Université Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : SIGNAL IMAGE PAROLE TELECOMS
Arrêté ministériel : 25 mai 2016

Présentée par
Ivan CASTILLO CAMACHO

Thèse dirigée par **Kai WANG**

préparée au sein du **Laboratoire Grenoble Images Parole Signal Automatique (GIPSA-lab)**
dans **l'École Doctorale Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)**

Méthodes d'initialisation des réseaux de neurones convolutifs pour la détection des manipulations d'images

Initialization Methods of Convolutional Neural Networks for Detection of Image Manipulations

Thèse soutenue publiquement le **06/05/2021**,
devant le jury composé de :

Alice CAPLIER
Professeure, Grenoble INP, GIPSA-lab, Présidente
Francisco GARCIA UGALDE
Professeur, Universidad Nacional Autónoma de México, Rapporteur
William PUECH
Professeur, Université de Montpellier, LIRMM, Rapporteur
Patrick BAS
Directeur de Recherche, CNRS, CRIStAL, Examinateur
Kai WANG
Chargé de Recherche, CNRS, GIPSA-lab, Directeur de thèse

Initialization Methods of Convolutional Neural Networks for Detection of Image Manipulations

Ivan Castillo Camacho

Abstract

Fake images and videos have engulfed mass communication media. This is not something recent, manipulations and forgeries have occurred since the advent of photography itself. These alterations can go from innocent retouches in an attempt to make an image visually attractive to the spread of misleading information or even the use of false media in legal instances. Accordingly, the creation of methods that can help us assure the authenticity of an image presented as non-modified is of paramount importance. In this thesis, we aim at detecting image manipulation operations using deep learning techniques. We present three methods showing the progression of our work under one common objective, *i.e.*, the design and test of Convolutional Neural Network (CNN) initialization methods for image forensic problems with a variance stability focus for the output of a CNN layer.

First, we carry out an extensive review of the state of the art in deep-learning-based methods for image forensics. From this review we can confirm that the first layer of a CNN has big impact on the final performance. Specifically, the initialization used on the first-layer filters plays an important role that should be in line with the image forensic task in hand.

As our first attempt to address this research problem, we propose a low-complexity initialization method for CNNs. Taking advantage of previous methods designed for the computer vision field, we extend the popular Xavier method to design a filter that would provide variance stability after a convolution operation. This method generates a set of random high-pass filters for the initialization of a CNN's first layer. These filters allow us to better identify forensic traces which usually lie towards the high-frequency part of the image.

This first approach constitutes a good starting point of our work. However, a wrong assumption, largely utilized in the research community, was made. This is corrected in our second method where we follow a different data-dependent approach and take into consideration the real statistical properties of natural images. Accordingly, we propose a scaling method for first-layer filters which can cope well with different CNN initialization algorithms. The objective remains in keeping the stability of the variance of data flow in a CNN. We also present theoretical and experimental studies

on the output variance for convolutional filter, which are the basis of our proposed data-dependent scaling.

Next we describe a revisited version of our first proposal now with a corrected assumption on the statistics of natural images. More precisely, we propose an improved random high-pass initialization method which does not explicitly compute the statistics of input data. We believe that such a "data-independent" approach has higher flexibility and broader application range than our second method in situations where the computation of input statistics is not possible.

Our proposed methods are tested over several image forensic problems and different CNN architectures.

Finally, during all this thesis work we took part in a challenge competition of image forgery detection organized by the French National Research Agency and the French Directorate General of Armaments. We explain in the Appendix the objectives of the challenge along with a brief description of our work conducted for each stage of the competition.

Acronym Table

ABC	Attribution Based Confidence
ANR	Agence Nationale de la Recherche
AWGN	Additive White Gaussian Noise
A-CNN	Attention-Convolutional Neural Network
A-RNN	Attention-Recurrent Neural Network
CEL	Cross Entropy Loss
CFA	Color Filter Array
CFFN	Common Fake Feature Network
CGI	Computer Graphics Image
CISDL	Constrained Image Splicing Detection and Localization
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
DEFALS	DEtection de FaLSifications dans des images
DGA	Direction Générale de l'Armement
EXIF	EXchangeable Image File
FCN	Fully Convolutional Network
GAN	Generative Adversarial Network
GLCM	Gray Level Co-occurrence Matrix
IFS-TC	Information Forensics and Security Technical Committee
IRHP	Improved Random High-Pass
JPEG	Joint Photographic Experts Group
LBP	Local Binary Pattern

LSTM	Long Short Term Memory
MFR	Median Filtering Residual
MPS	Maximal Poisson-disk Sampling
MSE	Mean Square Error
NIST	National Institute of Standards and Technology
PRNU	Photo Response Non-Uniformity
RELU	Rectified Linear Unit
RNN	Recurrent Neural Network
R-CNN	Region-Convolutional Neural Network
SIFT	Scale-Invariant Feature Transform
SRM	Spatial Rich Model
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
TANH	Hyperbolic Tangent

List of Figures

1.1 Example of biased story telling. 2
1.2 Image manipulation by the media. 3

2.1 Visual representation of some typical activation functions. 8
2.2 Example of an artificial neuron. 9
2.3 Visual representation of the CNN architecture proposed in [SZ14]. . . 13
2.4 Examples of image forgery during the BP oil spill. 14
2.5 Classification diagram for deep-learning-based image forensic works. . 16
2.6 Sample images from the NIST Nimble 2016 Dataset [NIS16]. 19
2.7 Architecture of the multi-domain convolutional neural network proposed in [Ame+17] for double JPEG compression detection. 24
2.8 An LSTM cell. 29
2.9 Bounding-box localization results of [Zho+18]. 31
2.10 Source-target disambiguation results of [WAN18a]. 35
2.11 Illustration of typical pipeline of image acquisition and forgery creation. 38
2.12 Visual comparison between a CGI and an image taken by a camera. . . 40
2.13 Example frame of a Deepfake video. 42

3.1 Shape and notations of the initialized high-pass filter. 52
3.2 Examples of images from Dresden database [GB10a] and the generated patches used in our experiments. 55

4.1 Output/input variance ratio for SRM filters. 64
4.2 Histogram of occurrences for output/input variance ratio for Xavier filters. 66
4.3 Curves of test accuracy for the data-dependent scaling method on the multi-class forensic problem. 76
4.4 Test accuracy curves for the JPEG binary problem. 78
4.5 CNN architecture comparison. 80
4.6 Curves of test accuracy for our small architecture. 81

5.1 Filter template and input notations for IRHP initialization method. . . 84

5.2 Histogram of occurrences of output-input variance ratio for our IRHP initialization. 87
5.3 Examples for real and GAN-generated images. 95

A.1 Ordered CNN output scores with a thresholding at 0.5. 106

List of Tables

2.1 Summary of datasets of original image data and falsified image data. . 20
2.2 Datasets of artificially generated data. 22
2.3 Image manipulation detection methods. 27
2.4 Multipurpose image falsification detection methods. 32
2.5 Targeted splicing detection methods. 34
2.6 Targeted detectors of copy-move and inpainting falsifications. 36
2.7 Camera identification methods. 39
2.8 CGI detection methods. 41
2.9 Deepfake detection methods. 45

3.1 Considered manipulation operations in the multiclass forensic problem. 54
3.2 Test accuracy for the multiclass forensic problem. 56
3.3 Test accuracy for the median filtering forensic problem with JPEG post-processing. 58

4.1 Test accuracy for the multi-class problem for our scaling method. . . . 74
4.2 Test accuracy for the multiclass problem with 30 filters. 77
4.3 Test accuracy for the binary JPEG forensic problem. 78
4.4 Test accuracy results for our smaller CNN architecture. 80

5.1 Considered image manipulation operations and their parameter settings in the multi-class forensic problem. 88
5.2 Test accuracy for the multi-class problem for IRHP method. 89
5.3 Values obtained in an example 5×5 filter. 90
5.4 Comparison of batch normalization and our scaling-based and IRHP initialization methods. 91
5.5 Comparison of test accuracy for SRM diagonal filters, the corresponding scaled version and our IRHP method, in the JPEG binary classification scenario. 92
5.6 Average test accuracy for the multi-class and JPEG binary problems with our proposed smaller CNN, for Bayar, SRM and our IRHP method. 94

5.7 Characteristics of test sets for the evaluation of the generalization capability of detecting GAN-generated images. 96
5.8 Generalization results for the different test sets with comparisons between Wang *et al.*'s ImageNet pre-trained weights and our IRHP initialization on the first layer. 97

List of Equations

2.1 Batch normalization . . . 9
2.2 Mean Square Error . . . 10
2.3 Cross entropy loss . . . 10
2.4 Softmax . . . 11
2.5 Bayar's normalization . . . 26
3.1 Variance of weighted sum . . . 51
3.2 Variance of filter output for the first approach . . . 52
3.3 Variance of w_i for the first approach . . . 53
3.4 Variance of w_i from uniform distribution . . . 53
3.5 Constant C value for the first approach . . . 53
3.6 Multi-class accuracy . . . 56
4.1 Dot product of W and X . . . 62
4.2 Variance of weighted sum divided into variance and covariance terms . . . 63
4.3 Variance of weighted sum with natural image statistics assumptions . . . 63
4.4 Sum of elements in a high-pass filters . . . 65
4.5 Variance of Xavier filter output with input X fixed statistics . . . 67
4.6 Mean of variance of output of Xavier filter . . . 68
4.7 Variance of variance of output of Xavier filter . . . 68
4.8 Skewness of variance of output of Xavier filter . . . 69
4.9 Expectation of $\mathrm{Var}^3(y)$ . . . 69
4.10 Further derivation for the expectation of $\mathrm{Var}^3(y)$ . . . 70
4.11 Scaling factor using the covariance-based method . . . 71
4.12 Scaling factor using the convolution-based method . . . 71
5.1 Variance of a sum of potentially correlated variables for the IRHP method with four terms . . . 85
5.2 Variance of a sum of potentially correlated variables for IRHP filter with further derivations . . . 85
5.3 Variance of IRHP filter with natural image statistics assumptions . . . 85
5.4 Variance of IRHP filter with further natural image statistics assumptions 85
5.5 Variance of w_i for the IRHP method . . . 86
5.6 Constant C value for IRHP method . . . 86

Contents

1 Introduction **1**
- 1.1 Truth in Images . . . 1
- 1.2 Objectives . . . 3
- 1.3 Thesis Organization . . . 4

2 Background Knowledge and State of the Art **7**
- 2.1 Deep Learning . . . 7
  - 2.1.1 Neural networks . . . 7
  - 2.1.2 Convolutional neural networks . . . 11
- 2.2 Image Forensics . . . 13
  - 2.2.1 Datasets . . . 16
  - 2.2.2 Manipulation detection . . . 21
  - 2.2.3 Falsification detection . . . 28
  - 2.2.4 Other specific forensic problems . . . 37
  - 2.2.5 Discussion . . . 44

3 Random High-Pass Initialization **47**
- 3.1 Weight Initialization of CNN . . . 47
  - 3.1.1 Common initializations of CNN . . . 48
  - 3.1.2 Common initializations of CNN for image forensics . . . 49
- 3.2 Our Random High-Pass Initialization . . . 50
  - 3.2.1 The proposed method . . . 50
- 3.3 Experimental Results . . . 53
  - 3.3.1 Multiclass forensics . . . 54
  - 3.3.2 Median filtering forensics with JPEG post-processing . . . 57
- 3.4 Summary and Discussion . . . 58

4 Data-Dependent Initialization **61**
- 4.1 Variance of Output of Convolutional Filter . . . 61
  - 4.1.1 Formulation . . . 62
  - 4.1.2 Convolutional filter initialized with high-pass filter . . . 63
  - 4.1.3 Convolutional filter initialized with Xavier initialization . . . 65

4.2 Scaling of Convolutional Filter . . . 70
4.2.1 Covariance-based method . . . 71
4.2.2 Convolution-based method . . . 71
4.3 Experimental Results . . . 72
4.3.1 Multi-class problem with CNN of Bayar and Stamm [BS18a] . 73
4.3.2 JPEG binary problem with CNN of Bayar and Stamm [BS18a] 77
4.3.3 Multi-class and binary problems on a different smaller CNN . 79
4.4 Summary and Discussion . . . 81

5 Revisiting the Random High-Pass Initialization 83
5.1 The Proposed Method . . . 83
5.2 Experimental Results . . . 87
5.2.1 Multi-class forensic problem . . . 88
5.2.2 Comparison with batch normalization . . . 90
5.2.3 On the selection of SRM filters . . . 91
5.2.4 Smaller CNN architecture . . . 93
5.2.5 Detection of GAN-generated images . . . 94
5.3 Summary and Discussion . . . 97

6 Conclusions and Perspectives 99
6.1 Summary of Contributions . . . 99
6.2 Perspectives . . . 101

A Challenge Competition for Image Forgery Detection 103
A.1 Challenge Description . . . 103
A.2 DEFALS 1st Stage . . . 104
A.3 DEFALS 2nd Stage . . . 106
A.4 Discussion . . . 108

B Author's Publications 109

Bibliography 111

1 Introduction

> *A picture is a secret about a secret, the more it tells you the less you know.*
>
> **— Diane Arbus**

1.1 Truth in Images

As writers, photographers, or anyone in these days with a smartphone and social networks, we have the power to highlight or deny. We can influence judgments people make on the crucial issues of our time. We had better be honest and accurate with every picture we select and think of the consequences of each picture taken. The roots of photography lie in reality and almost daily that reality is bent by forgers.

How can we expect the readers and viewers to believe in the stories told by newspapers or documentaries when photojournalism competitors are being disqualified for image manipulation [@ML15], Pulitzer prizes are being revoked [@Woo20] and government agencies share altered photographs [@Alj11] distorting the reality? Every time a photographer takes a false picture, someone manipulates original images or an editor publishes an untrue picture the credibility of images diminishes. Believing in the content of a picture is at stake as never before.

In some cases the point of view or the angle of a picture can foster one particular aspect of the scene, avoiding the whole story. This typically happens to benefit a certain group. In this case, images are neither staged nor altered. Obviously, the pictures resulting from this kind of approach are not lies nor are the whole truth. They are biased photographic story telling. Such is the case of recent manifestations in Mexico or France where non-doctored photos are shared, focusing only on a certain aspect and favouring one side of the reality. Figure 1.1 shows an example of a biased story telling where the top row shows two original pictures from 2020. The one on the left is a picture shared by journalists claiming $100,000$ assistants in a protest in Mexico. On the right we see a hotel webcam image of the whole square showing the real amount of people was probably lower than the number mentioned

Fig. 1.1.: Example of biased story telling. Top row shows on the left the image shared by Mexican media which claims 100,000 assistants. On the right we see an aerial angle of the square. Below, we see a photo shared by the French media showing policemen were in fire while in reality it was just the angle of the photo.

by journalists. Below we see a burning policeman, who seems to turn into a "human torch". In reality, the angle of view makes this picture misleading. According to videos of the scene, filmed from different angles, there was indeed an explosion with flames, but it would have occurred in front of the police, not on them.

Figure 1.2 shows a controversial image of 2006 Lebanon War for which the Reuters news agency dismissed a photographer collaborator after finding that he had manipulated an image of an Israeli air strike on Beirut. Adnan Hajj's photograph, which was distributed by the agency, showed a cloud of smoke rising over buildings in the Lebanese capital after an Israeli air strike. A day later, after a controversy unleashed in several blogs, Reuters withdrew the photograph when it was found that it had been manipulated with PhotoShop to make the smoke thicker. "The photographer denied that he had deliberately tried to manipulate the image, and argued that he was only trying to remove the dust marks and that he had made mistakes because of the bad light conditions he was working in," explained Moira Whittle, Reuters' public relations chief [@Edi08].

Examples like this, were tampered images which do not mirror what happens in reality and can have serious consequences on society. This would be even worse when considering the fact that the advanced technology for digital imaging has brought image alteration at our fingertips with hundreds of applications of easy use ready to be downloaded. The ultimate price could eventually be a further dilution of the public's acceptance of the photograph as a credible witness to events.

Fig. 1.2.: Image manipulation by the media. On the left we see the original picture, while the right one shows the alterations which clone and darken the smoke to exaggerate the damage.

One positive note on what technology and different tools brought is the example of independent groups that use open-source information, programs and social networks to investigate a wide range of issues, from journalistic controversies and crimes against humanity to tracking chemical weapons in conflict zones around the world. This is the case of *bellingcat*[1] which uses publicly available tools and information to understand the context and situation of different conflict situations. In many cases they have used open-source image forensic tools to aid their investigations. Furthermore, they not only have produced several investigations and won awards but also created public training for the different tools for the purpose of creating a bigger group of open-source journalists dedicated to pursuing the truth.

For these reasons, we believe in the necessity of developing algorithms that help us recover credibility of images, therefore bringing the truth closer to everyone.

1.2 Objectives

Our work mainly deals with image forensics using deep learning techniques. Traditional detection methods of image tampering resort to the detection of genuine and tampered blocks based on well-designed discriminative features. Such features are often borrowed from steganalysis and reflect local image statistics.

At the moment of designing this thesis project, almost all existing methods for detecting image manipulations were based on handcrafted features with the majority of them tested on medium-sized images of 256×256, leaving the detection performance on smaller patches, vital for spatially accurate forensics, rather unknown.

[1] https://www.bellingcat.com

Indeed since then research attempts were conducted by introducing some constraints into the deep learning paradigm to solve image forensic problems in a relatively ad-hoc manner. Others have used handcrafted high-pass filters at the beginning of deep neural networks without considerations on how the *stability* of the amplitude of the data flow in the network may get reduced. We have the intuition that after the image data passes through a first layer of high-pass filters, the filters' output becomes a weak signal. This would be detrimental to the training of Convolutional Neural Network (CNN) because the data flow shrinks.

As main objective, we focus on the detection of "routine" operations performed on image patches which are related to image's processing history (*e.g.*, medial filtering, noise addition, compression), because this piece of information can help further forensic analysis and because we think there is still room for improvement in this area. One of our goals is to obtain a good forensic performance on small patches, *e.g.*, patches of 64×64 pixels. This last point was of great importance in order to ensure a high spatial accuracy of modification/tampering detection. Additionally, existing methods mainly consider image manipulation operations with a relatively big amplitude for the induced modification. We consider more challenging scenarios where the manipulation operations to be detected introduce modifications of smaller amplitude.

In order to fulfill the thesis objectives, we dedicate our efforts to a fundamental part of the deep learning paradigm when applied to the image forensics field. More specifically, we focus on the study of the *weight initialization* of the first layer of a CNN for image forensic purposes.

1.3 Thesis Organization

The remainder of this thesis is organized as follows. First, in chapter 2 we present background knowledge in deep learning as well as a comprehensive review of the deep-learning-based methods for image forensics including also the datasets used by such methods. We divide the review into three large groups of forensic methods: 1) detection of routine image processing operations, 2) detection of malicious image falsifications, and 3) other specific forensic problems.

Then in chapter 3 we describe our first approach of CNN initialization for forensic detection of routine image processing operations. The proposed approach extends the well-known method of Glorot and Bengio [GB10b] to situations where we need to generate a group of random high-pass filters as initialization for a CNN's first

layer. As a first step of our thesis work, this method has a problematic yet largely used assumption which will be taken into account and corrected in our second data-dependent approach presented in chapter 4. In that chapter, we describe a practical initialization method based on a proper scaling of any given first-layer filters. Theoretical and experimental studies on the output variance for convolutional filters are also presented. The proposed data-dependent method can cope well with different first-layer initialization algorithms and different CNN architectures. Next, in chapter 5 we present a revised version of our first initialization method of chapter 3, now with correct assumption. This revised method is able to generate random high-pass filters without explicitly computing statistical properties of input data. The generated filters do not have the data flow shrinking problem at their output. Additional experimental results (*i.e.*, with new settings of comparisons, for new forensic problem and on new deep network) are also provided.

Finally, in chapter 6 we present some concluding remarks and suggest several future working directions to extend this thesis work.

In Appendix A we share our experience in an image forgery detection competition that we had the opportunity to participate in during the realization of this thesis. The competition occupied one part of our thesis working time but constitutes a unique experience of handling very piratical tasks of image forgery detection.

2 Background Knowledge and State of the Art

2.1 Deep Learning

Deep learning is part of a group of machine learning methods that are based on the so-called representation learning. An observation like an image can be represented in many ways but some of the representations make it easier to learn for tasks of interest (*e.g.*, "Is this image a human face?") based on examples, and research in this area attempts to define which representations are the "best" and how to create models to learn and recognize these representations. Nowadays deep learning has become the main algorithms used in the creation of applications and programs for image analysis and understanding.

The deep learning framework usually uses a hierarchical structure of artificial neural networks, which are built in a similar way to the neural structure of the human brain, with the neuron nodes connected together to simulate a neural network. This architecture can approach data analysis in a non-linear way. The striking advantage of deep learning is its ability to automatically learn useful features from available data, allowing us to bypass the tedious procedure of handcrafted feature design.

2.1.1 Neural networks

Neural networks as an initial step for the deep learning paradigm have been used to solve a wide variety of tasks (*e.g.*, in computer vision and speech recognition) that are difficult to solve using ordinary rule-based programming. They are structured groups of neurons or weights that are connected together to transmit signals. In a classification problem, the input values pass through the neural network resulting in output values that represent scores for the considered classes.

In these networks, the output value of a previous neuron is multiplied by a weight value. These weights can increase or inhibit the activation state of adjacent neurons. Similarly, at the exit of the neuron, there may be a limiting or threshold function, which modifies the resulting value or imposes a limit that must not be exceeded

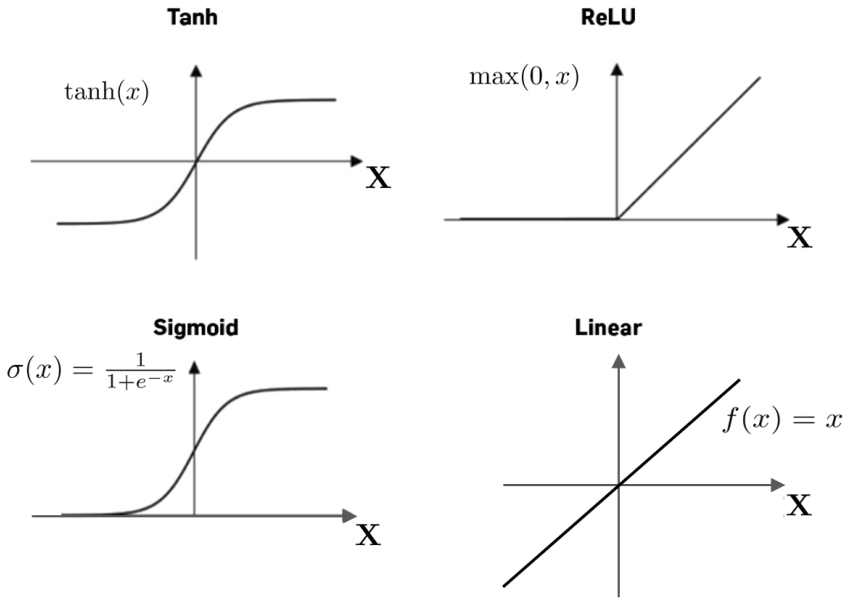

Fig. 2.1.: Visual representation of some typical activation functions.

before it spreads to another neuron. This function is known as the non-linearity or activation function. Neural network models are often organized into a number of layers of neurons. Inside the different layers of a neural network, one of the most common components is the fully-connected layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections.

Activation functions and artificial neurons

The non-linearity of a neural network is necessary to approximate complex patterns in data. Using activation functions serves this purpose by checking the value produced by a neuron and deciding whether outside connections should consider this neuron as activated or not. There are several activation functions that differ on the allowed threshold they impose. Some of the most common ones are Linear, Sigmoid, TANH [KK92], Rectified Linear Unit (RELU) [GBB11], etc. Figure 2.1 shows the graphs of these commonly used activation functions.

Each neuron computes the result of dot product of the input and its weights, then the sum of the dot product and a bias, and at last the output of the activation function. For instance, a layer of K neurons can be represented as $\max(0, W.X + b)$ where the function $\max(0, z)$ is an example of the RELU activation function that is applied element-wise, X is the image pixels flattened out in a single vector of shape $D \times 1$,

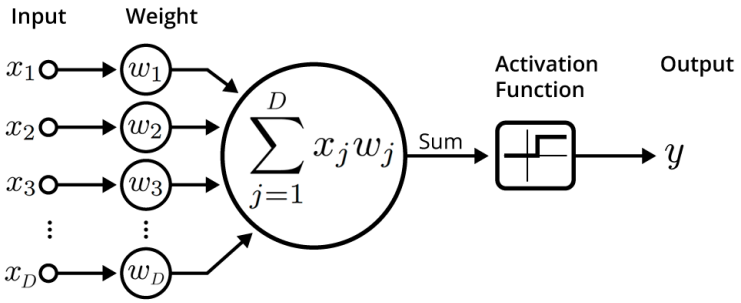

Fig. 2.2.: Example of an artificial neuron comprising the dot product, bias sum and non-linearity function. x_j and w_j represent respectively the input and the weights of the neuron, and y is the output.

W is the weight matrix of shape $K \times D$, and the $K \times 1$ vector b contains the bias values. Figure 2.2 shows an example of an artificial neuron.

Regularization

If a neural network performs very well with training samples but fails with the test dataset, usually the network suffers from the overfitting phenomenon. Regularization is a technique that helps to reduce it by penalizing for complexity. The objective of using regularization is to avoid this extreme fitting in terms of training predictions, because we know that usually the (slight) decrease of performance on training samples induced by regularization would result in a better generalization to the testing data.

Batch normalization [IS15] and Dropout [GG16] are the two most common techniques of regularization. The network training is usually done in a batch approach where a subset of the data called a batch is presented as input for the network. The process for batch normalization is given by Equation 2.1 (with j-th input dimension as an example), where $\mu_j, \sigma_j^2 \in \mathbb{R}$ are the mean and variance, respectively, of the j-th dimension across the batch, ϵ is some small constant that prevents division by 0, and $\gamma_j, \beta_j \in \mathbb{R}$ are learnable parameters for the j-th dimension. The mean and variance are often updated in a moving average fashion during training.

$$\hat{x}_j = \gamma_j \cdot \frac{x_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} + \beta_j. \tag{2.1}$$

Dropout function works by randomly setting the outgoing value of certain neurons to 0 at each update of the training phase. This is performed with a specified probability

p (a common value is $p = 0.5$). These neurons with outgoing value 0 are considered as ignored during a particular forward or backward pass.

Loss functions

To perform its automatic learning, usually an attempt is made to minimize a loss function that evaluates the network as a whole. If predictions from the network deviate too much from actual results, the output of the loss function would result in a high value, and it will be low if the network is doing a good job.

There are mainly two types of loss functions depending on the learning task. The first one known as classification loss is used when the prediction outputs probabilities of belonging to a defined set of categories, *e.g.*, 10 classes of images such as cats, dogs, flowers, etc. The second group is regression loss which is used when we want to model a function that explains certain data, *e.g.*, predicting the price of a house. In other words the choice changes depending on whether our target variables are numeric (for regression loss) or probabilistic (for classification loss).

Mean Square Error (MSE) is the most common regression loss function, it is also known as L2 loss. As shown in Equation 2.2, MSE is given by the average of squared differences between the target and the prediction. In this equation, $Y^{(i)}$ and $\hat{Y}^{(i)}$ (scalars in this case) represent the prediction and the target (*i.e.*, ground-truth) value respectively, while i is the index among the n training samples.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y^{(i)} - \hat{Y}^{(i)})^2. \tag{2.2}$$

Cross Entropy Loss (CEL) is the most common loss function for classification problems. Cross entropy measure is widely used when node activations can be understood as representing the probability that each hypothesis might be true, *i.e.*, when the output is a probability distribution. Equation 2.3 shows the case for a binary scenario, where each $Y^{(i)}$ is a 2-dimensional vector representing the probability of being the two classes, with $Y_0^{(i)} + Y_1^{(i)} = 1$. When the target is class 0 the second term in parentheses goes to 0, in contrast when the target is class 1 the first term disappears.

$$CEL = -\frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_0^{(i)} \log Y_0^{(i)} + (1 - \hat{Y}_0^{(i)}) \log(1 - Y_0^{(i)})\right). \tag{2.3}$$

An important aspect is that cross entropy loss penalizes heavily the predictions that are confident but wrong.

Classification layer

For a classification task, the most commonly used classification function is the Softmax function as shown in Equation 2.4, where the M-dimensional vector z is the output of the last layer before applying the classification function and contains the predicted values related to an M-class classification problem, with z_k the value for the k-th class. The Softmax function is applied to the vector z, performing a standard exponential to all elements of z and then normalizing the values by dividing them by the sum of all the exponentials. This ensures that all the M values after normalization sum up to 1, *i.e.*, $\sum_{k=1}^{M} \hat{z}_k = 1$.

$$\hat{z}_k = Softmax(z)_k = \frac{\exp(z_k)}{\sum_{k=1}^{M} (\exp(z_k))}. \tag{2.4}$$

Back-propagation

The core algorithm behind how neural networks learn is back-propagation. The main task is to compute the gradients which will be taken by the optimizer to train the neural network. The gradient vector will indicate how to adjust each weight in the network according to how much it contributes to the overall error. The process starts from the output layer and propagates backwards applying the chain rule to find the derivatives or gradients of the training cost with respect to any variable in the nested architecture [RHW86]. After the iterative process of feed-forward and back-propagation among the training dataset, the series of weights produce better predictions until a plateau is achieved in the cases where the neural network architecture is able to model the data.

2.1.2 Convolutional neural networks

CNN is also made up of neurons which have learnable weights. CNNs are designed with images as inputs instead of vectors. What distinguishes CNNs from any other neural network is that they use an operation called convolution in some of its layers, instead of using routine matrix multiplication that is generally applied. The convolution operation receives the image as input and applies a convolutional filter or kernel to it, resulting in an activation map that emphasizes certain properties of the input image.

In a CNN the entire architecture or model from the input image to the prediction values at the end continues with the approach of creating a differentiable function.

Similar to a neural network, the output part contains loss and classification layers. Guided by the loss function, all the weights of the whole architecture will be updated during the training phase via back-propagation [LeC+98].

The common architecture for a CNN is a progression of convolutional layers joined with other type of layers with each of them converting its inputs to an activation map using a differentiable operation. The main components in a CNN are convolutional layers, activation functions, pooling layers and fully-connected layers.

Inside a CNN for digital images, a convolutional layer contains kernels of several neurons in three dimensions: width, height and channel. This behavior matches with the characteristics of an image. The neurons in each layer will only be connected to a small region of the preceding and subsequent layer, instead of connecting them with all of the neurons in a fully-connected manner.

As explained before, the activation functions transform the combination of inputs, weights and biases in a non-linear manner. The activation functions are used to propagate the output of the nodes in one layer to the next layer. Many (but not all) of the non-linear transformations used in neural networks transform data at a convenient range. These types of functions allow the incorporation of non-linear modelling of input data into the network.

The pooling layer is usually placed after the activation function. Its main use lies in the reduction of the spatial dimensions: width and height of the input for the next convolutional layer. The operation of a pooling layer is also known as sampling reduction, since the reduction in size also leads to the loss of information. However, such a loss in general helps to both reduce computational complexity and alleviate overfitting problems. The most common cases for this layer can be realized by taking the maximum or the average value in each local window.

Figure 2.3 shows as an example of a well-known deep learning architecture designed by Simonyan and Zisserman [SZ14]. This model won the first place on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [OK15]. The details of this architecture are described as follows: There are two layers of 64 channels followed by a max-pool layer, two layers of 128 channels with max-pool, three layers of 256 channels with max-pool, three layers of 512 channels with max-pool, and again three layers of 512 with max-pool. After the stack of convolution and max-pooling layers, the activation map, also called feature map, has a size of $(7, 7, 512)$. This output is flattened to make it a feature vector. Then there are three fully-connected layers, and the third layer has an output of 1000 channels for 1000 classes of the ILSVRC

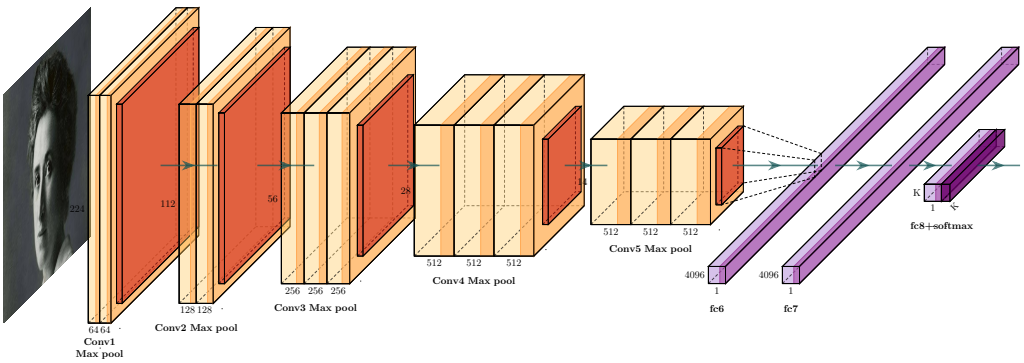

Fig. 2.3.: Visual representation of the CNN architecture proposed in [SZ14]. This figure was created by using the tool shared by Iqbal [@Iqb18].

challenge. The output of the third fully-connected layer is passed to a Softmax layer in order to normalize the final classification vector.

Deep learning architectures have been used profoundly in the computer vision field with different depths and characteristics, producing results superior to those of previous strategies. Recently they have also become a very popular tool in the image forensics field, as described in the next section.

