Foundations and Trends® in Computer Graphics and Vision

Deep Learning for Multimedia Forensics

Suggested Citation: Irene Amerini, Aris Anagnostopoulos, Luca Maiano and Lorenzo Ricciardi Celsi (2021), "Deep Learning for Multimedia Forensics", Foundations and Trends® in Computer Graphics and Vision: Vol. 12, No. 4, pp 309–457. DOI: 10.1561/0600000096.

Irene Amerini
Sapienza University of Rome
amerini@diag.uniroma1.it

Aris Anagnostopoulos
Sapienza University of Rome
aris@diag.uniroma1.it

Luca Maiano
Sapienza University of Rome
ELIS Innovation Hub
maiano@diag.uniroma1.it

Lorenzo Ricciardi Celsi
ELIS Innovation Hub
l.ricciardicelsi@elis.org

now
the essence of knowledge
Boston — Delft

Contents

Deep Learning for Multimedia Forensics

Irene Amerini[1], Aris Anagnostopoulos[1], Luca Maiano[1,2] and Lorenzo Ricciardi Celsi[2]

[1] *Sapienza University of Rome, Italy;*
[2] *ELIS Innovation Hub, Italy*

ABSTRACT

In the last two decades, we have witnessed an immense increase in the use of multimedia content on the internet, for multiple applications ranging from the most innocuous to very critical ones. Naturally, this emergence has given rise to many types of threats posed when this content can be manipulated/used for malicious purposes. For example, fake media can be used to drive personal opinions, ruining the image of a public figure, or for criminal activities such as terrorist propaganda and cyberbullying. The research community has of course moved to counter attack these threats by designing manipulation-detection systems based on a variety of techniques, such as signal processing, statistics, and machine learning. This research and practice activity has given rise to the field of *multimedia forensics*.

The success of deep learning in the last decade has led to its use in multimedia forensics as well. In this survey, we look at the latest trends and deep-learning-based techniques introduced to solve three main questions investigated in the field of multimedia forensics. We begin by examining the manipulations of images and videos produced with editing tools, reporting the deep-learning approaches adopted to

Irene Amerini, Aris Anagnostopoulos, Luca Maiano and Lorenzo Ricciardi Celsi (2021), "Deep Learning for Multimedia Forensics", Foundations and Trends® in Computer Graphics and Vision: Vol. 12, No. 4, pp 309–457. DOI: 10.1561/0600000096.

counter these attacks. Next, we move on to the issue of the source camera model and device identification, as well as the more recent problem of monitoring image and video sharing on social media. Finally, we look at the most recent challenge that has emerged in recent years: recognizing deepfakes, which we use to describe any content generated using artificial-intelligence techniques; we present the methods that have been introduced to show the existence of traces left in deepfake content and to detect them. For each problem, we also report the most popular metrics and datasets used today.

1

Introduction

Over the past years, online and multimedia content has passed traditional media as a preferred source of information, especially for young people (Richard Fletcher, 2020), and in the next few years visual content offered by social networks like Instagram could possibly overtake other platforms as a news source. The web and social media have favored the democratization of information and have allowed much more widespread dissemination of news (BBC-News, 2020). Although access to this content should have promoted the dissemination of reliable and validated content from multiple sources of information, the web and in particular social networks have also become a dangerous source of disinformation and dissemination of criminal content. Recently, fake videos of political leaders like Donald Trump, Vladimir Putin and North Korean leader Kim Jong-un have become increasingly realistic, opening up to the possibility of manipulating elections or public opinion (Staff, 2021 and Hao, 2020). Likewise, fake images and videos can be used for cyberbullying, military propaganda, or other criminal acts. All these problems have something in common. The widespread use of photo and video editing applications and the ease of use and retrieval of these tools have made multimedia manipulation a powerful instrument in the

hands of criminals and attackers. Fake news, fake political campaigns, and porn videos, as well as fraud attempts are becoming much easier to spread and produce with a high level of realism. Distinguishing between fake and real is becoming an extremely important but difficult task. When multimedia contents are published on the web, they can easily go viral on social media. Also, *deepfakes*, which consist of fake content artificially generated typically using modern deep-learning approaches, have received a lot of attention in the last few years thanks to the high level of realism reached by this technology. Sophisticated deep-learning architectures such as autoencoders (AE) and generative adversarial networks (GANs) can be used to create highly realistic fake images and videos. Building trust and enabling the assessment of the authenticity of multimedia content is no longer an option but a real necessity.

The area of *multimedia forensics* combines principles and approaches from diverse research areas such as computer vision and signal processing, when it comes to addressing the authenticity and source of an image or a video. The three topics that multimedia forensics investigates mostly are the following: (1) *forgery detection*, which involves the detection of the authenticity of an image or video as well as of the presence of any manipulations; (2) *source identification*, which is the reconstruction of the history of some digital content, addressing which camera model, brand, or even specific device has captured that content, or whether it has been downloaded from social media; (3) *deepfake detection*, defining a *deepfake* as any synthetic medium accounting for the replacement of a person in an existing image or video with someone else's likeness (see Fig. 1 for instance). Figure 1.1 shows these three main problems.

Researchers have been studying the problem of forgery detection for more than twenty years now. Every day, thousands of professionals around the world use editing tools such as GIMP, Photoshop, Lightroom, After Effects Pro, and Final Cut Pro X as basic applications for their work. Multimedia forensic researchers have tried to provide an immediate response to all such applications, developing new tools to spot fake content. These methods can be used to detect subtle modifications, such as double compression or blurring, as well as more sophisticated attacks that could be used to change the semantic of a content. The most widespread examples of these manipulations are *splicing* (an object is

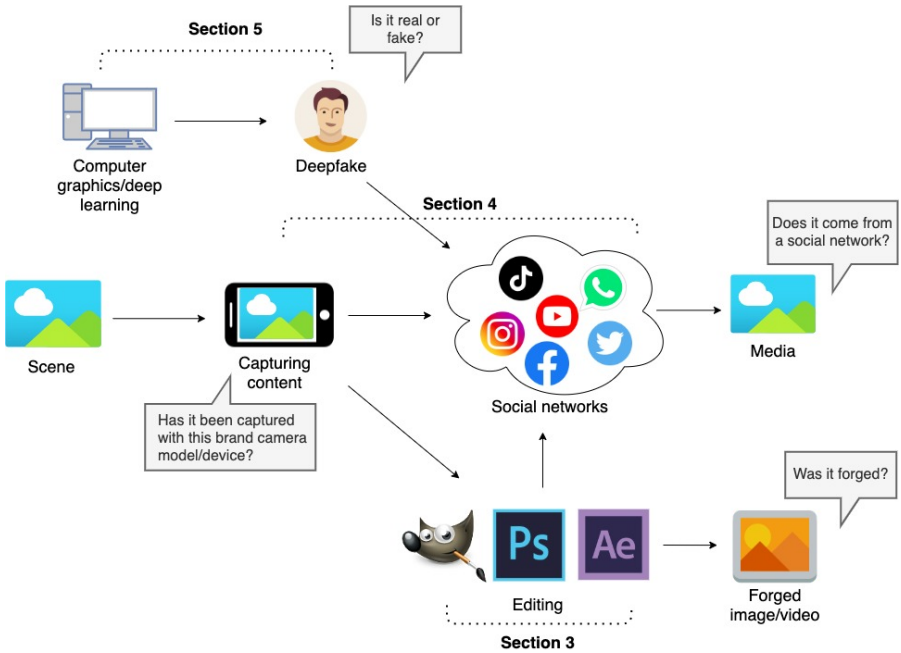


Figure 1.1: An overview of multimedia forensic investigations that we present in this work.

copied from an image and pasted into another picture), *copy–move* (the reproduction of an object into the same image), and *video-frame deletion* and *addition* in the case of video sequences. Recently, the advancements of artificially generated manipulations have attracted the attention of many researchers. *Deepfakes* are raising new alarms for the production of fake news, and their entry into the field of large technology giants has accelerated the design of new methods. Figure 1.2 shows some examples of the most recent fakes that spread out over the world.

