var}(w_1x_1 + w_2x_2 + ... + w_{\tilde{n}}x_{\tilde{n}} + C.x_{\tilde{n}+1}) \\ &= \mathrm{Var}(x_i)\left[C^2 + \frac{C^2}{\tilde{n}} + \tilde{n}.\mathrm{Var}(w_i)\right]. \end{aligned} \tag{2}$$

Here we still assume that w_i and x_i are mutually independent and follow iid. To have the second equality, we have mainly used the variance property of the product of two independent random variables [9] as detailed in the equation below:

$$\begin{aligned} \mathrm{Var}(w_ix_i) &= [\mathrm{E}(w_i)]^2\mathrm{Var}(x_i) + [\mathrm{E}(x_i)]^2\mathrm{Var}(w_i) + \mathrm{Var}(w_i)\mathrm{Var}(x_i) \\ &= \frac{C^2}{\tilde{n}^2}\mathrm{Var}(x_i) + 0.\mathrm{Var}(w_i) + \mathrm{Var}(w_i)\mathrm{Var}(x_i). \end{aligned}$$

Now, because we want to have the same variance for the input and output, it can be seen that the term inside the square brackets in Eq. (2) should be equal to 1. Then the variance of w_i can be obtained as:

$$\mathrm{Var}(w_i) = \frac{1 - C^2 - \frac{C^2}{\tilde{n}}}{\tilde{n}}. \tag{3}$$

In the meanwhile, we need to choose an adequate distribution of w_i, which should also be as simple as possible for the sake of easy numerical sampling. To this end, for w_i we choose in our case one of the simplest distributions with a prescribed expectation of $-\frac{C}{\tilde{n}}$ (to have a high-pass filter as discussed above), *i.e.*, the uniform distribution within the interval $[-\frac{2C}{\tilde{n}}, 0]$ or $[0, -\frac{2C}{\tilde{n}}]$, depending on the chosen sign of C. Therefore, the variance of w_i can be easily calculated as (with C being positive as example, but the final result is the same regardless of the sign of C):

$$\mathrm{Var}(w_i) = \int_{-\frac{2C}{\tilde{n}}}^{0} \left(x + \frac{C}{\tilde{n}}\right)^2 \frac{\tilde{n}}{2C} dx = \frac{C^2}{3\tilde{n}^2}. \tag{4}$$

By equalizing Eqs. (3) and (4), we obtain a simple quadratic equation which is guaranteed to have two valid real roots as:

$$(4 + 3\tilde{n})C^2 - 3\tilde{n} = 0 \quad \Longrightarrow \quad C = \pm\sqrt{\frac{3\tilde{n}}{4 + 3\tilde{n}}}. \tag{5}$$

This nicely implies that our initialization can be applied to arbitrary size of filters which comprise $\tilde{n}+1$ elements and which follow the template that we have chosen. In the case of 3×3 filter, the center element C is equal to $\pm\sqrt{\frac{6}{7}}$. Accordingly, we can easily get the uniform distribution for the numerical sampling of w_i. We initialize the CNN's first layer by using the value of C and drawing samples of w_i to build initialized high-pass filters, and the remaining layers are initialized with the conventional Xavier initialization. This is illustrated in Fig. 2. As shown in the next section, our method copes well with two CNNs to solve two different forensic problems of detecting image processing operations, with comparable or slightly better performance than existing methods.

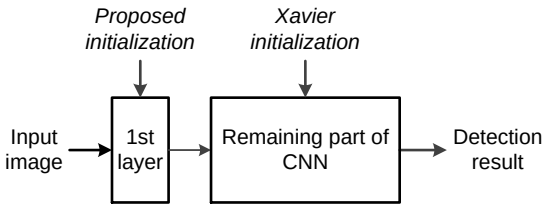

FIGURE 2 – Our CNN initialization scheme for forensic detection of image processing operations.