2.2 Image Forensics

Given our era of advanced technology and the high availability of image editing tools that make it extremely easy and fast to alter and create fake but realistic images, the trust of digital images has diminished. We can no longer easily accept an image as proof of an event without asking ourselves if the image has been modified. This has been in a continuous development together with the easy accessibility of tools used to create tampered contents and with the deep learning advancements which have led to an increase in the realism of fake images or videos [AFG19].

During the last years an evolution on disinformation has appeared to manipulate and disrupt public opinion. This disinformation comprises sophisticated campaigns aided by doctored images with the goal of influencing economic and societal events around the world. Different kinds of problems related to the usage of tampered images have appeared in different fields and will get worse as both digital cameras and software editing tools become more and more sophisticated.

Fig. 2.4.: Examples of image forgery during the BP oil spill. First row shows how the original image was modified by copying some screens over the initially blank ones. On the second row, the helipad was removed in the tampered version. Images were obtained from the following webpage: https://www.cbsnews.com/news/bp-and-the-gulf-oil-spill-misadventures-in-photoshop/.

In July 2010, British Petroleum (BP) came under public outcry over several doctored images of its Gulf of Mexico oil spill response, as images were tampered to indicate that BP staff were busier than they actually were. Figure 2.4 shows two pairs of the original (first column) and the tampered (second column) images.

A spokesman for the company eventually admitted that in one image (first row of Figure 2.4) two screens were actually blank in the original picture. On the second row of Figure 2.4, we see a photo taken inside a company helicopter which appeared to show it flying off the coast. It was later shown to be fake after internet bloggers identified several problems, which suggested that the helicopter was not even flying. The problems included part of a control tower appearing in the top left of the picture, its pilot holding a pre-flight checklist, and the control gauges showing the helicopter's door and ramp open and its parking brake engaged[1].

From this context, it is necessary to develop strategies and methods to allow the verification of the authenticity of digital images. Image forensics [Piv13] is the science that can help us to know if the image was acquired by the claimed device or if the current state corresponds to the original captured image, with the objective of

[1]Please refer to details presented at the following webpage: https://metro.co.uk/2010/07/22/bp-admits-to-doctoring-another-deepwater-horizon-oil-spill-image-456246/.

detecting and locating image forgeries. Image forensic techniques depend on the assumption that each stage of the image acquiring and processing, from the raw image to its compression, storage and post-processing, holds some inherent statistics and leaves a particular trace. It is then possible to infer the source of the image or decide whether it is authentic or tampered by identifying the existence, lack or inconsistency of forensic traces that are inherent to the image itself.

The research on this field started around 20 years ago and has seen recently a boost with the latest deep learning tools. This helps to restore some trust to digital images. Regardless of the big progress in the computer vision field using deep learning tools, the same strategies cannot be applied directly to the image forensics domain as the traces or fingerprints that we are looking for are normally not present in the visible domain. Most of the traces that we search are hardly perceptible by the human eyes. Therefore, certain strategies have been proposed to cope with this difference.

Early surveys on image forensics [Far09; SWL13; Piv13] naturally focused mainly on conventional feature-based methods. Recent surveys [ZZT19; Ver20] consider both conventional and deep-learning methods yet with a different focus or coverage from ours. For instance, [ZZT19] mainly considers the detection of copy-move, splicing and inpainting, while we cover more image forensic problems including also the detection of routine image processing operations, the detection of synthetic images, etc.; [Ver20] classifies existing methods from a machine learning perspective (*e.g.*, supervised, unsupervised and anomaly detection) with a special and timely focus on Deepfakes detection, while we classify with a rather comprehensive list of image forensic problems and focus on the particularities of deep network design for different problems. Other existing surveys have dedicated their reviews to presenting and analyzing the methods for one or several specific issues like copy-move (and splicing) detection [AHH17; ZWZ18], computer-generated image detection [NCY19], camera identification [WFT20] and image source identification [Yan+20], while we attempt to have a broader coverage.

In this chapter, we review existing deep-learning-based methods for a variety of image forensic problems. The research works presented in this survey can be classified into three large groups: the detection of image *manipulations* (*i.e.*, routine image processing operations like median filtering and JPEG compression), the detection of image *falsifications* which alter the semantic meaning of the image (*e.g.*, copy-move, splicing and inpainting) and other specific forensic problems. We pay attention to special designs of the deep models and special features used on the network input. Considering the rapid advancement in the image forensics field and the difference between our review and existing ones as discussed in the last

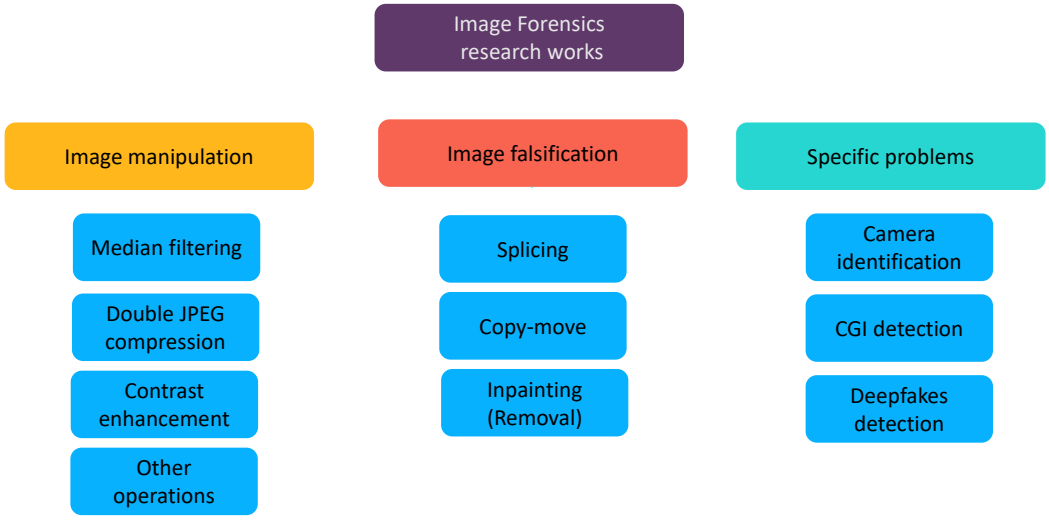

Fig. 2.5.: Classification diagram for deep-learning based image forensic works. "CGI" means computer graphics image.

paragraph, we believe that our survey can be helpful for the research community and is complementary to previous reviews. Our classification of research works on image forensics is illustrated in Figure 2.5. Each of the three main groups and its subsets are explained in one specific part. Before presenting the various forensic methods, we first present in the next section the datasets used for image forensics research which are vital for data-driven methods based on deep learning.

2.2.1 Datasets

Aside from the different models and different approaches, the access to a proper dataset is the first step and has a crucial role in the deep learning paradigm to make it work properly. This means using a dataset that corresponds to the results a researcher wants to predict. The dataset should match the problem context including the acquiring and any processing steps. Constructing a dataset is a time-consuming task which requires problem and context knowledge of the procedure to collect compatible data. If the dataset contains sufficient and adequate data and information, problems like over-fitting and underfitting could be mitigated. Furthermore, the usage of multiple available datasets is of paramount importance to obtain a more reliable benchmarking of existing and new methods. In this section, the publicly available datasets for different categories of image forensic tasks will be introduced. Different datasets are grouped according to the different image forensics categories for which they are used.

Original data

Datasets of pristine data used in the image forensics field (*e.g.*, in the manipulation detection area) often contain original uncompressed image data. In this way, researchers are able to recreate different manipulation operations and conduct experiments on an adequate and customized dataset. Some of these databases were originally designed for the purpose of benchmarking camera identification techniques.

One of the first works in this field is the UCID dataset [SS04] with 1,338 uncompressed images (version 2) in TIFF format stemming from a single camera. The BOSSBase 1.01 dataset [PP11] contains 10,000 grayscale uncompressed images, originally designed for research in the steganalysis field. In the Dresden image dataset [GB10a], 73 digital cameras with 25 different models were used to create 14,000 Joint Photographic Experts Group (JPEG) images. The RAISE dataset [Dan+15] contains 8,156 uncompressed high-resolution images of different categories such as landscape or indoor scenes. It comprises 4 subsets called RAISE-1K, RAISE-2K, RAISE-3K and RAISE-4K.

Some recent datasets introduced mobile phone cameras to their catalogue. A small number of devices was considered in the MICHE-I dataset [De +15] comprising 3,732 iris images from 3 different smartphones using both front and back cameras. The IEEE and Kaggle [@IEE18] organized a camera identification challenge in 2018 with a dataset captured from 10 different camera models (9 of 10 being smartphone cameras) with 275 images from each device.

The Vision dataset [Shu+17] also purposed for camera model identification and contained 34,427 images and 1,914 videos from 35 portable devices of 11 major brands, both in the native format and in their social platform version including Facebook, YouTube and WhatsApp. Some datasets like [Ber+19] and [@Ama18] are designed for a specific domain. For instance [@Ama18] is an ongoing collection of satellite images of all land on Earth produced by the LandSat 8 satellite. Other proposals like [NZ08], [Lin+14], [Den+09] [Zho+14] and [Xia+10], initially designed for object and scene detection, segmentation and recognition, were used in the image forensics field to create synthetic data. For example, the Microsoft COCO dataset [Lin+14], originally constructed for object and scene analysis and comprising more than 300,000 images in JPEG format, has been used to create different image forgeries. Another example is the SUN2012 dataset [Xia+10], composed of 130,519 images of different outdoor and indoor scenes, has been employed to create synthetic data for image forensics purposes.

Regarding the creation of Deepfakes (*i.e.*, fake images generated by deep neural networks), some well-known datasets of human faces have been used for network training, for instance the CelebA dataset [Liu+15] which contains around 200,000 faces with different annotations originally designed for facial image analysis. Stemming from CelebA dataset, CelebAHQ [Kar+17] is a set of high-resolution face images and is one of the first datasets used for training and evaluation of GAN (Generative Adversarial Net) models for face generation and editing.

Falsified data

To our knowledge, the first public datasets for detection of *splicing* (*i.e.*, a common image falsification in which one copies a part of an image and pastes it to another image) were the Columbia gray DVMM dataset [@NHC04] and the Columbia color splicing dataset [HC06]. The two datasets comprise respectively 1,845 grayscale images for the first one and 180 color spliced images for the second one, both with rather non-realistic random-like splicing falsifications. Two other well-known splicing datasets are the Casia V1 and V2 [DW11] with more realistic forgeries and post-processing operations on the V2 to cover the traces of splicing. In 2013, the IEEE Information Forensics and Security Technical Committee (IFS-TC) organized an image forensics challenge and released a dataset of pristine and forged images [@IEE14] with a set of different falsification techniques such as splicing and *copy-move* (*i.e.*, another common falsification in which one copies a part of an image and pastes it in the same image). Each fake image had an associated ground-truth binary map showing the regions that were falsified. As a small sub-dataset from the IFS-TC proposal, the DS0-1 dataset (also known as Carvalho dataset) [Car+13] contains forgeries created in a careful and realistic manner.

The National Institute of Standards and Technology (NIST) developed the Nimble [NIS16] and MFC [MFC17] datasets. The first one, often called NIST Nimble 2016, included three types of falsifications including splicing, copy-move and *inpainting* (*i.e.*, a third type of common falsification in which a part of an image is replaced and filled with realistic synthetic contents), with different levels of compression and post-processing. Figure 2.6 shows some example images from this dataset. The NIST MFC17 dataset [MFC17] included more challenging image forgeries but did not contain the different compressed versions.

The Realistic Tampered Dataset [KH17], also known as Korus dataset, comprises 220 splicing and copy-move forgeries of a realistic level. Authors included PRNU signatures and ground-truth maps. Other datasets have been created with a specific

Fig. 2.6.: Sample images from the NIST Nimble 2016 Dataset [NIS16]. Top row shows the original images, and bottom row shows from left to right falsifications of inpainting-based removal, copy-move and splicing.

purpose in mind. Regarding the double compression scenario, the VIPP dataset [BP12] was created to evaluate the detection of double JPEG compression artifacts which may be present for instance in the splicing falsification.

The use of datasets specific for copy-move falsification, such as [Wen+16] and [Tra+13], is not very common for the deep-learning-based detection methods. The main reason is that existing datasets are relatively small. Therefore, majority of research on deep-learning-based copy-move detection has created customized synthetic datasets which are derived from dataset of original images and which contain much more samples.

Table 2.1 shows a list of popular datasets for image forensics research including datasets of original data and falsified data. In the case of falsified data, we provide the number ratio of pristine and tampered images. Regarding the "Operations" columns we mention the main operations (mostly falsifications) contained in the dataset and the "Other" case mainly includes double JPEG compression.

Artificially generated data

In the case of artificially generated data, it is important to use datasets that contain realistic examples. Existing datasets considered different scenes of authentic images taken by a camera and artificially generated fake images either with conventional Computer Graphics Image (CGI) creation algorithms or recent Generative Adversarial Network (GAN) architectures.

Tab. 2.1.: Summary of datasets of original image data and falsified image data. "GT" stands for "Ground-truth". The "Content ratio" column shows the number of pristine/tampered images.

Type	Name	Format & size	Content ratio	Operations				GT mask
				Copy-move	Splicing	Inpainting	Others	
Original data	BossBase [PP11]	PGM 512x512	10K					N/A
	UCID [SS04]	TIFF 512x384, 384x512	1338					N/A
	Landsat [@Ama18]	TIFF 650x650; 5312x2988	Ongoing					N/A
	MIT SUN [Xia+10]	JPEG 200x200	130519					N/A
	NRCS [@Mac04]	TIFF,JPEG 1500x2100	11036					N/A
	MS COCO [Lin+14]	JPEG various	328K					N/A
	CelebA [Liu+15]	JPEG 64x64; 512x512	200K					N/A
	CelebAHQ [Kar+17]	JPEG 512x512	30K					N/A
	RAISE [Dan+15]	TIFF 4288x2848	8156					N/A
	Dresden [GB10a]	JPEG 3039x2014; 3900x2616	14K					N/A
	MICHE-I [De +15]	JPEG 640x480; 2322x4128	3732					N/A
	Kaggle Camera [@IEE18]	JPEG,TIFF 1520x2688; 4160x3120	2750					N/A
	Vision [Shu+17]	JPEG 960x720; 5248x3696	34427					N/A
Falsified data	Columbia gray [@NHC04]	BMP 128x128	1845/912		●			No
	IEEE IFS-TC[@IEE14]	PNG 1024x768; 3000x2500	1050/1150	●	●			Yes
	Casia v1 [DW11]	JPEG 384x256	1725/925		●			No
	Casia v2 [DW11]	JPEG,BMP,TIFF 240x160;900x600	7491/5123	●	●			No
	NIST Nimble 16 [NIS16]	JPEG 500x500; 5616x3744	560/564	●	●	●		No
	Coverage [Wen+16]	TIFF 400x486	100/100	●				Yes
	Columbia color [HC06]	TIFF 757x568; 1152x768	183/180		●			Yes
	Carvalho [Car+13]	PNG 2048x1536	100/100		●			Yes
	Realistic (Korus) [KH17]	TIFF 1920x1080	220/220	●	●			Yes
	CoMoFoD [Tra+13]	PNG,JPEG 512x512; 3000x2000	260/260	●				Yes
	VIPP [BP12]	JPEG 300x300; 3456x5184	68/69		●		●	Yes
	NIST MFC17 [MFC17]	RAW,PNG,BMP,JPEG, TIFF 128x104; 7952x5304	14156/3265	●	●	●	●	No

One of the first popular dataset of CGIs is the Columbia dataset [Ng+05] with 1,600 photorealistic computer graphics images. Afchar *et al.* [Afc+18] created a dataset with 5,100 fake images generated from videos downloaded from the internet. Rahmouni *et al.* created a dataset of CGIs coming from high-resolution video game screenshots. There are several online repositories for CGI [@ABV05], [@Cha20], [@Ltd20], [@Aut20] that have been used as datasets for different detection approaches.

A small dataset of 49 Deepfake and 49 original videos was created by Yang *et al.* [YLL19] using the FakeApp application. A bigger one is the Faceforensics dataset [Rös+18] comprising about 1,000 videos and their corresponding forged versions focused on expression swap created with the Face2Face model. The same authors extended the dataset [Ros+19] with 4,000 forged videos. Li *et al.* [Li+20c] created a dataset of 590 original videos and 5,639 Deepfake videos. In comparison to other face datasets [Liu+15; Kar+17], the diversity among genders, ages and ethnic groups is bigger. The IDIAP institute created DeepfakeTIMIT [Kor+19] also known as DF-TIMIT containing 620 videos where faces were swapped. This dataset was generated using the faceswap-GAN [@dee20] with 32 subjects and 2 subsets of different resolutions: low quality with 64x64 and high quality with 128x128.

Recently, Google in collaboration with Jigsaw and Facebook have created a Deepfake dataset to contribute to the relevant research. In 2019, Facebook created the DFDC dataset [Dol+19] for the Deepfake detection challenge with 4,113 Deepfake and 1,131 original videos from 66 subjects of diverse origins who gave their consent for the relevant data. At last, the DFD dataset [@Goo19] contains 3,068 Deepfake videos and 363 original ones from 28 individuals who consented to the data.

Table 2.2 summarizes the artificially generated datasets presented above. The "Content ratio" column shows the number of pristine/fake images.

From the next section, we present different kinds of deep-learning-based image forensic methods, starting by the detection of routine image manipulations.

2.2.2 Manipulation detection

We consider image manipulation as routine operations modifying or improving digital images with basic and benign image processing such as median filtering, JPEG compression or contrast enhancement. These operations may be used to enhance the visual quality of tampered images or to hide the traces of falsification operations that may leave an apparent fingerprint if used alone. In this subsection

Tab. 2.2.: Datasets of artificially generated data.

Type	Name	Size	Format	Content ratio	Media	
					Video	Image
CGI	Columbia [Ng+05]	700x500; 3000x2000	JPEG	1600/1600		●
	Rahmouni [Rah+17]	1920x1080; 4900x3200	JPEG	1800/1800		●
CGI & Deepfakes	Faceforensics [Rös+18]	480p	H.264	1000/1000	●	
Deepfakes	UADFV [YLL19]	294x500	MP4	49/49	●	
	Faceforensics++ [Ros+19]	480p,720p, 1080p	H.264	1000/4000	●	
	Afchar [Afc+18]	854x480	JPEG	7250/5100		●
	PGGAN [Kar+17]	64x64; 1024x1024	JPEG	-/100K		●
	Deepfake TIMIT [Kor+19]	64x64; 128x128	AVI	-/620	●	
	CelebDF [Li+20c]	Various	MP4	509/5639	●	
	DFDC [Dol+19]	180p; 2160p	H.264	1131/4113	●	
	DFD [@Goo19]	1080p	H.264	363/3068	●	

we introduce the most relevant strategies to detect some of the most common manipulation operations using deep learning as the core technique. We present both *targeted* (*i.e.*, aiming at a specific manipulation operation) and *general-purpose* (*i.e.*, aiming at various operations) detectors.

Median filtering detection

The early deep learning proposals in the literature of image forensics were focused on designing a specific strategy to cope with each forensic challenge individually. The goal of one of the first methods proposed in 2015 by Chen *et al.* [CKL15] was to detect median filtering manipulation.

In their paper, Chen *et al.* [CKL15] used a tailored CNN to detect median filtering with JPEG post-processing. The JPEG compression after median filtering made the forensic problem more challenging because the compression can partially remove the forensic traces of medial filtering. The tailored CNN took the Median Filtering Residual (MFR) rather than the raw pixel values as input for the first layer in the CNN. The MFR is the difference between a given image and its median filtered

version. Authors found that using this special input, the network achieved a better forensic classification performance, with a better detection accuracy when compared with handcrafted-feature-based strategies on small patches of 64x64 and 32x32.

More recently, Tang *et al.* [Tan+18] proposed to upscale the input with nearest neighbor interpolation in an attempt to enlarge the difference between manipulated and original patches. After this upscaling, the first two layers in the network are mlpconv layers introduced in [LCY13]. An mlpconv consists in a special layer for deep learning architectures that defines a group of convolutional layers with activation functions that can enhance the non-linear ability of the network. Specifically, it proposes to replace a traditional convolutional layer followed by a ReLU activation function with a Convolutional layer, ReLU activation function, convolutional layer with filters of shape 1x1 and a final ReLU activation function. In the case of median filtering detection, Tang *et al.* [Tan+18] made use of mlpconv to enhance the network's non-linearity to deal with the detection of median filtering non-linearity.

Both the above proposals [CKL15; Tan+18] rely heavily on having a special input for the network being either the MFR or an upscaled version, regardless of their differences in the network architecture.

Double JPEG compression detection

JPEG images are widely used in daily life as one of the most common image formats. Hence, most of the forensic tasks are related to JPEG images. Typically, inside a normal forgery creation process, an image is decompressed from JPEG to the spatial domain for falsification, and later recompressed again in JPEG format for storage and further use. For this reason, the image forensics community has dedicated important research efforts to the detection of double JPEG compression through the years. Detection and localization of double JPEG compression provides valuable information towards image authenticity assessment.

In double JPEG compression, double quantization of Discrete Cosine Transform (DCT) coefficients leaves special artifacts in the DCT domain, in particular, on histograms of block-DCT coefficients [PF04]. In [WZ16] and [VAK18] authors proposed to use as input the concatenation of DCT histograms for their CNNs. These approaches outperformed non-deep-learning methods, especially on small-sized images up to 64x64 pixels. Afterwards, Barni *et al.* [BBB17] found that CNN architectures could detect double JPEG compression with high accuracy when the input of the network was noise residuals or histogram features; this was tested on double compression with both different and same quantization matrix.

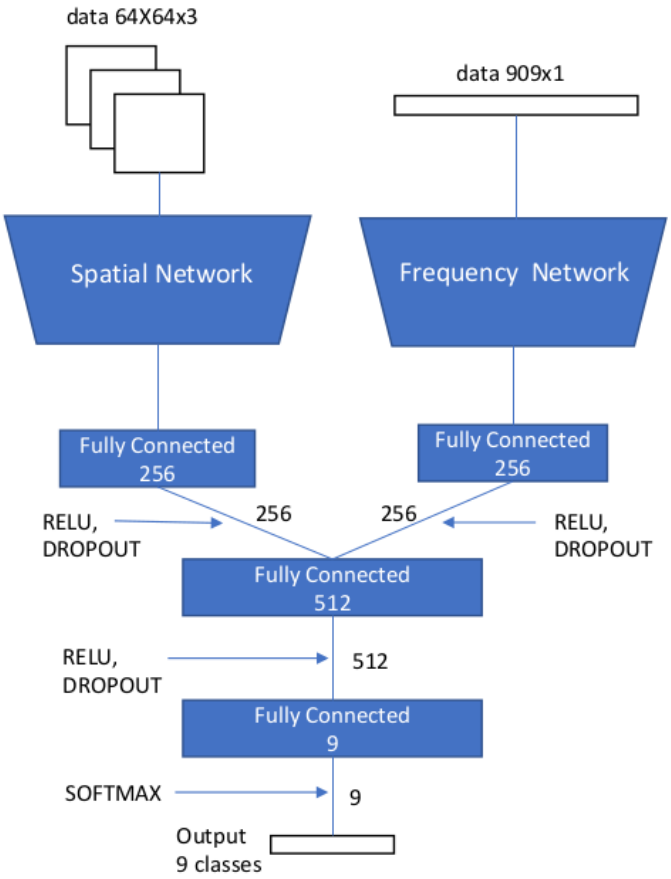

Fig. 2.7.: Architecture of the multi-domain convolutional neural network proposed in [Ame+17] for double JPEG compression detection.

In [Ame+17], Amerini *et al.* designed a multi-domain convolutional network to detect and localize double JPEG compression. They proposed to use both DCT coefficients and spatial features for the localization. The architecture achieved a better detection accuracy when compared to using only pixel values or DCT coefficients. In their implementation, two branches were used as inputs for the network, one receiving the image patches and the other the DCT coefficients. After several convolutional blocks (convolutional layer, activation function and pooling layer), both outputs are concatenated and fed to a final fully-connected layer followed by the classification layer for detecting different JPEG quality factors. Figure 2.7 shows the proposed multi-domain neural network. The architecture of the sub-network with the frequency-domain input has some similarities to the one in [WZ16], while the range of the bins in the DCT histogram is augmented.

The method proposed in [Par+18] extracted block-wise histogram-related statistical features under mixed quality factor conditions to achieve better accuracy and localization capability. The proposed CNN takes a multi-branch approach using

histogram features and quantization tables as inputs. The quantization branch is directly concatenated to the last max-pooling layer output and two fully-connected layers. Authors reported that the ability of the network to distinguish between single and double JPEG compressed blocks was dramatically improved by including quantization table branch.

The above presentation suggests that using special features as input for the first layer of CNN can achieve good detection performance and that in the case of using multiple inputs the multi-branch approach can combine them properly.

Contrast enhancement detection

Like median filtering, contrast enhancement is one of the routine image manipulations commonly applied to conceal the traces of tampering. In the case of a falsified image, it is common to have contrast differences between the background and the forged region, which may be caused by different lightning conditions. In this scenario, contrast enhancement is broadly used to remove or alleviate visual clues that would give away the forgery. Consequently, detecting the application of this operation has drawn researchers' attention in the image forensics field [SL10].

In [Bar+18] authors proposed a 9-layer CNN that is directly fed with 64x64 image pixel values with no special features, making the discriminative features self-learned by the network. Authors showed good robustness against JPEG compression post-processing over a wide range of quality factors by training the network with different contrast adjustments. The proposed architecture also generalized well to unseen tonal adjustments.

Sun *et al.* [Sun+18b] proposed to use the Gray Level Co-occurrence Matrix (GLCM) which is computed by accumulating the occurrence of the pixel intensity pairs between each pixel and its neighboring pixels. The GLCM was used as input for a shallow CNN of three convolutional groups for detecting contrast enhancement. The authors reported good results when an image is JPEG compressed after the contrast enhancement on 256x256 image patches. The proposed method outperformed the conventional ones in terms of the manipulation detection accuracy.

Using the GLCM as input of the network in a similar way, Shan *et al.* [Sha+19] also proposed a JPEG-robust forensic technique based on CNN to detect both local and global contrast enhancement. The adopted network architecture is one convolutional block (4 layers in one block) deeper than the one proposed in [Sun+18b]. Experimental results showed that Shan *et al.*'s method could efficiently detect both

local and global contrast enhancement in compressed images regardless of the order of contrast enhancement and JPEG compression.

General-purpose manipulations detection

The manipulation detection methods presented until now focus on the detection of a specific and targeted manipulation. This limits the application range of such methods because for creating a doctored image, several different processing operations can be applied to obtain a visually convincing result. For instance, in the case of splicing falsification, the forged part of the image can go through one or several basic operations such as rescaling, contrast enhancement and median filtering. Therefore, it is of great importance to develop general-purpose strategies that are capable of detecting different kinds of image manipulation operations.

As mentioned in previous subsections, the usage of special features in the CNN input in general leads to a better performance for image forensic problems. Following this approach, Bayar and Stamm [BS18a] proposed a new constrained filter for the first layer of a CNN to suppress the image contents for detecting various image processing operations. Their constrained network is forced to learn a set of high-pass filters by imposing a constraint on the weights of all the K first-layer filters in each forward pass of the learning process as shown in the following equation [BS18a]:

$$\begin{cases} w_k^{(1)}(0,0) = -1, \\ \sum_{m,n\neq 0} w_k^{(1)}(m,n) = 1, \end{cases} \tag{2.5}$$

where $w_k^{(1)}(m,n)$ denotes the weight at position (m,n) of the kth filter in the first layer (the indices m and n can be negative or positive), and $w_k^{(1)}(0,0)$ denotes the weight at the center of the corresponding filter kernel. In this manner the sum of all filter elements in each filter is 0, and the constrained first-layer filter operates like a high-pass one by effectively removing image content. This prediction error layer extracts and highlights the local dependency of pixels with its neighbours, which is an important piece of information from the forensics point of view. Experimental results in [BS18a] also showed that the usage of tanh as activation function outperforms the more common functions such as ReLU. The reason may be that tanh tends to preserve more information related to the sign of the values at the function input, without setting all negative values to be 0 as in ReLU. The sign information may be important for image forensic tasks.

Recently, Castillo Camacho and Wang [CW19] proposed a different initialization method for the first layer of a CNN to cope with a challenging setting of general-

Tab. 2.3.: Image manipulation detection methods. Network depth describes the number of convolutional blocks with C for a convolutional layer, or M for mlpconv layer, followed by an activation function and pooling layer, and the number of fully-connected blocks denoted by an F (fully-connected layer and activation function). MF stands for median filtering, DJPEG for double JPEG, CE for contrast enhancement, GIPO for general-purpose image processing operations, and approach is color coded as follows: D detection, L localization.

Problem	Method	Network depth	Input feature	Special CNN design	Input size	Approach
MF	[CKL15]	5C-2F	MFR	N/A	64x64, 32x32	D
	[Tan+18]	2M-3C	Upscaled values	mlpconv	64x64, 32x32	D
DJPEG	[VAK18]	4C-2F	DCT features	N/A	128x128	D
	[WZ16]	2C-2F	DCT features	Customized 3x1 kernels	64x64, 128x128,..., 1024x1024	D
	[BBB17]	3C-2F	Noise residuals or DCT features	N/A	64x64, 256x256	D
	[Ame+17]	2C-2F, 3C-1F	DCT features, pixel values	Two-branch CNN	64x64	D L
	[Par+18]	4C-3F, 3F	DCT features, quantization tables	Two-branch CNN	256x256	D L
CE	[Bar+18]	9C-1F	Pixel values	N/A	64x64	D
	[Sun+18b]	3C-2F	GLCM	N/A	256x256	D
	[Sha+19]	4C-2F	GLCM	N/A	256x256	D
GIPO	[BS18a]	5C-2F	Pixel values	Constrained 1st layer	256x256	D
	[CW19]	5C-2F	Pixel values	Special init. for 1st layer	64x64	D
	[CW20]	5C-2F, 6C	Pixel values	Scaling for 1st layer	64x64	D

purpose image manipulation detection. It is challenging because the considered manipulations are of small amplitude. Taking advantage of the milestone work of the famous Xavier initialization [GB10b], they proposed a way to create random high-pass filters that could operate without constrains. The method had a high detection rate for manipulations like median filtering, Additive White Gaussian Noise (AWGN) and resampling. Recently, the same authors [CW20] proposed a data-dependent scaling approach for first-layer filters initialized by different algorithms. The proposed approach took into account natural image statistics and could ensure the stability of the amplitude (*i.e.*, the variance) of data flow in a CNN, which was beneficial for general-purpose image manipulation detection. These two methods [CW19; CW20] will be detailed in subsequent chapters of this manuscript.

Table 2.3 summarizes existing deep-learning-based image manipulation detection methods, by considering different technical aspects in particular the input feature of the network and the specificity of CNN design.

2.2.3 Falsification detection

We consider image falsification as the creation of fake content in some part of the image to deceive viewers about the facts happened in the past. Different from routine image manipulation, image falsification is conducted intentionally to change the image's semantic meaning, often by inserting or removing certain content.

The most common image falsification techniques can be roughly divided into three broad categories: copy-move forgery where one part of the image (the source region) is copied and pasted into the same image as the fake part (the target region); splicing forgery where the tampered region in a host image was originally from a different image; and inpainting forgery which is sometimes considered as a subgroup of copy-move with the difference that the fake region in inpainting falsification is often constructed by using and combining small motifs at different locations of the same image. It is worth mentioning that the inpainting technique is traditionally used to reconstruct a lost or corrupted part of the image and that inpainting falsification is often applied for carrying out object removal in an image. Research on splicing detection is in general more active than copy-move and inpainting. This is probably because it is more convenient to create diverse splicing forgeries from a large pool of publicly available pristine images. Figure 2.6 shows, from left to right, examples of inpainting, copy-move and splicing forgeries. In the following, we will organize the presentation of deep-learning-based falsification detection methods into two groups: 1) *multipurpose* detectors which can detect different kinds of image forgeries among the above three categories and 2) *targeted* detectors which are focused on the detection of one specific falsification.

Multipurpose detectors

Multipurpose detectors are usually based on the general assumption that any image falsification introduces statistical deviation with respect to the authentic part, *i.e.*, within the fake region, around the fake region boundary, or both.