Parallel to this problem, the identification of the source has been carefully studied as a forensic analysis tool. This becomes extremely important today in a hyper-connected world where information spreads all over the web. In some scenarios, multimedia content may constitute proof in the court proceedings and it becomes necessary to prove not only the authenticity of an image or video but also the source of the image or video itself. First of all, when it comes to assessing the authenticity of

(a) Fake Mark Zuckerberg *Bill Posters UK, Instagram page* 2019.

(b) Fake Barack Obama (left) and the actor who is impersonating him (right) *You Won't Believe What Obama Says in This Video!, Youtube video* 2018.

(c) An actor (left) and a fake Donald Trump (right) *Trump: Deepfakes Replacement, Youtube video* 2018.

Figure 1.2: Some of the most recent fakes that spread out over the world.

an image or video, the most advanced techniques for forgery detection allow to identify dishomogeneities in the considered image or video as well as any tampered features responsible for introducing differences from the original image/patch, especially any differences that are not so evident to the naked eye. Source identification can then be used to determine if the content was captured with a specific camera model or brand and even with a specific device. This can be done by exploiting the sequence of processes that a camera uses to convert the input light hitting the lens into an output image or video. This operation leaves important traces on the acquired files that can be used for forensic purposes. With the widespread adoption of social media and messaging applications, the task of deciding whether an image or video has been downloaded from these platforms has become important as well.

Forensic problems have been studied for a long time and they have been surveyed in multiple works such as Stamm *et al.* (2013), Verdoliva (2020), and Yang *et al.* (2020b). For years, researchers with

different backgrounds have adopted signal-processing, computer-vision, and machine-learning techniques to solve the main challenges in this research field. Deep learning has recently come up with new designs that are capable of automatically learning both low- and high-level features to be analyzed to solve forensic problems.

In this survey, we present deep-learning methods for multimedia forensics, discussing the most important trends in both architectural and data-processing choices. We begin discussing different techniques used to manipulate content in Section 2. Next, we discuss image and video forgery techniques in Section 3. In Section 4, we review deep learning methods for source identification. Finally, in Section 5 we present the recent solutions for deepfake detection. Section 6 recaps the evaluation metrics considered throughout the cited works and Section 7 lists the datasets that have been mostly adopted for the above-mentioned tasks. Finally, in Section 8 we draw the conclusions.

2

Generating Fake Image and Video Content

The ultimate goal of multimedia forensics is to be able to detect and track modified and generated content. Although the purpose of this survey focuses on reporting the cutting-edge methods based on deep learning that the research community has developed in recent years, it is worth taking a step back at the various techniques that can be used to generate fake images and videos and manipulation attacks in general. In this section, we propose a fast introduction to different manipulations and generative techniques that will be cited in Section 3 and Section 5 of this review.

Before moving on, we want to clarify the difference between image or video *forgery* and *generation*. We refer to the term forgery whenever we apply some manipulation to an existing image or video via traditional editing software. This could be any kind of commonly used editing operation such as image resizing or color correction, or content editing operations such as adding a new object in an image or a frame in a video to alter its semantic content. In Sections 2.1 and 2.2 we discuss both image and video forgery attacks. When we refer to image and video generation we instead refer to generated content through methods based on computer graphics or on artificial intelligence. These last two types

of *attacks* (both multimedia forgery and generation can be thought of as attacks from the point of view of a multimedia-forensic researcher), are typically used to generate realistic faces that do not exist or modify in a realistic way the attributes of a source face in a video with that of a target person. Fakes generated with computer graphics and deep-learning approaches have much in common because they both lack the characteristic features that are typical of images and videos acquired by real cameras. We discuss these techniques in Section 2.3.

Following the classification of attacks that we described, we treat the detection of manipulated content in three ways: (1) as a classification problem where an image/video is classified as real or fake, (2) as a regression problem where you want to locate the manipulated area of an image or video, or (3) as an anomaly detection problem where a model is trained on authentic examples and learns to reveal these attacks as anomalies.

2.1 Multiple Compression

An image can be forged in many different ways. For example, you could simply correct the colors and dimensions of a photo to improve its visual quality before making it published on the Internet, or you could modify the content by introducing new elements from other images with the aim of changing the meaning of what has been caught in the image itself. Regardless of the type of operations performed on the image, it must be saved and recorded in a standard format. The same is true for videos. The traces of multiple compression (or *double compression* in the simplest case) do not tell us that a manipulation necessarily took place, as it is still possible that a photo was compressed before being shared in order to reduce the file size or exported in a format other than the original one, however, it can be useful information to be reconstructed to verify the genuineness of a multimedia content or to trace its source as will be discussed in Section 4.

Image compression. Whenever we edit an image or we simply export it with an editing application, this will be recompressed in the current or a new file format. For example, when a grayscale image undergoes JPEG compression, it is segmented into nonoverlapping

patches. Next, the discrete cosine transform (DCT) of each patch is computed and quantized. When the image is decompressed, each of the steps performed during the encoding process, except for quantization, is inverted. Therefore, each dequantized DCT coefficient is unlikely to correspond to its original value, causing distortion on the original image. These distortions left by *multiple compressions* can be exploited to extract some relevant information about the history of an image. Although this kind of traces typically suggest that an image may have undergone manipulation (see Section 3), this information can also be used to track image sharing on social media or to reconstruct the source of a picture as discussed in Section 4.

Video Compression. Multiple compression can also be used for videos. In fact, because the size of an uncompressed video file makes it difficult to store or share it on the Internet, virtually all digital video is compressed. Most of the compression algorithms used today share the same basic idea. Instead of processing the entire video at once, the encoder splits the video frame sequence into smaller segments. Each segment, known as a group of pictures (GOP), starts with an intra-frame (I-frame) which is independently encoded using a process similar to JPEG compression and continues with the predicted frames (P-frames) and bidirectional frames (B-frame). P-frames are predicted from preceding frames and B-frames can be predicted from I-frames or P-frames preceding or following them in the GOP. As a result of this structure, multiple compression leaves two kinds of traces: spatial and temporal. Spatial traces manifest to a single video frame and are very similar to image compression fingerprints. Temporal fingerprints are spread throughout the entire video and are really useful when some frames are added to or deleted from the video. In fact, this will cause a shift in the sequence of video frames, resulting in the formation of a new set of GOPs when the video is recompressed.

2.2 Manipulated Media

To detect the traces left by these editing operations, it is important to list the most common operations that we want to recognize.

Copy-move. A forger alters an image by replacing some part of the image with content copied from somewhere elsewhere within the same image. This is often done to hide the presence of an object by covering it with other elements in the image (also know as removal) or to create duplicates of a significant object within the image.

Splicing/cut-paste. Similar to copy-move, a forger alters an image with content copied from another image. Because the two images usually have different statistical properties, this kind of manipulation is typically easier to detect with respect to copy-move.

Resampling. Resampling is performed whenever an image is resized, rotated, or undergoes an affine transformation. Resampling is the mathematical technique used to create a new version of the image with a different size. For example, increasing the size of an image is called upsampling; reducing its size is called downsampling. When you downsample, you delete the information and therefore the details from the image by computing the convolution of the image with a low-pass filter (to avoid aliasing). However, when you upsample, you convolve the image with some interpolation kernel, which means that the editing software examines the colors and details of the original photo and creates new ones, which are then added to the existing details. In general, resampling can be performed with several filters like nearest neighbor, bilinear, bicubic, and windowed sinc functions. Even though the detection of resampling proofs in the entire image does not necessarily imply that the image has been manipulated, it surely indicates that the image has been processed. In fact, when creating a splice or copy-move image fake, it is often necessary to resize or rotate an object in the image to make the fake look visually realistic.