3 Experimental Results

In this section we present two sets of experiments that we performed to test and validate the proposed initialization. In order to ensure fair comparisons and show the utility of our method, we carry out experiments for the two forensic problems considered in [5] and [3]. We first test a multiclass forensic problem for identifying a group of image processing operations [5], and then conduct experiments for median filtering forensics with JPEG post-processing [3]. For fair comparisons, for each problem, we use CNN architectures from the original papers [5, 3] and apply our initialization at the first layer of these networks. The remaining layers of our CNNs are initialized with conventional Xavier initialization. In both experiments, for the CNN with our initialization, we use Adam [10] optimizer to train the CNN, and we let the network learn for 25 epochs. The implementation and the test of our method were based on Tensorflow® v1.7.0 with two Nvidia® GPUs of GeForce GTX 1080 Ti.

3.1 Multiclass Forensics

In this first experiment, we consider a multiclass classification problem. Following [5], our objective is to correctly classify six kinds of image patches: the unprocessed pristine patches and the processed patches by respectively five different operations listed in Table 1. Note that our parameter setting of operations is more challenging than those considered in [5]. The experiment was conducted on the Dresden image database [11], which comprises 1491 unprocessed high-resolution images and which was also used in [5]. The Dresden images were converted to be grayscale by the Python `rgb2gray` function and then divided into three groups with the ratio of 3:1:1, respectively for training, validation and testing. We conduct tests on patches of 64×64 pixels drawn from full-sized Dresden images. There are $100,008$ patches in the training set, *i.e.*, $16,668$ patches for each of the six classes. The validation and testing sets both have $32,022$ patches, with $5,337$ patches for each class.

Since we focus on the first layer initialization, for this first experiment we have borrowed the network design used by [5] (see the original paper for details of network architecture). The first layer of their network contains three constrained 5×5 high-pass filters. In our network, we only replace the first layer by three 5×5 filters initialized by our method presented in Section 2, while keeping the network architecture unchanged. Later, our network was trained without any constraint. For comparison purposes, we also tested a network with three fixed 5×5 high-pass filters which simply compute the prediction error of the center element in input by using a weighted sum (with equal weights) of its 24 neighbors "seen" by the filter. For the constrained CNN [5], we have used the implementation shared on-line by the authors [1].

TABLE 1 – Considered processing operations and their parameters in the multiclass problem. The parameter is randomly chosen for the last two operations. σ is the parameter of standard deviation.

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

TABLE 2 – Testing accuracy of the three methods (in %, average of 15 runs) for the multiclass forensic problem.

Method	Accuracy
Fixed high-pass filters	77.38
Constrained filters [5]	81.32
Our initialization	**87.56**

We measure the forensic performance by using the detection accuracy which is computed as the percentage of correctly classified patches among all the patches of the six classes. Table 2 presents the accuracy on the testing set of all the three methods for the setting of the network's first layer: our initialization, the constrained filters of [5], and the fixed high-pass filters. In order to enhance the significance and reliability of the results, we ran the experiments for 15 times for all three methods and report in Table 2 the average of 15 runs. We can see that our initialization has higher forensic accuracy than the other two methods. It is worth mentioning that we also tested with other parameter settings for this multiclass forensic problem (*e.g.*, with different JPEG quality factors). We find that our initialization performs consistently well for both easy and challenging problems, in general with higher accuracy, but less improvement compared with existing methods, under easy parameter settings (*e.g.*, with lower JPEG quality factors).

3.2 Median Filtering Forensics

To continue the test of proposed initialization, we now consider a second forensic problem and carry out experiments with a different network. The objective is to accurately detect median filtering applied before JPEG compression. This problem is challenging because JPEG post-processing can partially remove the traces of median filtering. Similar to the last experiment, we borrow network architecture from Chen *et al.*'s work

1. Code available at `https://gitlab.com/MISLgit/constrained-conv-TIFS2018/`.

TABLE 3 – Detection accuracy (in %, average of 5 runs) on the testing set of the median filtering forensic problem with JPEG post-processing of different quality factors of 50, 70 and 90. An additional JPEG compression of default quality factor of 75 was mistakenly introduced which makes the detection of median filtering even more difficult.