Zhang *et al.* [Zha+16] proposed to use an autoencoder [Kra91] which is a type of neural network taking an image as input and reconstructing it using fewer number of bits. Wavelet features were used as input for the network to detect and localize in a patch-wise manner the tampered regions. Besides wavelet features, local noise features originally proposed for steganalysis, like Spatial Rich Model (SRM) [FK12], have been largely used to solve image forensic problems with encouraging results. In SRM, a group of handcrafted filters were designed to extract local noise from

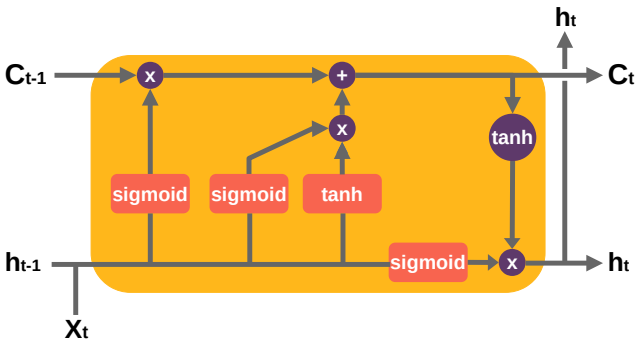

Fig. 2.8.: An LSTM cell. X_t is the input, h_{t-1} and h_t are the output of the previous and current block, C_{t-1} and C_t are the cell state on the previous and current block. An LSTM block can help to correlate neighboring blocks and search for inconsistencies when a forgery is present. This is achieved via gates of activation functions to determine if certain data is relevant for forwarding it or forgetting it.

neighboring pixels, and this often allows us to obtain disparities between forged and original areas. SRM filters have been used for creating a special input for CNNs. This is one important difference from CNNs used in computer vision tasks: it is considered beneficial for CNNs of image forensics tasks to use SRM filters as initialization for the first layer, instead of the random weights conventionally used in CNNs from the computer vision community. In [RN16], Rao and Ni proposed to use the 30 SRM filters as initialization for the first layer in a CNN to detect splicing and copy-move forgeries. The results from the pre-trained CNN were utilized in a Support Vector Machine (SVM) classifier for solving a binary problem (authentic/forged). In a similar approach based on steganalysis features, Cozzolino *et al.* [CPV17] proposed to use a shallow or short CNN to detect image forgeries on small patches.

In [Bun+17] and [Bap+19], authors made use of a Long Short Term Memory (LSTM) architecture for localizing with pixel level the tampered regions. An LSTM as proposed in [HS97] is a special type of Recurrent Neural Network (RNN) designed for sequences or time series data. An LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. Each block contains one or more recurrently connected memory cells and three multiplicative units – the input (sigmoid and tanh functions), output (sigmoid and tanh functions) and forget (a sigmoid function) gates – that regulate the flow of information into and out of the cell. Figure 2.8 shows an unrolled example of an LSTM block. The core strength of using LSTM in the image forensics field is to acquire from previous blocks the boundary information, which is decisive to obtain particular features to classify between original and tampered regions. In [Bun+17] experiments showed that both CNNs with Radon transform as input and LSTM based strategies were effective in exploiting resampling features to detect and localize tampered regions. Bappy *et*

al. [Bap+19] proposed an LSTM and an encoder-decoder network to semantically segment falsified regions in a tampered image.

Some researchers suggested that a CNN trained for detecting camera traces could be used to detect and localize image splicing. If an analyzed image contains patches of different sources, then the blocks can be clustered in different groups separating the suspicious area. Works in [Bon+17] and [CV18] made use of camera-specific features obtained by a CNN that focuses on them. Both methods analyzed patches and looked for traces of different cameras in the same image. Bondi *et al.* [Bon+17] used a clustering algorithm to create different groups of the authentic and suspicious areas. In [CV18] a noise residual called *Noiseprint* was extracted and utilized to check inconsistencies within a single image.

Yarlagadda *et al.* [Yar+18] used a GAN that included an adversarial feedback loop to learn how to generate some information in a realistic manner, with the objective to detect satellite image forgeries. There are two major components within GANs: the *generator* that takes a noise vector as input and outputs an image improved at each step with the knowledge of what a valid input should be, and the *discriminator* that tries to classify between real and fake (*i.e.*, created by generator) contents. Their proposed architecture was followed by an SVM to detect whether feature vectors come from pristine images or forgeries.

Recently, [Zho+18] and [WAN19] proposed the multi-branch CNNs to tackle the challenge of image forgery detection. Specifically, Zhou *et al.* [Zho+18] proposed a multi-branch Region-Convolutional Neural Network (R-CNN) which is a type of CNN typically used for object detection to coarsely locate the tampered regions in bounding boxes. Authors used pixel values in one branch with ResNet-101 architecture [He+16] and noise features obtained by SRM filters in the second branch. Wu *et al.* [WAN19] suggested a multi-branch CNN joined with an LSTM trained with a set of 385 different image manipulations. Their architecture named Mantra-Net generates a pixel-level detection mask reflecting the probability of a falsification. In the three input branches of Mantra-Net the first layers are initialized with SRM filters, high-pass constrained filters of Bayar and Stamm [BS18a], and normal random weights. Figure 2.9 shows example results of bounding-box localization of falsifications produced by Zhou *et al.*'s detector [Zho+18].

Very recently, Mara *et al.* [Mar+20] worked on a full-image CNN based on Xception architecture [Cho17] to detect and localize image falsifications. The proposed end-to-end network utilized the *Noiseprint* [CV18] as features extracted from the image input. Meanwhile, in [Zho+20] a GAN was proposed to generate falsified images avoiding the burdensome task of creating and labeling image forgery examples

Fig. 2.9.: Bounding-box localization results generated by using the implementation of [Zho+18] on NIST 16 dataset [NIS16]. Top and bottom rows show copy-move and splicing examples respectively. (A) is the original image, (B) is the falsified image, (C) is the ground-truth mask, and (D) is the localization result.

in a conventional way. With this big amount of synthetic examples, the proposed algorithm was able to segment and refine the focus on boundary artifacts around falsified regions during the training process.

Table 2.4 provides a summary of the various multipurpose falsification detection techniques. The summary includes the method reference, input for the network, initialization used in the first layer, input size, localization level, considered databases, and network type.

Targeted detectors

Targeted detectors, which are designed to detect only one type of image falsification, have been developed in parallel with multipurpose ones.

1) Splicing detection

Some early works dealing with splicing detection and localization were based on autoencoders. In [CV16], authors used SRM features as input for their autoencoder model. The method in [DAv+17] used the steganalysis features from SRM to analyze frames in a video with autoencoder and LSTM to detect splicing forgeries.

Wu *et al.* [WAN17] proposed a framework of Constrained Image Splicing Detection and Localization (CISDL) based on the well-known VGG-16 architecture [SZ14]. Using two input branches they calculated the probability that one image had been partially spliced to another one and localized the spliced region. Meanwhile, in [SRK18] and [LP18], a CNN without fully-connected layers known as Fully Convolutional Network (FCN) [LSD15] was used to predict a tampering map for a given

Tab. 2.4.: Multipurpose image falsification detection methods. Loc. level describes whether the localization is performed in a pixel-, block- or bounding-box-wise manner. Dataset is color coded as follows: C CASIA [DW11], M Smartphones, N NIST 16 [NIS16], I IEEE Forensics Challenge [@IEE14], V Coverage [Wen+16], K Kaggle Camera Challenge [@IEE18], L Columbia gray [@NHC04], D DRESDEN [GB10a], O Vision [Shu+17], A Landsat on AWS [@Ama18], U UCID [SS04], F MS COCO [Lin+14], B Columbia color [HC06], H Carvalho [Car+13], and S when it is an ad-hoc dataset created by authors of the original paper.

Method	Input feature	Init. first layer	Input size	Loc. level	Dataset	Network type
[RN16]	Pixel values	SRM filters	128x128	pixel	C L	CNN - SVM
[Zha+16]	Wavelet features	Random init.	32x32	block	C	Autoencoder
[CPV17]	Steganalysis features	Random init.	128x128	pixel	S M	CNN - SVM
[Bon+17]	Pixel values	Random init.	64x64	block	S D	CNN
[Bun+17]	Radon features	Random init.	64x64, 128x128	pixel	N	LSTM
[Bap+17]	Resampling features	Random init.	64x64	pixel	S N I V	CNN
[Zho+18]	Pixel values, noise features	Random init.	224x224	bbox	S F N C V	Multi-branch
[Yar+18]	Pixel values	Random init.	64x64	block	S A	GAN-SVM
[CV18]	Pixel values, Noiseprints	Random init.	44x44, 64x64	pixel	S D	CNN
[Bap+19]	Resampling features	Random init.	Resized 256x256	pixel	S N I V	LSTM
[WAN19]	Pixel values	SRM filters, Bayar filters, Random init.	256x256, 512x512	pixel	S D K B N C V	Multi-branch
[Mar+20]	Pixel values, Noiseprints	Random init.	960x720; 4640x3480	pixel	S O U	CNN incremental learning
[Zho+20]	Pixel values	Random init.	224x224	pixel	S V C H	GAN-CNN

image. In [SRK18], the proposed architecture has two exit localization branches. The first one was used for localizing the inner part of the spliced area and the second one for detecting the boundary between pristine and spliced regions. Concurrently, Liu *et al.* [LP18] made use of three FCNs to deal with different scales; moreover, conditional random field was used to combine the results of different scales.

Some approaches [Huh+18; Pom+18] attempted to detect anomalies or inconsistencies within tampered images. In [Huh+18], a Siamese CNN with a self-consistency approach to determine if contents had been produced by a single device was proposed. The proposed model could predict the probability that two patches had similar EXchangeable Image File (EXIF) attributes and output a "self-consistency" heatmap, highlighting image regions that had undergone possible forgery. In [Pom+18] au-

thors used transfer learning from a pre-trained residual network (ResNet-50) with illumination maps taken from input images to find hints of forgeries.

Recent strategies [KKR19; Bi+19] made use of U-Net [RFB15] architectures. In a U-Net, the features are captured by a size-reducing way of consecutive layers, then upsampled and concatenated with the first path in a *U*-shaped symmetric path, attempting to reduce loss and improve localization capability. In [KKR19], authors took advantage of U-Net architecture for the training of a GAN with image retouching generator, which helped a splicing localization model to learn a wide range of image falsifications. Meanwhile Bi *et al.* [Bi+19] proposed a method mainly based on U-Net as a segmentation network for splicing forgery detection.

Given the popularity of GANs in the computer vision field, some researchers have also started to use them for image forensics purposes. This is the case of [Bar+19] where the authors made use of a conditional GAN for the training of a detector to locate forgeries in satellite images. Liu *et al.* [Liu+19] proposed a deep matching CNN together with a GAN to generate probability maps in a CISDL scenario.

Special initialization of first layer was also considered for splicing detector. For example, Rao *et al.* [RNZ20] designed and implemented a CNN with the first layer of the network initialized with 30 SRM filters to locate splicing forgeries.

Table 2.5 summarizes the targeted detectors of splicing falsification. The considered properties of the detection methods are similar to those in Table 2.4.

2) Copy-move detection

Copy-move detection is one of the forensic techniques that have been studied with more balance between conventional and deep-learning approaches. As mentioned before, in a copy-move forgery, a part of the original image (source area) is copied and pasted at a different place (target area) of the same image. Before pasting, the target area can be transformed (rotation, scaling, shearing, *etc.*) to make the forgery visually realistic. Routine image manipulation (smoothing, contrast adjustment, *etc.*) can be applied locally or globally to enhance the visual quality. Copy-move is mainly used for falsifications where certain content needs to be disguised or cloned.

Probably the first proposal using a deep-learning approach to solve the copy-move detection problem was the method from Ouyang *et al.* [OLL17], which was based on the famous pre-trained AlexNet [KSH12] originally designed for image recognition. Authors generated forged images by choosing a random square from the upper left corner and copying it to the center. Although this method obtained decent results in this artificial scenario, the performance was diminished for realistic forgeries.

Tab. 2.5.: Targeted splicing detection methods. AE stands for autoencoder. Dataset is color coded as follows: C CASIA [DW11], M Smartphones, N NIST 16 [NIS16], T SUN 2012 [Xia+10], F MS COCO [Lin+14] , L Columbia gray [@NHC04], H Carvalho [Car+13], R Realistic (Korus) [KH17], W On the wild websites, B Columbia color [HC06], A Landsat on AWS [@Ama18], and S when it is an ad-hoc dataset created by authors of the method (information of source images used for dataset creation may be provided in the original paper).

Method	Input feature	Input size	Dataset	Network type	Backbone architecture
[CV16]	SRM features	768x1024	S	AE	Own
[WAN17]	Pixel values	256x256	S T F	CNN	VGG-16
[DAv+17]	SRM features	720x1280	S	AE-LSTM	Own
[SRK18]	Pixel values	224x224	S C L H N	FCN	VGG-16
[Huh+18]	EXIF metadata, pixel values	128x128	S L H R W	CNN	ResNet-v2
[Pom+18]	Illuminant maps	224x224	S L H	CNN-SVM	ResNet-v1
[LP18]	Pixel values	224x224	S B	FCN	VGG-16
[Bi+19]	Pixel values	384x384	S C B	CNN (U-Net)	ResNet-v1
[KKR19]	Pixel values	512x512	S	GAN (U-Net)	VGG-16
[Liu+19]	Pixel values	256x256	F	GAN	VGG-16
[Bar+19]	Pixel values	70x70	A	GAN	Pix2Pix
[RNZ20]	SRM features for 1st layer init.	128x128	S C H	CNN-SVM	Own

Wu *et al.* [WAN18b] proposed a CNN-based method which first divided the input image into blocks, then extracted special features, correlated features between blocks, localized matches between blocks and finally predicted a copy-move forgery mask. Furthermore, routine image manipulation operations to hide the forgery traces such as JPEG compression, blurring and AWGN were applied to training data as a means of data augmentation. The objective was to easily detect these manipulations as possible telltales of copy-move falsification. Very shortly after this piece of work, the same authors [WAN18a] proposed to use a different architecture with two exit branches to deal with the problem of source-target disambiguation where it is necessary to discern between source (original) and target (falsified) regions in a copy-move forgery. Another deep learning method for source-target disambiguation was proposed in [BPT19] where CNN with multi-exit branches was also used to identify source and target regions. This method was shown to be capable of learning special features focusing on the presence of interpolation artifacts and boundary inconsistencies. Figure 2.10 shows two examples of source-target disambiguation localization results generated by Wu *et al.*'s detector [WAN18a].

In [LGZ18] Liu *et al.* proposed one of the first copy-move detectors that used a CNN approach. Their proposal was partially based on conventional methods, by taking keypoints features such as Scale-Invariant Feature Transform (SIFT) or Speeded-Up

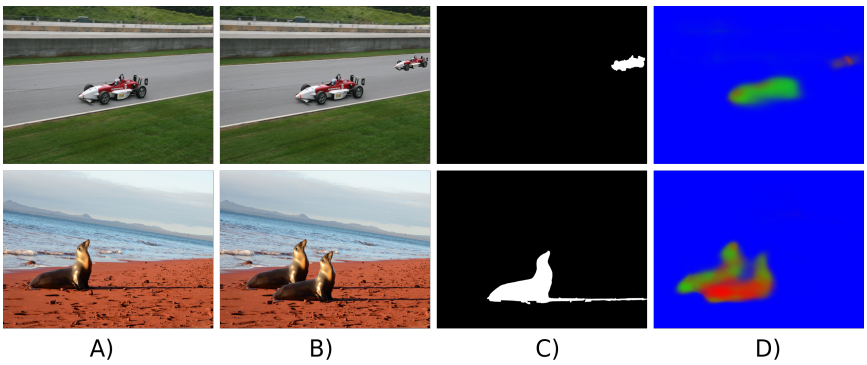

Fig. 2.10.: Source-target disambiguation results generated by using the implementation of [WAN18a] on images from the NIST 16 dataset [NIS16]. (A) is the original image, (B) is the falsified image, (C) is the ground-truth mask, and (D) is the localization result in which green and red color represents respectively the source (original) and target (falsified) region in a copy-move forgery.

Robust Features (SURF) as input for their network. One limitation was that this method had low performance when duplicated areas have a homogeneous content, because the keypoints could be hardly identified within such areas.

Very recently, Zhu *et al.* [Zhu+20] proposed an adaptive attention and residual based CNN to localize copy-move forgeries. The self attention module allowed neurons to interact with each other to find out which neurons should receive more attention. Experiments showed comparable results with previous deep-learning approaches, but the problem of source-target disambiguation was not addressed.

Illumination direction, contrast and noise are usually inconsistent in splicing forgery, so the tampering traces could be found rather easily by the CNN. However, the source and target regions are derived from the same image in copy-move, accordingly the illumination and contrast would be highly consistent, which raises a greater challenge for copy-move detection based on CNN. This may be one reason for the fewer published papers focused on copy-move when compared with splicing. The first part of Table 2.6 summarizes the existing deep-learning methods targeted at copy-move detection and localization.

3) Inpainting detection

The inpainting technique can create plausible image forgeries which are difficult to spot for the naked eyes. Different from copy-move where an image area is copied and pasted, in inpainting the falsified area is often filled with micro components (*e.g.*, blocks of 7 by 7 pixels) extracted from different places of the image. These small blocks usually represent a kind of micro-texture and are combined in inpainting in a visually convincing way. Although the inpainting method can be used for inoffensive

Tab. 2.6.: Targeted detectors of copy-move and inpainting falsifications. S-T disam. means source-target disambiguation. Dataset is color coded as follows: X Oxford [NZ08], U UCID [SS04], T SUN 2012 [Xia+10], circledmark[turqdos]F MS COCO [Lin+14], P ROME patches [Mat+15], M CoMoFoD [Tra+13], V Coverage [Wen+16], K Raise [Dan+15], D DRESDEN [GB10a], O Vision [Shu+17], G Imagenet [Den+09], L MIT Place [Zho+14], C CASIA [DW11], V MvTec [Ber+19] and S when it is an ad-hoc dataset.

Method	Input features	Input size	Localization level	Dataset	Backbone architecture
Copy-move					
[OLL17]	Pixel values	256x256	Detection	S X U	AlexNet
[WAN18b]	Pixel values	256x256	Pixel	S F T	VGG-16
[LGZ18]	Keypoints	51x51	Pixel	S P M C	Own
[WAN18a]	Pixel values	256x256	Pixel, S-T disam.	S F T	VGG-16
[BPT19]	Pixel values	64x64	Pixel, S-T disam.	S K D O	ResNet-V1
[Zhu+20]	Pixel values	256x256	Pixel	S C M V	VGG-16
Inpainting					
[Zhu+18]	High-pass residuals	256x256	Pixel	S L	Own
[WWN19]	Pixel values	128x128	bbox	S G	ResNet-V1
[LH19]	High-pass residuals	Various	Pixel	S G	ResNet-V1
[WNW20]	LBP, pixel values	256x256	Pixel and bbox	S F G	Own
[ZKS20]	Pixel values	256x256	Pixel	S V	Own
[LN20]	Pixel values	32x32	Pixel and bbox	S U	Own

purposes such as repairing partially deteriorated images, it is utilized likewise in forgery creation, for instance for object removal to falsify an image or for erasing visible watermarks. Some splicing or copy-move detection algorithms could be exploited to detect inpainting forgeries, but in general they do not consider the particularity of inpainting and their performance remains not as good as expected.

To our knowledge the first method targeted at inpainting detection was proposed by Zhu *et al.* [Zhu+18], where authors used an encoder-decoder network to predict the inpainting probability on each patch. Li and Huang [LH19] focused on detecting inpainting forgeries made by deep learning methods (also known as deep-inpainting). Image high-pass residuals were fed to a FCN in which transpose convolutional layers were initialized with bilinear kernel.

Wang *et al.* [WWN19] used a R-CNN, originally designed for object detection, to output a bounding box of the inpainted region along with a probability score. Very recently, the same authors [WNW20] designed a multitask CNN with two inputs, *i.e.*, a Local Binary Pattern (LBP) image as the first input and the pixel values as the second one, for inpainting detection. This new network could produce a bounding box of inpainted area together with an estimated mask of forgery.

In [ZKS20] authors proposed an anomaly detection method by randomly removing partial image regions and reconstructing them with inpainting methods to detect a forgery. Authors used a U-Net based encoder-decoder network to reconstruct the removed regions and output a tampering map in which each image is assigned an anomaly score according to the region with the poorest reconstruction quality. Meanwhile, Lu and Niu [LN20] published an object removal detection method by combining CNN and LSTM to detect inpainting with single and combined post-processing operations such as JPEG compression and Gaussian noise addition.

The second part of Table 2.6 provides a brief summary of the deep-learning-based forensic methods targeted at inpainting falsification.

2.2.4 Other specific forensic problems

This subsection is dedicated to the presentation about some other specific problems on which the image forensics research community has conducted extensive work. We divide them into three groups: 1) camera identification, 2) computer graphics image detection, and 3) detection of Deepfake images.

1) Camera identification

A typical image acquiring process is shown in Figure 2.11. First, the light rays are redirected by the lens, then different filters such as anti-aliasing can be applied before the Color Filter Array (CFA) divides the light into one of the red (R), green (G) and blue (B) components per pixel. A demosaicing step is performed afterwards to reconstruct the full-color scene from the input samples taken by the previous step. Depending on the camera model and software, several post-processing operations such as white balancing, gamma correction and JPEG compression can take place. These post-processing steps contribute with important and distinctive clues to the image forensics field. When the final output image of camera is falsified to create a forgery, additional traces unique for each falsification are usually left behind.

The challenge of verifying the authenticity of an image can be tackled from different perspectives. One of them is approached by answering the following question: given an image, is it possible to find out the model of the camera with which the image was taken? Despite the fact that camera model, date and time, and other information can be found in the EXIF or in the JPEG header, in general it is not possible to consider such information as reliable and legitimate because image metadata can be easily

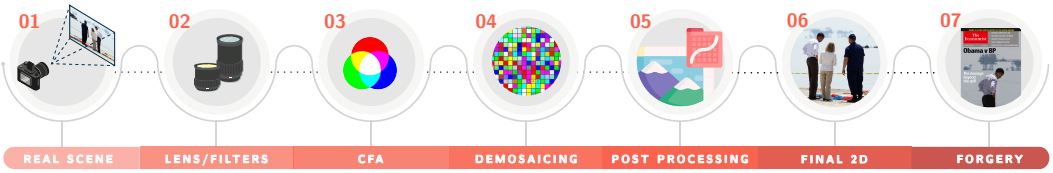

Fig. 2.11.: Illustration of typical pipeline of image acquisition and forgery creation.

modified. By contrast and as mentioned before, the traces of the post-processing steps carried out by each camera constitute important source of information that can be used to authenticate the image provenance in the image forensics field.

First deep learning methods for camera identification were mainly dedicated to classifying patches produced by different cameras. Bondi *et al.* [Bon+16] used a CNN followed by an SVM to classify patches coming from different unknown cameras. In addition, with the output of their CNN they looked for anomalies in an image to search for forgeries. Tuama *et al.* [TCC16] applied a high-pass filter in the first layer to suppress image content and obtain image residuals as input for a shallow CNN that was trained to learn to classify among different camera models. Due to the release of new camera models and the difficulty to keep an updated database, Bayar and Stamm [BS18b] suggested an open-set scenario which aimed to predict an unseen camera device. Authors used a constrained initialization for the first layer of a CNN to infer whether the image was taken by an unknown device.

Ding *et al.* [Din+19] proposed a multi-task CNN to predict information about brand, modes and devices from a patch. Authors used ResNet [He+16] blocks together with high-pass filter residuals as input for the network and with inputs of different sizes. In [Fre+19], authors used a shallow CNN for mobile camera identification in a multi-class challenging scenario. Experiments showed good forensic performance, but the performance diminished when devices came from a same manufacturer.

Methods in [CV19] and [SN20] both used Siamese network for this camera classification problem. There are multiple inputs in a Siamese network with the same architecture and same initial weights for each sub-network. Parameter updating is mirrored across all sub-networks. The purpose of this architecture is to learn the similarity of inputs. In [CV19], authors proposed a Siamese CNN to extract the camera unique fixed-pattern noise from an image's Photo Response Non-Uniformity (PRNU) to classify camera devices and furthermore trace device fingerprints for image forgery detection. Sameer and Naska [SN20] worked on the scenario where annotated data

Tab. 2.7.: Camera identification methods. ET means extremely randomized trees. Dataset is color coded as follows: D Dresden [GB10a], I MICHE-I [De +15], H Carvalho [Car+13], V VIPP [BP12], R Realistic (Korus) [KH17], N NIST 16 [NIS16], F MFC [MFC17], O Vision [Shu+17], M Smartphones, and S when it is an ad-hoc dataset.

Method	Input features	Initialization	Input size	Dataset	Network type
[TCC16]	High-pass residuals	Random init.	256x256	D	CNN
[Bon+16]	Pixel values	Random init.	64x64	S	CNN-SVM
[BS18b]	High-pass residuals	Bayar's constrained	256x256	D	CNN-SVM CNN-ET
[Fre+19]	Pixel values	Random init.	32x32	I	CNN-SVM
[CV19]	Pixel values	Random init.	48x48	H V R N F	Siamese
[Din+19]	High-pass residuals	Random init.	48x48	D M	Multi-scale CNN
[SN20]	Pixel values	Random init.	64x64	D O	Siamese

(*i.e.*, in this case image samples) were not available in big quantities and training had to be performed using a limited amount of samples per class. This approach is called *few-shot* learning and refers to learning and understanding a new model based on a few examples. For this few-shot learning approach, a Siamese network was used to enhance classification accuracy of camera models. The intuition behind the Siamese network for this challenge is to form pairs of image patches coming from the same camera models to improve the training.

Table 2.7 gives a brief summary of the deep-learning-based camera identification techniques. We can observe two interesting trends regarding the relevant research: 1) techniques like few-shot learning would be helpful in realistic scenarios in which we have a limited number of annotated samples and 2) the deep learning methods are promising techniques to deliver a good camera model classification performance and may further help in the search of anomalies for image forgery detection.

2) Detection of computer graphics images

Computer graphics techniques produce visually plausible images of fictive scenes. Despite the benefits of CGI in virtual reality and 3D animation, it can also be used as false information thus affecting real-life decisions, and this situation is augmented with the fast dissemination of content enabled by the Internet. Consequently, the challenge of discerning between a real photograph and CGI has been explored by image forensics researchers. Figure 2.12 shows how challenging it is to distinguish between CGI and an image taken by a camera.

Fig. 2.12.: Examples to show the difficulty of visually differentiating between CGI (on the left) and an image taken by a camera (on the right). The CGI is from Tumblr forum (https://hyperrealcg.tumblr.com/post/112323738189/title-a-land-where-dreams-take-wings-artist) and the camera image is from Reddit forum (https://www.reddit.com/r/EarthPorn/comments/4o9u03/no_filter_needed_grand_tetons_national_park_wy_oc/).

Rezende *et al.* [DRC17] took advantage of transfer learning from ResNet-50 model to classify small patches of CGI and real photographic images. Yu *et al.* [Yu+17] investigated for this CGI forensics problem the usage of a CNN without pooling layers. Authors of [Rah+17] and [Qua+18] proposed to use shallow CNNs in a patch-based manner. Rahmouni *et al.* [Rah+17] used a CNN with a customized pooling layer that computed statistics like mean and variance followed by an SVM to detect CGI patches. In order to classify a whole image, a weighted voting strategy was applied to combine the local probabilities on patches of sliding windows to produce a final label. Quan *et al.* [Qua+18] proposed an end-to-end approach using a Maximal Poisson-disk Sampling (MPS) method to crop patches in a loss-less manner from a full-sized image. Nguyen *et al.* [Ngu+18] continued with the sliding window approach to deal with high-resolution images using VGG-19 followed by multi-layer perceptron based CNN as classifier. In [Yao+18], authors proposed an approach for discriminating CGI using high-pass residuals as input to a CNN.

He *et al.* [He+18] designed a two-input CNN-RNN taking the color and texture from YCbCr color space on each input to detect CGIs. In [Tar+19] authors investigated the usage of an Attention-Recurrent Neural Network (A-RNN) to classify CGIs in a local-to-global approach following the sliding window strategy and using the simple majority voting rule to produce a decision on a whole image. Nguyen *et al.* [NYE19] studied the application of dynamic routing capsule networks [SFH17] based on the VGG-19 model for detecting CGI. Capsule networks were able to identify objects that hold spatial relationship between features.

More recently, Zhang *et al.* [Zha+20] proposed a CNN containing a special block at input called hybrid correlation module composed of a 1x1 convolution layer

Tab. 2.8.: CGI detection methods. Dataset is color coded as follows: M Smartphones, C Columbia CGI [Ng+05], R Rahmouni [Rah+17], K Raise [Dan+15] , H Carvalho [Car+13], F MesoNet [Afc+18], W Web images, N Faceforensics [Rös+18], H He [He+18], O Vision [Shu+17], T Artlantis [@ABV05], B Corona [@Cha20], Y Vray [@Ltd20], G Imagenet [Den+09], K Autodesk [@Aut20], and S when it is an ad-hoc dataset.

Method	Input size	Dataset	Network type	Backbone architecture
[DRC17]	224x224	G	CNN-SVM	ResNet-50
[Yu+17]	32x32	S C W	CNN	VGG-16
[Rah+17]	100x100	R	CNN-SVM	Own
[Qua+18]	30x30 to 240x240	C R	CNN	Own
[Yao+18]	650x650	R	CNN	Own
[Ngu+18]	100x100	R K	Two-input CNN-RNN	VGG-19
[He+18]	96x96	H W	CNN-RNN	ResNet-50
[Tar+19]	30x30 to 240x240	C K R	A-RNN	Own
[NYE19]	128x128	R F N R	Capsule	VGG-19
[Zha+20]	96x96	H	CNN	Own
[MT20]	224x224	H	CNN	DenseNet-201
[He+20]	32x32,64x64	H	Two-input A-CNN	Inception
[Qua+20]	233x233	K O T K B Y	Two-branch CNN	Own

followed by three blocks of convolutional layers, which would correlate channels and pixels in an attempt to detect CGIs. Meena and Tyagi [MT20] used the transfer learning approach from DenseNet-201 [Hua+17] followed by an SVM as classifier. In [He+20] authors made use of a shallow A-CNN with two inputs for CGI classification. Interestingly, the inputs for this network were pre-processed by a Gaussian low-pass filter as the authors wanted to focus on general patterns rather than local details. Quan *et al.* [Qua+20] designed a CNN combining SRM filters and Gaussian random weights as initializations for the first layer on a two-branch architecture. Authors also proposed to use the so-called negative samples created via gradient-based distortion to achieve a better generalization on test images created by unknown graphics rendering engines.

Table 2.8 summarizes the deep-learning-based CGI forensic techniques.

3) Deepfakes detection

Lately, GAN models have been used in various applications and have transformed a time-consuming task previously reserved to high-skilled experts now to a simple

Fig. 2.13.: Example frame of a Deepfake video. The tool used to generate this video is available at the following webpage: `https://faceswap.dev/`, and the full resulting video can be viewed at `https://www.youtube.com/watch?v=r1jng79a5xc`.

and fast operation. One of such applications is to create synthetic yet visually realistic images and videos. GAN-generated multimedia contents are commonly known as *Deepfakes*, making reference to the usage of a deep learning model and the fabricated synthetic results. Majority of cases have been used to replace a person (or a person's face) in an existing image or video with another person (or the face of this other person). Figure 2.13 illustrates the synthesis process realized by a GAN which replaces the face in the target (image on the left) by using a source (image in the middle) to generate the resulting frame (image on the right). Although benign material has been created for the illustrated example, this technique can have more serious impact in other situations, *e.g.*, to create political distress. Recently a big amount of research activities has been dedicated to detecting GAN-generated fake contents, mainly due to the easiness and impact of Deepfakes. In comparison with images, videos contain more information and different approaches have been proposed based on different kinds of clues for the detection of Deepfake videos.

First proposals in the literature [MCL18; Mar+18; Afc+18; Cha+19] focused on the detection of GAN-generated images created by a specific GAN model. In [MCL18], authors searched for statistical artifacts introduced by GAN with a pre-processing layer that extracted high-pass residuals. Marra *et al.* [Mar+18] tested the performance of some popular CNN-based image forensic methods for the detection of images created by GANs and shared in social networks. In [Afc+18], authors used a shallow CNN to detect Deepfakes and Face2Face [Rös+18] videos. Interestingly, Chan *et al.* [Cha+19] developed as first objective a GAN for video-to-video translation in dancing poses. Additionally, they developed a detector that would detect videos coming from their own model. In [Tar+18], authors compared several

popular sophisticated architectures and a shallow CNN. Experiments showed that the shallow CNN had better performance in detecting Deepfakes.

Güera and Delp [GD18] proposed to use a CNN for frame feature extraction and an LSTM for temporal sequence analysis to detect Deepfake videos which contained inconsistent frames. Amerini *et al.* [Ame+19] investigated the use of optical flow vectors to detect discrepancies in motion across several frames using the PWC-Net model [Sun+18a]. Optical flow is a vector computed on two consecutive frames to extract apparent motion between the observer and the scene itself. In a follow-up work [AC20], an LSTM was utilized in a sequence-based approach which exploited the dissimilarities between consecutive frames of Deepfake videos.

Other proposals like [CDY20],[LCL18], [Kor+19], [Aga+20] and [Mit+20] focused on the spatial coherence and temporal consistency among different physiological features. In [CDY20], authors designed a CNN to detect variations of heart rate extracted from face regions on different frames. Li *et al.*'s method [LCL18] was based on the observation that faces in Deepfake videos had a lower rate of blinking in comparison with real videos. This occurred in early GAN-generated videos for which the GAN was trained on faces with open eyes. Authors carried out a couple of pre-processing steps to locate the eyes and used this feature as input for an LSTM to detect a lower or higher rate of blinking as a telltale of Deepfake videos. Korshunov *et al.* [Kor+19] proposed an LSTM to search for anomalies between the audio and mouth movements. The method in [Aga+20] went in the same direction by comparing mouth shapes with the sound associated with M, B an P phonemes which required complete mouth closure and were in many cases incorrectly synthesized. Recently, Mittal *et al.* [Mit+20] went a step forward using a Siamese network to look for anomalies between the audio and video and combined it with the affective cues of both inputs to learn the differences between real and Deepfake videos.

