Multimedia Tools and Applications (2023) 82:17521–17566
https://doi.org/10.1007/s11042-022-13797-w

Image forgery detection: a survey of recent deep-learning approaches

Marcello Zanardelli[1] · **Fabrizio Guerrini**[1] · **Riccardo Leonardi**[1] · **Nicola Adami**[1]

Received: 11 August 2021 / Revised: 23 March 2022 / Accepted: 5 September 2022 /
Published online: 3 October 2022

Abstract
In the last years, due to the availability and easy of use of image editing tools, a large amount of fake and altered images have been produced and spread through the media and the Web. A lot of different approaches have been proposed in order to assess the authenticity of an image and in some cases to localize the altered (forged) areas. In this paper, we conduct a survey of some of the most recent image forgery detection methods that are specifically designed upon Deep Learning (DL) techniques, focusing on commonly found copy-move and splicing attacks. DeepFake generated content is also addressed insofar as its application is aimed at images, achieving the same effect as splicing. This survey is especially timely because deep learning powered techniques appear to be the most relevant right now, since they give the best overall performances on the available benchmark datasets. We discuss the key-aspects of these methods, while also describing the datasets on which they are trained and validated. We also discuss and compare (where possible) their performance. Building upon this analysis, we conclude by addressing possible future research trends and directions, in both deep learning architectural and evaluation approaches, and dataset building for easy methods comparison.

Keywords Image forgery detection · Image forensics · Deep learning · Copy-move · Splicing · DeepFake · Survey

✉ Marcello Zanardelli
m.zanardelli005@unibs.it

Fabrizio Guerrini
fabrizio.guerrini@unibs.it

Riccardo Leonardi
riccardo.leonardi@unibs.it

Nicola Adami
nicola.adami@unibs.it

[1] Department of Information Engineering, CNIT – University of Brescia,
Via Branze 38, 25134 Brescia, Italy

1 Introduction

The worldwide spread of smart devices, which integrate increasing quality cameras and image processing tools and "apps", the ubiquity of desktop computers, and the fact that all these devices are almost permanently connected with each other and to remotely located data servers through the Internet, have given ordinary people the possibility to collect, store, and process an enormous quantity of digital visual data on a scale just until recently quite unthinkable.

As a consequence, images and videos are often shared and considered as information sources in several different contexts. Indeed, a great amount of everyday facts are documented through the use of smartphones, even by professionals [64]. Massive sharing of visual content is enabled by a variety of digital technologies [79], such as effective compression methods, fast networks, and specially designed user applications. These latter, in particular, include Web platforms, e.g., social networks such as Instagram and forums like Reddit, that allow the almost instantaneous spreading of user generated images and video. On the other hand, user-friendly, advanced image editing software, both commercial like Adobe Photoshop [3], and free and open source like GIMP [33], not to mention smartphone-based apps that can apply basic image manipulations on the fly,[1] are widely available to everyone.

All these factors have contributed to the spread of fake or *forged* images and videos, in which the semantic content is significantly altered. Sometimes this is done for malevolent purposes, such as political or commercial ones [94]. As of 2022, all of the major social network platforms are struggling to filter manipulated data, and so avoid that such fake content, often directed to the most vulnerable users, could "go viral" [96]. Legal conundrums are also emerging regarding where to put the responsibility for the possibly damaging fallout of fake content spreading [34].

Such problems arise because most times humans are easily fooled by forgeries, and in some cases they are even demonstrably not able to detect any but the less sophisticated modifications undergone by visual content, due to the so-called change blindness cognitive effect [73, 93]. Thus, there is the need for carefully designed digital techniques.

Semantic alterations can be carried out on all types of digital media content, like video or even audio. However, the focus of the analysis presented in this paper is on methods and algorithms specifically designed for forgery detection on *still images*, which is by far the most common case.

In this context, the general problem of determining if a given image has not been altered so as to modify its semantics is referred to as image authentication, or image integrity verification [48]. If the emphasis is put on expressly establish if a given image has undergone a semantic alteration, or *forgery*, the same application is often referred to as *image forgery detection* in the literature [29]. The objective of this paper is to provide a survey of selected forgery detection methods, with particular attention to *deep learning (DL)* techniques that have since come to the fore.

Before starting our analysis on forgery detection methods, in the rest of this Section we frame why we think this comprehensive, performance-driven survey that describes the most recent DL methods is both timely and necessary. We first provide a broad overview of the considered application, mainly to fix some definitions. Next, we provide a concise summary

[1] For example Instagram, Snapseed, Prisma Photo Editor, Visage, and many more.

of the most commonly found types of forgery. We finally provide the organization of the remainder of the paper while also detailing the contributions of our present analysis.

1.1 Image forgery detection applications

Image forgery detection can mainly be divided into two categories: *active* and *passive*. Sometimes these methods also give a localization of the altered/forged areas of the image, and even provide an estimate of the original visual content.

Active methods for general visual content protection are based on technologies like *digital signatures* [74] and *digital watermarking* [6]. Digital signatures are straight cryptographic methods that authenticate the bit-stream. However, the authentication in this case is fragile, meaning that any change in the bit-stream invalidates the signature, and thus it is more tailored to alternative applications such as copyright protection. This is instead not desirable when verifying image semantic content, since alterations that does not change the semantics e.g., a mild amount of compression) should be tolerated. In other words, the authentication method needs to be robust. Another serious drawback is that the signature has to be attached as metadata to the image, and therefore could be discarded or sometimes even substituted by a malicious user.

To address these shortcomings, robust methods have been proposed. For example, robust digital watermarking embeds security information in the content itself by controlled imperceptible modifications. Ideally, an attacker should not be able to alter the content of an image without changing the embedded watermark, while being able to safely apply selected processing such as compression, thus allowing the consumer of the image to detect the manipulation.

Note that variants of the aforementioned approaches exist, namely, robust signatures (based on content hashing techniques) [87, 92], and fragile watermarking [21]. Sometimes these variants have been cleverly combined [66]. However, they still inherit the same problems associated to metadata presence and fragility that we have just outlined.

In the end, active methods have the advantage of being able to convey side information which may be useful to detect the attempted forgery, but they need the watermark or signature to be computed on the unaltered version of the image, ideally at acquisition level. This in turn requires the capturing camera to have specific hardware and/or in-board post-processing software. Furthermore, any entity interested in verifying the semantic content of a given image must be able to decode the authentication information, which means having access to the (private or public) key of the creator and/or the watermark detector. However, leaving to potentially malicious users both the security information embedding and the decoding devices is usually a threat to the entire framework.[2]

As an alternative, a trusted third party could be set up to verify the image integrity, for instance, a Web site able to embed and decode the watermarks. However, scalability problems prevent such architecture to be feasible for everyday images shared on the Internet. Recently, commercial solutions based on the blockchain paradigm aimed at image integrity have also appeared to get rid of the trusted third party presence, though little details at the present time are known of their inner workings.[3] Blockchain methods can be considered

[2]The on-board image signature algorithm developed by Nikon, for example, has been long compromised [10]. Another high profile case is Blu-Ray, which protection scheme used a combination of cryptography, digital signatures, and watermarking [77]

[3]For example, Photo Proof Pro by Keeex [46] and Numbers Protocol [76]

active only in the sense that a block needs to be generated for each protected image, but the image itself is released without modifications. To the best of the authors' knowledge, however, these techniques are not widespread for forgery detection. That may well be because, while the distributed ledger paradigm does not need a trusted third party, fragile authentication is unavoidable since in the end blockchain has a cryptographic core, and furthermore scalability issues are still present. Still, new solutions are being proposed in this field, for instance [47].

Conversely, passive methods do not need the presence of additional data attached to the image, and they are commonly known as *forensics* [81]. Their goal is thus to tell whether an image is authentic or not by analysing only the image itself, searching for traces of specific processing undergone by the image. In the case of massively shared, ordinary images, this solution has been traditionally considered the only feasible one.

Often, an attacker can apply one or a set of successive manipulations on the target image, either on the whole image or only on a tampered region, such as a semantic alteration, e.g. object duplication, JPEG compression, geometric transformations, up-sampling, filtering, e.g. contrast enhancement, and so on. When this chain of manipulations is used by an attacker to disguise the original forgery it is referred to as *anti-forensics*.

The task of determining the history of attacks that a target image has undergone is sometimes called image *philogeny* [70]. Of course, this is a more challenging problem than simply telling apart pristine and forged images, as it involves the detection of multiple kind of attacks while also determining the order in which they were performed. Let us consider, for example, a scenario in which the attacker can perform three different manipulations, and assume for simplicity that each attack is applied at most once. The number N of possible processing histories is thus the sum of simple dispositions of k attacks from the possible three, as in:

$$N = \sum_{k=0}^{3} D_{3,k} = \sum_{k=0}^{3} \frac{3!}{(3-k)!} = 16 \tag{1}$$

Note that $k = 0$ means that the image is pristine. As can be observed, the number of possible histories grows exponentially with the number of available attacks. A possible solution can be found in [14, 60, 61], where the authors formulated the problem of determining the processing history as a multi-class classification problem. Therein, each of the N histories corresponds to a class, and a fusion-decision algorithm tries to combine the outputs of multiple forgery detection methods by means of an agreement function, which aims to give a higher weight to decisions on which more forgery methods agree and less to the ones on which there is less consensus.

As a final note, there is another possible forensics application, that is the trustworthy attribution of the visual content to its creator, for example, the device that generated the image. The forensics traces could be present all the way back at the acquisition level e.g. the camera-specific acquisition noise, known as Photo Response Non Uniformity [32], or PRNU) down to the post-processing stage (that is, after the original image has been stored in digital form) [48].

Sometimes, however, forgery detection follows the "in-the-wild" assumption that the creator of a particular image is not safely attributable to any entity, and thus it is to be considered coming from a possibly anonymous, unreliable source.

1.2 Image forgery types

We now present the most common forgeries and manipulations found in the context of the just discussed applications. Visual examples are depicted in Fig. 1.

Copy-move The copy-move forgery is performed by copying one or more regions of an image and pasting them in the same image in different locations. Copy-move forgeries are typically used to hide information or to duplicate objects/people, thus severely altering the semantic content of the target image. An example of copy-move forgery is shown in Fig. 1a, where the right building tower has been inserted as a copy of the left one.

Splicing This forgery is similar to copy-move, with the difference that the pasted regions/objects are cut from one or more other images. A splicing forgery can be done in order to hide some content, or to show a fake situation. For example, in Fig. 1b, we can see an image in which two famous people are depicted together, but the picture has been shown to be the composition of two different images.

Inpainting This kind of attack consists in filling a region or a "hole" in the image with plausible content. Inpainting is typically employed to restore damaged patches in images. However, it can also be used by potential attackers as a malicious means to hide information from an image, or to remove a visible watermark. The filled region can either be copied from another part of the image, or synthesized with a specific algorithm, such as a GAN network (Generative Adversarial Network [35], see also below). Note that, in the former instance, this attack can be thought as a particular instance of copy-move.

A particularly interesting instance of inpainting is the reconstruction of deleted parts of faces, such as the eyes or the mouth. Promising results in this regard have been obtained by *Nvidia* [63] (an example is shown in Fig. 1c).

DeepFakes DeepFake is a particular kind of manipulation in which a deep learning model is employed to synthesize fake content in images or videos. The "deep" term is used to emphasize the difference between the pre-DL era, in which this task was manually done by experts with professional editing tools, and the current era, in which this is automatically done by deep models, such as GANs [35].

A typical application of DeepFake consists in the substitution of the face of a person with the face of another person (usually a VIP) taken from a second image or video (see Fig. 1e). In another kind of DeepFake attack the facial expressions of a donor person are extracted and applied to the target person in another image or video. This is usually done by means of synthetization methods (namely, GANs) or by merging algorithms that aims to maximise the realism of the obtained face.

Even if most of the time DeepFakes are created for entertainment/comedy purposes, there have been cases in which a VIP was shown to be in certain situations in which he/she never was, thus damaging his/her image and leading to scandals. As a matter of fact, the vast majority of DeepFakes with the latter purpose are created in the video domain, because this kind of media usually poses a bigger semantic threat to the attacked person/VIP, especially when an appropriate audio track is available and can be matched to the facial expressions of the talking person. Furthermore, a number of easy-to-use tools have been developed to produce convincing DeepFakes, such as *FakeApp*, *faceswap-GAN*, and that available at [27]. As a consequence, many DeepFake videos have been spreading through the Web in the last few years.

(a) An example of a copy-move forgery, taken from MICC-F600 dataset [13]. One of the front towers is a surreptitious duplicate.

(b) A high-profile, in-the-wild example of a splicing forgery, that widely circulated online in 2004. The figure of Jane Fonda, captured in an unrelated 1972 photo, has been spliced into a pre-existing John Kerry picture, taken in 1970 (image taken from [18]).

(c) An example of inpainting generated with Nvidia interactive demo at [79]. The inpainting mask (*i.e.*, the deleted region that must be filled by the inpainting algorithm) is shown in yellow in the left image. The original image is taken from the MICC-F2000 dataset [13].

(d) A photo-realistic CGI rendered image (Photo published on *Reddit*. Author: Hary G, username: *u/Hary1495*).

(e) Examples of DeepFakes generated from different sources (image courtesy of [3]). The first column shows the original frames, while the other five show the DeepFakes obtained using five different sources.

Fig. 1 Examples for each discussed forgery kind

DeepFakes for static images are less common, but they are still worthy of interest for forgery detection purposes. Note that this kind of attack can be thought as a particular case of the aforementioned splicing.

CGI-generated images/videos This approach consists in creating photo-realistic content as the rendering output of a computer graphics generated 3D scene. Thanks to the recent advances in the video-gaming industry and in the GPU technology, techniques such as ray-tracing have been much easier to implement, thus making possible to reach realism levels unthinkable just a few years ago (an example is shown in Fig. 1d). In fact, in recent years a certain number of graphic engines, such as *Unity* and the *Unreal Engine*, have been developed and can be freely (or rather cheaply) used by everyone. So, more and more convincing rendered images/videos are being produced every day.

Consequently, the images generated through these engines can be almost indistinguishable from images taken with a real camera, and, of course, this can be used for malicious intents by potential attackers that can use these renderings to depict false scenes. It is worth noticing, though, that in the case of CGI generated content a certain level of expertise is still required in order to produce convincing results.

In this case, there is no clear parallels with splicing since the generated scene is generated from scratch.

GAN-based face synthesization Last, we introduce a particularly popular kind of fake content generation approach, which consists in the creation of a realistic face of a completely non-existing person, employing the previously cited GAN networks. This is done by feeding the trained model with a vector of random sampled noise, which is converted by the model to a realistic face (theoretically) different from any existing one. Again, as for the previously discussed CGI generated content, the fake image is synthesized anew instead of being copied from another source.

In [45], *Nvidia* proposed a GAN architecture that is considered a breakthrough for this technology. Interactive demos based upon this original work can also be found on the Web, such as [39]. Apart from artifacts that can sometimes still be noticeable in the background, the produced faces are really convincing and they are hardly detectable as fake by the naked eye.

1.3 Contributions and paper organization

Since the early 2000s, a lot of approaches to image forgery detection have been proposed, and many excellent reviews can be found [11, 29, 38, 48, 84, 103]. However, deep learning techniques have proved to be a game-changer in many digital processing and computer vision applications, especially when a lot of training data are available [56, 62, 109]. Even if in the case of forgery detection this last assumption is not quite satisfied, nonetheless, as discussed in what follows, the best performance on standard benchmarks were obtained with algorithms that leverage DL models in one or more phases.

For this reason, we feel that it is very important to keep track of the breakthroughs made possible by deep learning in forgery detection. In particular, it is crucial that some degree of comparison between DL-based techniques that follow different perspectives is carried out. This is especially true since it is challenging to identify future (and even present) trends in a technology like DL, which is already vast and still expanding at a tremendous rate.