Contrast enhancement. This technique is typically used to adjust the lighting within an image. Contrast enhancement works by applying a nondecreasing nonlinear mapping to the values of a signal that can leave traces in the form of impulsive peaks and gaps introduced into an image's pixel values. Although it can be used to enhance a photo that is too dark or too light, it is sometimes applied in conjunction with splicing to mask the difference in brightness between the image from which the object introduced with the manipulation comes from.

Denoising/Median filtering. It is often used to denoise or smooth an image through a nonlinear operation that smoothes the signal while preserving its edge content. Through this technique a sequence of observations of adjacent signals all ends up assuming the same value, this operation leaves streaks in the signals.

2.3 Generated Media

The advancement of computer graphics and deep learning has led to the emergence of more advanced manipulation techniques. Today, most of the researchers' efforts focus on detecting manipulations on faces. As suggested by Rössler *et al.*, 2019, current manipulation methods can be grouped into two classes: facial expression manipulation and facial identity manipulation. The first class allows the transfer of a person's facial expressions another. The latter consists in the replacement of the face of a person with the face of another one. In the remainder of this section, we report the most common attacks that have been studied in recent years. Most of the attacks analyzed in recent years are collected in the richest datasets containing deepfakes: FaceForensics++ and the Deepfake Detection Challenge (DFDC) datasets (see Section 7.3).

Deepfake. The term Deepfakes has been widely adopted to indicate face replacement methods based on deep learning. This term was originated after a Reddit user named *deepfakes* claimed in late 2017 to have developed a machine learning algorithm that helped him to transpose celebrity faces into porn videos. However, it also represents the name of a specific manipulation method that originally spread through online forums. In a deepfake, the face of a target person is replaced by another face from a source video or image. There are various public implementations of DeepFakes available, most notably the FaceSwap Github (*Deepfakes Faceswap* 2018) and FakeApp (*Fake App* 2018). The model uses one shared encoder, but two separately trained decoders, one for each identity in the swap. This architecture forces the encoder to learn common features across both identities (e.g., lighting, pose, facial expressions); then each decoder learns person-specific features such as to produce a realistic swap.

Face2Face. Face2Face (introduced by Thies *et al.*, 2020) is a facial reenactment method that transfers the expressions of a source face to a target video while maintaining the identity of the target person. The original method is based on two video input streams and a manual keyframe selection. Through these frames, the model generates a dense reconstruction of the face that can be used to reproduce the face in new sequences.

FaceSwap. FaceSwap is a computer-graphics based method used to transfer the face from a source to a target. Ic computes facial landmarks in both source and target images, and it morphs the pixels from the source image by fitting a 3D template model to match the landmarks in the target image. This model is back-projected to the target image by minimizing the difference between the projected shape and the localized landmarks using the textures of the input image and blended with the image.

Neural Textures. According to Thies *et al.*, 2019, this approach consists in training on original videos to learn object-specific *neural textures* (learned feature maps that are trained as part of the scene capture process), which can be interpreted by a neural renderer. The model is trained with a photometric reconstruction loss and an adversarial loss.

This completes our brief introduction on the manipulations and generative techniques. We refer to them in the rest of our text. We start in the next section by presenting the state-of-the-art on image and video forgery techniques.

3

Forgery Detection on Images and Videos

The widespread adoption of easy-to-use photo editing tools has made image manipulation a powerful tool for criminals and attackers. Fake news and fake political-campaign videos, as well as fraud attempts, are becoming much easier to generate and spread, also with an extremely high degree of realism.

Convolutional neural networks (CNNs) achieve highly accurate results on object detection tasks. However, standard CNNs tend to learn features related to the image-specific content, whereas, image forensic usually requires to suppress the semantic content and capture pixel value dependencies left by editing operations. Therefore, the multimedia forensics community has seen an increasing interest in the development of new deep-learning solutions explicitly designed to solve the problem of forgery detection as much independently of the image-specific content as possible. To address this problem, they have proposed different architectures to analyze both low-level and high-level features left by image or video manipulations. In this section, we review the actual state-of-the-art solutions, describing both the methodological and the architectural choices that have been introduced in the last few years.

According to the definition given in Section 2, a *forgery* could be every kind of editing operation that is performed over multimedia content, therefore compromising the authenticity of the file. Such manipulations can be the result either of a manual forgery procedure, for instance, a manipulation carried out by a human with the help of software such as Photoshop or GIMP, or of an automatic computer-generated one, such as in the case of deepfakes or Computer-Generated Imagery (CGI). Multimedia forensics is aimed at recognizing all these sorts of forgeries and, in fact, we will examine all of them in this work. However, it is common to treat these problems as separate because they could leave different traces. Thus, in this section, we take care of manual forgeries traces left by photo editing operations, and we discuss the second group of computer-generated images or videos in detail in Section 5.

As new solutions for image forgery detection emerge, two important design choices seem to guide all recent publications: the network architecture and the set of features extracted. The rest of this section explores recent trends involving deep neural networks; Tables B.1 and B.2 summarize some of the most innovative solutions related to images and videos, respectively. We begin by discussing network architectures targeting a single typology of attack and referenced in the following as manipulation-specific architectures (see Section 3.1), then we move on to the most recent trends involving the detection of multiple kinds of different manipulations (referred to as multimanipulations from Section 3.2 onwards). Finally, we conclude the section by reviewing anomaly-based architectures (see Section 3.3), in which manipulations are treated as anomaly points, therefore possibly enabling the detection of general manipulations. Please see Figure 3.1 for the organization of topics in this chapter.

3.1 Manipulation-Specific Architectures

The simplest family of deep learning models for detecting fake images and videos is the one looking for a specific type of counterfeit attack. Most of these models are usually borrowed from other computer vision applications such as object detection and segmentation. We categorize the detectors into two families based on their architecture: (1) single-

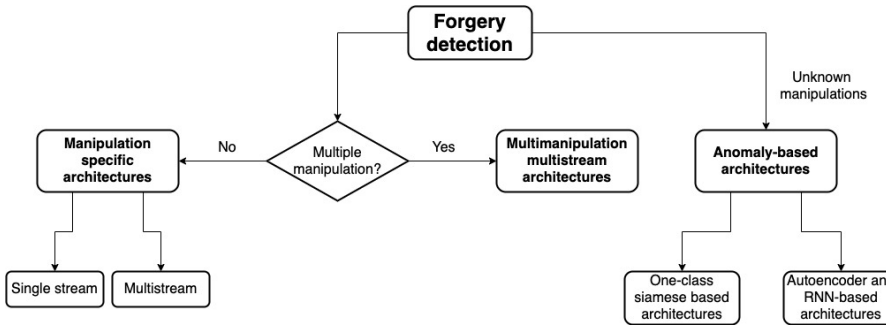


Figure 3.1: An overview of forgery-detection approaches.

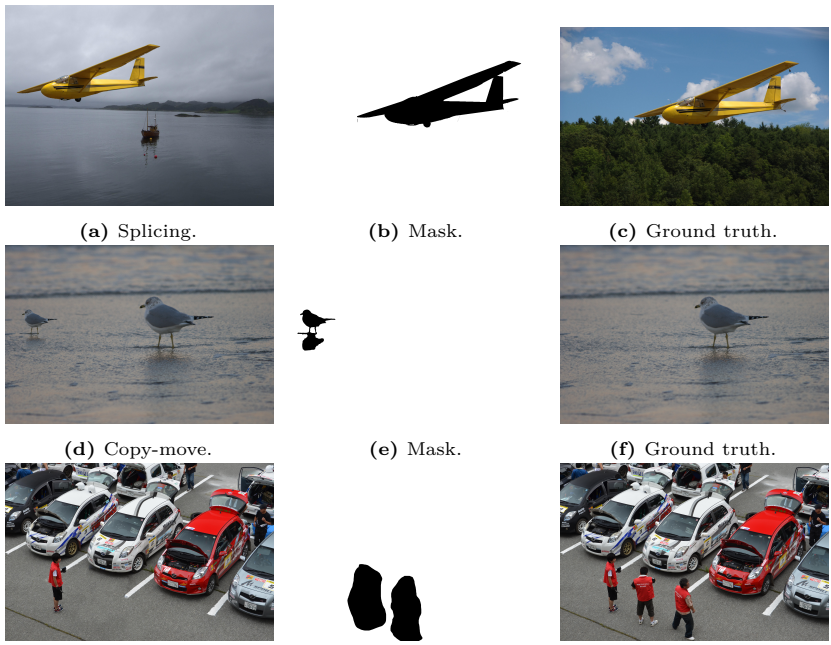
stream architectures, which are architectures composed of a single deep learning classifier and (2) multistream solutions, which combine two or more architectures targeting different forgery traces.