The signal-level artifacts introduced during the synthesis were investigated for the detection of fake contents. Li *et al.* [Li+20a] focused on artifacts at face boundaries by exploiting the fact that most existing face tampering methods shared a common blending operation. Meanwhile in [LL18], authors exploited the inconsistencies between warped face area and the surrounding background. The method in [Xua+19] adopted Gaussian noise extraction as a pre-processing step for a CNN, enforcing the network to learn more meaningful features about GAN traces.

In [Ngu+19] a multi-task CNN was proposed to detect fake faces and to segment tampered areas. Dang *et al.* [Dan+20] investigated the use of attention mechanism for the detection and segmentation of tampered faces. In [Din+20], authors used deep transfer learning for face swapping detection. Hsu *et al.* [HZL20] made use

of a so-called Common Fake Feature Network (CFFN) consisting of several dense units and a Siamese network for Deepfake detection. One limitation was that the CFFN may fail when the fake features of the results of a new GAN were significantly different from most of those used in the training phase.

To overcome data scarcity, [Fer+20] and [KW20] proposed some solutions. Fernandes *et al.* [Fer+20] used a Attribution Based Confidence (ABC) metric to detect Deepfake videos with a deep model only trained on original videos. Khalid and Woo [KW20] formulated the challenge as a one-class anomaly detection problem by using a Variational Autoencoder trained only on real face images and subsequently detected Deepfakes as anomalies.

More recently, Wang *et al.* [Wan+20] used the well-known ResNet50 with careful data preparation to study the artifacts left by GANs. Their method demonstrated good generalization performance on unseen Deepfake contents. In [Li+20b], authors designed a two-branch CNN to exploit the distribution differences between pixels in the face region and the background. Masi *et al.* [Mas+20] proposed a two-branch LSTM to combine color and frequency information. A multi-scale Laplacian-of-Gaussian operator was used in their method, which acted as a band-pass filter to amplify the artifacts.

Table 2.9 provides a summary of Deepfake detection methods presented above. Particularly, in the table we present the main cue used by each method, by grouping cues into several categories as spatial context, generator traces, physiology-inspired, inter-frame consistency, and anomaly classification.

2.2.5 Discussion

Through this survey, we provide a general understanding of the detection methods in the image forensics field. The main content of this chapter has recently been published in an international journal. Readers could extend their reading (*e.g.*, the recent survey of [TR20] with a different classification of methods and a presentation of performance metrics) to gain a better understanding of the recent advances in the rapidly evolving research field of image forensics.

In our survey presented in this chapter, we collected and presented a large number of deep-learning-based methods divided into three broad categories, with a focus on the different characteristics that are particular for the image forensic approaches. It can be observed that a pre-processing step to obtain a certain feature or a special initialization on the network's first layer have been used in many pioneer works

Tab. 2.9.: Deepfake detection methods. Dataset is color coded as follows: H CelebAHQ [Kar+17], Y CycleGAN [Zhu+17], N FaceForensics [Rös+18], A CelebA [Liu+15], P PGGAN [Kar+17], U UADFV [YLL19], T DeepfakeTIMIT [Kor+19], D DFDC [Dol+19], G DFD [@Goo19], M FaceForensics++ [Ros+19], V CelebDF [Li+20c], F MesoNet [Afc+18] and S when it is an ad-hoc dataset.

Method	Input size	Dataset	Network type	Backbone architecture	Image	Video	Cue				
							Spatial	GAN trace	Physiology	Inter-frame	Anomaly
[MCL18]	256x256	H	CNN	Own	●		●				
[Mar+18]	256x256	S Y	CNN	XceptionNet	●			●			
[LCL18]	224x224	S	CNN-LSTM	VGG16		●			●		
[GD18]	299x299	S	CNN-LSTM	Inception V3		●				●	
[Afc+18]	256x256	N	CNN	Inception		●		●			
[Tar+18]	1024x1024	A P	CNN	VGG16, ResNet110, *etc.*	●			●			
[LL18]	224x224	U T	CNN	VGG16, ResNet50,101	●	●	●				
[Cha+19]	256x256	S	CNN	Own		●		●			
[Ame+19]	224x224	M	CNN	PWC-Net		●				●	
[CDY20]	128x128	S M N U V	CNN	Own		●			●		
[Kor+19]	720x576, 512x384	S	CNN-LSTM	Own		●			●		
[Ngu+19]	256x256	N M	AE-CNN	Own	●	●	●				
[Xua+19]	128x128	H P	CNN	Own	●	●		●			
[Din+20]	224x224	S	CNN	ResNet18	●	●	●				
[Aga+20]	128x128	S	CNN	XceptionNet		●			●		
[Li+20a]	64x64	D M V G	CNN	XceptionNet	●	●		●			
[Dan+20]	299x299	S	CNN	XceptionNet, VGG16	●	●	●				
[AC20]	256x256	M	LSTM	Inception V3		●				●	
[Fer+20]	224x224	A V M	ABC-CNN	ResNet50	●			●			
[HZL20]	64x64	A	Siamese-CNN	Own	●			●			
[KW20]	100x100	M	VAE	One-Class VAE	●						●
[Li+20b]	224x224	S N T F	CNN	ResNet18	●		●				
[Mas+20]	224x224	M V	LSTM	Own		●	●				
[Mit+20]	-	T D	CNN	Own		●			●		
[Wan+20]	224x224	M	CNN	ResNet50	●		●				

and still exist in recent ones. It is interesting to see that these characteristics are mainly present in the manipulation, falsification, camera identification and CGI detection methods but scarcely seen in the Deepfake detection works. We have not found clear reasons to explain this observation, and it would be interesting to carry out theoretical studies and practical comparisons with and without the use of pre-processing step and with different initializations of the first layer, for

various forensic problems. This is a research opportunity to be explored in our future work. As any arms race scenario where two opponents, in this case a forger and a forensic investigator, try to make their respective actions successful, both sides will keep evolving with new technologies and challenges. Deep learning has brought a tremendous advance due to its ability to automatically learn useful features from available data and this strength has been used on both sides and their competition will be continued in the future.

One promising working direction is that it is beneficial to gain access to real-life forgery datasets that include ground-truth masks with a vast number of samples. Currently, depending on the forensic problem we want to study, existing datasets may have a limited number of examples or focus on a small range of devices or subjects. Although data scarcity has been tackled with the few-shot learning approach, the generalization problem may still be in game. In the case of Deepfakes detection, a very popular research topic as we see from the large number of recent works collected in this survey, high-quality datasets are becoming more and more available because of the involvement and commitment of big companies.

We also believe that although single works can obtain good performance, the combination of several domains or features will be of huge importance in the future. We have listed some works that combine the usage of image and audio features to detect Deepfakes, and probably these works would benefit from other features or strategies if properly combined. To this end, the availability of open-source implementation of existing methods is of paramount importance.

Another interesting future research topic is the development of counter-forensic methods which we believe have a right of existence. The creation of tools to deceive forensic detectors adds another interesting and important player in the game who challenges the detectors of fake multimedia contents. Such scenarios are particularly relevant for the face modification detection and Deepfakes detection because they can have very practical and security-related applications. The resilience of these detectors would be improved by considering the attacks of counter-forensic methods. The competition between the two sides would be an interesting topic to follow.

In all, we think that the image forensics research presents big challenges and opportunities for the future in which we hope to see more deep-learning-based methods to take better account of the particularities of the image forensics field.

3 Random High-Pass Initialization

> *The universe would be a mess if things were not random.*
>
> **— Servet Martinez**

As mentioned in the Introduction chapter, in this thesis manuscript we focus on the study of the weight initialization of the first layer of CNN. More specifically, we consider two important aspects of the initialization of convolutional kernels: 1) generating a set of *random high-pass filters* and 2) maintaining the *stability of data flow* at the input and output of kernels. The first point is related to the specificity of image forensic tasks, in our case the detection of basic image manipulation operations including median filtering, JPEG compression, noise addition, *etc.* As discussed in the previous chapter, in many cases forensic traces mainly reside in the high-frequency components of the image, so a high-pass filter is preferred to extract these discriminative traces. The second point is important for the efficient and effective training of CNN, so as to avoid the training difficulties caused by the exploding or vanishing of the data flow. In this chapter, we present our first attempt in this direction which was conducted at the beginning of this thesis work and which aimed at generating a group of random high-pass filters at the CNN's first layer by extending the well-know Xavier initialization [GB10b]. The results of this study were published and presented at the ACM IH&MMSec Workshop. As discussed later in this chapter, this first attempt has limitations and flaws mainly caused by an unrealistic assumption largely used in the literature, and improvements will be proposed in the subsequent chapters. Before presenting our first method, we provide an overview of some related works on weight initialization of CNN.

3.1 Weight Initialization of CNN

Deep architectures are needed if the objective is to learn complex functions that are able to predict high-level abstractions [LBL09]. However, every neural network

architecture, being small, deep or with different design, contains weights at various layers that need to be initialized. An easy approach would be to initialize all weights with zeros because we have limited knowledge on what the ideal weights should be at the beginning. Nevertheless, using this method would make every weight in the architecture calculate the same output, which will result in the same gradient and therefore the same updating for all of them. This means that the neural network would not achieve the so-called symmetry-breaking due to same initialization in all weight values [He+15].

As a first strategy to break symmetry on the gradients during the backward process, random values from a small interval were used. The intuition behind this was that by having heterogeneous values in the initialized weights, backpropagation algorithm would output mostly unique results and this would take full advantage of weights among different layers. In the case of a convolutional neural network where filters/kernels are used, we expect them to develop differently during the learning phase. By following this strategy, values of filters are often initialized with random samples drawn from a somewhat arbitrary uniform or normal distribution.

An issue that arises with this approach is that without any explicit amplitude control for the initialized weights, signal in neural network may become too small or too big and this will affect the network training. Indeed, from [BSF94] we know that when using deep networks, if the weights in a network start too small, then the signal shrinks as it passes through each layer until it is too tiny to be useful. On the contrary, when the weights in a network start too large, then the signal grows as it passes through each layer until it becomes uselessly massive.

3.1.1 Common initializations of CNN

In order to avoid the exploding or vanishing issue mentioned above and to ensure that the weights in the network will output signal of stable amplitude (as measured by its variance), Glorot and Bengio [GB10b] proposed to relate the amplitude of the initialized weights to the number of layer's inputs and outputs. Their method is usually called Xavier initialization (Xavier being the given name of the first author of [GB10b]) and considered as a milestone work in the literature of deep neural networks. One basic result of this initialization is that the more number of inputs a filter has, the smaller the initial values of the filter should be. The derivation of Xavier initialization is simple yet elegant and will be briefly presented in the next subsection. It is important to note that for the derivation it is assumed that both the filter inputs and the filter weights follow respectively a zero-mean independent

and identical distribution (iid) and that the input signal and the filter weights are mutually independent [GBC16; FJY18].

Another important study on weight initialization was presented by He *et al.* [He+15]. A new initialization method considering the usage of RELU activation function was proposed. Here we note that the amplitude of weights derived in [He+15] with RELU activation is bigger than that of Xavier initialization which implicitly considers linear activation function. This is because of the non-zero mean of RELU function when compared with linear activation. The weight initialization proposed by He *et al.* is commonly known as He initialization or Kaiming initialization (Kaiming being the given name of the first author of [He+15]). Both Xavier and Kaiming initializations are used in many deep learning papers and almost all machine learning implementation frameworks. Another important contribution of [He+15] was a thorough analysis concluding that if the initialization method produced weights of appropriate amplitude for the forward process, normally it would also be the case for the backward process, and *vice versa*.

3.1.2 Common initializations of CNN for image forensics

As previously described in section 2.2 some popular initialization methods in the image forensics field are the ones that use high-pass filters to remove the visual content and enhance traces left by different manipulations. Bayar and Stamm's constrained method [BS18a] and the SRM filters are two of the most commonly used initialization techniques with this approach.

The Bayar's method proposed a customized first layer which forced the CNN to learn a set of Laplacian-like filters (please refer to section 2.2.2 for some technical details). The constraint is enforced through simple filter normalization after each iteration of training and can give higher detection accuracy than CNNs without this constraint. Meanwhile SRM filters are a common initialization method for the first layer of a CNN used in several image forensic tasks, *e.g.*, the detection of manipulation operations [CLL17a], of splicing and copy-move forgeries [LGC18], and of inpainted images [LH19]. SRM filters, a group of handcrafted high-pass filters originally designed for steganalysis [FK12], are put at CNN's first layer as initialization and this often leads to very good forensic performances. Other methods such as the one used in [CKL15] proposed to set the CNN input as the so-called median filtering residual rather than as the raw image patch. Here, the residual means the difference between an image patch and its median filtered version, and this can be understood as a customized first layer of fixed non-linear high-pass filtering in the CNN.

In this manuscript, we describe three approaches that are the results of the evolution of our work on the design and test of CNN initialization methods for image forensics. The first approach defines the basic framework for random high-pass initialization of first-layer filters (this chapter), followed by a correction of an unrealistic assumption for the design of a data-dependent scaling approach for filters initialized by various algorithms (chapter 4), and finally a revisited and improved random high-pass initialization with correct assumption (chapter 5).

3.2 Our Random High-Pass Initialization

In the remaining part of this chapter, we present our first method of random high-pass initialization. This was conducted at the beginning of this thesis work and was motivated by the observations about the previous initialization methods used in image forensics field: methods like [CKL15] with a customized and fixed first layer are restrictive and in consequence may be sub-optimal for the considered forensic problem; Bayar's constrained method [BS18a] is rather computationally costly (constraint enforcing at each iteration) and might also be sub-optimal; initialization with SRM filters [FK12] may be a good choice but it does not consider the stability of the signal in and out of filters; at last popular methods from the computer vision community, *e.g.*, the well-known Xavier initialization [GB10b], in general do not cope well with forensic tasks because they do not generate high-pass filters at first layer, and this in practice usually results in lower performance compared to other methods mentioned above. We also notice that in many existing CNNs for forensics of image manipulations, the second to last layers are initialized with a conventional method, *e.g.*, the popular Xavier method [GB10b], while the first layer is a customized one. We believe that a simple technical solution can be developed so as to mitigate the decoupling of the initialization and design of the first and the remaining layers. Given the above observations, we propose a new initialization method for CNNs, which is actually an extension of the Xavier method in cases where we want to generate random high-pass filters. We present the details of our first approach in the next subsection.

3.2.1 The proposed method

As mentioned above, some previous methods used simple fixed high-pass filtering or constrained filters in the first layer of CNN [CKL15; BS18a]. Instead, we decided to derive a less restrictive and less ad-hoc method, *i.e.*, setting up a good initialization

followed by a free training. This was inspired by the milestone work of Glorot and Bengio [GB10b], 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, and this helps the signal to reach deep into the network. Indeed, as mentioned above, 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 [GB10b].

In a CNN, each filter/kernel performs locally a weighted sum of the input data "seen" by the kernel, using the kernel values as weights [GBC16; FJY18]. 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. (3.1), where the second equality holds under mild conditions that the N random variables $w_{i,i=1,2,...,N}$ (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 [GBC16; FJY18].

$$\begin{aligned} \mathrm{Var}(y) &= \mathrm{Var}(w_1x_1 + w_2x_2 + ... + w_Nx_N) \\ &= N\mathrm{Var}(w_i)\mathrm{Var}(x_i). \end{aligned} \tag{3.1}$$

The 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. (3.1) 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 manuscript we aim 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 (especially those coming from the computer vision community) tend to learn features representative of an image's content, contrary to forensic traces which are rather content-free and lie towards the high-frequency part of the images [CKL15; BS18a]. 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

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

Fig. 3.1.: Shape and notations of the initialized high-pass filter (left) and the corresponding input (right), with a 3×3 filter as example (see the main text for details).

high-pass and not very sensitive to forensic traces. Therefore, our objective here is to design a new initialization method which would be able to keep the variance stability of filter's input and output, and at the same time capable of generating a set of randomly initialized high-pass 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 Figure 3.1 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.

Regarding the derivation of our proposed method, first of all, 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 Figure 3.1 left). For the filter input, we set the notations as shown in Figure 3.1 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 $\tilde{N}$. This is presented in the following equation:

$$\begin{aligned} \mathrm{Var}(y) &= \mathrm{Var}(w_1 x_1 + w_2 x_2 + ... + w_{\tilde{N}} x_{\mathbb{E}(w_i)=\frac{C}{\tilde{N}}} + C.x_{\tilde{N}+1}) \\ &= \mathrm{Var}(x_i)\left[C^2 + \frac{C^2}{\tilde{N}} + \tilde{N}.\mathrm{Var}(w_i)\right]. \end{aligned} \tag{3.2}$$

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 [Goo60] as detailed in the equation below:

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

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. (3.2) should be equal to 1. Then the variance of w_i can be obtained as:

$$\mathrm{Var}(w_i) = \frac{1 - C^2 - \frac{C^2}{\tilde{N}}}{\tilde{N}}. \tag{3.3}$$

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):

$$\mathrm{Var}(w_i) = \int_{-\frac{2C}{\tilde{N}}}^{0} \left(x + \frac{C}{\tilde{N}}\right)^2 \frac{\tilde{N}}{2C} dx = \frac{C^2}{3\tilde{N}^2}. \tag{3.4}$$

By equalizing Eqs. (3.3) and (3.4), we obtain a simple quadratic equation which is guaranteed to have two valid real roots as:

$$(4 + 3\tilde{N})C^2 - 3\tilde{N} = 0 \quad \Longrightarrow \quad C = \pm\sqrt{\frac{3\tilde{N}}{4 + 3\tilde{N}}}. \tag{3.5}$$

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 randomly initialized high-pass filters, and the remaining layers are initialized with the conventional Xavier initialization. As shown in the next section, the proposed initialization copes well with two different CNNs to solve two different forensic problems of detecting image processing operations, with competitive performance when compared to existing methods.

3.3 Experimental Results

In order to test the proposed initialization, we carry out experiments for the two forensic problems considered in [BS18a] and [CKL15]. We first test a multiclass

Tab. 3.1.: Considered manipulation operations and their parameters in the multiclass forensic problem. The parameter is randomly chosen for the last two operations. σ is the parameter of standard deviation. The considered parameter setting is more challenging than in [BS18a].

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

forensic problem for identifying a group of image processing operations [BS18a], and then conduct experiments for median filtering forensics with JPEG post-processing [CKL15]. For each problem, we use CNN architectures from the original papers [BS18a; CKL15] and apply our initialization at the first layer of these networks. In both experiments, for the CNN with our initialization, we use the Adam optimization algorithm [KB14] with exponential decay of learning rate for the network training, and we let the network learn for 25 epochs. In this chapter, the implementation and test of our proposed method were based on Tensorflow® v1.7.0 with two Nvidia® GPUs of GeForce GTX 1080 Ti.

3.3.1 Multiclass forensics

We first consider a challenging multiclass classification problem for forensics of a group of image processing operations. Following [BS18a], our objective is to correctly classify six kinds of image patches: the unprocessed pristine patches and the processed patches by respectively five different manipulation operations listed in Table 3.1 (same operations as in [BS18a] but with differences as explained below). In fact, our parameter setting of operations is more challenging than those considered in [BS18a]. For example, we consider JPEG compression with quality factor higher than or equal to 90, while [BS18a] mainly considers quality factor of 70 and 80; we use median filtering with kernel size of 3, while [BS18a] uses kernel size of 3 and 5 or even bigger. This in part explains the accuracy decrease of Bayar and Stamm's method in this manuscript when compared with the performance reported in the original paper (results will be presented in the remaining of this subsection). We have chosen this challenging setting to show performance of forensic methods when used to solve a difficult problem. This challenging setting can be encountered in a real scenario in which for example inside a splicing operation a resampling could be performed with a small scaling factor of 1.1. In this case, the high similarity of the processed patch to the original one makes the forensic problem difficult. This

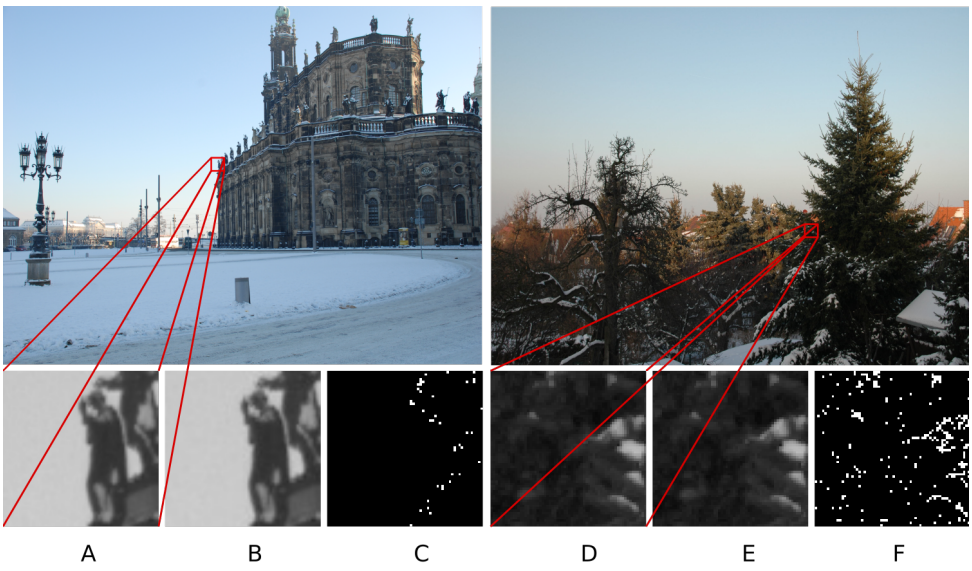

Fig. 3.2.: Examples of images from Dresden database [GB10a] and the generated patches used in our experiments. Bottom images on columns A an D show pristine patches while columns B and E show the JPEG compressed and median filtered patch respectively. Columns C and F show the thresholded absolute differences for the two pairs of pristine and manipulated patches. The threshold is set as 20.

is similar for weak noise addition with a parameter $\sigma = 1.1$, which could in some cases be applied to hide traces of other tampering operations.

The experiment was conducted on the Dresden image database [GB10a], which comprises 1491 unprocessed high-resolution images and which was also used in [BS18a]. 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. Here we conduct tests on patches of 64×64 pixels drawn from full-sized Dresden images. This small size of patches is another challenging factor, because the smaller a patch is, the less information it contains for carrying out correct classification. 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. Figure 3.2 shows on the first row two images of Dresden database. The bottom row shows both pristine and manipulated patches along with a binary representation of thresholded absolute difference between a pair of pristine and processed patches. We can notice that the difference between pristine and processed patches, both visually and computationally, is quite small, reflecting that what we consider is a challenging forensic problem.

Since we focus on the first layer initialization and its utility, for this first experiment we have borrowed the network design used by [BS18a] (see the original paper for

Tab. 3.2.: Detection accuracy, after 25 epochs, of the three methods regarding the design of network's first layer (in %, average of 15 runs), on the testing set of the challenging multiclass forensic problem.

Method	Accuracy
Fixed high-pass filters	77.38
Constrained filters [BS18a]	81.32
Our initialization	**87.56**

details of network architecture). The first layer of their network contains three constrained 5×5 Laplacian-like filters. In our network, we replace the first layer by three 5×5 filters initialized by our method presented in Section 3.2.1, 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 [BS18a], we have used the original Caffe implementation shared on-line by the authors[1]. For the training of network with our initialization, Adam [KB14] was used with a starting learning rate of $1e-3$. We used a schedule of exponential learning rate decay after each epoch with a parameter of 0.3, and we set a lower bound of $5e-7$ for the decayed rate.

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 as given in Equation 3.6, where M is the number of predictions (*i.e.*, the number of samples from all the six classes), $\tilde{y}_i$ is the ground-truth label of i-th sample, y_i is the predicted label, and $\mathbb{1}(.)$ is the indicator function.

$$\text{Accuracy} = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}(\tilde{y}_i = y_i). \tag{3.6}$$

Table 3.2 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 [BS18a], and the fixed high-pass filters. In order to enhance the significance of the results, we ran the experiments for 15 times for all three methods and report in Table 3.2 the average of 15 runs. We can see that our initialization provides satisfying forensic accuracy for this challenging multiclass forensic problem.

We would like to mention that due to some technical and implementation difficulties, in this subsection experiments of our method (implemented with TensorFlow) and

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

those of Bayar and Stamm's method (with the original Caffe implementation shared by the authors) were conducted on two different computers equipped with different deep learning development environments. We failed to install two environments on a same computer, probably because of our limited experience on deep learning development and the limited technical support of multiple environments at that time. Furthermore, we were not familiar with development using Caffe and were not able to implement a TensorFlow version of Bayar and Stamm's method that could reliably reproduce results of their original Caffe version. In addition, each method has its own properly tuned optimization algorithm and learning schedule. Therefore the comparisons were not completely fair and we do not claim that our method works better. The aim was to show the feasibility and utility of our method which extends the Xavier initialization and which is based on the idea of random high-pass initialization followed by free training without constraint. This point about experiments is improved in the next chapter to achieve fairer comparisons where we manage to create reliable PyTorch implementations for all considered methods.

3.3.2 Median filtering forensics with JPEG post-processing

To continue the test of proposed initialization, we now consider a second forensic problem and carry out experiments with a different database and 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 [CKL15] and replace the first layer of their fixed residual computation by our initialization and the original Xavier initialization [GB10b], respectively. Here we use our own implementation for the method and network of [CKL15] (to our knowledge the original authors' implementation is not available). Adam optimizer and the same optimizer parameters were used for our initialization, Xavier initialization and the method in [CKL15]. We conducted experiments on the UCID database [SS04] which is also used in [CKL15]. 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. Our careless mistake in

Tab. 3.3.: 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 [CKL15]	80.32	83.32	85.32
Xavier init. [GB10b]	71.56	75.32	84.23
Our init.	**83.55**	**85.77**	**88.63**

data preparation makes the forensic problem even more difficult, nevertheless in our opinion the problem still remains realistic and reasonable.

Table 3.3 presents the accuracy on the testing set at the end of the CNN training of 25 epochs for all the three scenarios. It can be noticed that our method has satisfying and competitive forensic performances. The difference of detection accuracy of Chen *et al.*'s method when compared to the original paper [CKL15] is probably due to difference and randomness in JPEG post-processing, experimental data, and network implementation and training.

3.4 Summary and Discussion

The preliminary results presented in this chapter show the feasibility and effectiveness of our initialization for the tested forensic problems. Our conjecture is that the proposed initialization provides a good initial status of the network and puts less restriction on the training of CNN. This would be useful for the CNN to explore interesting solution spaces. More precisely, the optimization problem involved in CNN training is in general highly complicated and non-convex, and the quality of the final result may depend heavily on the initial values of the CNN. Our initialization outperforms the conventional Xavier initialization, as shown by the results of median filtering forensics with JPEG post-processing, especially in the challenging settings of JPEG50+75 and JPEG70+75 (see Table 3.3). This may be due to that the former probably gives a better starting point in the optimization space, thus a more relevant subspace to explore. In fact, our initialization generates a high-pass filter which is useful to extract forensic traces, while the conventional Xavier initialization does not generate such a filter.

This is our first attempt in designing CNN initialization for image forensics, and there is much room for improvement as briefly discussed in the next paragraph.

Nevertheless, the simplicity of this first approach and the easy combination with popular initialization solutions, *i.e.*, the Xavier initialization, make the proposed method an eligible option for building and training CNNs for forensic tasks related to the detection of image processing operations.

For this first approach, we extend the well-known Xavier method to cases where we want to generate random high-pass filters for CNN kernels. Although satisfying results were obtained, this first method has limitations and flaws. We quickly realize later in our thesis work that the independent and identical distribution (iid) assumption of network input, *i.e.*, image pixels $\mathrm{X} = (x_1, x_2, ..., x_N)$ in our formulation, is unrealistic, though this assumption is largely used among the research community. Indeed, it has been demonstrated that neighboring pixels in natural images are *highly correlated* and not independent at all [SO01]. In addition, we are aware that the comparisons with existing methods in this chapter were not completely fair due to different deep learning development environments and the tuned network training of each method. Furthermore, we would like to thoroughly analyze the concrete behavior of popular initialization methods (including Xavier method and various high-pass filters) to understand the actual functioning of these methods and their impact on the properties of the output signal. These points are considered, handled and improved in the next chapter in which we propose a simple data-dependent scaling method for any given initialized first-layer filter to ensure the data flow stability in and out of the filter.

Data-Dependent Initialization

4

> *Flexibility is the key to stability.*
>
> **— John Wooden**

In this chapter, we use a corrected assumption of the statistics of natural images to propose a scaling method for popular initialization algorithms of CNN's first layer. To accomplish this, we focus on two subjects: 1) An analysis of the output variance of a convolutional filter and 2) a proposed scaling method to maintain a stable output amplitude. The first part is important to understand why common initialization algorithms may produce filters that shrink the input signal at their output, which would slow down the convergence speed of the learning process and reduce the final accuracy for the detection of image manipulation operations. Furthermore, in the second part we propose a scaling method based on the knowledge of statistical properties of input images. The proposed scaling method can improve the forensic performance of four different initialization algorithms when tested on different forensic scenarios and CNN architectures. Our proposed method was published and presented at the IWDW workshop. For the purpose of improving the quality of our work, from this chapter we test every method and architecture under the same develop environment to ensure a fair and reliable comparison. As discussed at the end of this chapter, a data-dependent proposal (*i.e.*, dependent on input signal properties) such as the one presented in this chapter may not be the preference for certain situations and this will be addressed in the next chapter. In the following, we start with an analysis of the output variance of a convolutional kernel in a CNN.

4.1 Variance of Output of Convolutional Filter

Our motivation continue in the line of having a stable data flow in CNN, as reflected by the stability of *variance* of the signal in and out a layer. Ideally, the variance of input and output of a layer should be *equal* to each other, which is beneficial for the training of CNN [GB10b; FJY18]. In this second approach, we first show that we can predict the variance of the output of a convolutional filter by using statistics of input signal and elements of the filter. Then, we present observations and understandings

regarding the output variance for the four initialization algorithms of convolutional filter which are mentioned in the last chapter. For the sake of brevity, the four algorithms are hereafter called Xavier [GB10b], SRM [FK12], Bayar [BS18a], and Castillo [CW19] (our first method presented in the previous chapter).

We observe potential limitations of the four initialization algorithms and have the intuition that the variance of output of a convolutional filter initialized by them may change substantially compared to the input. SRM [FK12] and Bayar [BS18a] do not take into account the output variance during the initialization, because the two algorithms put directly third-party SRM filters or normalized high-pass filters at first layer without modelling the input-output relation. Xavier [GB10b] and Castillo [CW19] consider the input-output relation and generate pseudo-random filters. These two algorithms are based on a statistical point of view and realized by drawing pseudo-random samples, so in practice properties of initialized filters may differ for different realizations. In addition, both algorithms assume that input is composed of random variables of independent and identical distribution (iid), but it is clear that pixels in natural images have *strong local correlations* [SO01]. In consequence, this unrealistic yet popular assumption used by Xavier [GB10b] and Castillo [CW19] may lead to unexpected or biased results for initialized filters. Therefore, it is interesting and important to experimentally and theoretically study the *actual output variance* of filters initialized by the above algorithms.