In this paper, we mainly focus our discussion on copy-move and splicing detection methods. Even if these attacks are not as recent as GAN-based ones or DeepFakes, they are very

prominent in the literature and lots of algorithms for their detection are still being published to date. The reason why these forgeries are so diffused is mainly because of their simplicity, both related to end user employment and experimental dataset building, but also because they are a very immediate threat to the image semantics integrity.

Even so, we discuss some of the DeepFake detection techniques, insofar as this kind of attacks can be seen of a special (and more sophisticated) case of splicing, or at least a manipulation that usually involves a source or donor image/video and a target one. However, since this work aims to give an overview on *image* forgery detection methods, we do not deal with approaches specifically designed for video content, i.e., that cannot be applied to single images. In fact, video-specific methods typically do not analyze each frame as a standalone image, but they also leverage temporal clues between different frames or, if available, inconsistencies between the audio track and the facial expressions. We refer the reader interested in DeepFakes seen as a standalone research field to the review in [102].

This paper is organized as follows. As stated before, the focus of this paper is on the most recent methods for copy-move and splicing detection that are specifically based upon DL. To better highlight the contrast with the previous state-of-the-art, it is useful to first recap in Section 2 several of the established forensics-based techniques for image forgery detection that instead follow traditional approaches. In Section 3, we describe the key-aspects of the deep learning based methods, including their applicability and their limitations, and we illustrate their properties such as the kind of attacks they can detect and whether they give or not the localization of the forged areas. We concurrently discuss the datasets on which they were trained/tested. Then, in Section 4 we discuss their performance, which are also directly compared when possible (that is, tested on the same benchmark dataset). Finally, in Section 5 we follow up on the previous discussion by drawing some conclusions, while providing some insights on what we think should be the most important future research directions.

2 Traditional passive forgery detection methods

We now briefly discuss some of the "conventional" passive image forgery detection approaches that have been proposed since the early 2000s. Of course, what we present here is not an exhaustive, nor in-depth review of these methods. For a more comprehensive review, see [29, 38], and [103].

Conventional passive methods leverage techniques from the fields of signal processing, statistics, physics, and geometry, and are usually also referred to as "classic" or "traditional" approaches. In fact, they come from the pre-DL era that we are currently in and, as such, they require little or no data to perform an eventual training phase. Those that still require data for training are typically based on traditional machine learning techniques, such as clustering, support vector machines (SVM), linear/logistic regression, random forests, and so on. Here, we still consider those as belonging to the classic methods, because they rely on models that have a relatively small number of parameters, and therefore do not require a great amount of data for training.

We think it is useful to briefly describe some of the traditional approaches, for the following two reasons:

1. As mentioned above, they typically do not require much data for training (or none, even). Of course, this is an advantage when it is hard or impossible to collect a good amount of labelled images to train a high parameterized deep learning model. Also,

most of these methods are not as computationally expensive, and thus can be easily deployed on commercial low-power hardware, like smartphones or tablets;
2. Some of the core ideas and principles these methods rely on can also be used in conjunction with deep learning models, in order to accelerate the training phase or to achieve better performance. For example, in [86], a SVM model is used as final classification phase applied on the output of a CNN. In [85], a YCbCr color space conversion and a DCT transform are used as pre-processing stages before a CNN. In [97], a CNN takes as input the Laplacian filter residuals (LFR) computed on the input images rather than the images themselves. All of these methods, among several others, are discussed in detail in Section 3.

Passive traditional methods can be usually grouped into five main categories. We discuss each separately in the remainder of this Section.

2.1 Pixel based

These methods rely on the fact that certain manipulations introduce anomalies that can affect the statistical content of the image at the pixels level. Some of these anomalies can be detected in the spatial domain, while others in the frequency domain or in a combination of both.

For copy-move attacks, it is common to observe a strong correlation between copied regions in the image but, due to the fact that these can be of any size and shape, it is computationally infeasible to explore the whole space of possible shape/size combinations. The authors of [31] have proposed a method based on the Discrete Cosine Transform (DCT). In particular, they divided the image into overlapping blocks and applied a DCT on each block. The DCT coefficients were used as feature vectors that describe each block. Duplicated regions then were detected by lexicograpycally ordering the DCT block coefficients and grouping the most similar ones. Another approach, proposed in [82], consisted in applying a Principal Component Analysis (PCA) on image blocks' features, and then comparing blocks representation in this reduced-dimension space. These approaches have been shown to be robust when minor variations in the copied regions are performed, like additive noise or lossy compression. However, in general these methods do not perform well in the case of geometric transformations like rotation or scaling.

Thus, let us consider now a situation in which a geometric transformation is used in order to make a copy-move attack more convincing. Geometric transformations usually involve some form of interpolation between neighbouring pixels, in particular, the most common techniques are bilinear or cubic interpolation. Depending on the chosen technique, a specific correlation pattern between these pixels is created. Statistical methods are then employed with the aim of finding these patterns in order to detect regions in which a geometric manipulation has been employed. An example of this approach is described in [83].

An example of frequency-based forgery detection is [28]. To detect spliced regions, the authors observed that, even if the boundary between the spliced region and the original image can be visually imperceptible, high-order Fourier statistics are affected by this kind of manipulation and thus can be used for detection.

Another common type of methods, specifically designed for copy-move attacks detection, are the key-point based methods. They typically require the following steps:

1. Key-points extraction. Key-points are variously defined as "points of interest" of the image, for example, local minima or maxima, corners, blobs, etc. Some of the most commonly employed key-points extraction processes include the well-known Scale

Invariant Feature Transform (SIFT) [65], Speeded Up Robust Features (SURF) [9], or Features from Accelerated Segment Test (FAST) [89];
2. Descriptors extraction. One or more feature vectors (descriptors) are extracted from each key-point. Usually, these vectors are a compact description of the region in the vicinity of the key-point. In addition to the SIFT/SURF feature values, Histogram of Gradients (HOG) and the FAST-based ORB [89] are other common ones;
3. Descriptors matching. In this step, descriptors are compared according to a distance (or a complementary similarity) function. If the distance of two or more descriptors is below a certain threshold, a match between these descriptors is declared;
4. Filtering step. In this phase, some form of filtering of the matching results is done in order to rule out weak matches. This can be done by different criteria, such as Lowe's ratio [65], in which a match is considered valid only if the distance between the two most similar descriptors is considerably smaller than that between the two next-best ones. Other criteria can be employed, for instance, based on the spatial relationship between the key-points.

One of the most cited key-point based methods for copy-move detection was proposed by Amerini et al. in [5]. The authors showed that these methods are quite robust even against rotation and scaling, but the performance are not as good when the copy-moved regions are too uniform. In fact, in this case only few key-points can be extracted, and consequently the matching phase provides weak results.

2.2 Format based

Usually, images captured by a digital camera are encoded in JPEG format. This means that the image is divided into 8×8 pixel blocks, which are then DCT transformed and quantized. As a consequence, specific artefacts are generated at the border of neighbouring blocks. The authors of [67] observed that image manipulations like copy-move or splicing result in alterations in the JPEG artefact pattern, and proposed a method in which they used a sample region (which is supposed authentic) of the target image to estimate the JPEG quantization table. Then, they divided the image into blocks, and a "blocking artefact" measure is computed for each block. A block is considered tampered if the score given by this measure is sufficiently distant from the average value on the whole image.

Obviously, a key limitation of these methods is that they are based on specific assumptions on the format of the stored image (e.g. JPEG), and therefore they are not universally applicable.

2.3 Camera based

The basic idea exploited by these methods is that every digital camera leaves a particular "footprint" or "signature" on each image they generate. This fact can also be useful to tie an image to a specific capturing device. In [32], the authors used a set of images taken by a known camera to estimate the parameters of the already mentioned PRNU, which is a camera specific multiplicative term that models the result of in-camera processing operations. These PRNU parameters are also extracted from the target image, which is supposed to be taken with the same camera, and compared with the previously estimated ones. The idea is that, if a splicing operation from a different camera type has been made, this results in a discrepancy between the estimated parameters.

One of the obvious limitations of this method is that it is camera-specific: this means that a different training set of images must be used for each type of camera in order to build its specific PRNU model. Also, this method is effective just for those splicing attacks in which the spliced region is extracted from a source image taken with a different camera with respect to the one used to acquire the target image, which is not always the case.

The authors of [41], instead, leveraged chromatic aberration to detect image forgeries. The phenomenon of chromatic aberration arises from the fact that photographic lenses are not able to focus light of different wavelengths on the same point on the camera sensor. In fact, from Snell's Law, the refraction index of a material depends on the wavelength of the incident light too. As a consequence, each point of the physical scene is mapped, in the RGB color channels, into points that are spatially slightly shifted one from another.

So, the authors of [41] built a model that approximates the effect of the chromatic aberration and estimated its parameters. Forged regions usually show inconsistencies with the estimated model, and can thus be detected. In this case as well, the main drawback is that this method is camera-specific. In fact, different cameras have different chromatic aberration levels (that typically depend on the kind of lenses), and consequently it is hard to set a specific threshold for the anomalies detection, if the camera from which the target image was taken is not known a priori.

2.4 Lighting based

Typically, when an attacker performs a copy-move or splicing attack, it is hard to ensure that the lighting conditions of the forged region is consistent with that of surrounding image. Compensating for this effect can be hard even using professional software like Adobe Photoshop. Therefore, the basic idea of lighting (or physics) based techniques is to build a global lighting model from the target image, and then to find local inconsistencies with the model as evidence of forgery.

Different lighting models were proposed, such as those in [40] and in [44], for which least squares error approaches are usually employed for parameters estimation. Techniques like Random Sample Consensus (RANSAC) [30] are sometimes used in order to make the model more robust to outliers. The positive aspect of these methods is their wide applicability. In fact, they are not based on assumptions on the type of camera that generated the image, and they can be used to detect both copy-move and splicing attacks. However, a downside of these methods is the fact that they are dependent on the physical context present in the image. In particular, if the lighting conditions are quite complex (for example, an indoor scene), a global lighting model cannot be estimated, and thus the method cannot be applied.

2.5 Geometry based

Geometry-based methods rely on the fact that a copy-move or a splicing attack usually results in some anomalies in the geometric properties of the 3D scene from which the image is obtained.

The authors of [43] proposed a method to estimate the so-called principal point through the analysis of known planar objects, and observed that this is usually near the center of the image. They also showed that a translation of an object in the image plane results in a shift of the principal point, and thus this fact can be used as evidence of forgery.

Another interesting approach was proposed in [42]. The idea was to consider specific known objects such as billboard signs or license plates and make them planar through a

perspective transformation. Once the reference objects are viewed in a convenient plane, it is possible, through a camera calibration, to make real world measurements, which can then be used to make considerations on the authenticity of the objects in the image.

Of course, these methods are based on strong assumptions on the geometry of the 3D scene. They also require a human knowledge of the real world measures taken from specific objects in the image. Consequently, their applicability is quite limited.

3 Deep Learning based methods

Deep learning methods have gained a huge popularity over the past decade, and indeed they have been applied to a great variety of scientific problems. This is due to the fact that they were shown to perform particularly well for classification problems, as well as regression and segmentation ones. For certain tasks, these methods can even outperform humans in terms of accuracy and precision. Another crucial factor that contributed to the spread of deep learning techniques is that, in contrast to conventional machine learning approaches, they do not require the researcher to manually create (craft) meaningful features to be used as input to the learning algorithm, which is often a hard task that requires domain-specific knowledge. Deep learning models, such as Convolution Neural Networks (CNN), are in fact capable of automatically extract descriptive features which capture those facets of the input data that are well tailored to the task at hand.

For image forgery detection too, deep learning techniques have been explored in the recent literature in order to achieve better accuracy than previously proposed, traditional methods. The techniques that we are considering can be grouped in distinct categories according to different criteria, in this case:

A) Type of detected forgery: copy-move, splicing, or both;
B) Localization property, i.e. if the considered algorithm is able to localize the forged areas. In the case of copy-move detection, an additional question is whether the algorithm is able to distinguish between the source region and the target one, i.e. the region on which the source patch is pasted. This property is useful, for example, in a scenario in which a forensic expert is asked to analyze a tampered image in order to interpret the semantic meaning of a copy-move attack;
C) Architecture type, that is, the algorithm is an end-to-end trainable solution, i.e. without parameters that need manual tweaking, or not.

As discussed in Section 1.2, DeepFakes can be regarded as a particular case of splicing attack. However, given the fact that the vast majority of DeepFake forgeries involve face manipulations, methods that aim to detect these attacks can leverage domain-specific knowledge e.g. face detection algorithms) that cannot be used by generic splicing detection algorithms. As such, different datasets need to be used for evaluating and comparing these methods. Therefore, DeepFake forgery detection performance cannot presently be directly compared with generic splicing detection algorithms. Consequently, in this paper, the discussion on the former methods is conducted separately, both in regard to employed datasets and experimental results.

For our analysis, we have selected some papers among the most recent ones that we think are particularly representative of those that can be categorized into at least one of the distinct groups that we have outlined above. A further principle that we have used for this selection is performance driven, with the added objective of being able to do a meaningful comparison (when possible), given in Section 4. These papers are described in some detail in this

Section, with the further objective of identifying if any trend in the DL overall architecture choice is emerging.

In particular, we have used the criteria A) and B) above to sort the presentation order of the papers. Methods [1, 4, 25, 78], and [105] are copy-move-only specific, and are presented first in Section 3.2. Then, methods [22, 68, 85, 86, 97, 105, 107], and [18], that are for both splicing and copy-move detection, are discussed next in Section 3.3.

Besides this first separation through criterion A), we sort the techniques in each subset using criterion B), namely, [1, 4], and [105] in the first subset possess the localization property and are discussed first. For the second subset, such property is verified by [85, 107], and [18], which are thus described before the others. Note that methods [1] and [105] are also able to distinguish the source from the target regions.

Regarding criterion C), which is not used for sorting the methods, we remark here that end-to-end architectures can be found in [25, 68, 105], and [78]. The reader is referred to Table 5 for a summary of the characteristics of the described techniques.

Finally, DeepFake specific methods are discussed in Section 3.4.

For each described method, we also discuss:

- which datasets, whether public benchmark or custom ones, were used for the experimental validation;
- the performance on one or more of the above datasets: metrics like accuracy, precision, localization accuracy, etc.

Therefore, before diving into a detailed overview of the deep learning based approaches, we proceed to first briefly describe in Section 3.1 some of the benchmark datasets that are typically used in the most recent literature for evaluation of the considered forgery detection methods, and summarize the employed performance metrics.

Finally, we mention that there are several other interesting works that involve deep learning as a means for forgery detection, which are however not analyzed here because their characteristics are a mixture of the representative works that we have selected. Some examples are [71] and [106]. In the former, a copy-move-only method is presented that leverages a pre-trained AlexNet (on ImageNet) as a block feature extractor and a subsequent feature matching step that allows to localize the copy-moved regions. In [106], instead, a technique for both copy-move and splicing detection is discussed, which is built upon the formulation of the forgery detection and localization task as a local anomaly detection problem. In particular, a "Z-score" feature is designed that describes the local anomaly level and is used in conjunction with a LSTM (long short term memory) model that is trained to assess local anomalies. Note that both of these methods satisfy criterion B), i.e. they give the localization of the forged areas.

As a further remark regarding the property of being able to distinguish between source and target regions, we refer the reader to the recently published work in [7], in which a DL-based method is presented as a post-processing phase to distinguish between source and target regions, starting from the localization mask of any copy-move forgery detection technique.

3.1 Datasets description

We now provide a comprehensive list of the benchmark datasets used by a majority of the proposed copy-move, splicing and DeepFake (confined by the previously stated purposes) detection methods. In fact, most of the deep learning methods that are presented in what

follows are trained and/or tested on either one of these datasets, or a custom one built upon the datasets themselves. The main characteristics of each dataset are summarized in Table 1. Evaluation metrics are discussed next in Section 3.1.1.