3.1.1 Single-Stream Architectures

The most straightforward detectors usually implement an established deep learning model. These detectors (see Figure 3.3), usually take as input a preprocessed image or patch residual, extracted through constrained layers or filters. Then, convolutional layers learn to map low-level features, which subsequently can be either fed to a classifier or used to reconstruct the so-called manipulation mask, that is, the image portion labeling the manipulated part of the image, as shown in Figure 3.2. Various works have adopted these network architectures to detect several types of processing applied to the images and videos, including different kind of forgeries. In this section, we present their use on resizing, splicing, image warping on faces, JPEG compression, and video frame drop.

We start by describing some general approaches for detecting image tampering.

Residual-based local descriptors have been proven successful in both image forgery detection (Cozzolino *et al.*, 2014a) and localization (Cozzolino *et al.*, 2014b), thus attracting a lot of interest in the forensic community. Such descriptors reveal anomalies by extracting Spatial Rich Model (SRM) based (as in Fridrich and Kodovsky, 2012) or



(a) Splicing. **(b)** Mask. **(c)** Ground truth.

(d) Copy-move. **(e)** Mask. **(f)** Ground truth.

(g) Removal. **(h)** Mask. **(i)** Ground truth.

Figure 3.2: An example of most common image manipulation techniques. (a) a portion of an image is copied and pasted into another image. (d) a portion of an image is reproduced inside the same image. (g) a portion of the image is removed. From Guan *et al.*, 2019.

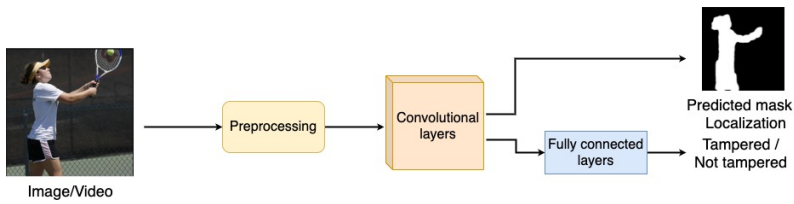


Figure 3.3: An example pipeline for manipulation-specific single-stream architectures. The input image, patch, or frame usually goes through a preprocessing step where a constrained convolutional layer or filter is applied. Next, the convolutional layers learn to extract relevant features, which subsequently can be either fed to a classifier or used to reconstruct the manipulation mask.

Subtractive Pixel Adjacency Matrix (SPAM) based (as in Pevný *et al.*, 2010) steganalysis features formed by high-order statistics collected

from image noise residuals. Once the image content is removed, the so-called residual noise contains specific micro-patterns that can allow for reliable analysis. In this respect, Cozzolino *et al.*, 2017 have shown that a class of residual-based characteristics can be extracted by means of convolutional neural networks.

A paper that has been the basis of multiple subsequent researches employing residuals is the work of Bayar *et al.*, 2017. It investigates the effect of several CNN design choices, tracing different guidelines for designing and training CNN architectures for multimedia forensics. As an initial step, the designer should pay particular attention to the choice of the first convolutional layer, which is very crucial for CNN performance. This layer is in charge of extracting the lowest-level features from the input data. Thus, the authors compare the performance of a predetermined high-pass filter (HPF) and of constrained convolutional layers (Bayar and Stamm, 2016; Bayar and Stamm, 2017b), discussing the important improvement brought by residual-based descriptors. Next, the authors claim that the image manipulation detection rate could be improved using deeper CNN architectures until the optimal depth is achieved. Additionally, the use of 1×1 convolutional filters after the highest-level feature maps i.e., the output of each convolutional layer) and average-pooling instead of max-pooling operations help to increase the detection rate (Xu *et al.*, 2016). The 1×1 filters learn the association between the highest-level feature maps in the network before the fully connected layers perform the classification. The average-pooling layer retains the most representative features from the deepest convolutional feature maps in the network, which could be destroyed by means of max-pooling. Finally, the authors show that batch normalization layers can be used to reduce overfitting. They test their solution on a set of experiments. At first, they evaluate the CNN's ability to detect a single manipulation (precisely, median filtering, Gaussian blurring, additive white Gaussian noise, resampling using bilinear interpolation and JPEG compression). Next, they evaluate the CNN's ability to be used as a multiclass classifier to perform general image-manipulation detection with different editing operations and parameters. Eventually, they prove the CNN's ability to detect a sequence of two different editing operation. To conduct these experiments, they create 10 different

databases, each one corresponding to one type of manipulation with different editing parameters. Each database consists of both manipulated and unaltered grayscale image patches. The proposed CNN achieves at least a 99.36% detection rate with all types of manipulations for single manipulation detection (with a training set of 25,000 manipulated and 25,000 authentic patches), and an overall manipulation identification rate of 99.66% for multiple manipulation detection (with a training set of 100,000 patches, 16,667 of which are unaltered). In Bayar and Stamm, 2018, the same research group later proposed a deeper and more sophisticated architecture, introducing important refinements on different components of the network.

Resizing high quality images tends to destroy precious high-frequency details, deeply affecting the performance of image-forensics tasks. In some cases, manipulated regions can be so small that they become practically invisible after a down-sampling operation. Still, some advanced attacks can only be detected based on the statistical analysis of micro-textures. To deal with this challenge, Marra *et al.*, 2019b suggest an architecture made of three blocks performing, respectively, patch-level feature extraction, feature aggregation, and classification. Their approach first divides the image into overlapping patches, subsequently it extracts the high-pass image residuals noise fingerprint (or Noiseprint) similar to photo response non uniformity (PRNU, see Section 4.3), and, then, it passes them through the aggregation pooling layer. As the authors suggest, the most appropriate type of pooling strictly depends on the problem of interest; therefore, they consider several forms and combinations of pooling. Finally, they aggregate the local information in a single descriptor for the whole image, which is then classified with a fully connected network. The network is trained end-to-end, forcing all the patches of the same image to share the same label, thus, forcing the network to learn how to manage differences between forged and pristine patches belonging to the same image to make the correct decision. To train the networks, they generate a suitable synthetic dataset splicing 81 objects manually cropped from the uncompressed images of the UCID dataset (introduced in Section 7.2) into background images taken from the Vision dataset (see Section 7.2). Next, they test the performance of their method on several datasets (a synthetic Dresden/FAU dataset

described in Section 7.2, Carvalho coming from Carvalho *et al.*, 2013, Korus introduced by Korus and Huang, 2016, NC2017, MFC2018, and MFC2019 described in Section 7.1) comparing it with the performance of reference methods. The proposed method achieves an average AUC of 82.4%. The performance of the model is consistently good in all cases including the NIST datasets, despite their great variety and the abundance of counter-forensic measures. Moreover, although the model is trained only on splicing manipulations, it works well also on all other localized manipulations that the authors tested against. Next, they compare the ROC curves, with and without fine tuning on the NIST dataset, suggesting that fine tuning on the development set grants further performance gains. With fine tuning, the AUC grows from 84.6% to 93.2% on NC2017, and from 83.8% to 90.2% on MFC2018.

Having presented the main works that address general types of manipulations (i.e., approaches developed for and tested against a variety of manipulation attacks), we now turn our attentions to specific attacks.