4.1.1 Formulation

As described in subsection 3.2.1, convolutional layers used in CNN contains a set of learnable filters (also called *kernels*) [FJY18]. During the forward pass, the kernel moves in a sliding-window manner across the input and computes a weighted sum of the local input data and the kernel. This procedure results in a so-called activation map comprising all the local results computed at every sliding movement of the kernel. Now assume that the kernel contains N scalars denoted by $\mathrm{W} = (w_1, w_2, ..., w_N)$, then the local input data involved in the computation also contains N scalars, denoted by $\mathrm{X} = (x_1, x_2, ..., x_N)$. It is easy to see that the local output y is simply the dot product of W and X, as:

$$y = \langle \mathrm{W}, \mathrm{X} \rangle = \sum_{i=1}^{N} w_i.x_i. \tag{4.1}$$

In Xavier [GB10b] and our first proposal described in the previous chapter, both w_i and x_i are assumed as independent random variables. As mentioned before, this

assumption may have limitations, *e.g.*, the (strong) correlations between different pixels x_i are not considered. In this second approach, we take a *new and more practical* point of view. Since we focus on a proper scaling of a given kernel, we assume that the kernel elements w_i are *known constants*, which can be generated by any initialization algorithm. In addition, we do not consider x_i as independent; instead, we consider them as *mutually correlated* random variables reflecting the natural image statistics [SO01]. With these assumptions and based on the property of variance of weighted sum of variables, we can compute the variance of the output y as presented in the following equation:

$$\begin{aligned} \mathrm{Var}(y) &= \mathrm{Var}\left(\sum_{i=1}^{N} w_i.x_i\right) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \mathrm{Cov}(x_i, x_j) \\ &= \sum_{i=1}^{N} w_i^2 \mathrm{Var}(x_i) + 2\sum_{1\le i<j\le N}\sum w_i w_j \mathrm{Cov}(x_i, x_j). \end{aligned} \tag{4.2}$$

The last expression just divides all the relevant terms into two groups: variance terms and covariance terms of the input signal components $(x_1, x_2, ..., x_N)$.
Furthermore, it is well-known that natural images have approximate *translation invariance* [SO01], implying that $\mathrm{Var}(x_i), i = 1, 2, ..., N$ are almost identical. In addition, the neighboring pixels are usually highly-correlated [SO01], which means that $\mathrm{Cov}(x_i, x_j)$ is close to $\mathrm{Var}(x_i)$. We approximate $\mathrm{Var}(x_i)$ by $\mathrm{Var}(x)$, the overall variance of input. Then we have the following approximation for Eq. (4.2):

$$\mathrm{Var}(y) \approx \mathrm{Var}(x)\left(\sum_{i=1}^{N} w_i^2 + 2\sum_{1\le i<j\le N}\sum w_i w_j Z_{ij}\right), \tag{4.3}$$

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

4.1.2 Convolutional filter initialized with high-pass filter

We first consider convolutional filter initialized as each of the 30 SRM filters[1] of shape 5×5 (so here $N = 25$). In order to test on real data for convolutional filter, we

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

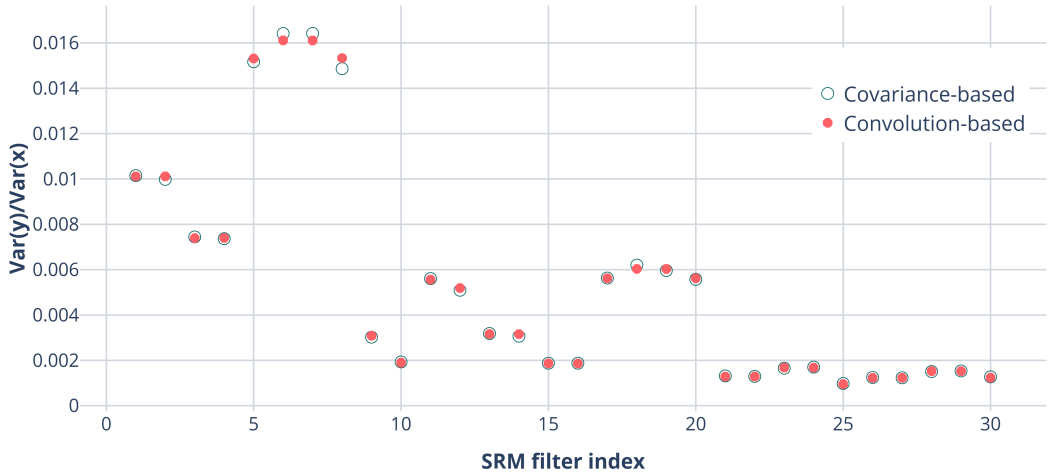

Fig. 4.1.: The value of $\mathrm{Var}(y)/\mathrm{Var}(x)$ for each of the 30 SRM filters obtained by using the covariance-based method of Eq. (4.3) and the convolution-based method.

take as input 64×64 grayscale image patches generated from the Dresden database [GB10a]. The image manipulation operations that we want to detect are listed in Table 3.1 (*i.e.*, the same manipulations of the multi-class forensics setting considered in the previous chapter). We then compute the variance of output of each SRM filter by two different methods: the first one with Eq. (4.3) and the second one with actual convolution between the input and the filter. Hereafter, we call the first as *covariance-based method* because Eq. (4.3) is mainly based on the covariance terms $\mathrm{Cov}(x_i, x_j)$ of the input signal components $(x_1, x_2, ..., x_N)$, and we call the second one as *convolution-based method*. For the first method, the covariance terms are estimated from 5×5 small patches (same size as SRM filters) which are randomly extracted from the aforementioned 64×64 Dresden image patches that have been converted to grayscale.

The results of $\mathrm{Var}(y)/\mathrm{Var}(x)$, *i.e.*, the ratio of output and input variance, are shown in Fig. 4.1. We can see that the amplitude of $\mathrm{Var}(y)/\mathrm{Var}(x)$ is very small for all 30 SRM filters, which lies basically in a range from 0 to 0.016 with a mean of about 0.005. The output of majority of SRM filters has a variance smaller than 1% of input variance, reflecting the signal shrinkage. It can also be observed that the two methods to obtain output variance give very close results of $\mathrm{Var}(y)/\mathrm{Var}(x)$, implying the coherence of the prediction by Eq. (4.3) with the practical convolution results.

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

footnote 1). It is then easy to deduce that $\sum_{j=1}^{N} w_j . \sum_{i=1}^{N} w_i = \sum_{i=1}^{N} \sum_{j=1}^{N} w_i . w_j = 0$. By dividing the $w_i . w_j$ terms into two groups, we obtain

$$\sum_{i=1}^{N} w_i^2 + 2 \sum_{1 \leq i} \sum_{< j \leq N} w_i w_j = 0. \tag{4.4}$$

The left-hand side of the above equation is almost same as the term in the bracket of Eq. (4.3), except that in the above Eq. (4.4) we replace Z_{ij} by 1. As mentioned earlier, for small natural image patches, we have the property that Z_{ij} are usually smaller than but very close to 1. This is verified by experiments on Dresden database where the minimum Z_{ij} value is 0.9573 for 5×5 small patches. It is not surprising that this minimum Z_{ij} value is attained between two pixel positions that are the farthest from each other within the 5×5 small patch. From the above analysis we can see that the term in the bracket of Eq. (4.3) is close to 0, which results in a small value of output variance $\mathrm{Var}(y)$ for 30 SRM filters. This intuitively explains the small $\mathrm{Var}(y)/\mathrm{Var}(x)$ values shown in Fig. 4.1.

Regarding the high-pass filter initialized by Bayar [BS18a] and Castillo [CW19], we simulated $10,000$ filters with both algorithms and calculated the variance of output of simulated filters. We also observe very small values of the ratio of output-input variance, with about 0.005 and 0.010 as the mean value of $\mathrm{Var}(y)/\mathrm{Var}(x)$, respectively for Bayar and Castillo. As we can see, our method proposed in the previous chapter results in a much smaller output variance than expected. This is mainly caused by the unrealistic iid assumption of input signal components $(x_1, x_2, ..., x_N)$ that we make in the last chapter. In fact, both Bayar and Castillo generate high-pass filters after initialization. For Bayar all filter elements sum up to zero after enforcement of filter normalization (*cf.*, Section 2.2.2 in particular Eq. (2.5)). For Castillo the sum of the two groups of filter elements, *i.e.*, a constant and a number of pseudo-random numbers, is approximately equal to zero because theoretically the mathematical expectation of this sum is zero (*cf.*, paragraphs below Fig. 3.1 in Section 3.2.1). Therefore, the above explanation with Eqs. (4.4) and (4.3) can still be used to intuitively understand the small output variance of Bayar [BS18a] and Castillo [CW19]. An improved version of our method Castillo, with corrected assumption about input signal, will be presented in the next chapter.

4.1.3 Convolutional filter initialized with Xavier initialization

We also carry out studies on the popular Xavier initialization [GB10b] which generates 5×5 filters filled with pseudo-random samples drawn from a zero-mean uniform

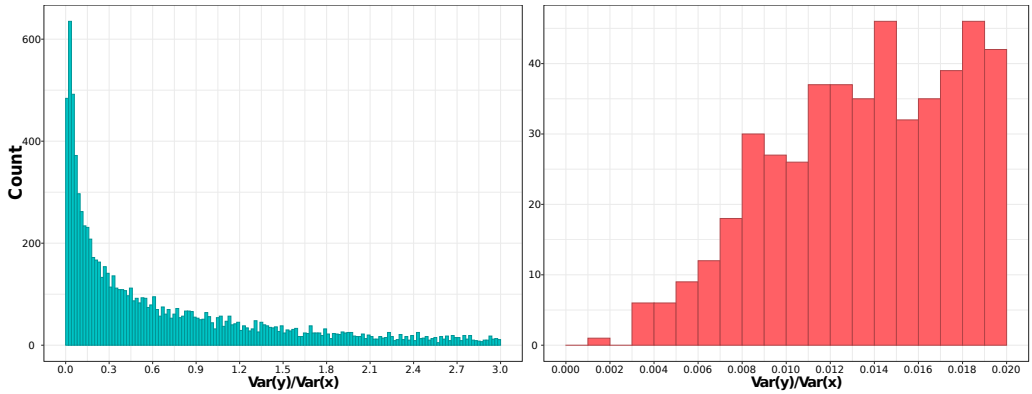

Fig. 4.2.: Two histograms of occurrences of output-input variance ratio $\mathrm{Var}(y)/\mathrm{Var}(x)$ for $10,000$ Xavier filters. Please refer to main text for detailed explanation.

distribution. We created $10,000$ Xavier filters using PyTorch and Figure 4.2 shows two histograms of occurrences of output-input variance ratio, *i.e.*, $\mathrm{Var}(y)/\mathrm{Var}(x)$: the left one is for the range of 0 to 3 with a bin width of 0.02, while the right one shows detailed occurrences for the first bin of the left histogram for the range of 0 to 0.02 with a bin width of 0.001. We computed the mean value of $\mathrm{Var}(y)/\mathrm{Var}(x)$ for the $10,000$ simulations of Xavier and found that the mean is close to 1 (desired value of Xavier); this is because of a long tail of big values that we do not completely show in Fig. 4.2. The left histogram of Fig. 4.2 does not have a peak around 1; instead, the highest occurrences occur near 0. This is a little surprising yet understandable according to Eq. (4.3). In fact, the elements of Xavier filter are drawn from a zero-mean distribution, so the bracket term in Eq. (4.3) tends to have a small value. However, due to numerical sampling and in particular considering the relatively low number of 25 pseudo-random samples (for 5×5 filter), it is possible for the bracket term to take big values in certain simulations. Experimentally this bracket term of Eq. (4.3) can be as big as 13 for some Xavier filters. In addition, from the right histogram of Fig. 4.2, for Xavier the occurrences of $\mathrm{Var}(y)/\mathrm{Var}(x)$ being very small values, *i.e.*, less than 0.01, is very low: 109 out of $10,000$ simulations (*i.e.*, around 1% probability). In contrast, the majority of this variance ratio is less than 0.01 for high-pass filter initializations as presented in last subsection.

Theoretical explanation for the histogram shape of Xavier simulations

We give further explanation for the shape of the histogram shown in Figure 4.2 left. This histogram of output-input variance ratio of the popular Xavier initialization is somehow 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 (which we do not completely show in the histogram) the mean value is around 1. In addition, we can seen that the ratio $\mathrm{Var}(y)/\mathrm{Var}(x)$ 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 we can predict the values of the mean, variance and skewness of the histogram, which are important statistical properties reflecting the shape of the histogram.

In our simulations, we generated $10,000$ Xavier filters of shape $[1, 1, 5, 5]$, meaning that we have one input, one output and a kernel of height and width of five. These filters were created using the PyTorch function `torch.nn.init.xavier_uniform_`. The interval 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 with N the total number of elements in a filter (here we have $N = 5 \times 5 = 25$). With these characteristics we can study the theoretical properties of the ratio of output-input variance and compare with the results of experimental simulations.

For this scenario of Xavier filter simulations, we consider the statistical properties of $\mathrm{X} = (x_1, x_2, ..., x_N)$ as fixed while $\mathrm{W} = (w_1, w_2, ..., w_N)$ 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 $\mathrm{Var}(y)$ is given by Equation 4.2 and approximated by Equation 4.3. The computation of $\mathrm{Var}(y)$ can be further approximated by replacing $\mathrm{Cov}(x_i, x_j)$ by $\mathrm{Var}(x)$, or equivalently by replacing the Z_{ij} terms by 1. This gives the following Equation 4.5 where we can notice that $\mathrm{Var}(y)$ is a random variable dependent on $w_{i,i=1,2,...,N}$ while $\mathrm{Var}(x)$ is the fixed overall variance of the input.

$$\begin{aligned} \mathrm{Var}(y) &= \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \mathrm{Cov}(x_i, x_j) \\ &\approx \mathrm{Var}(x) \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \\ &= \mathrm{Var}(x)(w_1 w_1 + \cdots + w_1 w_N + w_2 w_1 + \cdots + w_N w_N). \end{aligned} \tag{4.5}$$

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

1) Mean of $\mathrm{Var}(y)/\mathrm{Var}(x)$

We begin by calculating the mean of $\mathrm{Var}(y)$ as detailed in Equation 4.6 below. The last equation is obtained with help of the property of Xavier initialization [GB10b]

(also briefly presented in subsection 3.2.1) for which we have $\mathrm{E}(w_i^2) = \mathrm{Var}(w_i) = \frac{1}{N}$ (recall that w_i has zero mean). Additionally, many of the N^2 terms of $\mathrm{E}(w_i w_j)$ in the summation are equal to zero when $i \neq j$ (beacause $w_1, w_2, ..., w_N$ follow zero-mean iid). In the end, only the N terms of $\mathrm{E}(w_i w_i)$ contribute to the sum.

$$\begin{aligned}\mathrm{E}(\mathrm{Var}(y)) &\approx \mathrm{Var}(x)\mathrm{E}(w_1 w_1 + \cdots + w_1 w_N + w_2 w_1 + \cdots + w_N w_N) \\ &= \mathrm{Var}(x) \sum_{i=1}^{N}\sum_{j=1}^{N} \mathrm{E}(w_i w_j) = \mathrm{Var}(x) \sum_{i=1}^{N} \mathrm{E}(w_i w_i) \\ &= \mathrm{Var}(x) N \mathrm{E}(w_i^2) = \mathrm{Var}(x).\end{aligned} \tag{4.6}$$

The derived theoretical mean of output-input variance ratio, *i.e.*, $\mathrm{E}(\mathrm{Var}(y)/\mathrm{Var}(x))$, is actually the desired value 1 of Xavier initialization. However as we showed previously, the histogram of $\mathrm{Var}(y)/\mathrm{Var}(x)$ 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.

2) Variance of $\mathrm{Var}(y)/\mathrm{Var}(x)$

Ideally the variance of $\mathrm{Var}(y)$ should be very small, reflecting the stability of output variance in a large number of simulations. Nevertheless, we can see from the simulations that $\mathrm{Var}(\mathrm{Var}(y))$ would not be very small because the variance ratio can be a relatively small or a relatively big value. In order to calculate $\mathrm{Var}(\mathrm{Var}(y))$, we use the standard formula of the variance of the sum of (potentially) correlated random variables as shown in the third row of Equation 4.7 below.

$$\begin{aligned}\mathrm{Var}(\mathrm{Var}(y)) &\approx \mathrm{Var}\left(\mathrm{Var}(x) \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j\right) \\ &= \mathrm{Var}^2(x)\mathrm{Var}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j\right) \\ &= \mathrm{Var}^2(x)\left[\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} \mathrm{Cov}(w_i w_j, w_k w_l)\right] \\ &= \mathrm{Var}^2(x)\left[\frac{4}{5N} + \frac{2(N-1)}{N}\right].\end{aligned} \tag{4.7}$$

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 Equation 4.7, 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 indices 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 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 Equation 4.7 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 approximation made in the derivation, *i.e.*, we approximate all the $\mathrm{Cov}(x_i, x_j)$ terms by $\mathrm{Var}(x)$ in Equation 4.5. We can also observe that the variance of the output variance is not small, being about 1.952 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 to understand the shape of the histogram in Figure 4.2 left with 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 in Equation 4.8.

$$\mathrm{Skewness}(\mathrm{Var}(y)) = \frac{\mathrm{E}[\mathrm{Var}^3(y)] - 3\mathrm{E}[\mathrm{Var}(y)]\mathrm{Var}(\mathrm{Var}(y)) - \mathrm{E}^3[\mathrm{Var}(y)]}{(\mathrm{Var}(\mathrm{Var}(y)))^{\frac{3}{2}}}. \tag{4.8}$$

We have the formula of $\mathrm{E}(\mathrm{Var}(y))$ and $\mathrm{Var}(\mathrm{Var}(y))$ respectively in Equation 4.6 and Equation 4.7. Therefore, in order to calculate the skewness we now only need to compute the term $\mathrm{E}[\mathrm{Var}^3(y)]$. By using Equation 4.5 we can write this term as:

$$\begin{aligned}\mathrm{E}[\mathrm{Var}^3(y)] &\approx \mathrm{Var}^3(x)\mathrm{E}\left[\left(\sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j\right)^3\right] \\ &= \mathrm{Var}^3(x)\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{m=1}^{N}\sum_{n=1}^{N}\mathrm{E}\left[w_i w_j w_k w_l w_m w_n\right].\end{aligned} \tag{4.9}$$

In this last equation, there are N^6 terms in the summation, but similar to the derivation of the variance of $\mathrm{Var}(y)$ above, only several kinds of specific terms are not zero and contribute to the sum. These specific terms are $\mathrm{E}(w_i^6)$, $\mathrm{E}(w_i^2 w_j^4)$ and $\mathrm{E}(w_i^2 w_j^2 w_k^2)$. In fact, still using the hints presented above 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 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 Equation 4.10 below.

$$\begin{aligned}\mathrm{E}[\mathrm{Var}^3(y)] &\approx \mathrm{Var}^3(x)\left[\frac{N.27}{7N^3} + \frac{15N(N-1).27}{15N^3} + \frac{15N(N-1)(N-2)}{N^3}\right] \\ &= \mathrm{Var}^3(x)\left[\frac{105N^2 - 126N + 48}{7N^2}\right].\end{aligned} \tag{4.10}$$

Here with $N = 25$, we have $\mathrm{E}[\mathrm{Var}^3(y)] \approx 14.29\mathrm{Var}^3(x)$. With Equation 4.10 for $\mathrm{E}[\mathrm{Var}^3(y)]$, Equation 4.6 for $\mathrm{E}(\mathrm{Var}(y))$ and Equation 4.7 for $\mathrm{Var}(\mathrm{Var}(y))$, we can calculate the skewness of $\mathrm{Var}(y)$ by using Equation 4.8. 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 $10,000$ experimental simulations we obtain an empirical skewness value of about 2.811 for the output-input variance ratio. These results confirm the effectiveness 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 $\mathrm{Var}(y)/\mathrm{Var}(x)$, we demonstrate that with the new and correct assumption of strong correlation between 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 somehow 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 well as what is expected for Xavier initialization.

4.2 Scaling of Convolutional Filter

From results and analysis in Section 4.1, we can see that the variance of output of convolutional filter initialized by popular algorithms can be significantly smaller than the variance of input. This is particularly true for high-pass filter: the ratio of output-input variance $\mathrm{Var}(y)/\mathrm{Var}(x)$ is usually smaller than 0.01. The output signal after convolution operation substantially shrinks. This can be detrimental to the training of CNN, and as shown later in Section 4.3 the CNN training sometimes fails in such situations.

Using a data-dependent approach (*i.e.*, dependent on input data), we propose a simple yet effective scaling of the first-layer convolutional filter. The idea is to keep the variance stable after scaling for the input and output of any given filter generated by popular initialization algorithms. Corresponding to the two methods to compute output variance in Section 4.1, we propose two different ways to calculate the scaling factor s, as presented below. After obtaining the scaling factor, the elements of the given filter $\mathrm{W} = (w_1, w_2, ..., w_N)$ are properly scaled as $\widetilde{\mathrm{W}} = s.\mathrm{W}$. We then initialize the first-layer filter with the scaled version $\widetilde{\mathrm{W}}$.

4.2.1 Covariance-based method

From Eq. (4.3), it can be seen that in order to make $\mathrm{Var}(y)$ and $\mathrm{Var}(x)$ approximately equal to each other, we need to compensate for the effect of the term in the bracket. So the scaling factor is computed as:

$$s = \sqrt{1 \Big/ \left(\sum_{i=1}^{N} w_i^2 + 2 \sum_{1 \leq i} \sum_{< j \leq N} w_i w_j Z_{ij} \right)}. \tag{4.11}$$

In practice, we take random small patches of the same shape of the convolutional filter to be scaled (*e.g.*, 5×5) from a small portion of the training data. We then estimate the variance and covariance terms on these small patches to obtain the values of $Z_{ij} = \mathrm{Cov}(x_i, x_j)/\mathrm{Var}(x)$. Afterwards the scaling factor s is computed by using Eq. (4.11), and at last we obtain the scaled version $\widetilde{\mathrm{W}}$ of any given filter W from the considered four initialization algorithms.

4.2.2 Convolution-based method

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

$$s = \sqrt{\mathrm{Var}(\widehat{x})/\mathrm{Var}(\widehat{y})}. \tag{4.12}$$

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

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

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

4.3 Experimental Results

Several experiments are performed in order to test and show the efficiency of our proposed scaling. These experiments consider the four filter initialization algorithms mentioned earlier, two CNN architectures (CNN of Bayar and Stamm [BS18a] and a smaller CNN without fully-connected layer designed by ourselves), and two forensic problems (a multi-class problem of detecting a group of manipulation operations and a binary problem of detecting JPEG compression of high quality factor). For the multi-class problem, we also consider a different number of filters used in the first layer of the CNN of Bayar and Stamm [BS18a].

In the last chapter, due to technical difficulties we conducted part of the experiments in an independent environment with a different framework. From this chapter, we take a better approach with a fairer comparison by using the same deep learning framework, hardware and network optimization details for all testes methods. Consequently, the comparison with existing methods such as Bayar [BS18a] was carried out with the same environment as our proposed method. We have verified that the performance of our implementation is on par with or even slightly better than

the authors' original version. In the following, the implementation and experiments were all conducted using PyTorch v1.4.0 with Nvidia® 1080 Ti GPU.

The experimental data was created from the Dresden database [GB10a]. Full-resolution Dresden images are split for training, validation and testing with ratio of 3:1:1 and converted to grayscale. Patches of 64×64 pixels were randomly extracted from full-resolution grayscale Dresden images. This relatively small size of image patches makes the forensic problems more challenging.

4.3.1 Multi-class problem with CNN of Bayar and Stamm [BS18a]

We first consider the multi-class problem described in subsection 3.3.1 of classifying six different kinds of image patches: the original patches and the five classes of manipulated patches as explained in Table 3.1. The parameters for the resampling and JPEG compression manipulations are taken randomly from the specified sets in Table 3.1. The total number of patches in training set is $100,000$ ($\approx 16,667$ patches per class), while the number of patches in testing set is $32,000$ ($\approx 5,333$ patches per class). The number of training and testing samples is same as in [BS18a]. It is worth mentioning that the manipulations and their parameters in our study are more challenging than those in [BS18a]. The patch size is also smaller than [BS18a]: our patches are of 64×64 pixels, while [BS18a] mainly considers 256×256 patches.

We still use the successful CNN architecture of Bayar and Stamm [BS18a] in this set of experiments and initialize the three filters in the CNN's first layer with four different algorithms: Bayar [BS18a], SRM [FK12], Castillo [CW19], and Xavier [GB10b]. We carry out 5 runs for each algorithm and the corresponding two scaled versions. For SRM, for each run we randomly select 3 filters from the pool of 30 SRM filters. We compare each original initialization algorithm with their scaled versions obtained with the covariance-based method and the convolution-based method presented in Section 4.2. For fair comparisons, we make sure that for each run the scaled versions share the same "base filters" of the original version before performing scaling. We follow exactly the same training procedure described in [BS18a], including optimization algorithm, learning rate schedule, stopping criterion, *etc*.

For Bayar algorithm [BS18a], we have tested two variants of the scaling of the first-layer filters. The first one ("Bayar A.") follows closely the idea of Bayar's original constrained training strategy: we carry out scaling of the normalized high-pass filter at the beginning of each forward pass (please refer to Section 2.2.2 for detail of the

Tab. 4.1.: Test accuracy for the multi-class forensic problem (in %, average of 5 runs). The experiments were performed with four initialization algorithms and their scaled versions for first-layer filters of the CNN of Bayar and Stamm [BS18a]. In parentheses is the improvement of scaled version compared to the corresponding original version.

Initialization	Original version	Convolution-based scaling	Covariance-based scaling
Bayar [BS18a] A.	94.19	96.04 (+1.85)	96.02 (+1.83)
Bayar [BS18a] B.		96.15 (+1.96)	96.22 (+2.03)
Castillo [CW19]	93.71	96.45 (+2.74)	96.42 (+2.71)
SRM [FK12]	94.39	96.54 (+2.15)	96.55 (+2.16)
Xavier [GB10b]	93.48	94.61 (+1.13)	94.71 (+1.23)

normalization procedure proposed in [BS18a]). The second variant ("Bayar B.") is computationally cheaper and less complex: the scaling of normalized high-pass filter is only performed in the initialization, and we no longer impose normalization constraint during training. Our intuition behind the second variant is that with a proper scaling of initialized filters even a free training without the constraint of [BS18a] may provide satisfying performance.

The detection performances in terms of test accuracy (*i.e.*, classification accuracy on testing set) for this multi-class problem are presented in Table 4.1. The reported results are average of 5 runs with randomness, *e.g.*, different first-layer "base filters". However, for each run, the "base filters" are the same for the original and scaled versions: original version direct uses these filters, while scaled versions apply proper scaling on the "base filters" and then use the scaled ones.

We can see from Table 4.1 that the test accuracy of all the initialization algorithms is consistently and noticeably improved after scaling. Improvement of at least 1.13% and as high as 2.74% is obtained. We also observe that the results of the two scaling methods are very close to each other. We checked the computed scaling factors and found that they are indeed almost identical for the two methods. Furthermore, for the two scaling variants of Bayar [BS18a], variant B gives slightly better results, which is also computationally cheaper as it only performs scaling in initialization without enforcing any constraint during CNN training. This implies that with a good initialization after proper scaling, it might not be necessary to impose training constraint. It is worth mentioning that the results of Bayar in Table 4.1 are in general lower than those reported in [BS18a] because we now consider a more challenging forensic problem with more difficult manipulations and on smaller patches.

We can observe that our first proposal Camacho [CW19] does not show the same performance as presented in the last chapter in section 3.3 when compared to Bayar

[BS18a]. Results in the first column of Table 4.1 show that the original unscaled Bayar has now better performance than unscaled Camacho. We believe that this happens mainly for two reasons. First, the number of training epochs is increased. Previously, we trained the network for 25 epochs stopping when a plateau was reached and the increase of performance was small. This time, the number of epochs is much higher than in last chapter (25 vs. 60). Second, we used in the previous chapter a tuned optimizer with a specific learning rate schedule for our first proposal. In this chapter we follow exactly the network training setting of Bayar's original paper [BS18a] for all initialization algorithms, including network optimizer, learning rate schedule and stopping criterion. Using more epochs and the original hyper-parameters of Bayar and Stamm [BS18a] might be favorable for their approach, showing that Bayar was rather designed to be used in a longer training setting. However, it is worthwhile mentioning that after carrying out our proposed scaling to better release the potential of various initialization algorithms, the scaled version of Camacho performs better than scaled Bayar, as shown in the last two columns of Table 4.1. Lastly, our approach Camacho, as well as Xavier and SRM, has lower computational complexity than Bayar, because there is no constraint enforcement during the learning procedure for Camacho, Xavier and SRM. In practice, for the multi-class forensic problem, one run of Bayar takes in average 60% more time than the other three algorithms to complete a full training of 60 epochs. This is due to the fact that on each forward step the Bayar algorithm extracts the values of the filters in the first layer, normalizes them and saves them. This situation is similar for the variant A of scaled Bayar which takes much more time for one run of training when compared to the variant B of scaled Bayar, mainly because of the filter normalization operation added to each forward step.

Results in Table 4.1 also show that the three kinds of high-pass filters (especially SRM) indeed outperform Xavier, before and after scaling. This demonstrates the difference between forensics and computer vision tasks. Nevertheless, the performance of Xavier is also improved after scaling because as analyzed in subsection 4.1.3 Xavier can also result in small variance of output. In fact, for Xavier the probability to have $\mathrm{Var}(y)$ smaller than half of $\mathrm{Var}(x)$ is about 52.20% in our $10,000$ simulations.

We also observe that the proposed scaling helps to have quicker increase of forensic performance during CNN training. We show in Fig. 4.3 curves of test accuracy of SRM and Xavier (average of 5 runs), before and after the covariance-based scaling. It can be observed that the convergence speed is considerably improved for SRM. For both algorithms, the curve of scaled version is always above that of original version during the whole 60 epochs. It is also interesting to notice that the scaled Xavier performs slightly better than the original SRM.

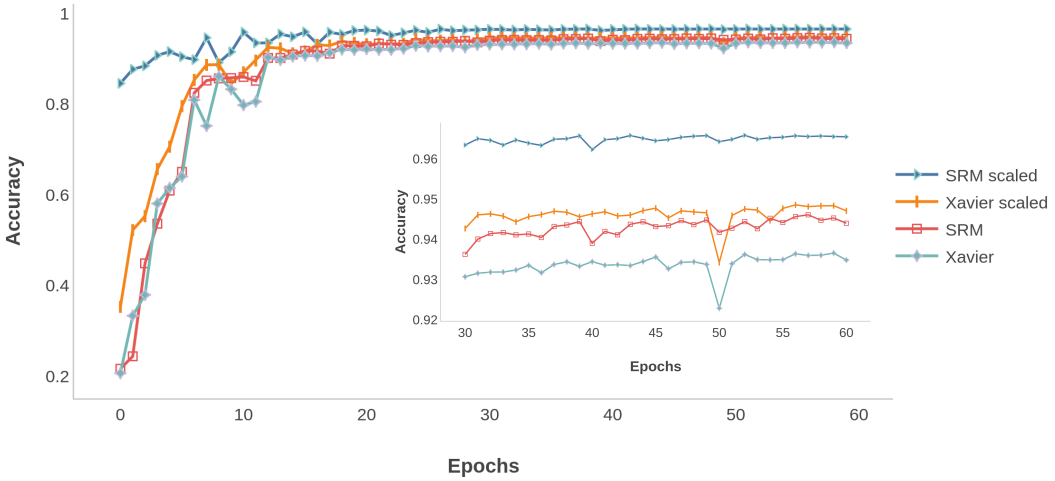

Fig. 4.3.: Curves of test accuracy (average of 5 runs) for the multi-class forensic problem, during the whole 60 training epochs of the CNN of [BS18a]. The curves are for SRM and Xavier, original version and scaled version by the covariance-based method.

As mentioned before in subsection 4.1.3, filters initialized with Xavier [GB10b] can produce output signal with a variance (much) smaller than input. We would like to test and understand the effect of our scaling method in such situations. Specifically, we compare three first-layer *customized Xavier filters* with an output variance smaller than 1/5 of the input variance, with and without our proposed scaling. This experiment is still performed for the same multi-class problem and with Bayar and Stamm's CNN. For customized Xavier filters, the averages of 5 runs for the test accuracy after 60 epochs are 93.76, 95.86 and 95.95, respectively for the unscaled version, the scaled version by convolution-based method and the scaled version by covariance-based method. We observe that with such customized Xavier filters the test accuracy is higher than conventional Xavier in the last row of Table 4.1 (accuracy values are 93.48, 94.61 and 94.71), and that the difference between unscaled and scaled customized Xavier is bigger with more than 2 percent improvement for test accuracy (93.76 vs. 95.86 or 95.95). Two possible explanations for these observations are: 1) customized Xavier filters with smaller output variance behave more like high-pass filters which are more effective in image manipulation detection; 2) our proposed scaling method is helpful for the network to make full use of a better initial status of first-layer filters after properly scaling them.

30 filters at first layer

Next, we present results for the same multi-class problem while changing the number of filters in the first layer of the CNN of [BS18a] to 30. We make this change for

Tab. 4.2.: Test accuracy for the multi-class forensic problem (in %, average of 5 runs). We still use the CNN of Bayar and Stamm [BS18a] but change the number of first-layer filters from 3 to 30.

Initialization	Original version	Convolution-based scaling	Covariance-based scaling
Bayar [BS18a] B.	94.91	96.11 (+1.20)	96.04 (+1.13)
Castillo [CW19]	94.11	96.31 (+2.20)	96.32 (+2.21)
SRM [FK12]	94.37	96.51 (+2.14)	96.49 (+2.12)
Xavier [GB10b]	91.80	96.03 (+4.23)	96.02 (+4.22)

two reasons: first, to test our approach with a different number of filters in the first layer; and second, to use all the 30 SRM filters which is a common practice in image forensics, *e.g.*, for detecting splicing and copy-move forgeries [LGC18]. For this scenario we still test the four initialization algorithms but only use variant B for scaled Bayar as it proved to obtain slightly better results while being computationally cheaper than variant A. The results are presented in Table 4.2. Again, we observe that scaling the filters with any of the two methods leads to consistently better test accuracy, with an improvement ranging from 1.13% to 4.23%. We notice from Tables 4.1 and 4.2 that after increasing the number of first-layer filters, 1) the original version of high-pass initialization (Bayar, Castillo and SRM) has slightly improved or comparable performance while the accuracy of Xavier decreases; and 2) the scaled version of Bayar, Castillo and SRM has comparable performance with the case of 3 filters while Xavier has noticeable improvement. We guess the reason for the good performance of scaled Xavier may be that with 30 filters there is more chance to have a very good filter which after scaling can improve the result. We plan to conduct further analysis in our future work to understand these observations.

4.3.2 JPEG binary problem with CNN of Bayar and Stamm [BS18a]

We notice in the multi-class problem that JPEG compression is the most difficult manipulation to detect. In this section we consider the binary classification between original patches and JPEG compressed patches with parameters in Table 3.1 (*i.e.*, very high quality factor between 90 and 100). This allows us to test the proposed scaling on a different challenging forensic problem. We use the CNN of Bayar and Stamm [BS18a]. The number of training and testing patches per class is the same as in last subsection. All CNN training settings are kept unchanged.

For this binary problem, we consider two initialization algorithms of Bayar [BS18a] and SRM [FK12], original and scaled versions (variant B for scaled Bayar). Table 4.3

Tab. 4.3.: Test accuracy for the binary JPEG forensic problem (in %, average of 5 runs). The experiments were performed with Bayar and SRM, original and scaled versions (variant B for scaled Bayar), on the CNN of [BS18a].