CASIA v1.0 (CASIA1) [24] It contains 1725 color images with resolution of 384×256 pixels in JPEG format. Of these, 975 images are forged while the rest are original. It contains both copy-move and splicing attacks;

CASIA v2.0 (CASIA2) [24] It contains 7491 authentic and 5123 forged color images with different sizes. The image formats comprise JPEG, BMP, and TIFF. This dataset is more challenging than CASIA1 because the boundary regions of the forged areas are post-processed in order to make the detection more difficult. It contains both copy-move and splicing attacks;

DVMM [101] It is made of 933 authentic and 912 spliced uncompressed grayscale images in BMP format, with fixed size of 128×128;

MICC-F220 [5] It is composed by 110 copy-moved and 110 original JPEG color images. Different kinds of post-processing are also performed on the copied patches, such as rotation, scaling, and noise addition;

MICC-F600 [5] It contains 440 original and 160 tampered color images in JPEG and PNG formats. The tampered images involve multiple copy-moved regions, which are also rotated. The image sizes vary between 722×480 and 800×600 pixels;

MICC-F2000 [5] It consists of 700 copy-moved and 1300 original JPEG images, each one with a resolution of 2048×1536 pixels;

Table 1 Benchmark copy-move/splicing datasets overview

Dataset	Ref.	Manipulations	# orig./forged	Size	Format
CASIA1	[24]	copy-move, splicing	750/975	384×256	JPG
CASIA2	[24]	copy-move, splicing	7491/5123	320×240 – 800×600	JPG, BMP, TIF
DVMM	[101]	splicing	933/912	128×128	BMP (grayscale)
MICC-F220	[5]	copy-move	110/110	480×722 – 1070×800	JPG
MICC-F600	[5]	copy-move	440/160	722×480 – 800×600	JPG, PNG
MICC-F2000	[5]	copy-move	1300/700	2048×1536	JPG
SATs-130	[16]	copy-move	10/120	various	JPG
CMFD	[17]	copy-move	0/48	various	JPG, PNG
CoMoFoD	[100]	copy-move	4800/4800	various	JPG, PNG
DS0-1	[19]	splicing	100/100	2048×1536	PNG
Korus	[49]	copy-move, splicing	220/220	1920×1080	TIF
DFDC	[23]	DeepFake	1131/4113 (videos)	various	MP4
FaceForensic++	[88]	DeepFake	1000/1000 (videos)	various	MP4
Celeb-DF	[59]	DeepFake	590/5639 (videos)	various	MP4

SATs-130 [16] It contains 130 images, generated by 10 source authentic images, with copy-moved regions of different sizes. Various JPEG compression levels are applied, therefore the images are stored in JPEG format;

CMFD [17] It is composed of 48 source images in which a total of 87 regions (referred by the authors as "snippets"), with different sizes and content (from smooth areas, (e.g.), the sky, to rough ones, (e.g.), rocks, to human-made, (e.g.), buildings) are manually selected and copy-moved. The authors also provide a software that allows to apply different post-processing steps on the forged images in a controlled way. The images are given in JPEG and PNG formats;

CoMoFoD [100] This dataset contains 4800 original and 4800 forged images, with copy-move attacks and post-processing operations such as JPEG compression, noise adding, blurring, contrast adjustment, and brightness change. The images are stored in PNG and JPEG formats;

DS0-1 [19] It contains 200 images, 100 of which are pristine and 100 are forged with splicing attacks. All the images are in PNG format at a resolution of 2048 $\times$ 1536 pixels. Color and contrast adjustment operations are applied as counter-forensic measures;

Korus [49, 50] This dataset is composed of 220 pristine and 220 forged RGB images in TIFF format. The dataset contains both copy-move and splicing attacks, performed by hand with professional editing software. The resolution of the images is of 1920 $\times$ 1080.

DFDC (DeepFake detection challenge on kaggle) [23] It contains 4113 DeepFakes videos created from a set of 1131 original ones, involving 66 subjects from various ethnicity and both genders. The video resolution varies from 180p to 2160p. All the videos are in MP4 format and the employed codec is H.264;

FaceForensic++ [88] It is an extension of the previous dataset FaceForensic, with a total of 1.8 millions images created with 4 different DeepFake state-of-art generation methods (*DeepFakes* [27], *Face2Face* [98], *FaceSwap* [51], and *NeuralTexture* [99]), starting from 4000 videos downloaded from *YouTube*. Compared to other previously proposed datasets, it is bigger by at least one order of magnitude. The dataset contains videos of different sizes, such as 480p, 720p, and 1080p. The videos are in MP4 format, and the codec used is again H.264.

Celeb-DF [59] The authors of this dataset specifically created it in order to overcome the lack of realism of a large portion of DeepFake videos in previously published datasets (such as the original FaceForensic). It comprises a total of 5639 DeepFake videos and 590 pristine videos in MPEG4.0 format (H.264 coded), with different resolutions and a standard frame rate of 30 fps. The average length is about 13 seconds (corresponding to a total of more than 2 millions of frames). Another feature that sets this dataset apart from previously proposed ones is how it includes a pronounced variety of ethnicity and equilibrium among genders.

3.1.1 Evaluation metrics

Performance metrics in the considered forgery detection applications are the same used for binary classification problems. There are two classes, authentic or forged, that can be attributed either to the whole image or at the pixel level (through appropriate masks).

Table 2 recaps the terminology for binary classification evaluation using the so-called confusion matrix. Starting from ground-truth classes and the labels output by the detection system, the 4 outcomes given as TP, FP, TN, and FN can be counted according to the concordance or discordance of the labels with the corresponding classes.

The sum of every element in Table 2 is equal to the total number of queries T, namely the population (or the number of objects in the ground-truth). Among these T queries, P have a positive ground-truth class and N have a negative ground-truth class, therefore $T = P + N$. In forgery detection, as in many other binary classification problems, each element in Table 2 is suitably divided by P or N, and thus express the corresponding fraction, or rate, as follows:

$$\begin{aligned} TPR &= TP/P & FPR &= FP/N \\ FNR &= FN/P & TNR &= TN/N \end{aligned} \tag{2}$$

Please note that in some papers the R (rate) part can be omitted, however, there is no possible confusion as the given number is in the [0,1] interval. Given the outcomes in Table 2 and the rates in (2), additional metrics can be obtained as follows:

$$\begin{aligned} \text{precision} &= \frac{TP}{TP + FP} \\ \text{recall} &= TPR = \frac{TP}{TP + FN} \\ \text{F}_1\text{score} &= 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \\ \text{accuracy} &= \frac{TP + TN}{T} \end{aligned} \tag{3}$$

An additional metric is the AUC (Area Under the ROC curve). The AUC is the two-dimensional area under the whole Receiver Operating Characteristic (ROC) curve, that plots FPR versus TPR varying the decision threshold of the detection algorithm.

These measures, or slight variations thereof, are extensively used in the papers described in what follows. There are commonly used synonyms for some of them, for example, the false alarm rate or fallout is the same as FPR and sensitivity is a synonym for recall. Such occurrences have been adjusted for clarity's sake.

Table 2 Confusion matrix and outcomes

		Ground-truth classes	
		Positive (P)	Negative (N)
Output labels	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative(TN)

3.2 Copy-move specific methods

According to the grouping and sorting criteria of the DL-based techniques discussed in this work, we begin in this Section by introducing copy-move only forgery detection methods.

3.2.1 R. Agarwal et al. [4]

The authors of [4] proposed a method specific for copy-move detection that uses deep learning in conjunction with a segmentation step and further feature extraction phases. First, the $M \times N$ input image is segmented with the Simple Linear Iterative Clustering (SLIC) procedure [2]. In order to do so, a 5-D feature vector is built for each pixel, by concatenating its RGB color values and spatial x, y coordinates. A clustering is then performed on these features, and the segmented patches (referred to as "super pixels") are given as output.

Then, multi-scale features are extracted from each super-pixel S_k with a VGGNet [95] network. This process involves the following steps:

– Given the segmented image, a binary mask BM for each super-pixel is obtained as:

$$BM_k(i, j) = \begin{cases} 1 & \text{if pixel } (i, j) \in S_k, \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

– Let $f \in \mathbb{R}^{M' \times N' \times D}$ be the output of the first convolutional layer, where M', N' are the spatial dimensions, and D is the number of output channels. $RF(l, m)$ denotes the receptive field on the input image in the (l, m) position. A continuous value mask $MConv_k \in \mathbb{R}^{M' \times N'}$ is then computed as follows:

$$MConv_k(l, m) = \frac{1}{|RF(l, m)|} \sum_{(u,v) \in RF(l,m)} BM_k(u, v). \tag{5}$$

The super-pixel-level feature map g_k is obtained by multiplying the output of the convolutional layer with the mask:

$$g_k(l, m, c) = f(l, m, c) \cdot MConv_k(l, m), \quad c = 1, .., D \tag{6}$$

– The previous steps are repeated for each convolutional stage of the VGGnet. By using Max-pooling after each convolutional layer, increasingly high-level features are extracted for each super-pixel (see Fig. 2).

Next, a "relocation" phase of the higher-levels features (with lower spatial resolution) is employed in order to find a pixel-level position of the features themselves in the input image. In this way, a set of key-points, with the corresponding multi-level features, is obtained for each patch.

Finally, a key-points matching phase is performed by comparing their associated features, and the copy-moved patches are identified by a further comparison of the super-pixels to which the key-points belong. This procedure is referred to as ADM (Adaptive Patch Matching) by the authors.

The VGGnet is trained on the MICC-F220 dataset. The same dataset is used for testing, though it is not specified which portion of it is used for training and which one is used for testing. The metrics used for evaluation are TNR, FNR, FPR, precision, TPR (recall), and accuracy. The reported results are:

– TNR: 97.1%;
– FNR: 9.2%;

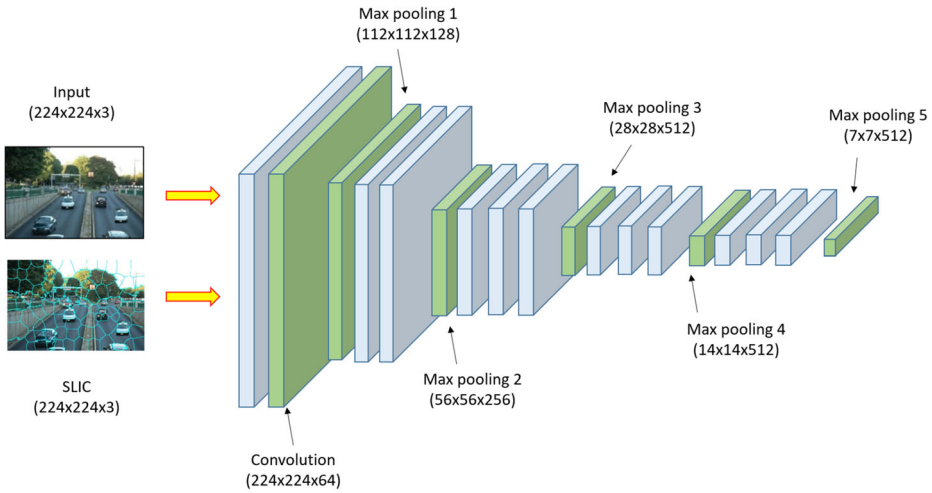

Fig. 2 In [4] the super-pixel segmentation map is given, along with the target image, as input to a VGGNet. Features at different levels are extracted for each of the input super-pixels. Finally, high-level features undergo a so called "relocation phase" to obtain a localization mask at the original resolution

- FPR: 55%;
- Precision: 98%;
- TPR: 89%;
- Accuracy: 95%.

Therefore, the reported accuracy of the method is high, but at the cost of a large number of false positives.

Also, it should be noted that the reported performance is relative to the MICC-F220 dataset, that only has 220 images, with a limited number of types of copy-move attacks. For these reasons, results obtained on just this dataset are not as statistically relevant as methods tested on other, more populated copy-move datasets, such as MICC-F2000 or CoMoFoD.

3.2.2 Y. Abdalla et al. [1]

The authors of [1] proposed a 3-branches method for copy-move detection. An overview of the considered architecture is shown in Fig. 3, which is in the end based on a GAN model. To recap, the GAN is composed of two different deep learning modules: the Generator (G) and the Discriminator (D).

- The generator is a Unet that takes as input an image I and gives as output a forged version of the image itself $I' = G(I)$;
- The discriminator is a CNN network that takes as input either an original image I or a generated image $I' = G(I)$. The output is a binary mask, in which each pixel is labelled as either authentic or forged.

The purpose of D is to discriminate between original pixels and pixels that were manipulated by G. Instead, G aims to generate forgeries $I' = G(I)$, with $I' \simeq I$, in order to fool the discriminator into wrongly classify the forged areas of I' as authentic. The training of the two modules can be seen as a competitive game between them, at the end of which the

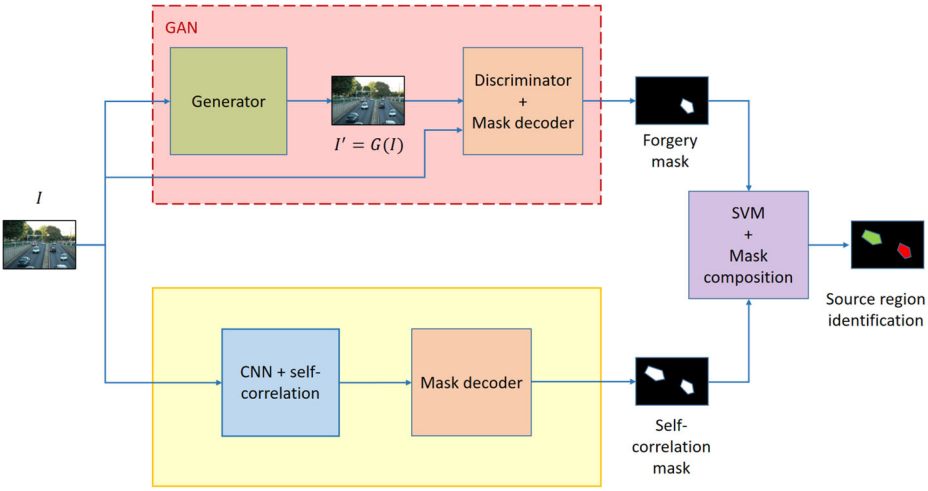

Fig. 3 Architecture of the GAN-based method in [1]. The upper branch implements a per-pixel binary classificator (forged/pristine), while the bottom one is used to find similarities between regions. The outputs of these branches are then combined to obtain the final output mask in which, if the image is considered forged, source and target regions can be distinguished

generator is able to create forgeries that are difficult to detect, and the discriminator is able to correctly classify them.

In addition to the described GAN network, the authors used a custom CNN model specifically designed to detect similarities between regions i.e. copy-moved areas). This CNN is composed of different convolutional layers as well as custom ones that perform a self-correlation operation on the input features. Then, different pooling steps are used to extract more compact features that are fed to fully connected layers. Finally, a mask-decoder layer is used to reconstruct, from the extracted features, a binary mask that represents the similar regions in the image.

As a final decision step in the forgery detection pipeline, a linear SVM model is used for classification. The SVM is fed with an input vector that combines the output of the GAN and the output of the similarity detection CNN. If the image is classified as copy-moved by the SVM model, an additional mask is given as output by comparing the two input binary masks obtained by the GAN and the custom CNN, in which not only the forged areas are labelled, but also the source region used for the copy-move attack is identified (with a different label).

Two datasets unrelated to forgery detection, namely, the CIFAR-10 [52] and MNIST [55] datasets, were used to pre-train and test the GAN network. In detail, the CIFAR-10 dataset contains 60,000, 32×32 color images categorized as 10 distinct classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck), while MNIST is composed of 60,000 grayscale images depicting handwritten digits. After the pre-training phase, the other two modules of the detection pipeline were trained and validated on a custom dataset composed of a total of 1792 pairs of forged and corresponding authentic images, sampled from MICC-F600 and two other datasets, the "Oxford Building Dataset" [80] and the "IM" [12].