Splicing. As stated in Section 2, *splicing* is one of the most used manipulation techniques, where the content of an image is copied into another image. Bi *et al.*, 2019 introduce a Ringed Residual U-Net (RRU-Net) network for splicing attacks. The network takes spliced images as input and automatically learns how to extract relevant features to distinguish between tampered and untampered artifacts. Because the tampered regions come from other images, different clues of the image characteristics between the tampered and untampered regions are utilized to identify and locate the manipulated regions. However, the gradient degradation problem will destroy the performance when the network architecture gets deeper (He *et al.*, 2015). This well-known problem arises when training deep networks where the gradient diminishes dramatically as it is propagated backward through the network. The error may become so small by the time it reaches layers close to the input of the model that it may have very little effect. This problem, frequently encountered in the training of neural networks, is known as the *vanishing gradient problem* (Hochreiter *et al.*, 2001). To address this issue, the authors construct the network implementing dilated convolution blocks (as in Yu and Koltun, 2015). Then, taking advantage of the

ideas of Hu *et al.*, 2017, Bi *et al.*, 2019 design an attention system and add on to the residual feedback to draw attention to the discriminative features of the input information. A sigmoid activation function is used to learn a nonlinear interaction between discriminative feature channels. The residual feedback consolidates the input feature information to make the differences of image essence attributes between the untampered and tampered regions be amplified. The authors evaluate the performance of the proposed method under various attacks, including JPEG compression and noise corruption. The network achieves an F1-score (see Section 6 for the definition of this evaluation metric) on splicing detection of 84.1% on Casia (see Section 7.1) and 91.5% on Columbia (as introduced in Section 7.1) datasets.

Face manipulation. Because a huge amount of selfies and face images are shared every day on social networks, image manipulation applied to faces is becoming a common tampering technique used for beautification and expression editing. Wang *et al.*, 2019a propose a methodology to detect retouched faces. First, they generate a dataset of realistic manipulations using Adobe Photoshop to automatically generate fake training data. Then, a dilated residual network variant (DRN-C-26) similar to the one proposed by Yu *et al.*, 2017a is trained with both high and low resolution images augmenting the data with several methods, including resizing methods (bicubic and bilinear), JPEG compression, brightness, contrast, and saturation. Lastly, they propose an unwarping operation on the manipulated image to make it more similar to the original one. They evaluate the model first on auto-generated fake images and next they test it with a real-world setting collecting data from a professional artist, tasked with the goal of making a subject more attractive or changing the subject's expression. In the first case, the models achieve a 93.7% accuracy and a 98.9% average precision (AP) on low-resolution images (400 pixels on the smaller side) up to a 97.1% accuracy and a 99.8% AP on higher-resolution images (700 pixels on the shorter side). With real-world images, the accuracy drops from 97.1% in the validation set to 90.0% and the AP drops from 99.8% to 97.4%.

Fernando *et al.*, 2019 introduce a hierarchical memory network (HMN) architecture, which is able to successfully detect faked faces.

A pretrained residual neural network (ResNet) extracts visual facial features from the input images, next the convolutional feature maps are rearranged in a sequence and passed through a bidirectional GRU (as in Cho *et al.*, 2014) to map their relationships. Since not all output components contribute equally to the representation of the dominant attributes of a face, the authors introduce an attention mechanism that learns to pay varying levels of attention to different parts of the feature map, therefore producing an output query vector that highlights the most relevant information. Finally, the memory module outputs information regarding the authenticity of the face region. The authors test their method on the FaceForensics (as introduced by Rössler *et al.*, 2018a), FaceForensics++ (as introduced by Rössler *et al.*, 2019), and FakeFace in the Wild (FFW, as introduced by Khodabakhsh *et al.*, 2018) datasets. The model achieves 99.43% accuracy on seen attacks, and 84.12% and 86.53% on Deepfake and FaceSwap unseen attacks on FaceForensics++ dataset.

Dang *et al.*, 2019 propose the use of an attention layer, which can be inserted into any network architecture to improve the selection of relevant feature maps of manipulated face images by focusing the network's attention on discriminative regions. It takes the high-dimensional features extracted by the convolutional network as input and estimates an attention map using a new technique called manipulation appearance model (MAM). MAM assumes that any manipulated map can be represented as a linear combination of a set of map prototypes calculated from predefined average map and basis functions of maps (i.e., the first 10 components of principal component analysis applied to 100 ground-truth manipulation masks computed from FakeApp). Next, the high-dimensional feature map is multiplied with the attention map and fed back into the backbone. The experiments on Xception (introduced by Chollet, 2016) and VGG16 (introduced by Simonyan and Zisserman, 2014) networks show that using an attention mechanism improves the detection on both backbones, with a slight increase of the AUC and True Detect Rate (TDR) compared to the same architectures without the attention mechanism.

Double JPEG compression. Many digital images are JPEG compressed instantly when captured or uploaded on the web, which, by

the lossy nature of such compression operation, eliminates or changes high-frequency signals within the image. Even though JPEG compression deletes some fine-grained traces, quantization still leaves traces, and researchers came up with techniques that make use of these traces to detect image manipulations. Mandelli *et al.*, 2020a show that CNNs are extremely delicate in the multimedia-forensic scenario. If we train a CNN considering only uncompressed images, it fails when applied to compressed ones. If we train a CNN only considering a specific JPEG grid alignment, it will fail on randomly cropped images. Conversely, computer-vision tasks involving image analysis and understanding are inherently more robust to JPEG compression, given that image visual quality remains good. Park *et al.*, 2018 exploit a CNN that takes the Discrete Cosine Transform (DCT) domain of the Y channel (i.e., the luminance component of the image converted into YCbCr color space) of an image block and its histogram feature as input. The network consists of four convolutional layers, three max pooling layers, and three fully connected layers. The quantization table from the JPEG header is concatenated with the last max pooling layer and two fully connected layer activations. The authors generated a dataset of 18,946 RAW images (starting from the previous works of Gloe and Böhme, 2010, Bas *et al.*, 2011, and Dang-Nguyen *et al.*, 2015) from 15 different camera models and split the images into 256×256 blocks. The single JPEG blocks were produced by compressing each RAW block with a randomly chosen quantization table, and the double JPEG blocks were produced by further compression with another random quantization table with a quality factor between 51 and 100. The experiments were performed using 1,026,387 patches for training and 114,043 patches for testing. The proposed method achieves 92.76% accuracy, with 90.90% true positive rate and 94.59% true negative rate on the test set. Similarly, Wang and Zhang, 2016 propose a network composed of two convolutional connections followed by two pooling connections and three full connections to classify the DCT coefficient histograms. Still in the same direction, Niu *et al.*, 2021 use a local estimate of the primary quantization matrix to distinguish between spliced regions taken from different sources. Splicing detection is achieved by recognizing the presence of more than one clusters according to the estimated primary quantization matrix. To

determine the number of clusters, the authors train a CNN taking as input the estimated quantization steps.

Until now, we have presented techniques for image manipulation. In the last few years, the forensics community has put its attention to video manipulation as well. Next we discuss some approaches.