Initialization	Original version	Convolution-based scaling	Covariance-based scaling
Bayar [BS18a] B.	88.27	90.80 (+2.53)	90.80 (+2.53)
SRM [FK12]	78.24	92.33 (+14.09)	92.44 (+14.20)

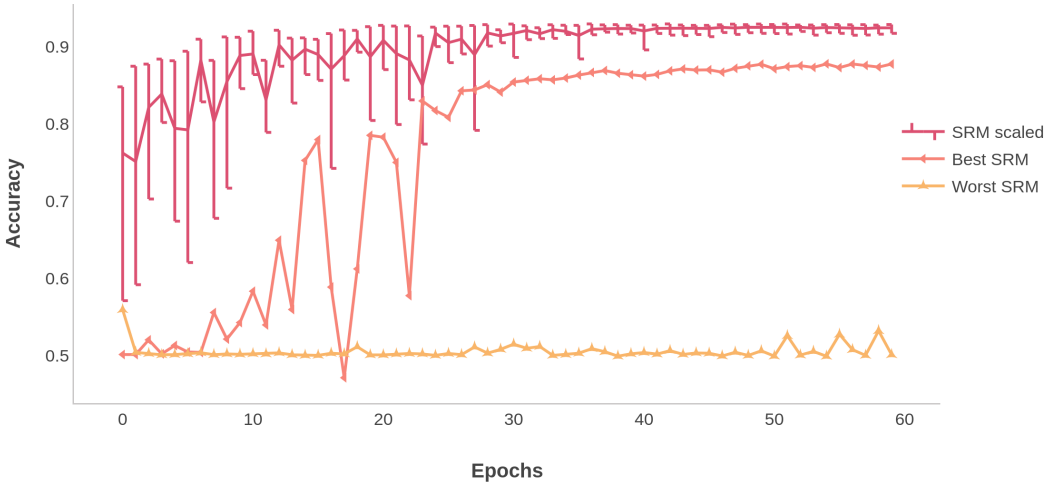

Fig. 4.4.: Curves of test accuracy for the JPEG binary forensic problem: scaled version of SRM (average of 5 runs) with bars of maximum and minimum accuracy at each epoch among 5 runs; and the best and worst runs of original version of SRM.

presents the obtained results (average of 5 runs). This challenging problem makes the original version of both Bayar and SRM struggle to achieve a good performance. Especially, training of SRM can occasionally fail, leading to accuracy close to random guess. Much better average test accuracy is achieved by scaled versions. For SRM [FK12], a boost of more than 14% is obtained with scaling.

We show in Fig. 4.4 some curves of test accuracy for scaled (covariance-based) and original SRM [FK12]. The curve of scaled version shows the maximum and minimum test accuracy together with the average at each epoch among the 5 runs. For the original version of SRM we can have very different results. Therefore, we show the best and the worst curves of test accuracy among all the 5 runs. As we can see the worst case does not improve during the whole procedure and the test accuracy remains close to 50%. The difference may come from the randomly selected three first-layer SRM filters in each run (certain SRM filters perform worse than others according to our observation, additional studies on this point will be presented in the next chapter). At last, we would like to mention that for each run, although we select randomly different SRM filters, the same selected filters are used to carry out

comparisons between the original and scaled versions. Therefore, even for filters that result in bad performance for the original version, we can obtain a better and satisfying performance after scaling them.

4.3.3 Multi-class and binary problems on a different smaller CNN

We then test both the multi-class and JPEG binary problems on a different CNN designed by ourselves. We first describe the architecture of this smaller network. Let Ck(M or A) denote a Convolutional-BatchNorm-Tanh(-MaxPool or -AveragePool) layer with k filters. For the first layer we use Hk which denotes a Convolutional layer with k filters. The architecture of our smaller CNN is H3-C40M-C25M-C20M-C15M-C6A. The first four layers have a kernel size of 5×5 while for the last two layers the kernel size is 1×1. All convolutional stride size is 1. The first layer and the last two layers do not have zero-padding, for the other layers the padding size is 2. This is a network without fully-connected layer. To compare with, the architecture proposed by Bayar and Stamm [BS18a] is H3-C96M-C64M-C64M-C128A-F200-F200-F6, where Fk denotes a fully-connected layer with k neurons and Tanh. The number of learnable parameters of the CNN of [BS18a] is about 337K, while our smaller CNN has about 41K parameters. Figure 4.5 shows a visual comparison of both architectures.

Using our smaller CNN, we test both the multi-class and the binary problems on a different CNN architecture. All the data preparation and experimental setting are the same as those described in Sections 4.3.1 and 4.3.2. For this set of experiments, we compare the original and scaled versions of Bayar [BS18a] and SRM [FK12] (variant B for scaled Bayar). Table 4.4 presents the obtained results. We can see that in all cases the scaled version leads to improved performance compared to the original version. The improvement of test accuracy goes from 0.93% for multi-class problem with Bayar, to 5.27% for binary problem with SRM.

We show in Fig. 4.6 curves of test accuracy of the small CNN architecture for SRM and Bayar (average of 5 runs), before and after the covariance-based scaling. In the case of SRM filters, the convergence speed is noticeably improved. For both initialization algorithms, the curve of scaled version is on average above that of original version during the whole training procedure.

Our objective in this subsection is to show that with a different CNN, our proposed scaling can still reliably improve the performance for different initialization algorithms and forensic problems. Meanwhile, it can be noticed that performance is

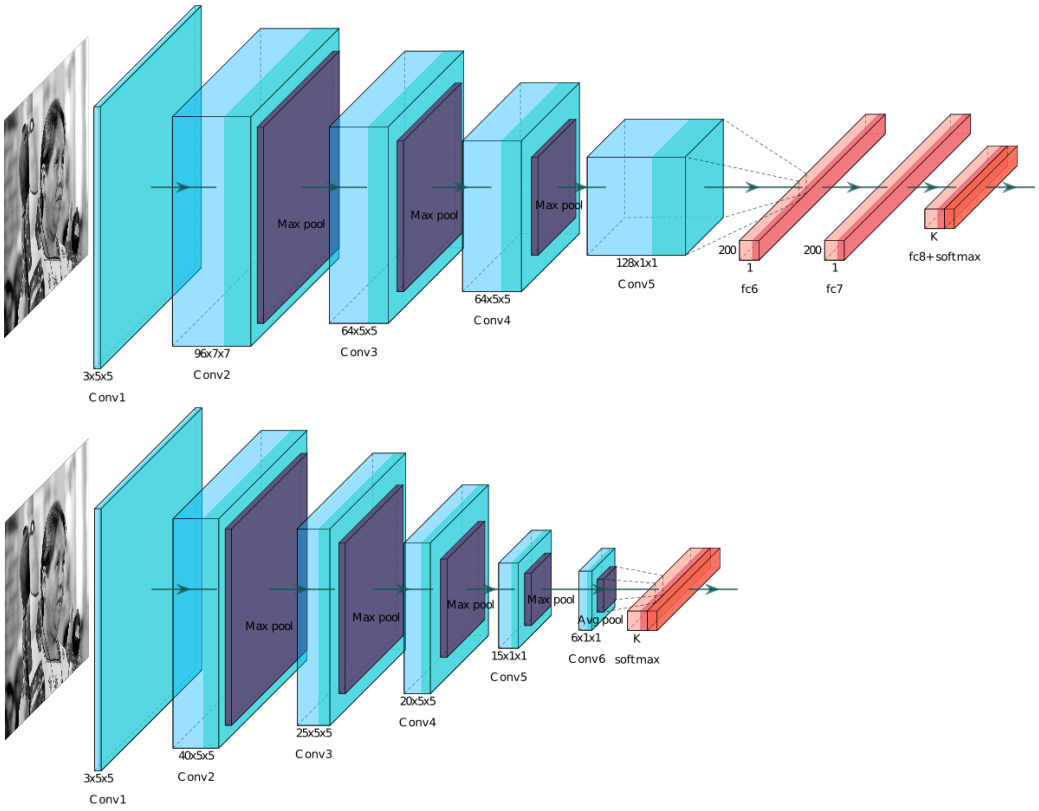

Fig. 4.5.: CNN architecture comparison. Figure on the top shows the CNN architecture of Bayar and Stamm [BS18a], while our smaller CNN is depicted below. Figure created with the tool PlotNeuralNet [@Iqb18].

Tab. 4.4.: Test accuracy for the multi-class and binary problems with our proposed smaller CNN without fully-connected layer (in %, average of 5 runs). The experiments were performed with Bayar and SRM, original version and convolution-based and covariance-based scaled versions.

Problem	Original version	Convolution-based scaling	Covariance-based scaling
Bayar [BS18a]			
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 [FK12]			
Multi-class	95.72	97.06(+1.34)	97.09(+1.37)
JPEG binary	89.85	95.11(+5.26)	95.12(+5.27)

better with our smaller CNN when compared to the network of [BS18a]. Our guess is that the forensic problems and/or the amount of data cope better with the smaller CNN's size (less parameters) and architecture (only comprising convolutional layers without fully-connected layer). Thorough analysis and understanding of the relationships between these factors is one part of our future work.

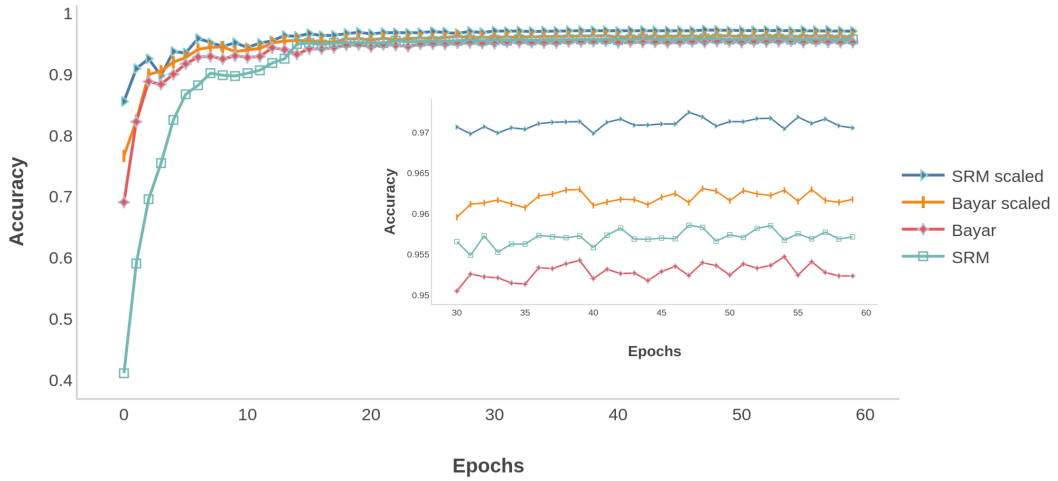

Fig. 4.6.: Curves of test accuracy using our smaller CNN architecture for the multi-class forensic problem (average of 5 runs). Results of original version and covariance-based scaled version are shown for Bayar and SRM.

4.4 Summary and Discussion

The experimental results presented in this chapter show the effectiveness of our data-dependent scaling (*i.e.*, scaling factor dependent on the input) for the tested forensic problems, initialization algorithms and CNN architectures. We believe that for the task of detecting image manipulation operations, the proposed scaling method provides a better initial status of the network with a stable amplitude of data flow in and out a convolutional filter. This results in a higher detection accuracy for the CNN. As originally explained by Glorot and Bengio [GB10b], having a constant variance for data flow during the CNN training procedure is vital for a good performance. Our approach follows this strategy and this may explain the improved results after scaling. It is also interesting to notice that even for filters not designed for image forensics (*e.g.*, Xavier [GB10b]), scaling them can lead to a final result comparable to unscaled customized filters (*e.g.*, SRM [FK12]) designed specifically for the considered forensic problems. Our proposed scaling is able to consistently improve all the initialization algorithms that we tested, as shown by the results in different forensic problems with different CNN architectures. Furthermore, the two scaling methods, *i.e.*, covariance-based and convolution-based, have comparable if not equal behavior, although as we mentioned before, obtaining the scaling factor with the covariance-based method is a computationally cheaper option. The simplicity of our method and the small computation time make our proposal a good candidate for the initialization of CNN models for image forensics

tasks. We believe that the proposed scaling method could also help in other image analysis tasks where customized filters are used for initializing CNN's first layer.

In this chapter, we also present theoretical and experimental studies which help to understand why the ratio of output-input variance for first-layer convolutional filter can be a (very) small value. In particular, it is very interesting and somehow surprising to see that experimentally even the popular Xavier initialization in general does not ensure a very good variance stability between input and output. We also provided theoretical derivation of the statistical properties of the histogram of output-input variance ratio for Xavier filters. We show that with the correct assumption on statistics of neighboring pixels in input, we can accurately predict the histogram's statistical properties and understand the shape of the histogram. To our knowledge, this is the first effort in the image forensics community to experimentally and theoretically study the *real behavior* of CNN initialization algorithm.

One possible limitation of our proposed data-dependent scaling method is that it needs to explicitly calculate the statistics of the input data or carry out convolution operation between input and convolutional filter to obtain the scaling factor. However in some practical applications, it is not possible to get access to a sufficient quantity of input data to reliably estimate the scaling factor for initialization. In order to propose an alternative solution and still in line with the basic idea of random high-pass initialization of our first proposal, in the next chapter 5 we will present a "data-independent" method which does not explicitly use the input data for the initialization and which can generate a group of random high-pass filters with rather stable signal variance at input and output. More precisely, we will revisit and improve our proposal in chapter 3 with a corrected and realistic assumption about the statistics of pixels in input images.

5 Revisiting the Random High-Pass Initialization

> *The beginning is the most important part of the work.*
>
> **— Plato**

In the previous chapter we described a data-dependent approach for the initialization of the first layer of a CNN in 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. 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 these reasons we believe that it is necessary and important to propose a corrected and improved random high-pass initialization approach which does not explicitly utilize the input data, *e.g.*, it does not need to calculate explicitly the variance and covariance statistics of input. For this final approach, we would like to revisit our first random high-pass initialization proposal with the corrected assumption of natural image statistics for the input X as presented in the remaining of this chapter.

5.1 The Proposed Method

Our objective is to propose an Improved Random High-Pass (IRHP) initialization method 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 5.1. The symbol C represents an unknown constant in the filter, $\mathrm{W} = (w_1, w_2, ..., w_N)$ are

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}

Fig. 5.1.: Shape and notations of the filter of our IRHP initialization method (left) and the corresponding input (right), with a 5×5 filter as example. C is an unknown constant and $w_{i,i=1,2,...,24}$ are scalar random variables following an appropriate distribution (see main text for details).

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. Despite the fact that this template 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 elements $w_{i,i=1,2,...,N}$. For the template shown in Figure 5.1 we have $N = 24$. As in our first proposal, 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 elements 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. In our first approach, both w_i and x_i were assumed as independent random variables; for the second proposal, we considered w_i as known constants (*i.e.*, elements of a given filter to be scaled) while x_i as correlated random variables. For this final proposal, $w_{i,i=1,2,...,N}$ are independent random variables, and $x_{i,i=1,2,...,N,N+1}$ are mutually correlated random variables reflecting 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, the output can be calculated as $y = w_1x_1 + w_2x_2 + ... + w_Nx_N + Cx_{N+1}$. Similar to the derivation in the last chapter 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:

$$\begin{aligned} \mathrm{Var}(y) = &\sum_{i=1}^{N} \mathrm{Var}(w_i x_i) + \mathrm{Var}(C x_{N+1}) \\ &+ 2 \sum_{1 \le i < j \le N} \mathrm{Cov}(w_i x_i, w_j x_j) + 2 \sum_{i=1}^{N} \mathrm{Cov}(w_i x_i, C x_{N+1}). \end{aligned} \tag{5.1}$$

In the above Equation 5.1, 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. Similar to the derivations in the previous chapters, for example by using the property of product of random variables and the expectation of w_i (*i.e.*, $\mathrm{E}(w_i) = -\frac{C}{N}$), we obtain the following Equation 5.2 for the variance of y:

$$\begin{aligned}\mathrm{Var}(y) = &\sum_{i=1}^{N}\mathrm{Var}(x_i)\left[\mathrm{Var}(w_i) + \frac{C^2}{N^2}\right] + C^2\mathrm{Var}(x_{N+1}) \\ &+ 2\sum_{1\le i}\sum_{<j\le N}\frac{C^2}{N^2}\mathrm{Cov}(x_i, x_j) - 2\sum_{i=1}^{N}\frac{C^2}{N}\mathrm{Cov}(x_i, x_{N+1}).\end{aligned} \tag{5.2}$$

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 [SO01] which was also utilized in the derivation in subsection 4.1.1), we can make some adjustments to Equation 5.2 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, 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 Equation 5.3:

$$\mathrm{Var}(y) \approx \mathrm{Var}(x)\left(\sum_{i=1}^{N}\left[\mathrm{Var}(w_i) + \frac{C^2}{N^2}\right] + C^2 + 2\sum_{1\le i}\sum_{<j\le N}\frac{C^2}{N^2}Z_{ij} - 2\sum_{i=1}^{N}\frac{C^2}{N}Z_{iN+1}\right). \tag{5.3}$$

Next, following the reasonable assumption of highly-correlated neighboring pixels in natural images [SO01], 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 Equation 4.4 in the last chapter). With this approximation applied to Equation 5.3, we obtain the following Equation 5.4:

$$\begin{aligned}\mathrm{Var}(y) &\approx \mathrm{Var}(x)\left(N\mathrm{Var}(w_i) + N\frac{C^2}{N^2} + C^2 + 2\frac{N(N-1)}{2}\frac{C^2}{N^2} - 2N\frac{C^2}{N}\right) \\ &= \mathrm{Var}(x)\left[N\mathrm{Var}(w_i)\right].\end{aligned} \tag{5.4}$$

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 Equation 5.4 that in order to make $\mathrm{Var}(x)$ and $\mathrm{Var}(y)$ close to each other, we should have the following property for filter elements w_i:

$$\mathrm{Var}(w_i) = \frac{1}{N}. \tag{5.5}$$

At first glance, the above Equation 5.5 happens to be the same as the one deduced for the Xavier initialization [GB10b], 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 elements W are random variables following zero-mean iid, while in our IRHP initialization method 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 Equation 5.5 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 Equation 5.6 to derive the value of C:

$$\mathrm{Var}(w_i) = \frac{C^2}{3N^2} = \frac{1}{N} \quad \Longrightarrow \quad C = \pm\sqrt{3N}. \tag{5.6}$$

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 5.1 left ($N = 24$), we have the value of C as $C = \pm\sqrt{72} \approx \pm 8.485$. We can notice that the absolute value of the derived values of C here is significantly larger than the one obtained by using Equation 3.5 of our first proposal in chapter 3 (still for a filter with a template of 5×5, by using Equation 3.5 we have $C = \pm\sqrt{\frac{72}{76}} \approx \pm 0.973$). This difference is due to the corrected assumption about the input X used in this chapter. The bigger absolute value of C (as well as the bigger absolute value of w_i because C and w_i are related) in our IRHP initialization method would mitigate the signal shrinkage problem at filter output and thus be beneficial for the network training.

Figure 5.2 left shows the histogram of the output-input variance ratio $\mathrm{Var}(y)/\mathrm{Var}(x)$ for $10,000$ simulated filters using our proposed IRHP initialization, while Figure 5.2 right shows a comparison between our IRHP method and the well-known Xavier method [GB10b]. As we can see, for our IRHP method the high occurrences no

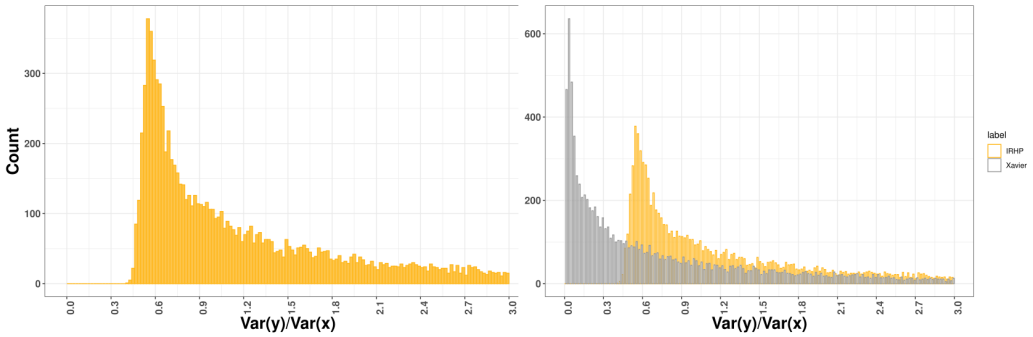

Fig. 5.2.: Histogram of occurrences of output-input variance ratio for our IRHP and the Xavier [GB10b] initialization methods. Figure on the left shows the histogram of occurrences of the variance ratio $\mathrm{Var}(y)/\mathrm{Var}(x)$ for $10,000$ simulations of our IRHP filters. On the right, we show the comparison between the histogram of our IRHP method and the histogram of $10,000$ Xavier filters.

longer happen in an interval of small values as what happens with Xavier method. Practically there is no occurrence for our method within the interval where Xavier method has the majority of occurrences. Indeed, the minimum value of output-input variance ratio for our IRHP method 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 method. By contrast, the Xavier method 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 IRHP initialization method, 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 various 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.

These preliminary results show the effectiveness of our new formulation for random high-pass initialization. In the next section we present experimental performances obtained for different image forensics problems after initializing the first layer of a CNN with the proposed IRHP initialization method.

5.2 Experimental Results

In this section we present several experiments performed in order to test and validate the proposed IRHP initialization in different scenarios.

Tab. 5.1.: Considered image manipulation operations and their parameter settings in the multi-class forensic problem. Parameters of the last two operations are chosen randomly from the given set.

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

For the purpose of carrying out fair comparisons, we begin by testing its utility in the same multi-class problem considered in section 4.3. Next, we conduct a comparative experiment with the very popular batch normalization technique under this same scenario. Subsequently, we illustrate the potential problem of randomly choosing a few handcrafted filters from a pool of such filters.

We also carry out tests of the proposed method on two other CNN architectures and one new forensic problem. First, we test our method on a smaller CNN architecture previously used in subsection 4.3.3 for both the multi-class and the JPEG binary forensic problems considered in the last chapter. Then, we show the performance of our IRHP initialization in a new forensic problem of detecting GAN-generated images using the well-known ResNet50 network.

All the experimental details regarding the deep learning library and GPU specifications remain the same as mentioned in section 4.3. Like in the last chapter, all methods under a comparison are implemented with same deep learning framework and trained with same optimizer and hyper-parameters.

5.2.1 Multi-class forensic problem

As previously tested in chapter 4, we present in this subsection the results for the multi-class forensic scenario where five image manipulation operations are applied to unprocessed pristine patches and the network is trained to differentiate among the six classes. The manipulation operations and parameter settings are the ones mentioned in subsection 3.3.1 and shown again in this chapter in Table 5.1 for readability reasons. Following the same settings, this experiment was developed using the dataset created previously from the Dresden database [GB10a]. There is no change in the percentage division for training, validation and testing sets with a 3:1:1 ratio. This results in around 100K patches for the training set and 32K patches for the validation and testing sets. The size of the patches is kept as 64×64 pixels.

Tab. 5.2.: Test accuracy (in %, average of 5 runs) for the multi-class forensic problem. We show results of different initialization methods on the first layer, as well as the scaled version of the first four methods.

Initialization	Original version	Convolution-based scaling
Bayar B. [BS18a]	94.19	96.15
Castillo [CW19]	93.71	96.45
SRM [FK12]	94.39	96.54
Xavier [GB10b]	93.48	94.61
Our IRHP init.	**96.35**	-

In this first scenario we use the network designed by Bayar and Stamm [BS18a]. The first layer on this network and for this scenario contains three filters of shape 5×5 which we initialize with our IRHP method presented in section 5.1. The rest of the network is kept unchanged in an attempt to isolate only the effect of the initialization used in the first layer. The optimization method together with the learning schedule remains untouched as the original proposal of [BS18a]. We compare the results of the IRHP initialization with the previous experiments in subsection 4.3.1: Bayar [BS18a], SRM [FK12], Xavier [GB10b] and our first proposal Castillo [CW19] presented in section 3.2. We only show the results of the convolution-based scaling as the results of the two scaling methods (convolution-based and covariance-based) are very close to each other. Scaled version "Bayar B." consists in scaling the normalized high-pass filter only in the initialization phase without any further constraint or scaling applied during the learning phase. We only present results of this version because it is computationally cheaper and obtains better results than the other version "Bayar A." as shown previously in subsection 4.3.1. Regarding our IRHP proposal, we do not try a scaled version given the fact that a random initialization without data-dependent scaling is our objective and filter values obtained from IRHP are in fact similar to the scaled ones as described later.

The test accuracy for each initialization method is presented in Table 5.2. We can observe that in comparison to the original unscaled version of all the other methods, our IRHP method has the highest test accuracy with an improvement of at least 1.96%. We can also see that the accuracy of IRHP is similar to but slightly lower than the results of scaled Castillo and scaled SRM. This is probably because the IRHP method does not ensure perfect equality of input and output variances, as illustrated in Figure 5.2. Nevertheless, the proposed IRHP initialization is computationally more efficient than initializations with scaled filters and avoids a pre-processing step of scaling factor computation with explicit use of input data. This follows the mainstream of "data-independent" random initialization approach for CNNs in the

Tab. 5.3.: Values obtained in an example 5×5 filter with our IRHP initialization. The center value C is 8.4853 and a uniform distribution of $U(-0.7071, 0)$ is used for drawing pseudo-random samples for non-center elements.

-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

research community. It is worth mentioning that "data-independent" is perhaps not a very rigorous term to describe our IRHP method because our method indeed considers the real statistics (with approximations) of the input data, however it does not explicit use the input data for initialization.

It is interesting to take a look at the values in filters generated by this new IRHP 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 high-pass filter with $+1$ and -1 as the only 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 elements have values close to 8.0. The center element in 5×5 filter of our IRHP method has a value of 8.4853, suggesting a good amplitude of filter element for our initialization. The details of the initial values in one simulation of our IRHP proposal are shown in Table 5.3.

5.2.2 Comparison with batch normalization

The stability of the data flow in the learning process of a CNN can be achieved in a complementary manner with 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. In the same sense, a reactive measure would be the usage of functions such as batch normalization [IS15] to re-center and re-scale our training batch during the learning stage as briefly described in section 2.1.1.

For this experiment, we show the comparison of batch normalization, the proposed scaling-based initialization of section 4.2 (for Xavier [GB10b] and SRM [FK12]) and our new IRHP initialization of section 5.1, in the multi-class forensic scenario. The network architecture and all other experimental settings remain unchanged as described in subsection 5.2.1.

Tab. 5.4.: Test accuracy (in %, average of 5 runs) for the multi-class forensic problem. The rows of "Xavier" and "SRM" present results for our scaling-based method and/or the batch normalization method, when combined together or applied separately for the first layer. The rows of "Our IRHP init." compare our IRHP method for two cases of with and without batch normalization at the first layer.

Initialization	Covariance-based scaling	Batch normalization	Test accuracy
Xavier [GB10b]	Yes	Yes	**95.04**
	No	Yes	94.36
	Yes	No	94.61
SRM [FK12]	Yes	Yes	96.53
	No	Yes	95.43
	Yes	No	**96.54**
Our IRHP init.	-	Yes	**96.51**
	-	No	96.35

Table 5.4 shows firstly three scenarios for Xavier and SRM initializations considering the different combinations of our scaling method and the batch normalization method on the first layer. Then, for our IRHP initialization we show the cases with and without batch normalization for the first layer. We can see that for Xavier and SRM, our scaling method alone works better than only using batch normalization, with an improvement of 0.25% and 1.11% for Xavier and SRM respectively. It is also interesting to notice that in the case of Xavier, jointly using both methods results in a higher accuracy. Our IRHP method also gets a small improvement when combined with batch normalization. For SRM filters, the performance is comparable when using both methods or using only our scaling proposal (*i.e.*, 96.53% vs. 96.54%).

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 IRHP method 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.

5.2.3 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 [FK12] filters. From the results in the previous

Tab. 5.5.: Comparison of test accuracy (in %) for 3 diagonal SRM filters, the corresponding scaled version and our IRHP method, in the JPEG binary classification scenario. 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 [FK12] diagonal filters	84.50	82.98
SRM [FK12] diagonal filters with convolution-based scaling	87.14	86.49
Our IRHP init.	**91.37**	**91.25**

chapter, we have observed that certain SRM filters perform less well than others for the image manipulation detection problem. This is the case of the JPEG binary problem results as shown in Table 4.3 and Figure 4.4. 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 IRHP initialization.

For this experiment we carry out tests with the same JPEG binary classification problem as described in subsection 4.3.2. The number of filters in the first layer is 3 and the CNN architecture is the one designed by Bayar and Stamm [BS18a]. We compare the performance of 3 diagonal SRM filters, the corresponding scaled version using the convolution-based method of section 4.2, and our IRHP proposal.

We show in the second column of Table 5.5 the average test accuracy of 5 runs for the three different methods 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 limited improvement of about 2.6% for average test accuracy when compared to the non-scaled ones. This is probably due to the limited forensic capability of these diagonal filters. By contrast, our IRHP method achieves higher accuracy. We also show in the last column of Table 5.5 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 IRHP method, which demonstrates the stability of its performance.

It is interesting to mention that in the worst run presented in Figure 4.4 in the last chapter 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 case in Figure 4.4). 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: first, the performance after scaling may be related to the overall forensic capability of all the 3 selected filters (non-diagonal SRM filters 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. 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 IRHP method.

5.2.4 Smaller CNN architecture

Testing our IRHP proposal on a different CNN architecture is important to show its efficiency. For this purpose, we perform experiments with our proposed smaller CNN architecture visually shown in Figure 4.5 in the previous chapter. Layer details are described in subsection 4.3.3. With this different architecture we test two problems as described previously, *i.e.*, the multi-class and the JPEG binary forensic problems. In this set of experiment, we compare our IRHP method with Bayar [BS18a] and SRM [FK12]. The number of patches for each dataset and all experimental settings remain unchanged as those of subsection 4.3.3.

Table 5.6 presents the obtained experimental results. For the sake of brevity, only the convolution-based scaling method is considered for scaled version. As we can see, the original unscaled SRM and Bayar have lower test accuracy than the proposed IRHP initialization. Regarding the scaled versions, our IRHP method performs better than scaled Bayar in both multi-class and JPEG binary problems. Scaled SRM achieves higher accuracy for both problems than our IRHP method. The latter has nevertheless its own advantages: different from the data-dependent scaling-based method, the IRHP 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.

Tab. 5.6.: Average test accuracy (in %, average of 5 runs) for the multi-class and JPEG binary problems with our proposed smaller CNN without fully-connected layer. The experiments were performed with Bayar and SRM (original version and convolution-based scaled version) and our IRHP initialization method.

Problem	Original version	Convolution-based scaling
Bayar [BS18a]		
Multi-class	95.24	96.17
JPEG binary	90.56	93.72
SRM [FK12]		
Multi-class	95.72	97.06
JPEG binary	89.85	95.11
Our IRHP init.		
Multi-class	**96.76**	-
JPEG binary	**94.45**	-

5.2.5 Detection of GAN-generated images

Finally we consider a new forensic problem with a popular deeper network. As we mentioned in section 2.2.4, GAN (Generative Adversarial Network) models have recently been on the news for their capability of creating synthetic and 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. We choose to carry out experiments with this 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 image synthesis tools. For this setting we use as reference the work recently published by Wang *et al.* [Wan+20], where the popular ResNet50 is used to analyze the artifacts left by GANs. We use the same dataset and code of [Wan+20][1] 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 [Kar+17]. Figure 5.3 shows examples of some image categories with the top row for natural examples and the bottom row for the GAN-generated ones. Each of the 20 categories comprises $18,003$ images of 256×256 pixels for

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

Fig. 5.3.: Examples for real and GAN-generated images of different categories in a binary classification scenario. Top row shows real images while synthetic ones are shown below. All images come directly from the shared database of [Wan+20] where the GAN-generated ones were created with ProGAN network [Kar+17].

each class being real or fake. 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.* [Wan+20], we use the popular ResNet50 as CNN architecture. In their original proposal, the network is pre-trained with ImageNet [Den+09]. 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 IRHP 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 [Wan+20] and ours) and all experimental settings are the same as those used in [Wan+20]. 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.* [Wan+20] and IRHP initialization for our method.

Experimentally we use Adam optimizer with a starting learning rate of $1e-4$ as proposed in [Wan+20]. 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.* [Wan+20] proved that by using these data augmentation techniques good *generalization* performance can be obtained on test 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

Tab. 5.7.: Characteristics of test sets for the evaluation of the generalization capability of detecting GAN-generated images. Source dataset is color coded as follows: L LSUN[Yu+15], G ImageNet [Den+09], S Style/Object transfer, A CelebA [Liu+15], F MS COCO [Lin+14], G GTA [Ric+16], R Raw Camera, T Standard SR Benchmark, M FaceForensics++ [Ros+19].

Model	Source dataset	Number of images
ProGAN [Kar+17]	L	8,000
StyleGAN [KLA19]	L	12,000
BigGAN [BDS18]	G	4,000
CycleGAN [Zhu+17]	S	2,600
StarGAN [Cho+18]	A	4,000
GauGAN [Par+19]	F	10,000
CRN [CK17]	G	12,800
IMLE [LZM19]	G	12,800
SITD [Che+18]	R	360
SAN [Dai+19]	T	440
Deepfake [Ros+19]	M	5,400

the detection performance of the model on "unseen" test data of Deepfake images generated by synthesis tools that remain unknown during the training phase.