The obtained detection performances on this composite dataset are as follows:

– F_1-score: 88.35%;
– Precision: 69.63%;

– Recall: 80.42%;

In conclusion, it would have been interesting if the authors evaluated the performances of their method on one of the public benchmark datasets (such as MICC-F2000, or CASIA2) rather than a custom, composite one. One aspect of this method that should be further noted is that it is one of the few that gives as output not only a localization of the forged areas, but also the source regions of the copy-move attacks.

3.2.3 Y. Wu et al. [105]

In this paper, a pure end-to-end deep neural network pipeline (referred to as *BusterNet* by the authors) is presented as a copy-move forgery detection solution. A key aspect of this method, such as in [1], is the fact that it is able not just to give a pixel-level localization of the copy-move attacks, but it also distinguishes between the source and the target region.

The detection pipeline is composed of two branches and a fusion module (see Fig. 4):

– The first branch, called *Mani-det*, is responsible for the detection of manipulations in the image, and it is composed of the following modules: a feature extractor, a mask decoder, and a binary classifier. The feature extractor is a standard CNN that coincides with the first 4 blocks of the VGG16 network [95].

  The mask decoder is used in order to restore the input resolution of the image, via a de-convolution process, and it uses the *BN-inception* and *BilinearUpPool2D* layers [104].

  The binary classifier, which is implemented as a convolutional layer followed by a sigmoid activation function, produces a binary manipulation mask, in which the pasted patches of the copy-move attacks are localized;
– The second branch, referred to as *Simi-det*, is used in order to generate a copy-move binary mask, in which similar regions in the input image are detected. In particular, the

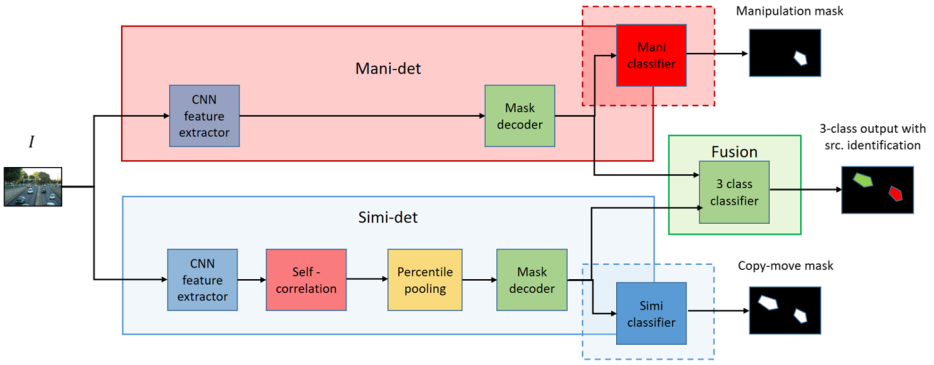

Fig. 4 Architecture of *BusterNet*. [105]. *Mani-det* branch is used to obtain a classification of each pixel of the input image as forged or pristine. *Simi-det* branch instead, aims to find similarities between pixels in the input image. Finally, a fusion module is employed that takes as input the outputs of the two branches and outputs a classification for each pixel: source, target or pristine

detection process can be summarized as follows: first, a CNN is used as feature extractor. Then, a self-correlation module is used to compute all-to-all feature similarities. These are given, as input, to a percentile pooling unit, which collects useful statistics. A mask decoder is used to up-sample the obtained tensor to the size of the input image. Finally, a binary classifier is applied in order to obtain the copy-move mask;
– The fusion unit takes as input the computed features from the two branches. It is constituted by a convolutional layer followed by a soft-max activation, that gives as output a three-class prediction mask: pristine, source region, and target region.

Note that the CNN networks used in the *Simi-det* and in the *Mani-det* branches have the same architecture, but they have different weights, since they are trained independently. The same applies for the mask-decoder and the binary classification modules.

In order to train their model, the authors built a dataset of 100,000 images by automatically performing copy-move operations from source pristine images. For each tampered image, they built three ground-truth pixel-level masks:

– A three-class mask $M_{s,t}$ with the following labels: pristine, source copy-move, and target copy-move;
– A binary mask M_{man} with the following labels: pristine and manipulated. Note that the source region here is considered pristine;
– A binary mask M_{sim} with the following labels: pristine and copy-move. Note that the source and target regions are both labeled as copy-move.

The authors adopted the following three-stage strategy for training:

1. Each branch is trained independently. In order to do so, the copy-move mask M_{sim} and the manipulation mask M_{man} are used, as ground-truth, for the *Simi-det* and *Mani-det* branches, respectively;
2. The weights of each branch are frozen and the fusion module is trained with the three-class mask $M_{s,t}$ as ground-truth;
3. A fine-tuning step is performed by un-freezing the weights of the two branches and training the whole network end-to-end.

The performances of the method were evaluated on CASIA2. As CASIA2 contains both copy-move and splicing attacks, the authors selected a total of 1313 copy-move-only images along with their authentic counterparts, thus obtaining a test-set of 2626 images. The authors used the following metrics: precision, recall, and F_1 score, and they computed them both at image level and at pixel level. For the latter, the authors used two different approaches: (i) aggregate TPR, FPR, and FNR over the whole dataset, and (ii) compute precision, recall, and the F_1 score for each image and then average the results over all of them. The obtained results are reported in Table 3.

Table 3 Performances of *BusterNet* [105] on CASIA2

Eval. method	Prec. %	Recall %	F_1 %
Image level	78.22	73.89	75.98
Pixel level (i)	77.38	59.15	67.05
Pixel level (ii)	55.71	43.83	45.56

3.2.4 M. Elaskily et al. [25]

In [25], a method for copy-move forgery detection is presented. It is purely DL-based, that is, no separate features are pre-computed. In detail, the authors built a CNN with the following architecture:

- Six convolutional layers, each one followed by a max pooling layer;
- A Global Average Pooling (GAP) layer, used to reduce the number of parameters of the network and to limit the probability of overfitting. This layer acts as a fully-connected dense layer;
- A soft-max classification with two classes: authentic or forged.

Therefore, the method does not give as output the localization of the forged regions, but only a global classification of the image. It has been evaluated on 4 benchmark datasets: MICC-F220, MICC-F600, MICC-F2000, and SATs-130. Since each of the listed dataset is quite too small to train a CNN, the authors merged them into a new one that could be more suitable for the training phase. The obtained dataset is thus composed of 2916 images: 1010 tampered and 1906 original.

The authors used the following metrics in order to evaluate the performance of the method: accuracy, TPR, TNR, FNR, and FPR. The metrics were evaluated by a k-fold (with $k = 10\%$) cross-validation. To elaborate, for each validation a random split of the composed dataset is performed: 90% for training and 10% for testing. Here, the 10% testing images is selected all from one of the 4 constituting sets of the composed dataset.

The obtained metrics are presented in Table 4, and they are actually really high. However, we observe that the testing was performed on a small percentage (10%) of the composed dataset, which contains images from all the 4 benchmark datasets themselves. As a consequence, test and training images are possibly highly correlated. Hence, they likely have similar kind of forgeries, that is, with similar dimensions and types of post-processing operations. It could have been interesting if the authors trained their model on one dataset, like MICC-F2000, and evaluated it on another one, such as MICC-F600, in order to better assess the robustness and generalization capability of the model.

3.2.5 J. Ouyang et al. [78]

The method presented in [78] is an end-to-end deep learning approach that features a CNN for binary classification (forged vs. authentic) of the whole image. The crucial aspect of this approach is the use of the transfer learning technique, as follows:

1. A CNN with the same architecture as AlexNet [53] is used as base-model;
2. The classification layer is changed in order to have two classes as output: authentic or forged;

Table 4 Performance metrics of [25]

Dataset	TPR %	TNR %	FPR %	FNR %
MICC-F220	100	100	0	0
MICC-F600	100	100	0	0
MICC-F2000	99.24	100	0	0.76

3. The weights of the AlexNet model trained on the ImageNet dataset [20] are used as initial weights for the training step;
4. A first training phase is carried out by freezing the weight values of the first levels of the network;
5. A second training phase (which is often referred to as "fine tuning") is performed by de-freezing all the network weights, and by using a smaller learning-rate value than the one used in the first training step (such as 10^{-5}).

Since, as already mentioned before, these public forgery detection datasets are not extensive enough for training a CNN without introducing overfitting issues, the authors artificially created copy-move operations by randomly selecting rectangles from an image and pasting them in different locations on the same image. By adopting this approach, they built the following datasets:

– "data1", that contains (i) all the 1338 color images from the UCID dataset [91], and (ii) a total of 10,000 forgeries obtained by applying the above discussed copy-move operations to the original images;
– "data2", that contains (i) all the 8189 color images from the Oxford flower dataset [75], and, again, (ii) a total of 10,000 forgeries obtained with copy-move operations on the original images.

The training of the network was done on both the "data1" and "data2" datasets. Data-augmentation with flipping and cropping operations was performed on the authentic images in order to balance the distribution of the two classes.

For the model performance evaluation, the "data1", "data2", and CMFD datasets were used. The obtained results are reported in terms of test detection error (which is the measure complementary to accuracy). They are as follows: 2.32%, 2.43% and 42% for "data1", "data2" and CMFD, respectively.

From these results it is clear that, even if the model performs well on the custom datasets, it has poor generalization capabilities for real-scenario forgeries, such as the ones contained in CMFD, likely due to its basic approach in generating forgeries. However, this simple approach could still be useful if richer copy-move datasets were available, or a more sophisticated algorithm could be used to build synthetic forgeries, such as a GAN network (see Section 3.2.2).

3.2.6 Amit Doegara et al. [22]

The authors of [22] proposed a simple yet effective method for copy-move detection.

A pre-trained AlexNet model [53] on MICC-F220 dataset is used to extract deep feature vectors of 4096 elements from the input images (note that, in order to obtain the feature vector, the classification layer of the AlexNet network is removed).

An SVM model is then fed with the extracted features and used to obtain a binary classification on the whole image: either pristine or forged.

The training process is carried out in two phases (see Fig. 5). First, the pretrained AlexNet CNN is used to extract features both from the pristine images and from the forged ones. As a pre-processing step, the images are resized to match the input dimension required by the AlexNet model, which is 227 × 227 pixels. Then, the SVM classifier is trained on the obtained dataset of features and corresponding binary labels.

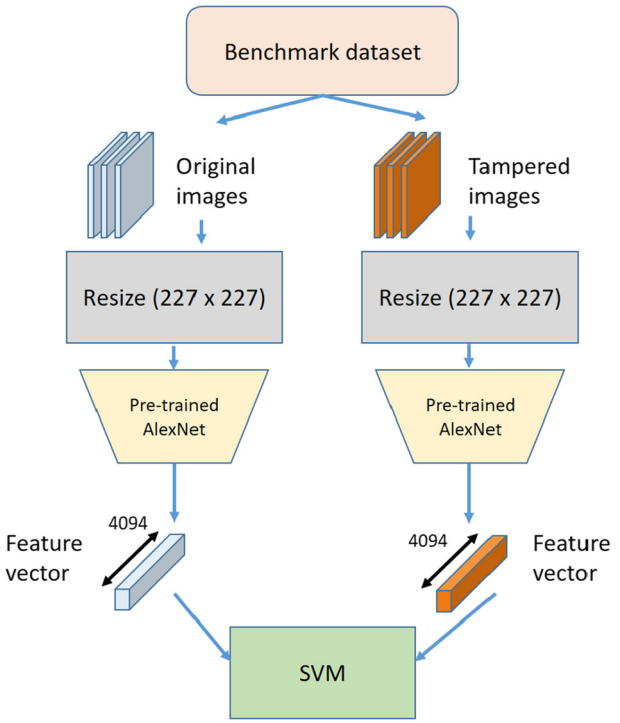

Fig. 5 Detection approach of [22]. A pre-trained AlexNet is used as feature extractor. The extracted features, either from pristine or forged images, are then used to train a SVM classifier to obtain the final decision on the input image: forged VS pristine

The authors evaluated their method on the MICC-F220 dataset, and it obtained the following results:

- FPR: 12.12%;
- TPR: 100%;
- Precision: 89.19%;
- Accuracy: 93.94%.

Even if the accuracy is quite high, there is still room for improvement as the number of false positives is not really low, especially if compared with other approaches, such as [5], in which the reported FPR ratio was of 8%, along with a TPR of 100%.

A final note on the choice of MICC-F220 dataset for performance evalutation is in order. This dataset is also used for pre-training the AlexNet model used by the authors. In the paper, it is not clear which portion of the dataset is used for training and which for testing. Therefore, it is not possible to evaluate if and how much the reported results are affected by bias due to correlation between training and testing sets. In order to clear up these issues, the authors could have used different datasets for either phase instead, such as MICC-F2000 or MICC-F600.

3.3 Copy-move and splicing methods

We now move on to discuss those methods designed to detect both copy-move and splicing forgeries.

3.3.1 Cozzolino and Verdoliva [18]

In this work, the authors presented a deep learning approach that aims to extract a camera model noise pattern (referred to as "noise print") as a means to detect forgeries.

A digital camera, due to the on-board processing operations carried out on the signal received from the sensor, leaves on the generated picture a distinctive pattern of artifacts that are model-specific. This can be exploited, in a forensic scenario, to estimate from which camera model a certain picture was taken from. This idea can also be applied for the purpose of forgery detection. For instance, in the case of a spliced image, if the patch used to create the composition was extracted from a photo taken by a different camera model, then inconsistencies between the camera model artifacts could be leveraged in order to detect the tampering.

A useful property of the camera noise pattern is that it is not space invariant. This means that two patches extracted at different locations from the same image are characterized by different noise artifacts. By exploiting this property, this method can also be used for copy-move detection, as the camera noise pattern at the target location of the copy-move attack is hardly consistent with the expected one at that particular location. The authors used the pre-trained denoising CNN presented in [108] as the starting point for their approach. This network was trained with a great number of paired input-output patches, where the input is a noisy image and the output is its corresponding noise pattern.

In order to estimate the camera model noise print, a further training of the previous architecture was performed. Since a mathematical model describing the camera noise pattern is not available, it is not easy to build a dataset with pairs of an input image and its corresponding desired camera noise print. In order to overcome this problem, the authors used the following key idea: patches extracted from images taken with the same camera model, and at the same location, should share similar camera noise print, while this should not be true for patches coming from different camera models or from different spatial locations. Following this insight, the authors built a Siamese architecture, in which two identical Residual CNNs (initialized with the optimal weights computed in the first training phase) are coupled and the prediction of one network is used as desired output for the other one and vice-versa. The overall architecture is shown in Fig. 6.

In the training phase, the two CNNs are fed with patches x_i^a and x_i^b, respectively. These patches can be:

1. extracted from images taken from different camera models;
2. extracted from images taken from the same camera model, but at different spatial locations;
3. extracted from images taken from the same camera model, at the same location.