Inter-Frame Video Forgery. *Inter-frame video forgery* can be applied to manipulate a video by introducing or removing frames to alter the content of the video for malicious purposes. Long *et al.*, 2017 propose a tool for video frame drop detection based on a 3D convolutional network (C3D) similar to the one proposed in Tran *et al.*, 2014. At training time, the C3D-based network takes 16-frame video clips extracted from a video as input to detect if there has been a frame drop. At test time, the video is decomposed into a sequence of continuous 16-frame clips that are fit into the network to obtain the output scores. The authors measure the performance of the proposed approach on the YFCC100m (Kalkowski *et al.*, 2015) and the Nimble Challenge 2017 (see Section 7.1) datasets, achieving 99.83% Area Under the ROC Curve (AUC) – see Section 6 for the definition of AUC – and 96.0% AUC, respectively. Analyzing the predictions of their method, the authors comment on the failed cases. Some false positives are caused by camera shakes during the video capture and some false negatives occur when the scene has almost no visible changes between two frames. The same C3D network is used by Bakas and Naskar, 2018, who introduce an initial pixel-wise difference layer to generate spatio-temporal features. This pixel difference provides temporal information about a video sequence. Because inter-frame forgery is a form of temporal domain forgery in videos, therefore feeding raw image pixels directly as inputs to CNN does not provide efficient performance. The authors generate test forged video sequences from the UCF101 dataset (as introduced by Soomro *et al.*, 2012). On the experiments, the proposed method achieves a maximum accuracy of 99.35% on frame insertion forgery detection, a maximum accuracy of 95.89% on frame deletion detection, and an accuracy between 97.86% to 98.4% for frame duplication forgery detection. In a later work, Long *et al.*, 2018 propose a coarse-to-fine CNN for frame duplication detection and localization based on the I3D network intro-

duced by Carreira and Zisserman, 2017. The pretrained I3D network is used to extract a 1024-dimensional feature vector for 64 frame sequences because the input for the standard I3D network is 64 RGB-data and 64 flow-data. The network is used to obtain the candidate duplicate sequences at a coarse level for a faster search through longer videos. Then, at a finer level, they apply a Siamese architecture (introduced in Section 3.3.1) composed of two ResNet networks (He *et al.*, 2015), to detect duplication at the frame level and to obtain accurate corresponding pairs of duplicated and selected original frames. Finally, when a duplication is detected, a temporal localization is determined with an I3D-based inconsistency detector to distinguish the duplicated frames from the selected 16-frames input video clips. The authors evaluate the proposed C2F-DCNN method on a self-collected video dataset and the Media Forensics Challenge 2018 (MFC18) dataset (which consists of two subsets: Dev dataset and Eval dataset). The method achieves an AUC score between 81.46 to 84.05 on the self-collected dataset, 99.66% on MFC18-Dev and 98.02% on MFC18-Eval.

Yao *et al.*, 2017 convert the input video frames to motion residual images by means of an absolute difference algorithm. The proposed algorithm works in three steps: (1) it converts each frame of the video sequence into a gray-scale image, (2) starting from the second grayscale image, it subtracts its previous grayscale image from the current ones, and (3) finally it takes the absolute value of the subtracted result to obtain absolute difference images. A max-pooling layer and a high-pass filter are used to make the network robust to the variations on motion residual values of the frame absolute difference image and reduce the impact caused by video object motion between video frames. Next, five convolutional layers each one followed by batch normalization, ReLU, and average pooling extract relevant feature maps. Finally, a fully connected layer and a softmax layer classify the feature maps. On the SYSU-OBJFORG data set (Chen *et al.*, 2016) the model achieves 96.79% frame accuracy and 94.08% recall.

As we have seen, deep-learning architectures can be effectively adopted to spot fake images or videos. Single-stream neural networks are surely the most simple and standard application of deep-learning, but most complex models could be designed to analyze several traces

or inputs left in the manipulated multimedia file. The next section will discuss more complex architectures for specific attack detection.

3.1.2 Multistream Architectures

Manipulation techniques usually leave subtle traces that cannot be easily detected by extracting only a specific type of feature maps. Most commonly in forensics problems, it is useful to analyze an image or video by extracting different types of information and combining them together. Therefore, most of the time, multistream networks are used to exploit both low-level and high-level features left by these attacks. To do this, the network is typically structured with two or more backbones in parallel (see Figure 3.4). Each stream analyzes the input looking for a specific trace. The models that we review in this section have been proposed to recognize image forgeries such as double JPEG compression, copy-move, and splicing, and to detect video forgeries such as double MPEG compression and video temporal splicing.

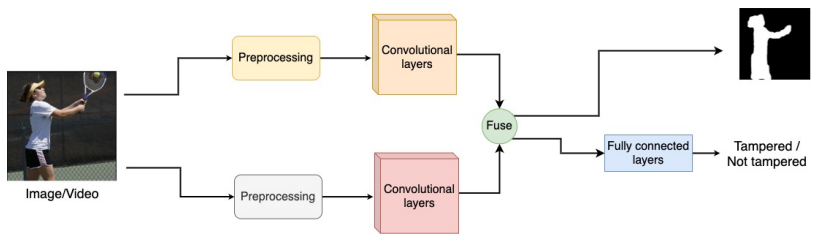


Figure 3.4: Manipulation-specific multistream architectures. The input image, patch, or video frame can be analyzed with two or more backbones in parallel. The input can be preprocessed in different ways to combine both low-level and high-level features left by forgeries.

Double JPEG compression. Amerini *et al.*, 2017b design and compare the effectiveness of three approaches and different inputs to detect double JPEG compression. First they use a spatial domain-based CNN made of two convolutional blocks and two fully connected layers to perform image forgery detection starting from the RGB color images. Second, they propose a frequency domain-based CNN taking the histogram of the DCT coefficient as input as done in Wang and Zhang, 2016. Last, they introduce a two-stream multidomain based network combining the two previous pieces of input information on

RGBs patches and on DCT histograms. The CNN is trained to learn the inter-modal relations between feature maps coming from the RGB-domain, and the frequency-domain features from the histogram of DCT coefficients extracted from the second stream. The fully connected layers of the two networks are joint together for classification. The authors test their proposal on the the UCID dataset) by considering 8 diverse JPEG quality factors (QF) ranging from 60 to 95 and extracting 28,944 64×64 image patches for test. Results (over 95% accuracy when QF is higher than 80) suggest that there is a significant improvement of the performance on the multidomain approach, suggesting that the two inputs provide complementary information that the third, hybrid model is able to correlate and exploit.

Copy-move. Copy-move forgery is a very common and easy to perform image manipulation technique. Because the cloned image patch comes from the same photo, the image characteristics remain largely consistent; this makes this forgery attack more difficult to detect. In this respect, Wu *et al.*, 2018 introduce a novel two-stream deep neural architecture called BusterNet. The first stream called Mani-Det is designed to detect manipulated regions and the second branch, called Simi-Det, is used to detect cloned regions. The Mani-Det network extracts features from the input image using the first four blocks of the VGG16 architecture (Simonyan and Zisserman, 2014), then it uses deconvolution similarly to Noh *et al.*, 2015 so as to restore the original resolution applying BN-Inception and BilinearUpPool2D (Wojna *et al.*, 2017). The output of this network is a pixel-level manipulation mask. The Simi-Det network takes an input image, extracts features similarly to Mani-Net, computes feature similarity via a self-correlation module based on Pearson correlation, and collects meaningful statistics to identify matched patches via a percentile pooling layer. This layer standardizes and sorts the similarity score vector by only picking those scores at percentile ranks of interests. Then, Simi-Net performs up-sampling so as to restore the original resolution and classification similarly to Mani-Net. Finally, the Fusion classification-network combines the information coming from those streams to predict pixel-level copy-move masks differentiating pristine, source copy, and target copy classes. The authors evaluated the performance of the proposed network architecture on the CASIA

and CoMoFoD (Tralic *et al.*, 2013) datasets, with the model achieving 75.98% F1-score.

In a similar fashion, Barni *et al.*, 2019 propose a two-branch CNN architecture, called DisTool. The first branch, named 4-Twins Net, consists of two parallel Siamese networks and the second stream is still another Siamese network. Initially, the authors cast the problem into a hypothesis testing framework whose goal is to decide which region between the two nearly duplicate regions detected by a generic copy-move detector is the original one. Thus, they carry out a preliminary step before running the two networks to identify the input copied and the original regions. Next, the 4-Twins Net takes as input the two candidate regions and is trained to exploit the noninvertibility of the copy-move process caused by the interpolation artefacts often associated to the copy-move operation. The second Siamese-network branch is designed to identify inconsistencies present at the boundary of the input copy-moved region. The soft outputs of the two branches are finally merged through a simple fusion module. DisTool achieves 75.86% and 86.26% of accuracy on CASIA and Grip (Cozzolino *et al.*, 2015b) datasets.