The results obtained by Wang *et al.* [Wan+20] 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 IRHP method. There is no room for improvement in this part. 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 test sets coming from a number of state-of-the-art GANs for creating synthetic images with style transfer applied to a set of source images. The network architectures and training settings of these GANs are all different. The image sources and the resulting number of images for each test set are shown in Table 5.7. These test sets are shared on-line by Wang *et al.* [Wan+20].

To make a fair comparison, we follow the original idea of Wang *et al.* [Wan+20] to train the networks with the ProGAN dataset, and then test the generalization capability of trained networks on the different test sets listed in Table 5.7. This represents a real-world scenario where diverse and unknown synthesis 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 [Wan+20] data augmentation techniques can improve the generalization performance, and we apply these techniques in the experiments. Table 5.8 presents the generalization results with two different data augmentation options: "Case A." where only

Tab. 5.8.: Generalization results for the different test sets with comparisons between Wang *et al.*'s [Wan+20] ImageNet pre-trained weights and our IRHP 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 test set and each case in **bold**. For all scenarios networks were trained with the ProGAN dataset. Results are reported in terms of Average Precision (in %) as used in [Wan+20], and mAP is the mean Average Precision of all test sets.

	Case A.		Case B.	
Test set	**ImageNet weights init.**	**Our IRHP init.**	**ImageNet weights init.**	**Our IRHP init.**
ProGAN [Kar+17]	**100.0**	**100.0**	**100.0**	**100.0**
StyleGAN [KLA19]	**99.0**	98.8	98.5	**98.6**
BigGAN [BDS18]	82.5	**84.7**	88.2	**89.0**
CycleGAN [Zhu+17]	90.1	**93.5**	96.8	**97.0**
StarGAN [Cho+18]	**100.0**	**100.0**	95.4	**96.3**
GauGAN [Par+19]	74.7	**75.7**	**98.1**	**98.1**
CRN [CK17]	66.6	**73.6**	98.9	**99.4**
IMLE [LZM19]	66.7	**73.9**	**99.5**	**99.5**
SITD [Che+18]	**99.6**	**99.6**	92.7	**95.8**
SAN [Dai+19]	53.7	**53.9**	63.9	**66.1**
Deepfake [Ros+19]	**95.1**	93.4	66.3	**69.3**
mAP	84.4	**86.1**	90.8	**91.7**

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.* [Wan+20] where the first layer is initialized with pre-trained ImageNet weights and those obtained by our IRHP initialization.

As we can see from the results in Table 5.8, initializing the first layer with our IRHP method leads to a better performance for the majority of the test sets. In both "Case A." and "Case B." the mean Average Precision (mAP) is better for our method. 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.

5.3 Summary and Discussion

In this chapter we proposed a revised random high-pass initialization approach for the cases where statistics of input cannot be obtained beforehand. To our knowledge, this is the first random initialization method for image forensic tasks

which considers the realistic statistics of input images. Through the results of a variety of experimental scenarios, we showed the benefit and applicability of our IRHP method. Data stabilization as a way to obtain better forensic performance is achievable via the proposed initialization method. From the experimental results presented in this chapter we can see that our IRHP method provides a slightly lower but similar performance when compared to the previously proposed scaling method in chapter 4, while the IRHP initialization does not explicitly use the input data thus has a higher flexibility and a better practical applicability.

Our IRHP proposal outperforms existing conventional [GB10b] and image forensics [FK12; BS18a] initialization methods as shown by the results of multi-class and JPEG binary forensic scenarios. Different CNN architectures were considered and tested in order to prove the effectiveness of our proposal. Additionally, we showed that batch normalization by itself does not achieve the same level of results as our proposal. We also conducted some experiments to show that our IRHP method does not present the same risk as we may have when randomly selecting handcrafted filters from a pool of such filters. Finally, a new forensic problem of the detection of GAN-generated images was tested using the well-known ResNet50 architecture. Better generalization performances on test sets of fake images from unknown synthesis tools were obtained by our initialization method when compared to pre-trained ImageNet weights. Future efforts shall be devoted to theoretical studies to fully understand the experimental results/observations and to improve the performance of such random high-pass initialization.

Conclusions and Perspectives

6

> *It is good to have an end to journey toward; but it is the journey that matters, in the end.*
>
> **— Ursula K. Le Guin**

6.1 Summary of Contributions

Image forensics research is a necessary field to ensure the authenticity of images. This manuscript was dedicated to the study of initialization methods of convolutional neural networks used for detecting image manipulation operations.

Designing a CNN initialization method is a challenging assignment, especially when dealing with a specific problem. Almost all of the initialization methods existing nowadays for the computer vision field are based on the content of the image and not on the traces left by image manipulation operations. Such traces, usually left on the high-frequency parts, are not taken into account in common methods. In addition, better assumptions related to the statistics of natural images can be used to design new initialization methods.

In chapter 2 we provided a comprehensive review of the state of the art for deep-learning-based methods in the image forensics field. With this review, we could observe the use of different strategies for the input layer in a CNN. Some of them rely on pre-processing steps to create special features that will be fed to the networks, while others employ handcrafted steganalysis filters as initialization of convolutional filters in the first layer. As a result of this general review, we recognize the need for well-designed filter initialization methods for CNN-based image forensics.

In chapter 3 we introduced our first attempt to propose a new initialization method used in CNN for forensics of image processing operations. This proposal used as foundation the well-known Xavier initialization, for which comparable input and output variances for a convolution operation are the objective for a proposed filter. We developed the same idea for the image forensics field where a random high-pass filter is a better fit. As an early proposition we showed the simplicity of the

design and the feasibility in different forensic problems. We tested the proposed method on two problems with different CNN architectures demonstrating that an initialization properly designed for the given forensic problem is of great importance. As our first step there were several opportunity areas involving technical aspects and mathematical assumptions on the data which were addressed in later chapters.

Next, in chapter 4 we took a corrected assumption on the statistics of natural images where pixels show a strong local correlation. Additionally, we carried out studies on popular initializations including the Xavier method [GB10b] and the SRM filters which have been largely used in various image analysis tasks. We showed that these methods result in a signal shrinkage at filter output, which may deteriorate the convergence speed and the final performance of the network. With this new perspective, we developed a data-dependent method in which we calculated the statistics of the training data to obtain a scaling factor for the filter in use. After applying scaling of filters generated by the tested initialization methods, we can ensure a stable data flow for the first-layer filter of a CNN with comparable variances for input and output. The calculation time for this scaling factor was minor in relation to the average time commonly spent for training a CNN.

We performed a number of experiments in different scenarios and showed an improvement for all tested filters when scaled with our method. In addition, the performance of our scaling proposal were tested using different CNN architectures to prove its efficiency. It was interesting and a little surprising to observe that popular initialization methods, though originally designed to maintain a stable data flow, in practice result in a signal shrinkage at filter output. With the correct assumption of natural images statistics we were able to explain this interesting observation.

Finally, we proposed in chapter 5 a revised random high-pass initialization with the corrected assumption about the statistics of input images. This final proposal is aimed for the scenarios where we are not able to carry out reliable calculation on the training data. Furthermore, this kind of initialization appears to be the main trend in the machine learning field. With this improved random initialization we obtained better performance than existing conventional and image forensics initialization methods. Moreover, a deeper CNN architecture was tested for a new forensic problem of detecting GAN-generated fake images, and results showed that our revised random initialization can achieve better generalization performance.

We believe that these efforts will prove useful in the research of image manipulation detection. We are aware that further research and studies should be performed to fully and theoretically understand some experimental observations. Nevertheless, we hope that our work can bring the truth in images a little closer to everyone.

6.2 Perspectives

With every achievement, several new paths are opened to continue the progress. In this final section we mention some of the future perspectives that we believe are necessary or promising and that are derived from this thesis work. We have divided them in two groups. The first group presents the short- to medium-term lines that could be envisioned from the results we obtained. For the second group we include the possible options that could gain inspiration from our work but would take longer time or are with an objective more or less different from ours.

Shot-term and medium-term perspectives

Image forensics is a vast field that includes many different problems with their own characteristics. In this manuscript we mainly dedicated our efforts to one of them, the detection of image manipulation operations. Future works may investigate different problems such as the detection of copy-move and splicing, or other forgeries that may leave traces in the high-frequency part of images.

In addition, we see the need to test our proposals using more CNN architectures. Vanilla and residual architectures were tested in this manuscript but other models such as multi-branch, LSTM or even deeper architectures could be tested. Moreover, the combination with other optimizers and loss functions could be analyzed.

In some image forensics works, several initialization methods are combined for the first layer. Consequently, a mix of our data-dependent and "data-independent" approaches could be performed and tested. The combination of different initializations on different input branches could also be tested.

Finally, we would like to consider both forward and backward passes of CNN training to improve our proposed initializations. Additionally, our derivations do not explicitly consider the potential impact of an activation function after a convolutional filter. Therefore, a need of research for a stable output in this more general case for our proposals is envisioned. Furthermore, even if the trend has shifted to the use of RELU as the default option, improved derivations could be performed for several other activation functions that might be more appropriate for image forensic tasks, leaving the opportunity of having more options.

Long-term perspectives

Future works may consider all layers in the network. Our work is supported by the idea that the first layer has big importance because it is the one receiving directly the input data. But there is work to be done to understand the synergy for all layers.

In the GAN-generated image detection problem we tested the generalization capability. Although improved result could be obtained with our random high-pass initialization, the generalization performances on some of the test sets were still limited. Therefore, we see the need of further research to understand the performance of our initializations on "unknown" test sets.

Majority of works in the image forensics field use well-known architectures from the computer vision field and apply them directly. It would be an interesting point to investigate further on the selection of the deep network architecture, in order to make an educated choice specific for image forgery detection tasks. The recent work from [AD21] on this subject would be a good starting point.

Additionally, the usage of transfer learning or pre-trained weights could be further investigated for the image forensics field. Object recognition models are now widely used but the use of weights trained for the detection of a large number of manipulation operations such as the work of [WAN19] could be tested in combination with our initializations for the first layer.

Moreover, the shape and template of the high-pass filter in our derivation could be investigated in more depth. Different technical options for the shape, template and element values could be designed on a problem-dependent basis.

As we mentioned in the last chapter, we see the need to conduct further research in an attempt to design a random initialization method with a histogram of output-input variance ratio centered as closely as possible at 1. Studies on the theoretical feasibility and the practical implementation would be desirable.

Finally, based on our experience gained from the DEFALS challenge (please refer to Appendix A for some details), we see the need of new CNN-based forensic methods that can work on high-resolution images without the need of cropping or resizing. Furthermore, the experience with the datasets used for the competition showed that very high-quality and realistic forgeries are difficult to detect for almost all the state-of-the-art methods we tested. Consequently, we believe that it is necessary to investigate forensic solutions for this kind of highly professional forgeries.

A Challenge Competition for Image Forgery Detection

Besides the research work presented in this manuscript, we also dedicated around 6 months to the image forensics challenge DEtection de FaLSifications dans des images (DEFALS). In this appendix we briefly present the two stages entailed in the challenge and describe the tested methods as well as the obtained results.

At each stage, training and testing datasets were released separately in order to develop and evaluate different image forensic methods. Several state-of-the-art methods were tested on these datasets but unfortunately they did not show good performance. This led us in both stages to the use of a combination strategy of different methods in order to obtain a better forensic accuracy.

A rather satisfying performance was obtained for our team in the overall competition with better results for the binary detection task. In the following we start by describing the objectives and characteristics of the challenge.

A.1 Challenge Description

The Direction Générale de l'Armement (DGA), in partnership with the Agence Nationale de la Recherche (ANR), launched in 2017 the DEFALS challenge. The main objectives as described in the challenge's webpage[1] are two-fold: 1) to initiate and advance research in image analysis for integrity verification purposes and 2) to foster closer links between the academy and private companies on this subject.

Four teams were selected to participate in the challenge, which was organized in two stages with a common objective of designing and implementing a system that would allow the automated detection of manipulations and falsifications in images. In the challenge we use the same definitions of manipulation and falsification as presented in section 2.2 and categorized in Figure 2.5. Our team called "REVEAL" consisted of 6 members: Patrick Bas, CNRS Researcher at CRIStAL laboratory and leader

[1] http://defals.fr/

of our team; Kai Wang, CNRS Researcher at GIPSA-lab; François Cayre, Assistant Professor at Grenoble Institute of Technology; 2 PhD students at GIPSA-lab being Ivan Castillo Camacho and Ludovic Darmet; and Gaëtan Le Guelvouit, Research Engineer at B<>Com, the industrial partner in our team.

As mentioned above, for each stage the organizer provided within two different time intervals a training dataset and a testing dataset, respectively for the development and evaluation of forensic methods. The difference between the two datasets is that the training set is provided with ground-truth labels (for the binary detection task in both stages) and the ground-truth localization masks of falsifications (only for the localization task of the second stage). Each team had to develop their systems using the training set, and provide their results on the testing set for evaluation purposes. The falsifications in these datasets were performed by a hired professional team to create highly realistic and detailed forgeries in many cases with high-resolution images, *e.g.*, of 3000×4000 pixels.

Among the falsifications in the datasets, there were three main types being 1) splicing, 2) copy-move and 3) inpainting. Moreover, various manipulation operations were also present such as color enhancement, scaling, blurring, *etc.* In some cases several falsifications and/or manipulations were applied to the same image. Competition rules stated that images with manipulation operations only were considered as authentic. The images supplied in these datasets included photos of persons and pictures of a great variety of scenes coming from indoor and outdoor natural scenes, urban scenes and landscapes. The datasets were distributed with a large range of image sources such as smartphones, compact cameras and professional digital cameras. Unfortunately, due to copyright restrictions, we are not allowed to visually show DEFALS images in this manuscript. In the following we briefly present our strategies and methods adopted and utilized during the two stages of the challenge competition, starting with the first stage.

A.2 DEFALS 1st Stage

The first round of DEFALS took place between February and March 2019. The objective was to carry out a binary classification to detect whether tested images were falsified or not. For this matter we were provided with a testing set of about 700 images with different falsifications.

As mentioned before, we followed different approaches that we combined to obtain the final result. As a first strategy, we developed a CNN capable of classifying small

patches and subsequently giving a final probability of falsification for the full-sized image. The basic architecture was taken from the network designed by Chen *et al.* [CLL17b]. It consists of one first convolutional layer initialized with 4 SRM filters followed by another vanilla convolutional layer with 32 filters. After these two layers, the network has 6 groups of layers and each group comprises a convolutional layer, a TANH activation function and a pooling layer. The number of filters used on the first group is 64, with doubled number of filters in subsequent groups until the sixth and last group of 2048 filters. The final group uses a global average pooling layer instead of the max pooling as used in the first five groups. Finally, there is a flattening layer with Softmax to produce the final output. The original network was designed to be used with grayscale patches, and we slightly modified it to allow input of 3 channels. This allowed us to accept color patches as input so that the network could cope better with the DEFALS datasets which only included color images.

For this network, we created a dataset made up of patches of 256×256 pixels with the objective of training a two-class detection network to classify between: 1) patches coming from original images and manipulated ones and 2) patches coming from falsified and falsified + manipulated images. In this case, images from the second class were fewer than the first. To compensate this unbalance, we made sure to use all patches from the second class on each training epoch while randomly selecting equal number of patches from the first class.

Different approaches of extracting patches from full-sized images were tested, such as using boundary patches in an attempt to localize the falsified region, but the results were not satisfying for that kind of methods. In the end, the best performance was achieved by using patches *randomly extracted* from the two groups of images mentioned above. This was because the falsified images in the training and testing sets had undergone some specific post-processing operations which altered the global color information and which luckily helped the CNN to achieve good results.

Figure A.1 shows the probability of falsification in a descending order achieved by the CNN-based strategy alone. The testing set in fact included about 90 falsified images. As we can see from Figure A.1, a simple and intuitive thresholding of the output scores at a value of 0.5 would allow us to well estimate the number of falsified images and identify them.

Another strategy was the use of steganalysis features, specifically the color SRM features and another kind of differential features extracted from the YCrCb color space. Patches were still extracted randomly from the two classes of images. Like in the CNN-based strategy, a simple averaging of scores from patches was performed to obtain a final falsification probability for the full-sized image.

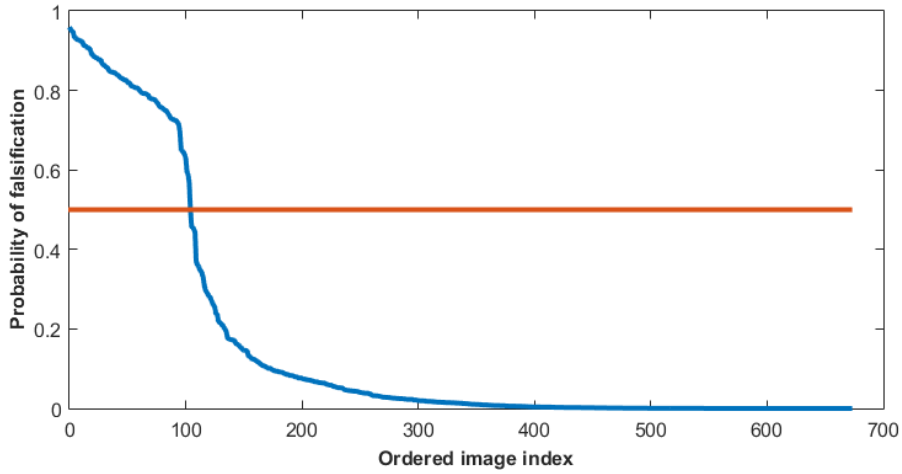

Fig. A.1.: Ordered CNN output scores (*i.e.*, probability of falsification, blue curve) with a thresholding at 0.5 (red horizontal line).

After obtaining falsification probability scores of an image having been falsified or not with both strategies, *i.e.*, CNN-based detector and steganalysis-feature-based detectors with two different kinds of features, we combined the results with the BORDA fusion technique which worked as consensus-based voting system resulting in a ranking of falsified images along with the final suspicious level. For the final results of this first round of challenge, we obtained the first place.

A.3 DEFALS 2nd Stage

The second round of DEFALS took place between February and April 2020. The objective of this stage was divided in two parts: 1) a binary classification of falsified images and 2) the localization of the falsified areas. For method development, we were provided with a training set comprising authentic images as well as high-quality falsified images and the corresponding ground-truth masks of falsifications. The evaluation was carried out on a separate testing set. The image falsifications for this stage mainly included copy-move, inpainting, and splicing forgeries.

In the released datasets for this second stage, there were no apparent traces of post-processing operations on falsified images. In addition, the falsified regions were in many cases of very small size, *e.g.*, a letter, a bird or a leaf in a big scene, which made the forensic analysis very difficult. Fortunately, we found information leak from the image meta-data, more precisely coming from the dimensions of falsified

images. This allowed us to obtain almost perfect classification (*i.e.*, close to 100% accuracy) for the binary detection task of falsified images.

As mentioned in section 2.2 several deep learning options exist for the detection and localization of image falsifications. We tested the original implementations of [Bap+19], [WAN19] and [Zho+18]. In all cases the maximum size of input for the networks was 512×512. Because of the high-resolution images in the DEFALS datasets, the memory usage was prohibitive for these networks if full-resolution images were used (*e.g.*, of 3000×4000 pixels). We tried initially to resize the images but this pre-processing resulted in a poor performance probably because resizing might partially remove falsification traces. Therefore, we created another dataset as in the first stage comprising patches of 256×256 pixels cropped from the full-sized images in order to train and fine-tune the different existing networks.

More precisely, we extracted boundary patches from the falsified images. These patches were taken with a range of 25% to 75% of the patch pixels within the falsified region according to the ground-truth masks. In the cases where no patch satisfied the above condition, which implied that the falsified region was very small, we took some patches that completely comprised the whole falsified region. After training and/or fine-tuning the networks with patches we noticed that for the splicing falsification the networks achieved rather good results at patch level. The reason might be that edges between spliced and original regions were usually sharper and relatively easy to detect. However, we encountered the problem of very high false positive when combining patch-level results to obtain the localization result on full-sized image. Unfortunately we were not able to fully understand and improve this point.

Regarding copy-move detectors, the best option that we tested was the one designed and implemented by Cozzolino *et al.* [CPV15]. Nevertheless, one limitation was that this detector did not support the source-target disambiguation, while this was necessary to obtain better results in the challenge because only the target region was considered as falsified in the localization mask.

At last, we found and used a simple method to combine the splicing and copy-move results. Images were first divided into two groups which were considered to have respectively the splicing and the copy-move falsification. The division was realized based on the binary detection results of falsified images and the copy-move identification results. Afterwards, the localization maps of the two groups of images were produced by the splicing detector and the copy-move detector, respectively. This simple method allowed us to carry out automated detection and localization, however the obtained localization performance was limited with better results for copy-move than splicing. The poor performance of splicing localization was mainly

caused by a high false positive rate when combining patch-level results to get the predicted mask of the full-sized image.

Following the strategies presented above, we obtained the first place again for the binary classification task. However, our score for the localization task was not as good as expected which gave us the third place. In general, the localization performance was rather limited for all participating teams. This may be due to the lack of easily detectable forensic traces in high-quality falsifications as those considered in the second round of the challenge.

A.4 Discussion

Being able to participate in a challenge like DEFALS was a unique experience. The learning curve during each stage was a fast-paced evolvement with lots of new knowledge. We also realize that there is a gap between academic research outcome and forensic methods that can be applied in practice. As mentioned above, almost all existing state-of-the-art methods have poor performance on DEFALS images. Meta-data information and post-processing operations were the main clues that we utilized to achieve a good performance in the binary classification task in the two stages of the challenge. However, it is very likely that such ad-hoc detectors in general fail in real-world forensic scenarios.

CNNs may be a qualified solution but it heavily depends on the data that we use to train the network. The appropriateness and quality of the training data is probably the key to the success of CNN-based forensic methods and more efforts should be devoted to the relevant research. We can also see that detection of high-quality falsifications in high-resolution images should be one of the mains future topics of the image forensics research. In fact many of the existing solutions were focused on synthetic and small-sized images, while the resolution of images, even for images acquired by mobile phones, becomes larger and larger and the visual quality of falsifications are continuously improved. In this sense, the DEFALS datasets were an excellent example that could hopefully be shared openly in the future.

Finally, the need for dedicated hardware such as video cards of big memory and high computational cost for this challenge was a restriction that we would like to reconsider in our future work. This would give the opportunity to open new lines of research that would focus on the usage of low-memory and low-cost solutions.

Author's Publications

B

1. International Journal Paper

    a) Castillo Camacho, I. and Wang, K., A comprehensive review of deep-learning-based methods for image forensics, *Journal of Imaging, Special Issue of Image and Video Forensics* (https://www.mdpi.com/journal/jimaging/special_issues/image_video_forensics), vol. 7, no. 4, pp. 69:1-69:39, 2021.

2. International Conferences Papers

    a) Castillo Camacho, I. and Wang, K., A simple and effective initialization of CNN for forensics of image processing operations, in *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security (ACM IH&MMSec)*, pp. 107-112, Paris, France, July 2019.

    b) Castillo Camacho, I. and Wang, K., Data-dependent scaling of CNN's first layer for improved image manipulation detection, in *Proceedings of the International Workshop on Digital-forensics and Watermarking (IWDW)*, pp. 208-223, Melbourne, Australia (on-line), November 2020.

3. National Conference Paper

    a) Castillo Camacho, I. and Wang, K., A simple and effective CNN initialization for forensic detection of image processing operations, in *Proceedings of the Colloque Gretsi - Communauté Francophone du Traitement du Signal et des Images*, 4 pages, Lille, France, August 2019. (same technical content as ACM IH&MMSec paper)

4. International Journal Paper Under Preparation

    a) Castillo Camacho, I. and Wang, K., Convolutional neural network initializations for image manipulation detection, 2021. (under preparation, extension of IWDW paper and including technical contents of Chapters 4 and 5)

Bibliography

[Afc+18] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen. “Mesonet: a compact facial video forgery detection network”. In: *Proceedings of the IEEE International Workshop on Information Forensics and Security*. 2018, pp. 1–7 (cit. on pp. 21, 22, 41, 42, 45).

[Aga+20] S. Agarwal, H. Farid, O. Fried, and M. Agrawala. “Detecting deep-fake videos from phoneme-viseme mismatches”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 1–9 (cit. on pp. 43, 45).

[AFG19] S. Agarwal, H. Farid, and Y. Gu. “Protecting world leaders against deep fakes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 38–45 (cit. on p. 13).

[AD21] A. A. Ahmed and S. M Darwish. “A meta-heuristic automatic CNN architecture design approach based on ensemble learning”. In: *IEEE Access* 9 (2021), pp. 16975–16987 (cit. on p. 102).

[AC20] I. Amerini and R. Caldelli. “Exploiting prediction error inconsistencies through LSTM-based classifiers to detect deepfake videos”. In: *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*. 2020, pp. 97–102 (cit. on pp. 43, 45).

[Ame+19] I. Amerini, L. Galteri, R. Caldelli, and A. Del Bimbo. “Deepfake video detection through optical flow based CNN”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 1–3 (cit. on pp. 43, 45).

[Ame+17] I. Amerini, T. Uricchio, L. Ballan, and R. Caldelli. “Localization of JPEG double compression through multi-domain convolutional neural networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 1865–1871 (cit. on pp. 24, 27).

[AHH17] K. Asghar, Z. Habib, and M. Hussain. “Copy-move and splicing image forgery detection and localization techniques: a review”. In: *Australian Journal of Forensic Sciences* 49.3 (2017), pp. 281–307 (cit. on p. 15).

[Bap+17] J. Bappy, A. K. Roy-Chowdhury, J. Bunk, L. Nataraj, and BS. Manjunath. “Exploiting spatial structure for localizing manipulated image regions”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2017, pp. 4970–4979 (cit. on p. 32).

[Bap+19] J. Bappy, C. Simons, L. Nataraj, BS. Manjunath, and A. K. Roy-Chowdhury. "Hybrid LSTM and encoder–decoder architecture for detection of image forgeries". In: *IEEE Transactions on Image Processing* 28.7 (2019), pp. 3286–3300 (cit. on pp. 29, 30, 32, 107).

[BBB17] M. Barni, L. Bondi, and N. Bonettini. "Aligned and non-aligned double JPEG detection using convolutional neural networks". In: *Journal of Visual Communication and Image Representation* 49 (2017), pp. 153–163 (cit. on pp. 23, 27).

[Bar+18] M. Barni, A. Costanzo, E. Nowroozi, and B. Tondi. "CNN-based detection of generic contrast adjustment with JPEG post-processing". In: *Proceedings of the IEEE International Conference on Image Processing*. 2018, pp. 3803–3807 (cit. on pp. 25, 27).

[BPT19] M. Barni, Q. Phan, and B. Tondi. "Copy move source-target disambiguation through multi-branch CNNs". In: *arXiv preprint arXiv:1912.12640* (2019) (cit. on pp. 34, 36).

[Bar+19] E. Bartusiak, S. Yarlagadda, D. Güera, P. Bestagini, S. Tubaro, F. Zhu, and E. J. Delp. "Splicing detection and localization in satellite imagery using conditional gans". In: *Proceedings of the IEEE Conference on Multimedia Information Processing and Retrieval*. 2019, pp. 91–96 (cit. on pp. 33, 34).

[BS18a] B. Bayar and M. C. Stamm. "Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection". In: *IEEE Transactions on Information Forensics and Security* 13.11 (2018), pp. 2691–2706 (cit. on pp. 26, 27, 30, 49–51, 53–56, 62, 65, 72–80, 89, 92–94, 98).

[BS18b] B. Bayar and M. C. Stamm. "Towards open set camera model identification using a deep learning framework". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. 2018, pp. 2007–2011 (cit. on pp. 38, 39).

[BSF94] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166 (cit. on p. 48).

[Ber+19] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. "MVTec AD–A comprehensive real-world dataset for unsupervised anomaly detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9592–9600 (cit. on pp. 17, 36).

[Bi+19] X. Bi, Y. Wei, B. Xiao, and W. Li. "RRU-Net: The ringed residual U-Net for image splicing forgery detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 1–10 (cit. on pp. 33, 34).

[BP12] T. Bianchi and A. Piva. "Image forgery localization via block-grained analysis of JPEG artifacts". In: *IEEE Transactions on Information Forensics and Security* 7.3 (2012), pp. 1003–1017 (cit. on pp. 19, 20, 39).

[Bon+16] L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro. "First steps toward camera model identification with convolutional neural networks". In: *IEEE Signal Processing Letters* 24.3 (2016), pp. 259–263 (cit. on pp. 38, 39).

[Bon+17] L. Bondi, S. Lameri, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro. "Tampering detection and localization through clustering of camera-based CNN features". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 1855–1864 (cit. on pp. 30, 32).

[BDS18] A. Brock, J. Donahue, and K. Simonyan. "Large scale GAN training for high fidelity natural image synthesis". In: *arXiv preprint arXiv:1809.11096* (2018) (cit. on pp. 96, 97).

[Bun+17] J. Bunk, J. Bappy, et al. "Detection and localization of image forgeries using resampling features and deep learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 1881–1889 (cit. on pp. 29, 32).

[CW19] I. Castillo Camacho and 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*. 2019, pp. 107–112 (cit. on pp. 26, 27, 62, 65, 73, 74, 77, 89).

[CW20] I. Castillo Camacho and 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*. 2020, pp. 1–15 (cit. on p. 27).

[Car+13] T. De Carvalho, C. Riess, E. Angelopoulou, H. Pedrini, and A. de Rezende Rocha. "Exposing digital image forgeries by illumination color classification". In: *IEEE Transactions on Information Forensics and Security* 8.7 (2013), pp. 1182–1194 (cit. on pp. 18, 20, 32, 34, 39, 41).

[Cha+19] C. Chan, S. Ginosar, T. Zhou, and A. A Efros. "Everybody dance now". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 5933–5942 (cit. on pp. 42, 45).

[CLL17a] B. Chen, H. Li, and W. Luo. "Image processing operations identification via convolutional neural network". In: *arXiv preprint arXiv:1709.02908* (2017) (cit. on p. 49).

[CLL17b] B. Chen, H. Li, and W. Luo. "Image processing operations identification via convolutional neural network". In: *arXiv preprint arXiv:1709.02908* (2017) (cit. on p. 105).

[Che+18] C. Chen, Q. Chen, J. Xu, and V. Koltun. "Learning to see in the dark". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3291–3300 (cit. on pp. 96, 97).

[CKL15] J. Chen, X. Kang, and Y. Liu. "Median filtering forensics based on convolutional neural networks". In: *IEEE Signal Processing Letters* 22.11 (2015), pp. 1849–1853 (cit. on pp. 22, 23, 27, 49–51, 53, 54, 57, 58).

[CK17] Q. Chen and V. Koltun. “Photographic image synthesis with cascaded refinement networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1511–1520 (cit. on pp. 96, 97).

[Cho+18] Y. Choi, M. Choi, M. Kim, J. Ha, S. Kim, and J. Choo. “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8789–8797 (cit. on pp. 96, 97).

[Cho17] F. Chollet. “Xception: Deep learning with depthwise separable convolutions”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1251–1258 (cit. on p. 30).

[CDY20] U. A. Ciftci, I. Demir, and L. Yin. “Fakecatcher: Detection of synthetic portrait videos using biological signals”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–17 (cit. on pp. 43, 45).

[CPV15] D. Cozzolino, G. Poggi, and L. Verdoliva. “Efficient dense-field copy–move forgery detection”. In: *IEEE Transactions on Information Forensics and Security* 10.11 (2015), pp. 2284–2297 (cit. on p. 107).

[CPV17] D. Cozzolino, G. Poggi, and L. Verdoliva. “Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection”. In: *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*. 2017, pp. 159–164 (cit. on pp. 29, 32).

[CV18] D. Cozzolino and L. Verdoliva. “Camera-based image forgery localization using convolutional neural networks”. In: *Proceedings of the European Signal Processing Conference*. 2018, pp. 1372–1376 (cit. on pp. 30, 32).

[CV19] D. Cozzolino and L. Verdoliva. “Noiseprint: a CNN-based camera model fingerprint”. In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 144–159 (cit. on pp. 38, 39).

[CV16] D. Cozzolino and L. Verdoliva. “Single-image splicing localization through autoencoder-based anomaly detection”. In: *Proceedings of the IEEE International Workshop on Information Forensics and Security*. 2016, pp. 1–6 (cit. on pp. 31, 34).

[DAv+17] D. D’Avino, D. Cozzolino, G. Poggi, and L. Verdoliva. “Autoencoder with recurrent neural networks for video forgery detection”. In: *Electronic Imaging* 2017.7 (2017), pp. 92–99 (cit. on pp. 31, 34).

[Dai+19] T Dai, J. Cai, Y. Zhang, S. Xia, and 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 (cit. on pp. 96, 97).

[Dan+20] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. K. Jain. “On the detection of digital face manipulation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5781–5790 (cit. on pp. 43, 45).

[Dan+15] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. “RAISE: A raw images dataset for digital image forensics”. In: *Proceedings of the ACM Multimedia Systems Conference*. 2015, pp. 219–224 (cit. on pp. 17, 20, 36, 41).

[De +15] M. De Marsico, M. Nappi, D. Riccio, and H. Wechsler. “Mobile iris challenge evaluation (MICHE)-I, biometric iris dataset and protocols”. In: *Pattern Recognition Letters* 57 (2015), pp. 17–23 (cit. on pp. 17, 20, 39).