The input pair (x_i^a, x_i^b) is assigned, as expected output, a positive label $y_i = +1$ ("similar camera noise print") in the third case, while a negative label $y_i = -1$ ("different camera noise print") in the first and second cases. The output of the Siamese architecture is obtained by means of a binary classification layer that takes as input the noise print extracted by the

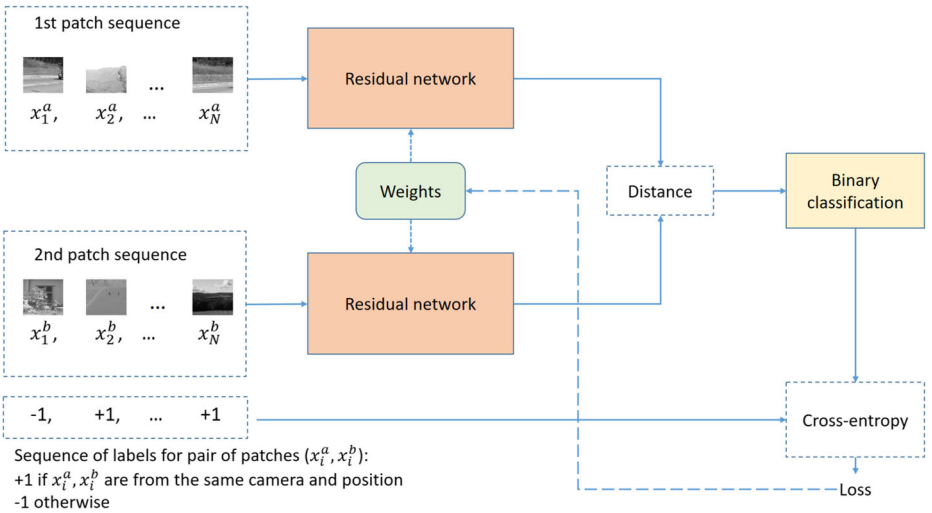

Fig. 6 Architecture of the Siamese network proposed in [18]. Two residual networks (with shared weights) are trained to extract noise patterns that are given as input to a binary classificator. The model learns to extract similar noise patterns for positive labels (patches from same cameras) or different ones for negative labels (patches from different cameras and/or different spatial locations)

two CNNs. This output is then compared to the expected label y_i and the error is back-propagated through the network. This way, the network is pushed towards generating a similar noise print for patches from the same camera model (and at the same location), and different ones for patches corresponding to different camera models and/or locations. As a result, the network learns to enhance the specific model artifacts and discard the irrelevant features, while reducing the high level scene content of the images. Once the network is trained, the noise print can be obtained as output of one of the two CNNs from an input target image.

In order to detect and localize forgeries, the authors used the EM (Expectation - Maximization) algorithm. With the assumption that the pristine and manipulated parts of the target image are characterized by different camera noise models, the algorithm searches for anomalies with respect to the dominant model. This is done by extracting features from the noise print image at a regular sampling grid, that are then used to train the EM algorithm. A heat-map with the probability of manipulation for each pixel is given as output.

The authors tested their method on 9 different datasets for forgery detection, containing many kind of tampering, such as copy-move, splicing, inpaiting, face-swap, GAN generated patches, and so on. Here, we only report the results on the DS0-1 [19] and Korus [49] datasets, as they contain only splicing and copy-move attacks (with possible post-processing operations). The obtained F_1-score is 78% for DS0-1 and 35% on Korus. The authors also computed the AUC score, which is 82.1% and 58.3%, respectively.

3.3.2 Y. Zhang et al. [107]

The authors of this paper proposed the following approach for image forgery detection:

1. Feature extraction and pre-processing. The image is first converted into the YCbCr color space, then it is divided into 32 × 32% overlapping patches. For each component

of the YCbCr space a total of 450 features are extracted from each patch by leveraging the 2-D Daubechies Wavelet transform;

2. The extracted features from each patch are used to train a 3-layers Stacked AutoEncoder (SAE), which is an unsupervised model. On top of the SAE, an additional MLP (Multi-Layer Perceptron) is employed for supervised learning and fine tuning;
3. Context learning. In order to detect forged regions that span across multiple $32 \times 32\%$ patches, each patch-level prediction from the MLP is integrated with the predictions of the neighboring patches. Specifically, for each patch p, a neighbouring patch set $\mathbf{N}(p)$ with cardinality $k+1$ is defined as:

$$\mathbf{N}(p) = [y_p^0, y_p^1, \ldots, y_p^k] \tag{7}$$

where y_p^0 is the output feature of the SAE for the patch p, and y_p^i, with $i \geq 1$ is the feature of its i-th neighbouring patch;

4. Finally, a binary output $Z(p)$ (forged/authentic) is obtained by computing the average of the MLP predictions of the neighbouring patches and comparing it to a threshold, as follows:

$$Z(p) = \begin{cases} 1 & \text{if } \frac{1}{k+1} \sum_{y_p^i \in \mathbf{N}(p)} \text{MLP}(y_p^i) >= \alpha \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where the authors set $k = 3$ and $\alpha = 0.5$.

For the training and testing stages of the model, a total of 1000 images were randomly extracted both from the CASIA1 and the CASIA2 datasets. In particular, 770 images were used for training and the remaining 230 for testing. The authors manually built a pixel-wise ground-truth mask for each image in order to train their model at the patch level. Likewise, a patch-level ground-truth mask for each of the test image was also built, as shown in Fig. 7.

In order to evaluate the performance, the authors used the following metrics: accuracy, FPR (fallout), and precision, where the usual rates are again defined at patch-level. The method can be applied for copy-move detection, as well as splicing detection. Note that this method gives a coarse localization of the forged areas (at patch-level).

The reported performance is 43.1%, 57.67% and 91.09% for fallout, precision, and accuracy metrics, respectively. Even if these performance are not quite satisfactory at a first glance, it should be considered that these metrics are evaluated at patch level, and hence are most restrictive than the the same metrics evaluated at image level.

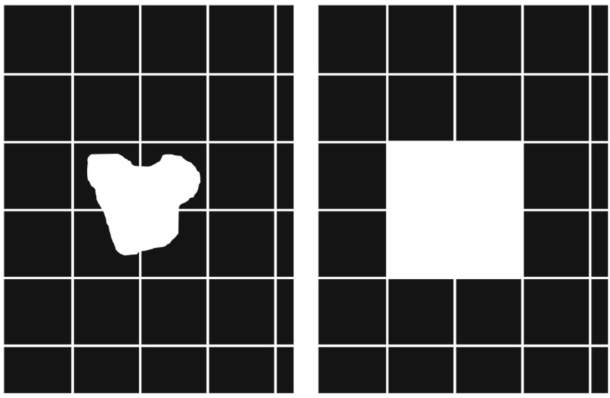

Fig. 7 Construction of patch-wise ground-truth from the pixel-level mask as in [107]

3.3.3 N. H. Rajini [85]

This technique involves two separate CNN models that are used for different purposes in the forgery detection pipeline. It is able to detect both splicing and copy-move attacks. A schematic view of the method is shown in Fig. 8, and it can be summarized as follows:

1. Pre-processing stage. The image is first converted into the YCbCr space. Then, a Block DCT is applied on each Y, Cb, and Cr component. In order to reduce the effect of the actual image content, horizontal and vertical de-correlation is computed from the DCT coefficients. Finally, a set of features are extracted from these values by means of a Markov Random Chain model;
2. Forged/authentic decision. The extracted features are given as input to the first CNN model, which gives a binary classification of the image as either forged or authentic;
3. Type of attack recognition. In the case that the image is recognized as forged, a second CNN is then employed to classify the type of attack: copy-move or splicing;
4. Post-processing. If a copy-move attack is detected by the second network, further features are extracted and used in order to localize the forged regions.

The authors evaluated their method on the CASIA2 dataset. In particular, they used 80% of the images for training and the remaining 20% for testing. The procedure was repeated 50 times with differently extracted training and testing sets, and the reported performance were computed as an average between all the experiments. The TPR, TNR, and accuracy are used as evaluation metrics.

Although the described method can provide as output the localization of the forged areas, the authors only reported performance at a global level (that is, the forged *vs.* non forged image assessment). The obtained results are the following:

– 98.91%, 99.16%, and 99.03% for TPR, TNR, and accuracy, respectively, in the case of copy-move attacks;

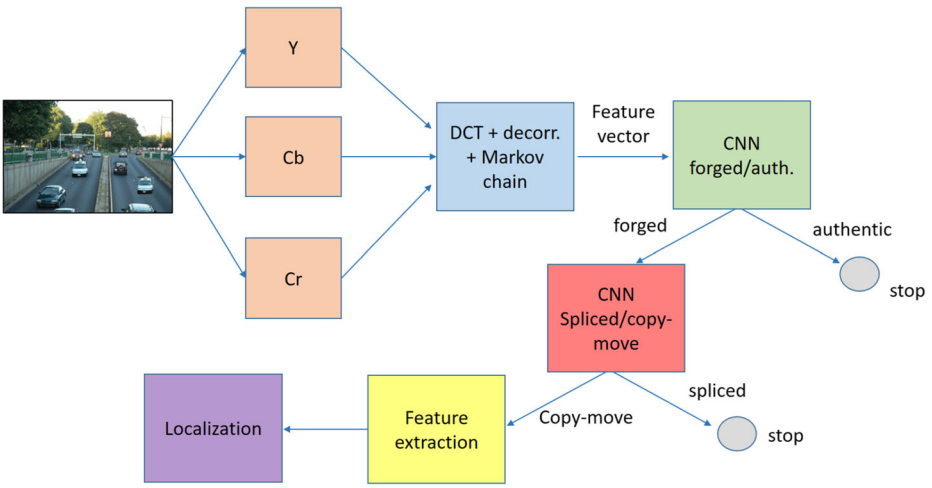

Fig. 8 Multi-step strategy proposed in [85]. First, features are extracted from the YCbCr converted image to classify the image as authentic or forged. If the image is classified as forged, a CNN is used to distinguish between copy-move and splicing attacks. Finally, in the case of copy-move attack, another feature extraction and localization procedure is employed to obtain a map of the forged regions

– 98.98%, 99.24%, and 99.11% for TPR, TNR, and accuracy, respectively, in the case of splicing attacks.

The reported performance metrics are really high. In addition, they are meaningful from a statistically point of view, as they are evaluated on the sizable CASIA2 dataset. It would have been interesting, though, if the authors evaluated the localization accuracy of their method too, in a similar manner to [107].

3.3.4 F.Marra et al. [69]

The authors proposed a full-resolution, end-to-end deep learning framework for forgery detection.

Typically, due to limited memory resources, deep learning models, such as CNNs, are designed to take as input images with small sizes. So, in order to process high resolution images, either a resize to match the network input size or a patch-level analysis (with possible overlapping) is needed. For computer-vision tasks in which only a high level understanding of the image content is required, such as object recognition, this is usually not an issue. But, for the purpose of forensic analysis, resizing is not recommended, as it tends to destroy important information that is usually stored at high frequencies. Patch-level analysis can also be a limiting factor, as usually the context of the whole image is important as well for forgery detection purposes.

In order to address these problems, the authors built a deep learning framework that takes as input full-resolution images and perform image-level predictions: "forged" or "pristine". The framework is composed of three consecutive blocks:

1. Patch-level feature extraction. This is a CNN that takes as input a patch extracted from the target image and gives as output a feature vector;
2. Future aggregation module. This block takes as input the extracted feature vectors from the overlapping patches and aggregate them together in order to obtain an image-level feature. The authors considered different methods for feature aggregation, such as average pooling, min/max pooling, and average square pooling;
3. Decision step. It is a binary classification process, that was implemented with two fully-connected layers.

The whole framework is trained end-to-end. This is not the case for other similar approaches, in which the patch feature extractor, the feature aggregation module, and the classification layers are trained independently one from the others.

Note that, when an input large size image is processed during training, a great amount of memory is required to simultaneously store all the overlapping patches and to compute their corresponding feature vectors. Also, in the forward pass, the activations in all the intermediate layers need to be memorized for the computation of the loss gradients (needed to update the network weights) in the subsequent back-propagation pass. In order to solve this issue, the authors exploited the gradient check-pointing strategy [13]. This technique consists in saving the activations only at certain check-point layers during the forward pass. In the back-propagation phase, the activations are re-computed up to the next check-point layer and used to compute the gradients. As a consequence, less memory is required at the cost of an increased computational time during the back-propagation.

The authors evaluated their method on the DSO-1 and Korus datasets, obtaining an AUC score of 82.4% and 65.5%, respectively.

3.3.5 Y. Rao et al. [86]

An overview of the architecture of this method is shown in Fig. 9. It starts by taking an input RGB image of size $M \times N$ and dividing it into $p \times p$, $p = 128$, overlapping patches $X_i, i = 1, \ldots, T$, where T is the total number of patches. Each patch X_i is given as input to a 10-layer CNN that gives a softmax binary output Y_i, as follows:

$$Y_i = f(X_i) \in \mathbb{R}^2 \tag{9}$$

The Y_i vector represents a compact feature that describes the patch i. A global feature vector is then obtained by concatenating the Y_i of each image patch:

$$\mathbf{Y} = [Y_1 ... Y_T] \in \mathbb{R}^{T \times 2} \tag{10}$$

A mean or max function is then applied for each of the 2 dimensions:

$$\hat{Y}(k) = \text{Mean/Max}\{Y_1(k) ... Y_T(k)\}, k = 1, 2 \tag{11}$$

Finally, $\hat{Y}$ is given as input to a SVM classifier that performs a global two-class prediction on the whole image: authentic vs. forged.

A key aspect of this technique is the following: in order to suppress the image perceptual content and instead focus the detection phases on the subtle artefacts introduced by the tampering operations, the authors initialized the first CNN layer weights with a set of high-pass filters that are used for residual maps computation in SRM (Spatial Rich Models). This step also has the benefit of speeding up the training phase of the network.

The CNN was trained on the CASIA1, CASIA2, and DVMM datasets. This method can be applied both for splicing and copy-move detection, because the CNN and the SVM are trained on the aforementioned datasets, which contain both type of forgeries. Note that the SVM classification step is only used for the CASIA datasets.

The detection performance, in terms of accuracy, is 98.04%, 97.83%, 96.38% on CASIA1, CASIA2, and DVMM datasets, respectively. These accuracy values are objectively high. This is true in particular in the case of CASIA2, which is the dataset with not

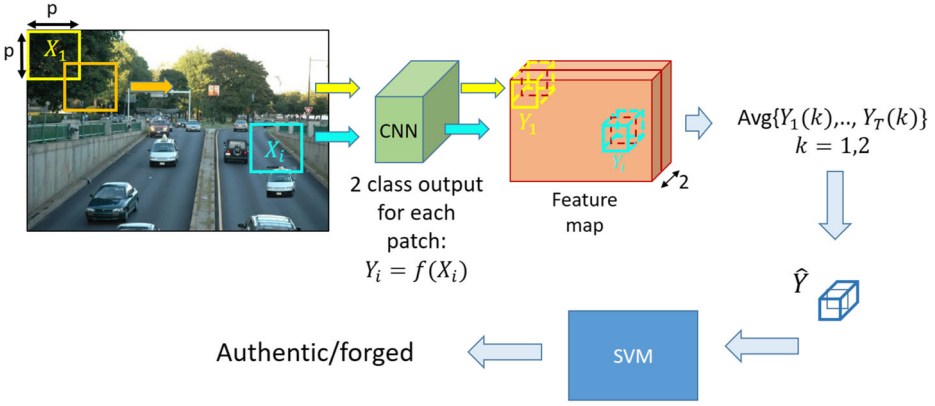

Fig. 9 Architecture of the technique in [86]. Overlapping patches are extracted from the input image and feature vectors are extracted from each of them. A global feature, computed by averaging along the spatial dimension, is then fed to an SVM model, which is used to obtain the final global classification: forged VS authentic

only the most images (and consequently it is the most statistical relevant, as we said before), but it also contains both splicing and copy-move attacks. It should be noted, though, that this method only gives a global binary prediction on the image, and no localization of the forged areas is performed.

3.3.6 M. T. H. Majumder et al. [68]

The approach described in [68] is also based on a CNN to classify an image as authentic or forged. In contrast to the previously discussed methods, however, in which deep learning networks were composed of a high number of layers, in this case a shallow CNN model, composed of just two convolutional layers, was employed. Also, no max-pooling steps were used for dimensionality reduction, as this goal was achieved by exploiting large convolutional filters, with size of 32 by 32 and 20 by 12 for the first and the second layer, respectively.

This strategy is based on the following idea: in deep neural networks, complex high-level features are learnt at deeper levels, while more simple visual structures, such as edges and corners, are learnt at the first ones. Hence, in order to detect the artefacts introduced by forgery operations, low-level features are more likely to be useful. As a consequence of this choice, the number of parameters of the network is limited, thus allowing for training with less over-fitting risk.