Splicing. Salloum *et al.*, 2018 propose a solution to detect splicing manipulations. They implement a VGG16 architecture with skip connections based on the work by Long *et al.*, 2014 and train it in a multitask fashion. The network takes in input spliced RGB images and then it separates into two branches: one branch learns how to perform classification and the other branch learns the edge or boundary of the spliced region.

Video double compression. Multi-stream networks are also employed for video-forgery detection. Double compression is a possible signal of a video alteration. Nam *et al.*, 2019 propose an I-frame based network (IF-N), which performs double compression detection starting from decompressed input intra-coded frames (I-frames). They also describe a two-stream network (TS-N), which learns inter-modal relations between features extracted by the P-frame based network (PF-N) presented in He *et al.*, 2017 and IF-N. Randomly generated pairs of P-frames and I-frames from a given video clip are fed into two streams. The features learned from the two streams are combined to form a joint feature, which is subsequently fed into a fully connected network for

classification. The network achieves an average accuracy of 93.24% on several datasets generated by the authors.

Video splicing detection. Verde *et al.*, 2018 employ a two-stream architecture to detect video temporal splicing leveraging video codec traces as an asset for forensic purposes. Specifically, they focus on the detection of video temporal splicing. Each video frame is split into nonoverlapping color patches and fed into a CNN tailored to detect whether each patch comes from a video encoded using MPEG2, MPEG4, H264, or H265. One CNN is trained to extract codec-related information and a second one to infer the compression quality level. Finally, the architecture concatenates the features produced by the two CNNs into a vector for each frame and analyzes the inconsistencies between adjacent feature vectors in a temporal domain to detect and localize attacks. The authors tested their model on a dataset composed by 100 spliced and 100 original videos, providing an AUC score of 96.0%.

Multi-stream architectures can analyze the input image or video extracting several features maps, thus allowing for a more sophisticated analysis of the input. Until now, we have discussed architectures detecting specific attacks. However, multistream detectors can be used to spot multiple manipulations at the same time, and have been applied to both recognize and distinguish between them. We present these models in the next section.

3.2 Multimanipulation Multistream Architectures

The goal of generic multistream architectures is to detect every kind of manipulated content without focusing on specific manipulation techniques. In this respect, different features are usually extracted in parallel and combined to increase detection performance. A common choice is to use a stream that analyzes the input in the spatial domain while the other looks for forgeries in the frequency domain. Hence, the general pipeline for detecting different manipulation classes involves two or more streams that analyze the input in different ways; subsequently the output of these streams is combined and fed to a classifier or subjected some other convolution or deconvolution operations. Figure 3.5 shows an example of these architectures.

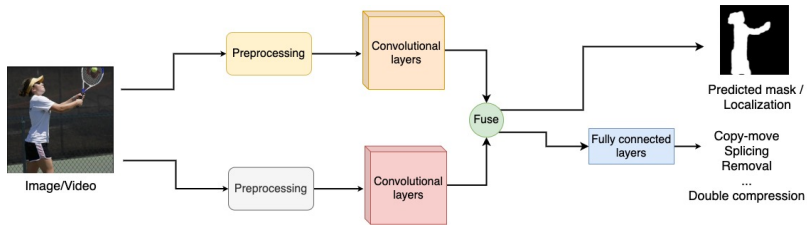


Figure 3.5: Multimanipulation multistream architectures. Multiple features are extracted and combined to increase the detection performance. Differently from multistream manipulation-specific architectures, the network learns to predict several manipulation classes.

The manipulation of an image could leave different types of traces in the form of low-level features, such as noise imperfections, and high-level features, such as pixel imperfections between tampered and authentic regions. A number of solutions combine both these features in a two-stream neural network to find tampering artifacts.

Zhou *et al.*, 2018a propose a two-stream Faster R-CNN architecture based on the Faster R-CNN (as introduced by Ren *et al.*, 2015 to detect manipulated regions in an image. The model consists of an RGB channel that extracts tampering artifacts such as strong-contrast difference or unnatural boundaries between the authentic and manipulated region, and an SRM noise stream (inspired by Fridrich and Kodovsky, 2012) that detects the inconsistency between authentic and tampered regions. A bilinear pooling layer takes as input the features coming from the two streams to further incorporate spatial co-occurrence of these two streams. Experimental results on several datasets such as NIST 2016, CASIA, Cover (see Section 7.1), and Columbia show the robustness of this solution to JPEG compression and resizing attacks. The model achieves 93.4% AP on NIST 2016 (96.0% on splicing, 93.9% on removal, and 90.3% on copy-move). Still a two-stream architecture is perfected by Zhou *et al.*, 2018b to detect face tampered pictures. The first stream, a face classification stream, is a CNN based on GoogleNet (Szegedy *et al.*, 2015) trained to classify whether a face image is tampered or authentic. The second stream is a patch triplet stream trained on steganalysis features of the image patches. This stream uses a triplet loss to model the traces left by in-camera processing and local noise characteristics. The

triplet loss function helps the network to learn to reduce the distance between patches from the same image in the learned embedding space and increase the distance between two patches from different images. A support vector machine (SVM), according to the framework introduced by Cortes and Vapnik, 1995, is trained to classify the learned features on each patch. Finally, the scores of two streams are fused to recognize a tampered face. The method achieves 92.7% AUC on the test dataset introduced by the authors.

Some multistream architectures analyze low-level features in image patches to find discrepancies in the content. Bappy *et al.*, 2017 present an architecture that extracts patches by sliding windows and then feeds them to two convolutional layers, which extract low-level features. A long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) examines the correlation between the blocks in the 2D map and learns the boundary transformation between blocks. A fully connected layer reconstructs the segmented area of the manipulation. Stemming from this work, Bappy *et al.*, 2019 introduce an encoder–decoder architecture, which allows to understand the appearance, shape and spatial relationship (context) between manipulated and authentic regions. When the network receives as input a new RGB image, it first extracts resampling features from patches. These features pass through an LSTM stream, which learns the fingerprints of a manipulation in the frequency domain. In parallel, an encoder extracts low-level features from the input image in the RGB-domain. Finally, by combining the two streams, the decoder learns the finer details of authentic and manipulated classes. Mazaheri *et al.*, 2019 improve this framework by introduction skip connections, that is, reformulate the layers as learning residual functions with reference to the layer inputs like He *et al.*, 2015. Unlike the work by Bappy *et al.*, 2019, a U-Net architecture (inspired by Ronneberger *et al.*, 2015) is used for the encoder–decoder network, showing the effect of layer fusion. Skip connections help traverse information in deep neural networks. Usually, manipulated regions in an image have smooth boundaries. In this respect, skip connections take advantage of early layers in CNN which are rich in spatial details and help to find the layers which are rich in features for forgery detection. The authors tested the model tested on NIST 2016 and CASIA datasets, achieving 85.7% and 81.4% AUC, respectively.

Even though these solutions achieve state-of-the-art performance, an intrinsic limitation of this approach is the high number of samples needed to train such models. Unfortunately, although researchers in the last years have created different multimedia forensic datasets, none of these datasets is still big enough to be compared with the ones typically used for object detection challenges. An additional intrinsic limitation of these models is the required coherence between training and testing set is almost impossible to reproduce for real-world applications due to the infinite number of variables that can be encountered.

3.3 Anomaly-Based Architectures

Usually, multimedia forensic approaches focus on identifying or classifying a particular forensic trace. The main drawback of this approach is that it required training samples from a particular trace. Editing tools and techniques have been evolving continually during the last years, making it even harder to consider every possible manipulation. Therefore, training learning models can be a challenging problem because of the lack of sufficient amounts of manipulated training samples. An alternative approach to training models on data samples is to consider tampered contents as anomalies, which can be recognized by comparing them with authentic images or videos.