[DRC17] E. De Rezende, G. Ruppert, and T. Carvalho. “Detecting computer generated images with deep convolutional neural networks”. In: *Proceedings of the SIBGRAPI Conference on Graphics, Patterns and Images*. 2017, pp. 71–78 (cit. on pp. 40, 41).

[Den+09] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and 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 (cit. on pp. 17, 36, 41, 95, 96).

[Din+19] X. Ding, Y. Chen, Z. Tang, and Y. Huang. “Camera identification based on domain knowledge-driven deep multi-task learning”. In: *IEEE Access* 7 (2019), pp. 25878–25890 (cit. on pp. 38, 39).

[Din+20] X. Ding, Z. Raziei, E. C. Larson, E. V. Olinick, P. Krueger, and M. Hahsler. “Swapped face detection using deep learning and subjective assessment”. In: *Eurasip Journal on Information Security* 2020.1 (2020), pp. 1–12 (cit. on pp. 43, 45).

[Dol+19] B. Dolhansky, R. Howes, B. Pflaum, N. Baram, and C. C. Ferrer. “The deepfake detection challenge (DFDC) preview dataset”. In: *arXiv preprint arXiv:1910.08854* (2019) (cit. on pp. 21, 22, 45).

[DW11] J. Dong and W. Wang. *CASIA Tampered Image Detection Evaluation (TIDE) Database, v1.0 and v2.0*. 2011 (cit. on pp. 18, 20, 32, 34, 36).

[Far09] H. Farid. “A survey of image forgery detection”. In: *IEEE Signal Processing Magazine* 2.26 (2009), pp. 16–25 (cit. on p. 15).

[FJY18] L. Fei-Fei, 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”). 2018 (cit. on pp. 49, 51, 61, 62).

[Fer+20] S. Fernandes, S. Raj, R. Ewetz, J. Singh Pannu, S. Kumar Jha, E. Ortiz, I. Vintila, and M. Salter. “Detecting deepfake videos using attribution-based confidence metric”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 308–309 (cit. on pp. 44, 45).

[Fre+19] D. Freire-Obregón, F. Narducci, S. Barra, and M. Castrillón-Santana. “Deep learning for source camera identification on mobile devices”. In: *Pattern Recognition Letters* 126 (2019), pp. 86–91 (cit. on pp. 38, 39).

[FK12] J. Fridrich and J. Kodovsky. “Rich models for steganalysis of digital images”. In: *IEEE Transactions on Information Forensics and Security* 7.3 (2012), pp. 868–882 (cit. on pp. 28, 49, 50, 62, 73, 74, 77–81, 89–94, 98).

[GG16] Y. Gal and Z. Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *Proceedings of the International Conference on Machine Learning*. 2016, pp. 1050–1059 (cit. on p. 9).

[GB10a] T. Gloe and R. Bohme. “The Dresden image database for benchmarking digital image forensics”. In: *Proceedings of the ACM Symposium on Applied Computing*. 2010, pp. 1584–1590 (cit. on pp. 17, 20, 32, 36, 39, 55, 64, 73, 88).

[GB10b] X. Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256 (cit. on pp. 4, 27, 47, 48, 50, 51, 57, 58, 61, 62, 65, 67, 73, 74, 76, 77, 81, 86, 87, 89–91, 98, 100).

[GBB11] X. Glorot, A. Bordes, and Y. Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 315–323 (cit. on p. 8).

[GBC16] I. J. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016 (cit. on pp. 49, 51).

[Goo60] L. A. Goodman. “On the exact variance of products”. In: *Journal of the American Statistical Association* 55.292 (1960), pp. 708–713 (cit. on p. 52).

[GD18] D. Güera and E. J Delp. “Deepfake video detection using recurrent neural networks”. In: *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance*. 2018, pp. 1–6 (cit. on pp. 43, 45).

[He+16] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE/CVG Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778 (cit. on pp. 30, 38).

[He+15] K. He, X. Zhang, S. Ren, and J. Sun. “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1026–1034 (cit. on pp. 48, 49).

[He+18] P. He, X. Jiang, T. Sun, and H. Li. “Computer graphics identification combining convolutional and recurrent neural networks”. In: *IEEE Signal Processing Letters* 25.9 (2018), pp. 1369–1373 (cit. on pp. 40, 41).

[He+20] P. He, H. Li, H. Wang, and R. Zhang. “Detection of Computer Graphics Using Attention-Based Dual-Branch Convolutional Neural Network from Fused Color Components”. In: *Sensors* 20.17 (2020), p. 4743 (cit. on p. 41).

[HS97] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 29).

[HZL20] C.-C. Hsu, Y.-X. Zhuang, and C.-Y. Lee. “Deep fake image detection based on pairwise learning”. In: *Applied Sciences* 10.1 (2020), p. 370 (cit. on pp. 43, 45).

[HC06] Y.-F. Hsu and S.-F. Chang. “Detecting image splicing using geometry invariants and camera characteristics consistency”. In: *Proceedings of the International Conference on Multimedia and Expo*. 2006 (cit. on pp. 18, 20, 32, 34).

[Hua+17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. “Densely connected convolutional networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4700–4708 (cit. on p. 41).

[Huh+18] M. Huh, A. Liu, A. Owens, and A. A. Efros. “Fighting fake news: Image splice detection via learned self-consistency”. In: *Proceedings of the European Conference on Computer Vision*. 2018, pp. 101–117 (cit. on pp. 32, 34).

[IS15] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015) (cit. on pp. 9, 90).

[KK92] B. L. Kalman and S. C. Kwasny. “Why tanh: choosing a sigmoidal function”. In: *Proceedings of the International Joint Conference on Neural Networks*. Vol. 4. 1992, pp. 578–581 (cit. on p. 8).

[Kar+17] T. Karras, T. Aila, S. Laine, and J. Lehtinen. “Progressive growing of GANs for improved quality, stability, and variation”. In: *arXiv preprint arXiv:1710.10196* (2017) (cit. on pp. 18, 20–22, 45, 94–97).

[KLA19] T. Karras, S. Laine, and 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 (cit. on pp. 96, 97).

[KW20] H. Khalid and S. S. Woo. “OC-FakeDect: Classifying deepfakes using one-class variational autoencoder”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 656–657 (cit. on pp. 44, 45).

[KB14] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *Proceedings of the International Conference on Learning Representations*. ICLR, 2014, pp. 1–15 (cit. on pp. 54, 56).

[KKR19] V. V. Kniaz, V. Knyaz, and Fabio Remondino. “The point where reality meets fantasy: Mixed adversarial generators for image splice detection”. In: *Proceedings of the Advances in Neural Information Processing Systems*. 2019, pp. 215–226 (cit. on pp. 33, 34).

[Kor+19] P. Korshunov, M. Halstead, D. Castan, M. Graciarena, M. McLaren, B. Burns, A. Lawson, and S. Marcel. “Tampered speaker inconsistency detection with phonetically aware audio-visual features”. In: *Proceedings of the International Conference on Machine Learning*. 2019, pp. 1–5 (cit. on pp. 21, 22, 43, 45).

[KH17] P. Korus and J. Huang. “Multi-scale analysis strategies in PRNU-based tampering localization”. In: *IEEE Transactions on Information Forensics and Security* 12.4 (2017), pp. 809–824 (cit. on pp. 18, 20, 34, 39).

[Kra91] M. A. Kramer. “Nonlinear principal component analysis using autoassociative neural networks”. In: *AIChE Journal* 37.2 (1991), pp. 233–243 (cit. on p. 28).

[KSH12] A. Krizhevsky, I. Sutskever, and G. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Proceedings of the Advances in Neural Information Processing Systems*. 2012, pp. 1097–1105 (cit. on p. 33).

[LBL09] H. Larochelle, Y. Bengio, and J. Louradour. “Exploring strategies for training deep neural networks”. In: *Journal of Machine Learning Research* 10.1 (2009), pp. 1–40 (cit. on p. 47).

[LeC+98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 12).

[LH19] H. Li and 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 (cit. on pp. 36, 49).

[LZM19] K. Li, T. Zhang, and J. Malik. “Diverse image synthesis from semantic layouts via conditional imle”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4220–4229 (cit. on pp. 96, 97).

[Li+20a] L. Li, J. Bao, T. Zhang, H. Yang, D. Chen, F. Wen, and B. Guo. “Face X-ray for more general face forgery detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5001–5010 (cit. on pp. 43, 45).

[Li+20b] X. Li, K. Yu, S. Ji, Y. Wang, C. Wu, and H. Xue. “Fighting against deepfake: Patch&pair convolutional neural networks (PPCNN)”. In: *Companion Proceedings of the Web Conference*. 2020, pp. 88–89 (cit. on pp. 44, 45).

[LCL18] Y. Li, M.-C. Chang, and S. Lyu. “In ictu oculi: Exposing AI created fake videos by detecting eye blinking”. In: *Proceedings of the IEEE International Workshop on Information Forensics and Security*. 2018, pp. 1–7 (cit. on pp. 43, 45).

[LL18] Y. Li and S. Lyu. “Exposing deepFake videos by detecting face warping artifacts”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 46–52 (cit. on pp. 43, 45).

[Li+20c] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu. “Celeb-DF: A large-scale challenging dataset for DeepFake forensics”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3207–3216 (cit. on pp. 21, 22, 45).

[LCY13] M. Lin, Q. Chen, and S. Yan. “Network in network”. In: *arXiv preprint arXiv:1312.4400* (2013) (cit. on p. 23).

[Lin+14] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft COCO: Common objects in context". In: *Proceedings of the European conference on computer vision*. 2014, pp. 740–755 (cit. on pp. 17, 20, 32, 34, 36, 96).

[LP18] B. Liu and C. Pun. "Locating splicing forgery by fully convolutional networks and conditional random field". In: *Signal Processing: Image Communication* 66 (2018), pp. 103–112 (cit. on pp. 31, 32, 34).

[LGC18] Y. Liu, Q. Guan, and Y. Cao. "Image forgery localization based on multi-scale convolutional neural networks". In: *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*. 2018, pp. 85–90 (cit. on pp. 49, 77).

[LGZ18] Y. Liu, Q. Guan, and X. Zhao. "Copy-move forgery detection based on convolutional kernel network". In: *Multimedia Tools and Applications* 77.14 (2018), pp. 18269–18293 (cit. on pp. 34, 36).

[Liu+19] Y. Liu, X. Zhu, X. Zhao, and Y. Cao. "Adversarial learning for constrained image splicing detection and localization based on atrous convolution". In: *IEEE Transactions on Information Forensics and Security* 14.10 (2019), pp. 2551–2566 (cit. on pp. 33, 34).

[Liu+15] Z. Liu, P. Luo, X. Wang, and X. Tang. "Deep learning face attributes in the wild". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3730–3738 (cit. on pp. 18, 20, 21, 45, 96).

[LSD15] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440 (cit. on p. 31).

[LN20] M. Lu and S. Niu. "A detection approach using LSTM-CNN for object removal caused by exemplar-based image inpainting". In: *Electronics* 9.5 (2020), p. 858 (cit. on pp. 36, 37).

[Mar+18] F. Marra, D. Gragnaniello, D. Cozzolino, and L. Verdoliva. "Detection of GAN-generated fake images over social networks". In: *Proceedings of the IEEE Conference on Multimedia Information Processing and Retrieval*. 2018, pp. 384–389 (cit. on pp. 42, 45).

[Mar+20] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi. "A full-image full-resolution end-to-end-trainable CNN framework for image forgery detection". In: *IEEE Access* 8 (2020), pp. 133488–133502 (cit. on pp. 30, 32).

[Mas+20] I. Masi, A. Killekar, R. M. Mascarenhas, S. P. Gurudatt, and W. AbdAlmageed. "Two-branch recurrent network for isolating deepfakes in videos". In: *arXiv preprint arXiv:2008.03412* (2020) (cit. on pp. 44, 45).

[Mat+15] P. Mattis, M. Douze, Z. Harchaoui, J. Mairal, F. Perronin, and C. Schmid. "Local convolutional features with unsupervised training for image retrieval". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 91–99 (cit. on p. 36).

[MT20] K. B. Meena and V. Tyagi. “A deep learning based method to discriminate between photorealistic computer generated images and photographic images”. In: *Proceedings of the International Conference on Advances in Computing and Data Sciences*. 2020, pp. 212–223 (cit. on p. 41).

[MFC17] NIST MFC. *Media Forensics Challenge 2018*. 2017 (cit. on pp. 18, 20, 39).

[Mit+20] T. Mittal, U. Bhattacharya, R. Chandra, A. Bera, and D. Manocha. “Emotions don’t lie: A deepfake detection method using audio-visual affective cues”. In: *arXiv preprint arXiv:2003.06711* (2020) (cit. on pp. 43, 45).

[MCL18] H. Mo, B. Chen, and W. Luo. “Fake faces identification via convolutional neural network”. In: *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*. 2018, pp. 43–47 (cit. on pp. 42, 45).

[Ng+05] T. Ng, S. Chang, J. Hsu, and M. Pepeljugoski. “Columbia photographic images and photorealistic computer graphics dataset”. In: *Columbia University, ADVENT Technical Report* (2005), pp. 205–2004 (cit. on pp. 21, 22, 41).

[Ngu+19] H. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen. “Multi-task learning for detecting and segmenting manipulated facial images and videos”. In: *Proceedings of the IEEE International Conference on Biometrics Theory, Applications and Systems*. 2019, pp. 1–8 (cit. on pp. 43, 45).

[Ngu+18] H. H. Nguyen, T. N.-D. Tieu, H.-Q. Nguyen-Son, V. Nozick, J. Yamagishi, and I. Echizen. “Modular convolutional neural network for discriminating between computer-generated images and photographic images”. In: *Proceedings of the International Conference on Availability, Reliability and Security*. 2018, pp. 1–10 (cit. on pp. 40, 41).

[NYE19] H. H. Nguyen, J. Yamagishi, and I. Echizen. “Capsule-forensics: Using capsule networks to detect forged images and videos”. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. 2019, pp. 2307–2311 (cit. on pp. 40, 41).

[NCY19] X. Ni, L. Chen, and Y. Yao. “An evaluation of deep learning-based computer generated image detection approaches”. In: *IEEE Access* 7 (2019), pp. 130830–130840 (cit. on p. 15).

[NZ08] M. Nilsback and A. Zisserman. “Automated flower classification over a large number of classes”. In: *Proceedings of the Indian Conference on Computer Vision, Graphics & Image Processing*. 2008, pp. 722–729 (cit. on pp. 17, 36).

[NIS16] NIST. *Nimble Datasets*. 2016 (cit. on pp. 18–20, 31, 32, 34, 35, 39).

[OK15] H. Su O. Russakovsky J. Deng and J. Krause. “ImageNet large scale visual recognition challenge”. In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252 (cit. on p. 12).

[OLL17] J. Ouyang, Y. Liu, and M. Liao. “Copy-move forgery detection based on deep learning”. In: *Proceedings of the International Congress on Image and Signal Processing, BioMedical Engineering and Informatics*. 2017, pp. 1–5 (cit. on pp. 33, 36).

[PP11] T. Filler P. Bas and T. Pevny. “Break our steganographic system: the ins and outs of organizing BOSS”. In: *Proceedings of the International Workshop on Information Hiding*. 2011, pp. 59–70 (cit. on pp. 17, 20).

[Par+18] J. Park, D. Cho, W. Ahn, and H. Lee. “Double JPEG detection in mixed JPEG quality factors using deep convolutional neural network”. In: *Proceedings of the European Conference on Computer Vision*. 2018, pp. 636–652 (cit. on pp. 24, 27).

[Par+19] T. Park, M. Liu, T. Wang, and 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 (cit. on pp. 96, 97).

[Piv13] A. Piva. “An overview on image forensics”. In: *ISRN Signal Processing* 2013 (2013), pp. 1–22 (cit. on pp. 14, 15).

[Pom+18] T. Pomari, G. Ruppert, E. Rezende, A. Rocha, and T. Carvalho. “Image splicing detection through illumination inconsistencies and deep learning”. In: *Proceedings of the IEEE International Conference on Image Processing*. 2018, pp. 3788–3792 (cit. on pp. 32, 34).

[PF04] A. Popescu and H. Farid. “Statistical tools for digital forensics”. In: *Proceedings of the International Workshop on Information Hiding*. 2004, pp. 128–147 (cit. on p. 23).

[Qua+18] W. Quan, K. Wang, D.-M. Yan, and X. Zhang. “Distinguishing between natural and computer-generated images using convolutional neural networks”. In: *IEEE Transactions on Information Forensics and Security* 13.11 (2018), pp. 2772–2787 (cit. on pp. 40, 41).

[Qua+20] W. Quan, K. Wang, D.-M. Yan, X. Zhang, and D. Pellerin. “Learn with diversity and from harder samples: Improving the generalization of CNN-Based detection of computer-generated images”. In: *Forensic Science International: Digital Investigation* 35 (2020), 301023:1–12 (cit. on p. 41).

[Rah+17] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen. “Distinguishing computer graphics from natural images using convolution neural networks”. In: *Proceedings of the IEEE Workshop on Information Forensics and Security*. 2017, pp. 1–6 (cit. on pp. 22, 40, 41).

[RN16] Y. Rao and J. Ni. “A deep learning approach to detection of splicing and copy-move forgeries in images”. In: *Proceedings of the IEEE International Workshop on Information Forensics and Security*. 2016, pp. 1–6 (cit. on pp. 29, 32).

[RNZ20] Y. Rao, J. Ni, and H. Zhao. “Deep learning local descriptor for image splicing detection and localization”. In: *IEEE Access* 8 (2020), pp. 25611–25625 (cit. on pp. 33, 34).

[Ric+16] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. “Playing for data: Ground truth from computer games”. In: *Proceedings of the European Conference on Computer Vision*. 2016, pp. 102–118 (cit. on p. 96).

[RFB15] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional networks for biomedical image segmentation”. In: *Proceedings of the International Conference on Medical image computing and computer-assisted intervention*. 2015, pp. 234–241 (cit. on p. 33).

[Ros+19] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. “Faceforensics++: Learning to detect manipulated facial images”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 1–11 (cit. on pp. 21, 22, 45, 96, 97).

[Rös+18] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. “Faceforensics: A large-scale video dataset for forgery detection in human faces”. In: *arXiv preprint arXiv:1803.09179* (2018) (cit. on pp. 21, 22, 41, 42, 45).

[RHW86] D. Rumelhart, G. Hinton, and R.J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536 (cit. on p. 11).

[SFH17] S. Sabour, N. Frosst, and G. Hinton. “Dynamic routing between capsules”. In: *Proceedings of the Advances in Neural Information Processing Systems*. 2017, pp. 3856–3866 (cit. on p. 40).

[SRK18] R. Salloum, Y. Ren, and C-C. K. Kuo. “Image splicing localization using a multi-task fully convolutional network (MFCN)”. In: *Journal of Visual Communication and Image Representation* 51 (2018), pp. 201–209 (cit. on pp. 31, 32, 34).

[SN20] V. U. Sameer and R. Naskar. “Deep siamese network for limited labels classification in source camera identification”. In: *Multimedia Tools and Applications* 79.37 (2020), pp. 28079–28104 (cit. on pp. 38, 39).

[SS04] G. Schaefer and M. Stich. “UCID – An uncompressed colour image database”. In: *Proceedings of the SPIE: Storage and Retrieval Methods and Applications for Multimedia*. 2004, pp. 472–480 (cit. on pp. 17, 20, 32, 36, 57).

[Sha+19] W. Shan, Y. Yi, R. Huang, and Y Xie. “Robust contrast enhancement forensics based on convolutional neural networks”. In: *Signal Processing: Image Communication* 71 (2019), pp. 138–146 (cit. on pp. 25, 27).

[Shu+17] D. Shullani, M. Fontani, M. Iuliani, O. Al Shaya, and A. Piva. “VISION: a video and image dataset for source identification”. In: *EURASIP Journal on Information Security* 2017.1 (2017), p. 15 (cit. on pp. 17, 20, 32, 36, 39, 41).

[SO01] E. P. Simoncelli and B. A. Olshausen. “Natural image statistics and neural representation”. In: *Annual Review of Neuroscience* 24.1 (2001), pp. 1193–1216 (cit. on pp. 59, 62, 63, 85).

[SZ14] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014) (cit. on pp. 12, 13, 31).

[SL10] M. C. Stamm and K. R. Liu. “Forensic detection of image manipulation using statistical intrinsic fingerprints”. In: *IEEE Transactions on Information Forensics and Security* 5.3 (2010), pp. 492–506 (cit. on p. 25).

[SWL13] M. C. Stamm, M. Wu, and K. R. Liu. “Information forensics: An overview of the first decade”. In: *IEEE Access* 1 (2013), pp. 167–200 (cit. on p. 15).

[Sun+18a] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. “PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8934–8943 (cit. on p. 43).

[Sun+18b] J. Sun, K. Seung-Wook, L. Sang-Won, and K. Sung-Jea. “A novel contrast enhancement forensics based on convolutional neural networks”. In: *Signal Processing: Image Communication* 63 (2018), pp. 149–160 (cit. on pp. 25, 27).

[Tan+18] H. Tang, R. Ni, Y. Zhao, and X. Li. “Median filtering detection of small-size image based on CNN”. In: *Journal of Visual Communication and Image Representation* 51 (2018), pp. 162–168 (cit. on pp. 23, 27).

[Tar+19] D. B. Tariang, P. Senguptab, A. Roy, R. S. Chakraborty, and R. Naskar. “Classification of computer generated and natural images based on efficient deep convolutional recurrent attention model”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 146–152 (cit. on pp. 40, 41).

[Tar+18] S. Tariq, S. Lee, H. Kim, Y. Shin, and S. S. Woo. “Detecting both machine and human created fake face images in the wild”. In: *Proceedings of the International Workshop on Multimedia Privacy and Security*. 2018, pp. 81–87 (cit. on pp. 42, 45).

[TR20] R. Thakur and R. Rohilla. “Recent advances in digital image manipulation detection techniques: A brief review”. In: *Forensic Science International* 312 (2020), 110311:1–110311:13 (cit. on p. 44).

[Tra+13] D. Tralic, I. Zupancic, S. Grgic, and M. Grgic. “CoMoFoD—New database for copy-move forgery detection”. In: *Proc. of the International Symposium on Electronics in Marine*. 2013, pp. 49–54 (cit. on pp. 19, 20, 36).

[TCC16] A. Tuama, F. Comby, and M. Chaumont. “Camera model identification with the use of deep convolutional neural networks”. In: *Proceedings of the IEEE International Workshop on Information Forensics and Security*. 2016, pp. 1–6 (cit. on pp. 38, 39).

[Ver20] L. Verdoliva. “Media forensics and deepfakes: an overview”. In: *arXiv preprint arXiv:2001.06564* (2020) (cit. on p. 15).

[VAK18] V. Verma, N. Agarwal, and N. Khanna. “DCT-domain deep convolutional neural networks for multiple JPEG compression classification”. In: *Signal Processing: Image Communication* 67 (2018), pp. 22–33 (cit. on pp. 23, 27).

[WZ16] Q. Wang and R. Zhang. “Double JPEG compression forensics based on a convolutional neural network”. In: *EURASIP Journal on Information Security* 2016.1 (2016), p. 23 (cit. on pp. 23, 24, 27).

[Wan+20] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and 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 (cit. on pp. 44, 45, 94–97).

[WNW20] X. Wang, S. Niu, and H. Wang. “Image inpainting detection based on multi-task deep learning network”. In: *IETE Technical Review* (2020), pp. 1–9 (cit. on p. 36).

[WWN19] X. Wang, H. Wang, and S. Niu. “An image forensic method for AI inpainting using faster R-CNN”. In: *Proceedings of the International Conference on Artificial Intelligence and Security*. 2019, pp. 476–487 (cit. on p. 36).

[Wen+16] B. Wen, Y. Zhu, R. Subramanian, T. Ng, X. Shen, and S. Winkler. “COVERAGE—A novel database for copy-move forgery detection”. In: *Proceedings of the IEEE International Conference on Image Processing*. 2016, pp. 161–165 (cit. on pp. 19, 20, 32, 36).

[WFT20] J. Wu, K. Feng, and M. Tian. “Review of imaging device identification based on machine learning”. In: *Proceedings of the International Conference on Machine Learning and Computing*. 2020, pp. 105–110 (cit. on p. 15).

[WAN18a] Y. Wu, W. Abd-Almageed, and P. Natarajan. “Busternet: Detecting copy-move image forgery with source/target localization”. In: *Proceedings of the European Conference on Computer Vision*. 2018, pp. 168–184 (cit. on pp. 34–36).

[WAN17] Y. Wu, W. Abd-Almageed, and P. Natarajan. “Deep matching and validation network: An end-to-end solution to constrained image splicing localization and detection”. In: *Proceedings of the ACM international conference on Multimedia*. 2017, pp. 1480–1502 (cit. on pp. 31, 34).

[WAN18b] Y. Wu, W. Abd-Almageed, and P. Natarajan. “Image copy-move forgery detection via an end-to-end deep neural network”. In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*. 2018, pp. 1907–1915 (cit. on pp. 34, 36).

[WAN19] Y. Wu, W. AbdAlmageed, and P. Natarajan. “ManTra-Net: Manipulation tracing network for detection and localization of image forgeries with anomalous features”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9543–9552 (cit. on pp. 30, 32, 102, 107).

[Xia+10] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. “Sun database: Large-scale scene recognition from abbey to zoo”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2010, pp. 3485–3492 (cit. on pp. 17, 20, 34, 36).

[Xua+19] X. Xuan, B. Peng, W. Wang, and J. Dong. “On the generalization of GAN image forensics”. In: *Proceedings of the Chinese Conference on Biometric Recognition*. 2019, pp. 134–141 (cit. on pp. 43, 45).

[Yan+20] P. Yang, D. Baracchi, R. Ni, Y. Zhao, F. Argenti, and A. Piva. “A survey of deep learning-based source image forensics”. In: *Journal of Imaging* 6.9 (2020), pp. 1–24 (cit. on p. 15).

[YLL19] X. Yang, Y. Li, and S. Lyu. “Exposing deep fakes using inconsistent head poses”. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. 2019, pp. 8261–8265 (cit. on pp. 21, 22, 45).

[Yao+18] Y. Yao, W. Hu, W. Zhang, T. Wu, and Y.-Q. Shi. “Distinguishing computer-generated graphics from natural images based on sensor pattern noise and deep learning”. In: *Sensors* 18.4 (2018), pp. 1–11 (cit. on pp. 40, 41).

[Yar+18] S. K. Yarlagadda, D. Güera, P. Bestagini, F. M. Zhu, S. Tubaro, and E. J. Delp. “Satellite image forgery detection and localization using gan and one-class classifier”. In: *Electronic Imaging* 2018.7 (2018), pp. 1–9 (cit. on pp. 30, 32).

[Yu+15] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. “Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop”. In: *arXiv preprint arXiv:1506.03365* (2015) (cit. on p. 96).

[Yu+17] I.-J. Yu, D.-G. Kim, J.-S. Park, J.-U. Hou, S. Choi, and H.-K. Lee. “Identifying photorealistic computer graphics using convolutional neural networks”. In: *Proceedings of the IEEE International Conference on Image Processing*. 2017, pp. 4093–4097 (cit. on pp. 40, 41).

[ZKS20] V. Zavrtanik, M. Kristan, and D. Skčaj. “Reconstruction by inpainting for visual anomaly detection”. In: *Pattern Recognition* (2020). (in press) (cit. on pp. 36, 37).

[Zha+20] R. Zhang, W. Quan, L. Fan, L. Hu, and D.-M. Yan. “Distinguishing Computer-Generated Images from Natural Images Using Channel and Pixel Correlation”. In: *Journal of Computer Science and Technology* 35 (2020), pp. 592–602 (cit. on pp. 40, 41).

[Zha+16] Y. Zhang, J. Goh, L. Win, and V. Thing. “Image region forgery detection: A deep learning approach”. In: *Proceedings of the Singapore Cyber-Security Conference*. 2016, pp. 1–11 (cit. on pp. 28, 32).

[ZWZ18] Z. Zhang, C. Wang, and X. Zhou. “A survey on passive image copy-move forgery detection”. In: *Journal of Information Processing Systems* 14.1 (2018), pp. 6–31 (cit. on p. 15).

[ZZT19] L. Zheng, Y. Zhang, and V. L. Thing. “A survey on image tampering and its detection in real-world photos”. In: *Journal of Visual Communication and Image Representation* 58 (2019), pp. 380–399 (cit. on p. 15).

[Zho+14] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. “Learning deep features for scene recognition using places database”. In: *Proceedings of the Advances in Neural Information Processing Systems*. 2014, pp. 487–495 (cit. on pp. 17, 36).

[Zho+20] P. Zhou, B. Chen, X. Han, M. Najibi, A. Shrivastava, S. Lim, and 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 (cit. on pp. 30, 32).

[Zho+18] P. Zhou, X. Han, V. Morariu, and L. S. Davis. "Learning rich features for image manipulation detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1053–1061 (cit. on pp. 30–32, 107).

[Zhu+17] J.-Y. Zhu, T. Park, P. Isola, and 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 (cit. on pp. 45, 96, 97).

[Zhu+18] X. Zhu, Y. Qian, X. Zhao, B. Sun, and Y. Sun. "A deep learning approach to patch-based image inpainting forensics". In: *Signal Processing: Image Communication* 67 (2018), pp. 90–99 (cit. on p. 36).

[Zhu+20] Y. Zhu, C. Chen, G. Yan, Y. Guo, and Yongfeng Y. Dong. "AR-Net: Adaptive attention and residual refinement network for copy-move forgery detection". In: *IEEE Transactions on Industrial Informatics* 16.10 (2020), pp. 6714–6723 (cit. on pp. 35, 36).

Webpages

[@ABV05] ABVENT. *Artlantis Gallery*. 2005. URL: https://atl.artlantis.com/en/gallery/ (visited on Nov. 7, 2020) (cit. on pp. 21, 41).

[@Alj11] Aljazeera. *Iran Missile Photo 'Doctored'*. 2011. URL: https://www.aljazeera.com/news/2008/7/11/iran-missile-photo-doctored (visited on Oct. 6, 2020) (cit. on p. 1).

[@Ama18] Amazon Web Services Inc. *Landsat on AWS*. 2018. URL: https://aws.amazon.com/public-datasets/landsat (visited on Nov. 18, 2020) (cit. on pp. 17, 20, 32, 34).

[@Aut20] Autodesk Inc. *Autodesk A360 Rendering Gallery*. 2020. URL: https://gallery.autodesk.com/a360rendering/ (visited on Nov. 7, 2020) (cit. on pp. 21, 41).

[@Cha20] Chaos Czech a.s. *Corona Renderer Gallery*. 2020. URL: https://corona-renderer.com/gallery (visited on Nov. 7, 2020) (cit. on pp. 21, 41).

[@dee20] deepfakes@Github. *Faceswap Github*. 2020. URL: https://github.com/deepfakes/faceswap (visited on Nov. 18, 2020) (cit. on p. 21).

[@Edi08] Editor & Publisher Staff. *UPDATE: Reuters Fires Photog Over Doctored Pictures*. 2008. URL: https://web.archive.org/web/20060908091324/http://editorandpublisher.com/eandp/news/article_display.jsp?vnu_content_id=1002950988&imw=Y (visited on Oct. 6, 2020) (cit. on p. 2).

[@Goo19] Google AI. *Deepfakes Detection Dataset*. 2019. URL: https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html (visited on Nov. 18, 2020) (cit. on pp. 21, 22, 45).

[@IEE14] IEEE IFS-TC. *IEEE IFS-TC Image Forensics Challenge Dataset*. 2014. URL: http://ifc.recod.ic.unicamp.br/fc.website/index.py (visited on Nov. 18, 2020) (cit. on pp. 18, 20, 32).

[@IEE18] IEEE Signal Processing Society. *IEEE's Signal Processing Society - Camera Model Identification Competition*. 2018. URL: https://www.kaggle.com/c/sp-society-camera-model-identification (visited on Nov. 18, 2020) (cit. on pp. 17, 20, 32).

[@Iqb18] H. Iqbal. *PlotNeuralNet v1.0.0*. 2018. URL: http://doi.org/10.5281/zenodo.2526396 (visited on Nov. 7, 2020) (cit. on pp. 13, 80).

[@Ltd20] CGworld Pte Ltd. *Learn V-Ray Gallery*. 2020. URL: https://www.learnvray.com/fotogallery/ (visited on Nov. 7, 2020) (cit. on pp. 21, 41).

[@Mac04] H. Macdonald. *NRCS Photo Gallery*. 2004. URL: http://serc.carleton.edu/introgeo/interactive/examples/morrisonpuzzle.html (visited on Nov. 7, 2020) (cit. on p. 20).

[@ML15] Y. Ming and O. Laurent. *World Press Photo Disqualifies 20% of Its Contest Finalists*. 2015. URL: https://time.com/3706626/world-press-photo-processing-manipulation-disqualified/ (visited on Oct. 6, 2020) (cit. on p. 1).

[@NHC04] T. Ng, J. Hsu, and S. Chang. *A Data Set of Athentic and Spliced Image Blocks*. 2004. URL: http://www.ee.columbia.edu/ln/dvmm/downloads/AuthSplicedDataSet/AuthSplicedDataSet.htm (cit. on pp. 18, 20, 32, 34).

[@Woo20] P. Wood. *Pulitzer Board Must Revoke Nikole Hannah-Jones' Prize*. 2020. URL: https://www.nas.org/blogs/article/pulitzer-board-must-revoke-nikole-hannah-jones-prize (visited on Oct. 6, 2020) (cit. on p. 1).