The CASIA2 dataset was used both for training and testing. The authors trained their shallow network multiple times in an independent fashion, using different pre-processing strategies, such as: raw input (that is, no pre-processing), DCT-based transformation, and YCbCr space conversion. They showed that the best results were obtained without any kind of pre-processing.

To further reduce the risk of overfitting, real-time data augmentation was applied during training, with transformations such as shearing, zooming, and vertical and horizontal flipping. An accuracy of 79% was obtained with this training strategy, and, as we said, without pre-processing.

As a comparative experiment, the authors also applied the aforementioned transfer learning technique, by using two deep learning models with a high number of layers that were pre-trained on the ImageNet dataset: the VGG-16 [95] and the well-known ResNet-152. Despite the fact that these models perform well on standard image classification problems, they were not able to transfer the acquired knowledge to this specific task, and a substantial underfitting issue was observed in the training phase. The outcome of this test validated the choice of a shallow model instead of a deep one.

The main contribution of this work is therefore the usage of a shallow network, in which low-level features are exploited as a mean to detect subtle artefacts generated by tampering (rather than high-level ones), which thus can be used for the forgery detection task. Also, the authors showed that large convolutional filters can be exploited in place of max-pooling layers to reduce the number of network parameters, therefore reducing the risk of overfitting. Despite this, the obtained accuracy still leaves room for improvement.

3.3.7 R. Thakur et al. [97]

In [97], a filtering scheme based on image residuals is exploited. Therefore, the residuals, rather than the raw images, are fed as input to a CNN network for classification (as usual, original/forged). This approach is tailored to pursue high frequencies in the image data,

which, as often assumed even by the other approaches, carry most of the possible tampering traces. The image residuals are computed as follows:

1. The image is resized at the 128×128 size, and converted to grayscale;
2. The second-order median filter residuals (SDMFR) are then calculated as follows. Given an image, a first median filtering is applied:

$$y(i, j) = \text{med}_w[x(i, j)] \tag{12}$$

where w is a 5×5 window and $x_{i,j}$ is the (i, j) pixel intensity. Then, a second median filtering is applied to the median-filtered image:

$$z(i, j) = \text{med}_w[y(i, j)] \tag{13}$$

Finally, the residuals are obtained by subtracting the second order median filtered image from the first order filtered image:

$$MFR(i, j) = z(i, j) - y(i, j) \tag{14}$$

3. Laplacian filter residuals (LFR) are also computed, with the following algorithm. Let:

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{15}$$

be the Laplacian kernel filter. The Laplacian-filtered image is obtained by convolving the original image with K, that is:

$$L(i, j) = (x * K)(i, j) \tag{16}$$

The residuals are then calculated as the difference between the filtered image and the original one:

$$LFR(i, j) = L(i, j) - x(i, j) \tag{17}$$

Both the SDMFR and the LFR residuals are fed to the CNN classification network as a combined input. The CNN model comprises 6 convolutional layers, each one followed by a max pooling step (except the first one). Two fully connected layers are then used before the final binary softmax classifier.

The authors trained and tested their network on two different datasets: the CoMoFoD and the BOSSBase [8]. In the case of the first dataset, a split of 70% and 30% has been made for training and validation, respectively. In the case of the second one, as it is composed of 10,000 raw pristine images, the authors applied median filtering to each image in order to simulate a tampering operation, thus obtaining a total of 20,000 images (half authentic and half filtered). Then, they split the obtained dataset into 70% for training and 30% for validation.

The accuracy obtained on both datasets is high: 95.97% for the CoMoFoD dataset, and 94.26% for the BOSSBase. However, it could have been interesting if the authors tested their method, without retraining, also on other benchmark datasets for forgery detection, such as CASIA2, MICC-F2000 or MICC-F600, in order to assess its generalization capability.

3.4 DeepFake methods

We now present a few of the most recent DeepFake-specific detection methods, that achieved the best results on the previously introduced datasets for DeepFakes detection evaluation (see Section 3.1). The selection has been made according to the criteria previously outlined, namely, suitability for the still images case.

3.4.1 A. Rössler et al. [88]

In [88], the authors developed a method to detect image DeepFakes that is based upon the *XceptionNet* architecture proposed by *Google* in a previous paper [15]. The main peculiarity of this model is the employment of a custom layer, called *SeparableConv*, whose purpose is to decouple the depth-wise convolution from the spatial one, thus reducing the number of weights of the model itself.

The detection pipeline can be summarized as follows: a state-of-art face detection/tracking method [98] is used to extract the face region from the image/frame, which is cropped as a slightly larger rectangle than the size of the face in order to include some contextual information.

The obtained bounding box is then fed to a modified XceptionNet for binary classification. In order to do this, the final fully-connected layer of the original XceptionNet is substituted with a fully-connected layer with binary output.

The authors adopted the following transfer-learning strategy to train the model:

1. The weights of each layer from the original XceptionNet are initialized with the ImageNet ones, while the fully-connected layer is random initialized;
2. The network is trained for 3 epochs, with all the weights freezed except the ones in the fully-connected layer;
3. All the weights are un-freezed and the network is trained for other 15 epochs (fine-tuning step).

The authors released three different versions of their model: the first one is trained on uncompressed videos, while the second and the third one were trained on videos compressed with H.264 codec at quantization levels of 23 and 40, respectively. We denote these variants as Xception_a, Xception_b, and Xception_c, respectively.

While Xception_a achieved the best results on FaceForensic++ dataset, with a detection accuracy of 99.7%, its performance dropped when evaluated on DFDC and CelebDF, with accuracy scores under 50% in both cases. Xception_b achieved the best accuracy on DFDC (72.2%), while Xception_c performed better on CelebDF, with an accuracy of 65.5%.

3.4.2 Huy H. Nguyen et al. [72]

In this paper [72], a novel forgery detection framework, called *Capsule-Forensic* was proposed. Its main feature is that it uses a particular kind of neural network, *Capsule Network* (first introduced in [37]), as the binary detector, instead of the more usual convolutional neural networks.

Capsule Networks were designed in order to efficiently model hierarchical relationships between objects in an image, and to infer not only the probability of observation of objects, but also their pose estimation.

The main idea behind Capsule Networks is the concept of "capsule". A capsule is an ensemble of neurons that describe a set of properties for a given object. In contrast to single neurons, in which the scalar output represents the probability of observation of a certain feature, the output of a capsule is an activation vector, in which each element represents the activation of one of the capsule's neurons, i.e., the value corresponding to the associated feature.

Capsules are arranged in different layers in a hierarchical fashion: a parent capsules receives, as input, the output of different child capsules. The connections between child and parent capsules i.e., which outputs are kept and which are discarded for the next layer) are

not fixed at the beginning, such as for max/average pooling layers (usually employed in standard CNNs), but they are dynamically computed by means of a routing by agreement algorithm.

Thanks to this procedure, child capsules whose predictions are closest to the predictions of certain parent ones become more and more "attached" to these parents, and a connection can be considered established. The interested reader is referred to the original paper for a more detailed explanation on how the hierarchical connections are built.

Among the advantages of Capsule Networks compared to CNNs, a remarkable fact is that they have less parameters, as neurons are grouped in capsules and the connections between layers are between capsules and not directly between neurons. Also, thanks to the presence of pose matrices, they are robust against viewpoint changes under which objects are seen in the image. This is not true for CNNs, that need to be trained on lots of possible rotations and transformations in order to generalize well to unseen transformations.

The proposed method is designed for different forensics tasks, such as (i) DeepFakes detection, and (ii) computer-generated frame detection, both for image and video content.

The detection pipeline (shown in Fig. 10) comprises the following elements:

- Pre-processing phase. It depends on the specific forensic task at hand, e.g., for DeepFakes detection it involves a face detection algorithm in order to extract the face region, while for CGI detection it consists in patch extraction from the input image. For video content the frames are separated and fed one by one to the subsequent steps;
- Feature extraction. This is done by using the first layers of a VGG_19 network pre-trained on ILSVRC dataset [90]. These weights are fixed during training;
- Capsule Network. It is the core of the detection method, involving three primary capsules (children) and two output capsules "Real" and "Fake" (parents). The predicted label is computed as in (18):

$$\hat{y} = \frac{1}{M} \sum_{i=1}^{M} \text{softmax}\left(\begin{bmatrix} \mathbf{v}^1 \\ \mathbf{v}^2 \end{bmatrix}_{:,i} \right), \tag{18}$$

  where $\mathbf{V}^1 \in R^M$ and $\mathbf{V}^2 \in R^M$ represent the output capsules, and M is their dimension;
- Post-processing phase. As the pre-processing step, this is task-specific: the scores are averaged among patches for computer generated image detection, or among frames for video input.

The achieved detection accuracy is very high on FaceForensic++, with a score of 96.6%, but it is lower on the more challenging datasets DFDC and CelebDF, with accuracies of 53.3% and 57.5%, respectively.

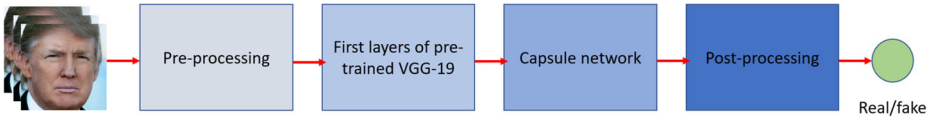

Fig. 10 Overview of method [72]. Note that pre-processing and post-processing stages are task-dependent, e.g. for DeepFake detection in the former a face tracking algorithm is used to extract and normalize the face region, while for CGI detection this step consists in the extraction of overlapping patches

3.4.3 Y. Li et al. [57]

In [57], the authors proposed a deep learning method to detect DeepFakes based on the following observation: typically, DeepFakes generation algorithms tend to leave distinctive artifacts in the face region due to resolution inconsistencies between the source image/video and the target one. In particular, GAN-synthesized face images are usually of a fixed low resolution size and, in order to be applied to the target video, an affine warping needs to be performed in order to match the source face to the facial landmarks of the target face. If the resolutions of the source and target videos do not match, or if the facial landmarks of the target person are far from the standard frontal view, these artifacts are more and more evident.

The authors trained four different CNNs, namely a VGG-16, a ResNet50, a ResNet101, and a ResNet152 to detect these kinds of artifact. In particular, they used a face-tracking algorithm to extract regions of interest containing the face as well as the surrounding area, which are then fed to the networks. The reason why also a portion of the surrounding area is included is to let their model learn the difference between the face area, that contains artifacts in the case of positive (fake) examples, and the surrounding one, which does not contain artifacts.

The authors used the following training strategy. Instead of generating positive examples by means of a GAN-syntesization algorithm, which in turn requires a good amount of time and computational resources to train and run, they generated positive examples by simulating the warping artifacts with standard image processing approaches, starting from negative (real) images. The processing steps are summarized as follows:

1. The face region is extracted with a face tracking algorithm;
2. The face is aligned and multiple scaled versions are created by down/up-sampling the original one. Then, one scale is randomly selected and Gaussian-smoothed. This has the effect of simulating the mismatch in resolutions between source and target videos;
3. The smoothed face is then affine-warped to match the face landmarks of the original face;
4. Further processing can be done in order to augment the training data, such as brightness change, gamma correction, contrast variations, and face shape modifications through face landmarks manipulation.

The detection accuracy obtained are: 93.0% for FaceForensic++, 75.5% for DFDC and 64.6% for CelebDF.

4 Performance comparison

In this Section we proceed to compare the previously described forgery detection methods from a performance perspective.

We begin by comparing techniques specific for copy-move and splicing, while DeepFake detection algorithms are discussed in a separate section. In fact, even if the DeepFake methods that we previously discussed can be seen as a particular kind of splicing attack, they are mostly performed on faces. As a consequence, DeepFake detection techniques must be evaluated with datasets specialized on face manipulations, while the standard splicing datasets, such as CASIA, contain pictures of generic scenes. Furthermore, these methods

can successfully exploit domain specific knowledge, such as face landmarks, mouth/eyes-based features, and so on, while of course this is not the case for generic splicing detection algorithms.

4.1 Splicing and copy-move methods

In Table 5 the performance of all previously discussed copy-move and splicing detection techniques are reported. For each method, we also indicate the type of detected attacks (splicing, copy-move, or both) and the capability or lack thereof to give as output the localization of the forged areas.

As a first comment, from the sparseness of the table it is easy to see that it is very challenging to compare the different techniques strictly in terms of performance. This is due to a number of reasons. The first and most obvious one is that approaches designed specifically for copy-move detection cannot be easily evaluated on CASIA (both v1.0 and v2.0) datasets, as these also contain splicing attacks (an exception can be made for method [105], that was evaluated on a copy-move-only subset of the dataset itself, see Section 3.2.3). In this case, copy-move specific datasets, such as MICC-F220, MICC-F600, and MICC-F2000 should be considered for evaluation.

The second reason is that the presented methods, especially in the case of copy-move specific ones, are mostly not trained nor tested on the same benchmark sets. This is due to the fact that some of the standard datasets are either too small for training a highly parameterized deep learning model, or contain only naive attacks (such as MICC-F220, in which copy-moved regions are square or rectangular patches). For this reason, different authors instead built their own custom datasets to fulfill their specific requirements, either by merging together the benchmark ones or by artificially generating them. However, the downside of this approach is the difficulty of comparing the results achieved by other techniques.

Therefore, the comparison between different techniques, when it is possible, is performed by grouping them on the basis of specific criteria, such as the type of detected attacks, the dataset used for evaluation, and the localization property.

We start by focusing our analysis on the methods designed for copy-move only forgeries, then proceed to both copy-move and splicing detection techniques, and conclude with DeepFake specific ones.

4.1.1 Copy-move detection methods

We start the present analysis by first comparing methods [4, 25], and [22], as they have been all tested on the MICC-F220 dataset. The first method achieved a slightly better accuracy and a considerable better FPR (see Table 4) than the other two, along with a considerably better accuracy. In addition, [25] has been shown to achieve perfect results on MICC-F600 and almost perfect ones on MICC-F2000, which are more significant evaluation datasets (see Section 3.2.4). However, it should be considered that [25] only gives as output a global decision on the authenticity of the image, while [4] also provides the location of the forgery.

Regarding the forgery localization property, it is worth noting that the techniques presented in [1] and [105] allow not only to detect the copy-moved regions, but also to distinguish them from the source patches used to perform the attack. This property is useful in real forensic scenarios, in which it is important to understand the semantic aspects of an image manipulation.