Method	JPEG50+75	JPEG70+75	JPEG90+75
Chen [3]	80.32	83.32	85.32
Bayar [5]	81.56	84.65	86.32
Xavier init.	71.56	75.32	84.23
Our init.	**83.55**	**85.77**	**88.63**

[3] and replace the first layer of their residual computation by our initialization, the constrained approach [5] and the original Xavier initialization [6], respectively. Here we use our own implementation for the methods in [3] (original authors' implementation is unavailable to our knowledge) and [5] (working with a new network architecture). For [5], we follow the training procedure described in the original paper. Adam optimizer was used for our initialization, Xavier initialization, and the method in [3]. These three methods shared the same Adam optimizer parameters under each testing scenario. The training set has $16,668$ patches, and the validation and testing sets both have $5,556$ patches, with equal number of patches for the two classes with or without 3×3 median filtering applied before JPEG compression. We use 64×64 patches and consider three JPEG compression quality factors of 50, 70 and 90. It is worth mentioning that we mistakenly introduced a second JPEG compression of default quality factor of 75 when saving patches on hard drive. So in the end, there are two JPEG compression post-processing operations on patches with or without 3×3 median filtering, denoted respectively by JPEG50+75, JPEG70+75 and JPEG90+75. Therefore, our careless mistake in data preparation makes the forensic problem even more difficult, but we believe that the problem still remains realistic and reasonable.

Table 3 presents the testing accuracy at the end of the CNN training of 25 epochs for all the three scenarios. It can be noticed that our method has satisfying performance, comparable or slightly better then existing methods. The difference of detection accuracy of Chen *et al.*'s method when compared to that reported in [3] is probably due to difference and randomness in JPEG post-processing, experimental data, and network implementation and training.

4 Conclusion and Future Work

In this paper, we propose a new and simple initialization method used in CNN for forensics of image processing operations. The effectiveness of the proposed method is demonstrated by experiments for two different detection problems with different CNNs. In the future, we would like to further improve the initialization, by introducing more diversity in the design of filter template and by making it dependent on the input data. Like in existing CNN-based methods [3, 5], currently in our scheme the proposed initialization is applied to the first layer while the remaining layers are initialized with a conventional method, *e.g.*, the original Xavier initialization [6]. An adaptive initialization according to the layer's depth would be a promising future working direction. It would also be interesting to apply and adapt our initialization for solving other forensic problems.

Références

[1] A. Piva. *An overview on image forensics*. ISRN Signal Processing, vol. 2013, pp. 1–22, 2013.

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

[3] J. Chen, X. Kang, Y. Liu, and Z. J. Wang. *Median filtering forensics based on convolutional neural networks*. IEEE Signal Processing Letters, vol. 22, no. 11, pp. 1849–1853, 2015.

[4] H. Li, W. Luo, X. Qiu, and J. Huang. *Identification of various image operations using residual-based features*. IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 1, pp. 31–45, 2018.

[5] B. Bayar and M. C. Stamm. *Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection*. IEEE Transactions on Information Forensics and Security, vol. 13, no. 11, pp. 2691–2706, 2018.

[6] X. Glorot and Y. Bengio. *Understanding the difficulty of training deep feedforward neural networks*. Proceedings of the International Conference on Artificial Intelligence and Statistics, pp. 249–256, 2010.

[7] I. J. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning*. MIT Press, 2016.

[8] F.-F. Li, J. Johnson, and 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 2019-06-06), 2018. [Online]. Available at: http://cs231n.github.io/neural-networks-2/

[9] L. A. Goodman. *On the exact variance of products*. Journal of the American Statistical Association, vol. 55, no. 292, pp. 708–713, 1960.

[10] D. P. Kingma and J. Ba. *Adam: A method for stochastic optimization*. Proceedings of the International Conference on Learning Representations, pp. 1–15, 2014.

[11] T. Gloe and R. Böhme. *The Dresden image database for benchmarking digital image forensics*. Proceedings of the ACM Symposium on Applied Computing, pp. 1584–1590, 2010.