A further interesting feature of [1] is the adoption of a GAN network to generate increasingly hard-to-detect forgeries, that are used to train the discriminator network. This is an

Table 5 Copy-move and splicing detection methods performance comparison. The best results for each dataset are highlighted in bold

Method	Detected attacks	Localization	CASIA1 Acc.%	CASIA2 Acc.%	MICC-F220 Acc.%	**MICC-F600** Acc.%	MICC-F2000 Acc.%	Other perf.
Agarwal and Verma [4]	Copy-move	Yes	–	–	99.11	–	–	55% FPR on MICC-F220
Abdalla et al. [1]	Copy-move	Yes + Source id.	–	–	–	–	–	88.35% F1-score on custom dataset
Wu et al. [105]	Copy-move	Yes + Source id.	–	76.65	–	–	–	75.98% F1-score on CASIA2
Elaskily et al. [25]	Copy-move	No	–	–	**100**	**100**	**99.7**	–
Ouyang et al. [78]	Copy-move	No	–	–	–	–	–	43% det. error on CMFD
Doegar et al. [22]	Copy-move	No	–	–	93.94	–	–	–
Cozzolino and Verdoliva [18]	Splicing Copy-move	Yes	–	–	–[a]	–	–	82.1% AUC on DS0-1 and 58.3% AUC on Korus
Zhang et al. [107]	Splicing Copy-move	Block-wise	91.09[b]	91.09[b]	–	–	–	–
Rajini [85]	Splicing Copy-move	Yes	–	**99.07**[c]	–	–	–	–
Marra et al. [69]	Splicing Copy-move	No	–	–	–	–	–	82.4% AUC on DS0-1 and 65.5% AUC on Korus
Rao and Ni [86]	Splicing Copy-move	No	**98.04**	97.83	–	–	–	96.38% acc. on DVMM
Majumder and Alim Al Islam [68]	Splicing Copy-move	No	–	79	–	–	–	–
Thakur and Rohilla [97]	Splicing Copy-move	No	–	–	–	–	–	95.97% acc. on CoMoFoD

[a]The accuracy score is sub-par compared to the performance obtained on the datasets used by the authors. This was probably due to the fact that MICC-F220 is a dataset of small-sized JPEG compressed images, which are very different from the images used to train the method

[b]The accuracy is computed on a dataset obtained by randomly selecting a total of 1000 images from CASIA1 and CASIA2

[c]This value was obtained as average of the splicing detection accuracy (99.03) and the copy-move detection accuracy (99.11)

original approach to address the problem of data-scarcity that plagues many different existing standard datasets. However, from a performance point of view, it is hard to compare this method to the other ones, as it was evaluated on a custom dataset and not on one of the benchmark datasets. This is not the case for [105], which was evaluated on CASIA2. Note that, even if its accuracy is slightly worse than [68], it has the source plus target localization property mentioned before, while the latter gives as output only a global classification on the image.

4.1.2 Splicing and copy-move detection methods

These techniques fit the best in a general application context, in which the type of attack is not known a priori, so it is better to cover as many attacks as possible. In particular, we consider the methods tested on CASIA2, which is likely the most significant dataset for copy-move and splicing detection evaluation, both for its sheer size and for the various applied post-processing operations.

Among the methods that we discussed, the one presented in [85] obtained the best overall accuracy. It also gives as output the localization of the forged areas, which as we mentioned is of course relevant in many application contexts. Looking at its forgery detection pipeline, it features both a pre-processing stage, in this case based on YCbCr space conversion and DCT compression, as well as a post-processing phase that through further features extraction allows to perform localization. Therefore, the good performance that it achieved indicate that an exclusively end-to-end deep learning model, without any pre-processing or post-processing, could be indeed a sub-optimal choice for the task of forgery detection.

On the same note, another comment can be made about the method in [68]. Even if its performance are worse than the others in terms of accuracy, the proposed approach is quite interesting because it involves a "shallow" deep learning model. This allows reducing not only the number of network parameters (and consequently the training time), but also the risk of over-fitting. This idea is in contrast to the common trend in computer vision to use ever deeper networks to achieve high accuracy on specific datasets, that usually cannot be achieved on slightly different ones, which is a clear indicator of over-fitting issues.

A remark should be made on the approach proposed in [18]. This method has a wide applicability even outside the field of forgery detection. In fact, the possibility to extract the noise camera pattern and suppress the high-level scene content of a target image is of great utility in other forensic scenarios as well as for sophisticated camera-specific denoising applications. It is important to also note that the authors evaluated the performance of their algorithm on different datasets, which contain a wide set of forgery attacks such as copy-move, splicing, inpainting, GAN-synthesized content, face-swap, etc., thus proving its wide applicability and robustness. Still, it would have been interesting to have the detection results on other more classic benchmark data, such as the CASIA2, thus allowing a better comparison with other existing methods.

4.1.3 DeepFake detection methods

In Table 6, the performance of DeepFake detection methods are reported.

As it can be immediately observed from the table, there is not a method that performs better on all three considered benchmark datasets: [57] reports the best accuracy on DFDC, while [88](a) performs better on FaceForensic++, and [88](c) achieved the highest accuracy on Celeb-DF. It must be considered, though, that FaceForensic++ was built by the same authors of [88] (all three versions). As such, it is, to some extent, expected that these are

Table 6 DeepFake detection methods performance comparison. The best results for each dataset are highlighted in bold

Method	DFDC acc.%	FaceForensic++ acc.%	Celeb-DF acc.%
Rössler et al. [88] (a)	49.9	**99.7**	48.2
Rössler et al. [88] (b)	72.2	**99.7**	65.3
Rössler et al. [88] (c)	69.7	95.5	**65.5**
Nguyen et al. [72]	53.3	96.6	57.5
Li and Lyu [57]	**75.5**	93.0	64.6

the methods that perform better on that particular dataset. Nonetheless, [88](c) still has the best results on Celeb-DF, while [88](b) has only slightly worse performance than [57] on DFDC, thus showing how the XceptionNet-based strategy can be the to-go choice for its generalization capability on different datasets.

Finally, we observe that, when evaluated against challenging and realistic datasets such as Celeb-DF, DeepFake detection methods still need to be improved, as the best accuracy obtained is just around 65%. This allows us to infer that the research field of DeepFake detection is still lagging behind, especially considering the fact that DeepFake generation algorithms are still largely improving year after year.

5 Conclusions

In this work we provide a survey of some of the recent AI-powered methods (from 2016 onward) for copy-move and splicing detection that achieve the best results in terms of accuracy on the standard benchmark datasets. Several reviews and surveys have been published on this topic, but most concerned mainly traditional approaches like those based on key-points/blocks, segmentation, or physical properties. Instead, we focused our analysis on recently published, deep learning based methods, because they have been shown to be more effective in terms of performance and generalization capability than the traditional approaches. As a matter of fact, they are able to achieve really high accuracy scores on the benchmark datasets.

We separated the performance analysis between copy-move only, both copy-move and splicing, and DeepFake detection methods. In the case of copy-move only detection, the method in [25] shows an almost perfect accuracy on all three standard benchmark datasets (MICC-F220, MICC-F600, and MICC-F2000). The technique presented in [4] is able to achieve a similar accuracy, while also giving the identification of both the copied regions and the original ones used as source for the attacks. In the case of both copy-move and splicing detection, similar results were achieved on the CASIA2 dataset. In particular, method [85] shows the best accuracy and gives the localization of the forged regions as well.

Concerning DeepFake detection, from the reported performance (see Table 6) we infer that there is not a clearly winning approach, in particular no method is general enough for different kinds of DeepFake content. However, we can conclude that the XceptionNet-based models proposed in [88] are able to achieve better performance on at least two out of the three considered benchmark datasets.

From a general point of view, it can be easily inferred from the DL-based methods surveyed in this paper that a clear trend has not yet emerged. Most works have been more or

less independently proposed, in the sense that the vast possibilities offered by DL architectures are still being explored, without a clear winning strategy indication. Nonetheless, we showed that, in the case of splicing and copy-move detection methods, the best accuracy scores were obtained by the techniques that involve some form of pre-processing and post-processing in addition to a deep learning network. For this reason, we think that this appears to be the most promising approach, and so we believe that further research should be conducted on algorithms that combine deep learning approaches with traditional techniques from all over the field of (statistical) signal processing.

As a further consideration, it can be noted that in the case of techniques aimed at "classic" forgery detection (splicing and copy-move), most of state-of-art methods are able to achieve good performance (on different datasets). Instead, this is not the case for newer challenges like DeepFake detection, whose methods report accuracy performance which is still not satisfactory on complex datasets, like Celeb-DF. As such, further research efforts and ideas still need to be explored in this particular direction.

Further remarks are in order on the problem of performance evaluation of deep learning based methods. Different authors built custom datasets or merged different ones in order to train and test their algorithms. While this can be a solution to overcome issues of data-scarcity (over-fitting), it makes the comparison with other methods more difficult, or even impossible. Even when the same dataset is used to evaluate different approaches, the authors do not always specify which and how many images were used as testing set.

This problem could be addressed by building a custom dataset for training, and using one or possibly more benchmark datasets in their entirety for testing. In this way, not only it would be possible to easily compare different deep learning based approaches, but also to compare them to traditional, non-learning based ones.

Of course, building a custom dataset with thousands of images, with realistic forgeries and post-processing operations on the forged areas, such as blurring, JPEG-compression, smoothing, and so on, is not a simple undertaking. For this reason, we point out that another possible future research direction could be the automation of this task, for example by leveraging a GAN network (as done in [1]), or encoder-decoder models such as a Unet.

A wholly different comment on the subject of datasets building should also be made on the meaning of the forgery attacks currently contained in the benchmark datasets. As these have always been generated in a laboratory environment (whether manually or not), they typically contain copy-move and splicing attacks that hardly bring a particular semantic value to the altered images. For example, when a tree is copied and pasted in a wood landscape, or a cloud is pasted into a blue sky, the obtained image could hardly be used for malicious purposes. This is clearly not the case for many manipulated images that can be found on the Web. Let us consider for example the splicing shown in Fig. 1b: the fact that the 2004 presidential election candidate John Kerry was (falsely) immortalized together with pacifist actress Jane Fonda, who was viewed by many as an anti-patriotic celebrity, could have seriously influenced the elective campaign (in this case, the image was shown to be false, but not quickly enough to avoid some damage to the candidate's reputation).

Of course, in such real-world cases, the context adds a lot to the meaning of the forgery, and thus it can hardly be taken into account by a forensic tool without a human supervision. Nevertheless, we feel that it could be interesting to build a database that collects more realistic, manually made, "in-the-wild" forgeries, like the ones that routinely spread on social media these last years, and so present potentially malicious attacks from a purely semantic point of view. Also, this database should contain, for each forgery, the associated ground-truth mask, in order to better assess and compare the forgery localization capability of the forensic tools.

We would like to conclude adding a final, more philosophical observation. As is typical in the case of security-related fields, attackers usually embody, in their attacks, ideas and "hacks" that are specifically designed to counterpoise the latest state-of-art detection methods, e.g., so-called *adversarial attacks* [26, 36, 54], which are used to fool deep learning classification systems. For example, a possible strategy to achieve this confusion consists in using a certain CNN architecture as a discriminator in a GAN model, in order to produce synthesized content which is, by construction, hard to be detected as fake by that particular CNN. Another interesting example of this kind involves DeepFake detection: in [58], the authors observed that, in DeepFake videos, it was common to see unnatural eye-blinking pattern (or no blinking at all), because DeepFake generation algorithms were trained mostly on pictures of people with open eyes. As expected, attackers immediately adapted DeepFake methods in order to generate realistic eye-blinking, either by including pictures of people with closed eyes during training, or by synthetically correct this issue altogether.

As a consequence, it is probably an illusion to consider a certain forgery detection method to be "safe" forever, even if it has been shown to achieve great detection accuracy on different datasets. For this reason, we think that continuous research efforts should be made in order to develop methods that can, at least to some extent, keep up with the attackers' pace in developing more and more sophisticated and hard-to-detect forgeries. One possible strategy, that tries to anticipate potential attacker moves, could be to actively implement new forgery techniques while developing detection algorithms, this way understanding and leveraging their flaws and thus to allow the creation of possible counter-measures.

Author Contributions The authors contributed equally to this work.

Funding Open access funding provided by Università degli Studi di Brescia within the CRUI-CARE Agreement. No funding was received to assist with the preparation of this manuscript.

Availability of Data and Material No additional data or material has been used for this work other than the referenced papers.

Code Availability No code has been developed by the authors for this work.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Abdalla Y, Iqbal T, Shehata M (2019) Copy-move forgery detection and localization using a generative adversarial network and convolutional neural-network. Information 10(09):286. https://doi.org/10.3390/info10090286

2. Achanta R, Shaji A, Smith K, Lucchi A, Fua P, Süsstrunk S (2010) Slic superpixels. Technical report, EPFL
3. Adobe Photoshop. https://www.adobe.com/it/products/photoshop.html. Accessed 16 Mar 2022
4. Agarwal R, Verma O (2020) An efficient copy move forgery detection using deep learning feature extraction and matching algorithm. Multimed Tools Appl 79. https://doi.org/10.1007/s11042-019-08495-z
5. Amerini I, Ballan L, Caldelli R, Del Bimbo A, Serra G (2011) A SIFT-based forensic method for copy-move attack detection and transformation recovery. IEEE Trans Inf Forensics Secur:1099–1110. https://doi.org/10.1109/TIFS.2011.2129512
6. Arnold MK, Schmucker M, Wolthusen SD (2003) Techniques and applications of digital watermarking and content protection. Artech House
7. Barni M, Phan QT, Tondi B (2021) Copy move source-target disambiguation through multi-branch cnns. IEEE Trans Inf Forensics Secur 16:1825–1840
8. Bas P, Filler T, Pevnỳ T (2011) Break our steganographic system the ins and outs of organizing BOSS. In: International workshop on information hiding, pp 59–70. https://doi.org/10.1007/978-3-642-24178-9_5
9. Bay H, Ess A, Tuytelaars T, Van Goo L (2008) Speeded-up robust features (surf). Comp Vision Image Underst 110(3):346–359. https://doi.org/10.1016/j.cviu.2007.09.014. Similarity Matching in Computer Vision and Multimedia
10. Blog post on Elcomsoft, April 2011. https://blog.elcomsoft.com/2011/04/nikon-image-authentication-compromised/. Accessed 16 Mar 2022
11. Birajdar GK, Mankar VH (2013) Digital image forgery detection using passive techniques: a survey. Digit Investig 10(3):226–245. https://doi.org/10.1016/j.diin.2013.04.007
12. Cao Z, Gao H, Mangalam K, Cai Q-Z, Vo M, Malik J (2020) Long-term human motion prediction with scene context. In: Vedaldi A, Bischof H, Brox T, Frahm J-M (eds) Computer vision - ECCV, pp 387–404
13. Chen T, Bing X, Zhang C, Guestrin C (2016) Training deep nets with sublinear memory cost
14. Chen J, Liao X, Qin Z (2021) Identifying tampering operations in image operator chains based on decision fusion. Sig Process Image Commun 95:116287. https://doi.org/10.1016/j.image.2021.116287
15. Chollet F (2017) Xception: deep learning with depthwise separable convolutions, pp 1800–1807. https://doi.org/10.1109/CVPR.2017.195
16. Christlein V, Riess C, Angelopoulou E (2010) On rotation invariance in copy-move forgery detection. In: 2010 IEEE international workshop on information forensics and security, pp 1–6. https://doi.org/10.1109/WIFS.2010.5711472
17. Christlein V, Riess C, Jordan J, Riess C, Angelopoulou E (2012) An evaluation of popular copy-move forgery detection approaches. IEEE Trans Inf Forensics Secur 7(6):1841–1854. https://doi.org/10.1109/TIFS.2012.2218597
18. Cozzolino D, Verdoliva L (2020) Noiseprint: a cnn-based camera model fingerprint. IEEE Trans Inf Forensics Secur 15:144–159. https://doi.org/10.1109/TIFS.2019.2916364
19. de Carvalho TJ, Riess C, Angelopoulou E, Pedrini H, de Rezende Rocha A (2013) Exposing digital image forgeries by illumination color classification. IEEE Trans Inf Forensics Secur 8(7):1182–1194. https://doi.org/10.1109/TIFS.2013.2265677
20. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) ImageNet: a large-scale hierarchical image database. In: CVPR. https://doi.org/10.1109/WIFS.2010.5711472
21. Dittmann J (2001) Content-fragile watermarking for image authentication. In: Security and watermarking of multimedia contents III, vol 4314, pp 175–184. International Society for Optics and Photonics. https://doi.org/10.1117/12.435398
22. Doegar A, Dutta M, Gaurav K (2019) Cnn based image forgery detection using pre-trained alexnet model. Electronic
23. Dolhansky B, Howes R, Pflaum, Baram N, Ferrer C (2019) The deepfake detection challenge dfdc preview dataset
24. Dong J, Wang W, Tan T (2013) Casia image tampering detection evaluation database. In: 2013 IEEE China summit and international conference on signal and information processing, pp 422–426. https://doi.org/10.1109/ChinaSIP.2013.6625374
25. Elaskily M, Elnemr H, Sedik A, Dessouky M, El Banby G, Elaskily O, Khalaf AAM, Aslan H, Faragallah O, El-Samie FA (2020) A novel deep learning framework for copy-move forgery detection in images. Multimed Tools Appl 79. https://doi.org/10.1007/s11042-020-08751-7

26. Eykholt K, Evtimov I, Fernandes E, Li B, Rahmati A, Xiao C, Prakash A, Kohno T, Song DX (2018) Robust physical-world attacks on deep learning visual classification. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 1625–1634
27. Faceswap. https://github.com/deepfakes/faceswap. Accessed 16 Mar 2022
28. Farid H (1999) Detecting digital forgeries using bispectral analysis. AI Lab, Massachusetts Institute of Technology, Tech Rep AIM-1657
29. Farid H (2009) Image forgery detection: a survey. Signal Proc Mag IEEE 26(04):16–25. https://doi.org/10.1109/MSP.2008.931079
30. Fischler M, Bolles R (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24(6):381–395. https://doi.org/10.1145/358669.358692
31. Fridrich J, Soukal D, Lukás J (2003) Detection of copy move forgery in digital images. Proc. Digital Forensic Research Workshop
32. Fridrich J, Chen M, Goljan M (2007) Imaging sensor noise as digital x-ray for revealing forgeries. In: Proceedings of the 9th international workshop on information hiding, Sant Malo, France, pp 342–358. https://doi.org/10.1007/978-3-540-77370-2_23
33. Gimp. https://www.gimp.org/. Accessed 16 Mar 2022
34. Goldman E (2018) The complicated story of FOSTA and Section 230. First Amend L Rev 17:279
35. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. Adv Neural Inf Process Syst 3
36. Goodfellow I, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples, vol 12. arXiv:1412.6572
37. Hinton GE, Krizhevsky A, Wang SD (2011) Transforming auto-encoders. In: Honkela T, Duch W, Girolami M, Kaski S (eds) Artificial Neural Networks and Machine Learning – ICANN 2011. Springer, Berlin, pp 44–51
38. Huynh TK, Huynh KV, Le-Tien T, Nguyen SC (2015) A survey on image forgery detection techniques. In: The 2015 IEEE RIVF international conference on computing & communication technologies-research, innovation, and vision for future (RIVF). IEEE, pp 71–76. https://doi.org/10.1109/RIVF.2015.7049877
39. Interactive Web demo: Whichfaceisreal. https://www.whichfaceisreal.com/index.php. Accessed 16 Mar 2022
40. Johnson MK, Farid H (2005) Exposing digital forgeries by detecting inconsistencies in lighting. In: Proceedings of the ACM multimedia and security workshop, New York, NY, pp 1–10. https://doi.org/10.1145/073170 1073171
41. Johnson MK, Farid H (2006) Exposing digital forgeries through chromatic aberration. In: Proceedings of the ACM multimedia and security workshop, Geneva, pp 48–55. https://doi.org/10.1145/1161366.1161376
42. Johnson MK, Farid H (2006) Metric measurements on a plane from a single image. Tech Rep TR2006-579
43. Johnson MK, Farid H (2007) Detecting photographic composites of people. In: Proceedings of the 6th international workshop on digital watermarking, Guangzhou. https://doi.org/10.1007/978-3-540-92238-4_3
44. Johnson MK, Farid H (2007) Exposing digital forgeries through specular highlights on the eye. In: Proceedings of the 9th international workshop on information hiding, Saint Malo, France, pp 311–325. https://doi.org/10.1007/978-3-540-77370-2_21
45. Karras T, Laine S, Aila T (2019) A style-based generator architecture for generative adversarial networks, pp 4396–4405. https://doi.org/10.1109/CVPR.2019.00453
46. Keek. https://keeex.me/products/. Accessed 16 Mar 2022
47. Koptyra K, Ogiela MR (2021) Imagechain—application of blockchain technology for images. Sensors 21(1):82. https://doi.org/10.3390/s21010082
48. Korus P (2017) Digital image integrity–a survey of protection and verification techniques. Digit Signal Process 71:1–26. https://doi.org/10.1016/j.dsp.2017.08.009
49. Korus P, Huang J (2016) Evaluation of random field models in multi-modal unsupervised tampering localization. In: 2016 IEEE international workshop on information forensics and security (WIFS), pp 1–6. https://doi.org/10.1109/WIFS.2016.7823898
50. Korus P, Huang J (2017) Multi-scale analysis strategies in prnu-based tampering localization. IEEE Trans Inf Forensic Secur
51. Kowalski M (2016) https://github.com/MarekKowalski/FaceSwap/. Accessed 16 Mar 2022
52. Krizhevsky A, Nair V, Hinton G (2009) Cifar-10 (Canadian Institute for Advanced Research)
53. Krizhevsky A, Sutskever I, Geoffrey H (2012) Imagenet classification with deep convolutional neural networks. Neural Inf Process Syst 25. https://doi.org/10.1145/3065386

54. Kurakin A, Goodfellow I, Bengio S (2016) Adversarial examples in the physical world
55. LeCun Y, Cortes C (2010) MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. Accessed 16 Mar 2022 [cited 2016-01-14 14:24:11]
56. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444. https://doi.org/10.1038/nature14539
57. Li Y, Lyu S (2018) Exposing deepfake videos by detecting face warping artifacts
58. Li Y, Chang MC, Lyu S (2018) In ictu oculi: exposing ai created fake videos by detecting eye blinking, pp 1–7. https://doi.org/10.1109/WIFS.2018.8630787
59. Li Y, Yang X, Qi H, Lyu S (2016) Celeb-df: a large-scale challenging dataset for deepfake forensics, pp 3204–3213. https://doi.org/10.1109/CVPR42600.2020.00327
60. Liao X, Li K, Zhu X, Liu KJR (2020) Robust detection of image operator chain with two-stream convolutional neural network. IEEE J Sel Top Signal Process 14(5):955–968. https://doi.org/10.1109/JSTSP.2020.3002391
61. Liao X, Huang Z, Peng L, Qiao T (2021) First step towards parameters estimation of image operator chain. Inf Sci 575. https://doi.org/10.1016/j.ins.2021.06.045
62. Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, Van Der Laak JA, Ginneken BV, Sánchez CI (2017) A survey on deep learning in medical image analysis. Med Image Anal 42:60–88. https://doi.org/10.1016/j.media.2017.07.005
63. Liu G, Reda F, Shih K, Wang TC, Tao A, Catanzaro B (2018) Image inpainting for irregular holes using partial convolutions
64. López-García X, Silva-Rodríguez A, Vizoso-García AA, Oscar W, Westlund J (2019) Mobile journalism: systematic literature review. Comunicar Media Educ Res J 27(1). https://doi.org/10.3916/C59-2019-01
65. Lowe D (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vision 60:91–. https://doi.org/10.1023/B:VISI.0000029664.99615.94
66. Lu C-S, Liao H-YM (2001) Multipurpose watermarking for image authentication and protection. IEEE Trans Image Process 10(10):1579–1592. https://doi.org/10.1109/83.951542
67. Lukás J, Fridrich J (2003) Estimation of primary quantization matrix in double compressed jpeg images. Proc Digital Forensic Research Workshop. https://doi.org/10.1117/12.759155
68. Majumder MTH, Alim Al Islam ABM (2018) A tale of a deep learning approach to image forgery detection. In: 2018 5th international conference on networking, systems and security (NSysS), pp 1–9. https://doi.org/10.1109/NSysS.2018.8631389
69. Marra F, Gragnaniello D, Verdoliva L, Poggi G (2020) A full-image full-resolution end-to-end-trainable cnn framework for image forgery detection. IEEE Access:1–1.
70. Moreira D, Bharati A, Brogan J, Pinto A, Parowski M, Bowyer KW, Flynn PJ, Rocha A, Scheirer WJ (2018) Image provenance analysis at scale. IEEE Trans Image Process 27(12):6109–6123
71. Muzaffer G, Ulutas G (2019) A new deep learning-based method to detection of copy-move forgery in digital images. In: 2019 Scientific meeting on electrical-electronics biomedical engineering and computer science (EBBT), pp 1–4. https://doi.org/10.1109/EBBT.2019.8741657
72. Nguyen H, Yamagishi J, Echizen I (2019) Use of a capsule network to detect fake images and videos
73. Nightingale SJ, Wade KA, Watson DG (2017) Can people identify original and manipulated photos of real-world scenes? Cognitive Research: Principles and Implications 2(1):1–21. https://doi.org/10.1186/s41235-017-0067-2
74. Nikolaidis N, Pitas I (1996) Copyright protection of images using robust digital signatures. In: 1996 IEEE international conference on acoustics, speech, and signal processing conference proceedings, vol 4. IEEE, pp 2168–2171. https://doi.org/10.1109/ICASSP.1996.545849
75. Nilsback M, Zisserman A (2008) Automated flower classification over a large number of classes. In: 2008 Sixth Indian conference on computer vision, simage processing, pp 722–729. https://doi.org/10.1109/ICVGIP.2008.47
76. Numbersprotocol.io. https://numbersprotocol.io/. Accessed 16 Mar 2022
77. Online article on Arstechnica, May 2007 https://arstechnica.com/uncategorized/2007/05/latest-aacs-revision-defeated-a-week-before-release/. Accessed 16 Mar 2022
78. Ouyang J, Liu Y, Liao M (2017) Copy-move forgery detection based on deep learning. In: 2017 10th international congress on image and signal processing, BioMedical engineering and informatics (CISP-BMEI), pp 1–5. https://doi.org/10.1109/CISP-BMEI.2017.8301940
79. Passarella A (2012) A survey on content-centric technologies for the current internet CDN and P2P solutions. Comput Commun 35(1):1–32. https://doi.org/10.1016/j.comcom.2011.10.005
80. Philbin J, randjelović R, Zisserman A (2007) The Oxford Buildings Dataset. https://www.robots.ox.ac.uk/vgg/data/oxbuildings/. Accessed 16 Mar 2022
81. Piva A (2013) An overview on image forensics. International Scholarly Research Notices 2013. https://doi.org/10.1155/2013/496701

82. Popescu AC, Farid H (2004) Exposing digital forgeries by detecting duplicated image regions. Tech. Rep. TR2004-515
83. Popescu AC, Farid H (2005) Exposing digital forgeries by detecting traces of re-sampling. IEEE Trans Signal Process 53(2):758–767. https://doi.org/10.1109/TSP.2004.839932
84. Qureshi MA, Deriche M (2015) A bibliography of pixel-based blind image forgery detection techniques. Signal Process Image Commun 39:46–74. https://doi.org/10.1016/j.image.2015.08.008
85. Rajini NH (2019) Image forgery identification using convolution neural network. Int J Recent Technol Eng 8
86. Rao Y, Ni J (2016) A deep learning approach to detection of splicing and copy-move forgeries in images. In: 2016 IEEE international workshop on information forensics and security (WIFS), pp 1–6. https://doi.org/10.1109/WIFS.2016.7823911
87. Roy S, Sun Q (2007) Robust hash for detecting and localizing image tampering. In: 2007 IEEE international conference on image processing, vol 6. IEEE, pp VI–117. https://doi.org/10.1109/ICIP.2007.4379535
88. Rössler A, Cozzolino D, Verdoliva L, Riess C, Thies J, Nießner M (2019) Faceforensics++: learning to detect manipulated facial images
89. Rublee E, Rabaud V, Konolige K, Bradski G (2011) Orb: an efficient alternative to sift or surf. In: 2011 International conference on computer vision, pp 2564–2571. https://doi.org/10.1109/ICCV.2011.6126544
90. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg A, Fei-Fei L (2014) Imagenet large scale visual recognition challenge. Int J Comput Vision 115. https://doi.org/10.1007/s11263-015-0816-y
91. Schaefer G, Stich M (2003) UCID: an uncompressed color image database. In: Yeung MM, Lienhart RW, Li CS (eds) Storage and retrieval methods and applications for multimedia 2004, vol 5307. International Society for Optics and Photonics, SPIE, pp 472–480. https://doi.org/10.1117/12.525375
92. Schetinger M, Chang S (1996) A robust content based digital signature for image authentication. In: Proceedings of 3rd IEEE international conference on image processing, vol 3. IEEE, pp 227–230. https://doi.org/10.1109/ICIP.1996.560425
93. Schetinger V, Oliveira MM, da Silva R, Carvalho TJ (2017) Humans are easily fooled by digital images. Comput Graph 68:142–151. https://doi.org/10.1016/j.cag.2017.08.010
94. Shen C, Kasra M, Pan P, Bassett GA, Malloch Y, F O'Brien J (2019) Fake images: the effects of source, intermediary, and digital media literacy on contextual assessment of image credibility online. New Media & Society 21(2):438–463. https://doi.org/10.1177/1461444818799526
95. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
96. Spohr D (2017) Fake news and ideological polarization: Filter bubbles and selective exposure on social media. Bus Inf Rev 34(3):150–160. https://doi.org/10.1177/0266382117722446
97. Thakur R, Rohilla R (2019) Copy-move forgery detection using residuals and convolutional neural network framework: a novel approach. In: 2019 2nd international conference on power energy, environment and intelligent control PEEIC, pp 561–564. https://doi.org/10.1109/PEEIC47157.2019.8976868
98. Thies T, Zollhöfer M, Stamminger M, Christian T, Nießner M (2018) Face2face: real-time face capture and reenactment of rgb videos. Commun ACM 62:96–104. https://doi.org/10.1145/3292039
99. Thies J, Zollhöfer M, Nießner M (2019) Deferred neural rendering: image synthesis using neural textures. ACM Trans Graph 38:1–12. https://doi.org/10.1145/3306346.3323035
100. Tralic D, Zupancic I, Grgic S, Grgic M (2013) Comofod — new database for copy-move forgery detection. In: Proceedings ELMAR-2013, pp 49–54
101. Various. Columbia image splicing detection evaluation dataset - list of photographers, 2004. https://www.ee.columbia.edu/ln/dvmm/downloads/AuthSplicedDataSet/photographers.htm. Accessed 16 Mar 2022
102. Verdoliva L (2020) Media forensics and deepfakes: an overview. IEEE J Sel Top Signal Process:1–1. https://doi.org/10.1109/JSTSP.2020.3002101
103. Warif NBA, Wahab AWA, dris MYI, Ramli R, Salleh R, Shamshirband S, Choo K-KR (2016) Copy-move forgery detection: Survey, challenges and future directions. J Netw Comput Appl 75:259–278. https://doi.org/10.1016/j.jnca.2016.09.008
104. Wojna Z, Ferrari V, Guadarrama S, Silberman N, Chen LC, Fathi A, Uijlings J (2017) The devil is in the decoder. In: British machine vision conference (BMVC), pp 1–13
105. Wu Y, Abd-Almageed W, Natarajan P (2018) Busternet: detecting copy-move image forgery with source/target localization. In: Proceedings of the European conference on computer vision (ECCV), pp 168–184. https://doi.org/10.1007/978-3-030-01231-1_11

106. Wu Y, AbdAlmageed W, Natarajan P (2019) Mantra-net: manipulation tracing network for detection and localization of image forgeries with anomalous features. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 9535–9544. https://doi.org/10.1109/CVPR.2019.00977
107. Zhang Y, Goh J, Win LL, Vrizlynn T (2016) Image region forgery detection: a deep learning approach. In: SG-CRC, pp 1–11. https://doi.org/10.3233/978-1-61499-617-0-1
108. Zhang K, Zuo W, Cheng Y, Meng D, Zhang L (2017) Beyond a gaussian denoiser: residual learning of deep cnn for image denoising. IEEE Trans Image Process 26(7):3142–3155. https://doi.org/10.1109/TIP.2017.2662206
109. Zhang Q, Yang LT, Chen Z, Li P (2018) A survey on deep learning for big data. Information Fusion 42:146–157. https://doi.org/10.1016/j.inffus.2017.10.006

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com