3.3.1 One-Class Siamese-Based Architectures

Siamese networks (as introduced by Bromley *et al.*, 1993) have been widely adopted in the past few years showing really promising results. These networks, sometimes also called twin neural networks, share the same weights while working in parallel on two different input vectors to compute comparable output vectors. An efficient anomaly-detection pipeline can be constructed by feeding Siamese networks with images, frames, or patches from authentic images. The network learns how to distinguish inputs coming from the same or different camera models, image or video, thus allowing to recognize forgeries as anomalies. The strength of such an approach is that it is possible to build a model

that does not require forged samples at training time, thus allowing for simple training datasets without loss of fake detection. Figure 3.6 shows an example pipeline.

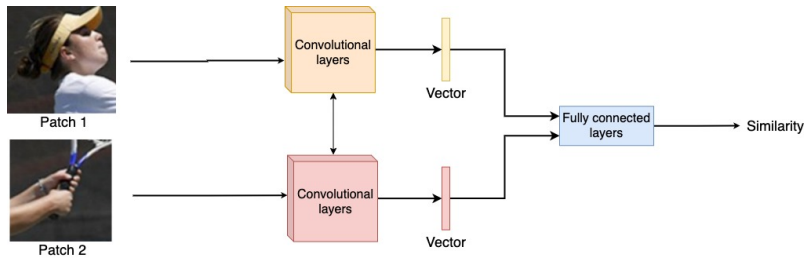


Figure 3.6: An example pipeline for one-class Siamese-based architectures. Two or more convolutional neural networks are used in a Siamese configuration (i.e., sharing weights and biases) to detect inconsistencies between two inputs. Typically, the inputs are real image or frame patches. The output vectors predicted by the CNNs are compared to estimate their similarity. In this configuration, forgeries are detected as anomalies with respect to real samples.

Low-level feature similarity can be used on pristine image patches to train neural networks capable of capturing strange anomalies. Mayer and Stamm, 2020 propose a two-part deep-learning system called forensic similarity. A Siamese feature extractor grasps low-level features from image patches and subsequently a 3-layer neural network estimates the similarity between two patches. The authors evaluate the system accuracy of determining whether two image patches are: (1) captured by the same camera model, (2) manipulated by the same or different edit operation, and (3) manipulated by the same or different manipulation parameter. The system is accurate even on unknown forensic traces that were not used to train the system. Mayer and Stamm, 2019 propose a solution that can be used to spot localized image tampering such as splicing and airbrushing. The authors propose a graph structure where small image patches are represented by graph vertices and edges connecting pairs of vertices are assigned according to their forensic similarity. Communities in the forensic similarity graph correspond to the tampered and unaltered regions in the image. Forgery detection is performed by identifying whether multiple communities exist, and forgery localization is performed by partitioning the communities. The

results on Columbia (see Section 7.1), Carvalho, and Korus (Korus and Huang, 2017) datasets show that the proposed technique exceeds prior-art forgery detection and localization performance on the benchmark datasets.

Other works, such as Cozzolino and Verdoliva, 2020; Cozzolino Giovanni Poggi Luisa Verdoliva, 2019; Cozzolino and Verdoliva, 2018a, use Siamese networks to suppress the content of an image and extract a camera-model fingerprint called Noiseprint, which is a fingerprint that identifies the camera model technology that captured a content; for example, Olympus TG-6 or Canon EOS 6D Mark II. At training time, the Siamese network needs to know only whether two patches come from the same camera and position or not; in this way they take advantage of hidden spatial dependencies to associate a Noiseprint to each image. Being a fingerprint of the camera model (such as the PRNU model), the Noiseprints can be used for a large variety of forensic tasks, such as source identification or forgery detection. As an example, other papers, including Cozzolino *et al.*, 2015a and Rössler *et al.*, 2019, use Noiseprint features in combination with other learning models to detect image manipulations. Experiments in these works show that the Noiseprint-based methods outperform most of the PRNU-based techniques, providing good results even when the fingerprint is estimated on a very small number of images.

Inconsistency can also be observed in the metadata of a manipulated image. Exploring this information, Huh *et al.*, 2018 propose to use EXIF metadata for training a Siamese model to predict whether a pair of image patches are consistent with each other, i.e., they share the same value for each of n metadata attributes. The Siamese network uses two shared ResNet 50 each producing a 4096 dimensional feature vector. These vectors are concatenated and passed through a four-layer fully connected network with 4096, 2048, 1024 units, followed by the final output layer. The network is trained on 128×128 real image patches randomly sampled from 400,000 Flickr photos. Although the consistency score can be noisy for any single pair of patches, aggregating many observations leads to a stable estimate of overall image consistency. Even though state-of-the-art performance is achieved on several datasets – Columbia, Carvalho and Korus –, two drawbacks should be considered:

(1) the model is not easily interpretable, i.e., it is not clear which visual clues the model uses to solve the task, and (2) it is still affected by design decisions that went into the self-supervised task, such as the ways that EXIF tags were balanced during training time.

Siamese architectures have shown a high level of flexibility, making it possible to apply these networks to various problems related to the evaluation of the authenticity of an image or video. These models have been successfully used with different methodologies and demonstrated their effectiveness in defining a similarity measure between input patches, as well as in the ability to automatically extract a fingerprint, which proves or refutes the authenticity of multimedia content and in the analysis of metadata. As such, Siamese networks could be a powerful method to help multimedia forensics researchers to develop general forgery detectors, independently from specific manipulation techniques.

3.3.2 Autoencoder-Based and Recurrent-Neural-Network–Based Architectures

Like Siamese architectures, autoencoders, and recurrent neural networks (RNNs), such as LSTMs, are often used for the recognition of anomalies with respect to a single training class. Autoencoders can effectively learn a latent space representation, which can be forced to separate authentic images for anomalous, fake ones. The idea is that inliers (real examples) tend to share similar statistics, while outliers (fake images) have a much more widespread distribution. If trained on real examples, the autoencoder will learn the inlier dominant distribution. Therefore, in the testing phase, the positive examples are reproduced with good accuracy, while the anomalies give rise to big errors, allowing a reliable detection. RNNs can be successfully applied as well, to capture the inconsistency of different input images, frames, or patches. Figure 3.7 shows an example pipeline of these architectures.

Cozzolino and Verdoliva, 2016 propose an image splicing localization method based on an autoencoder. Using the same local features adopted in their previous work (Cozzolino *et al.*, 2015a) they remodel the problem in terms of anomaly detection, adopting the approach proposed by Xia *et al.*, 2015. The proposed method involves a feature extraction step

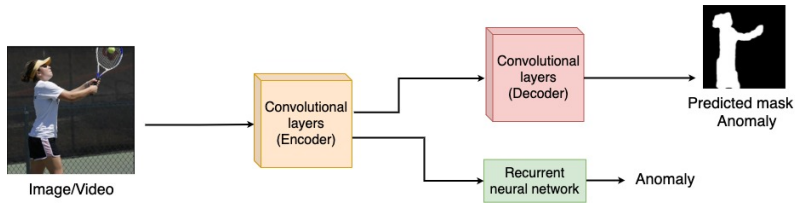


Figure 3.7: An example pipeline for one-class autoencoder-based and recurrent-neural-network-based architectures. An CNN is usually fed with an input image. The CNN encodes the input in a latent representation that can than be used to detect an anomaly. The encoder can be followed by a decoder CNN or a recurrent neural network that eventually predicts inconsistencies.

to preprocess the input image before applying it to the autoencoder. Initially their approach computes image residuals through high-pass filtering (third order derivative). The residual (i.e. the noise extracted by the image content through the previous operation) is split into K patches of $w \times w$ pixels with stride s and a histogram of co-occurrence, that is, the histogram describing the distribution of pixel values, is computed on each patch. The histogram of co-occurrences is processed through a square-root nonlinearity (Cozzolino *et al.*, 2015a) and in the end normalized to zero mean and unit norm. These histograms are used as input to the network. The autoencoder has a single hidden layer followed by hyperbolic tangent. The hidden layer has $H > K$ hidden neurons, in fact by decreasing H in such a way that $H < K$ the authors observed a large number of false alarms, which can be explained by the fact that the characteristics associated with splicing are not scattered and a bottleneck autoencoder would devote some of its scarce resources to encode negative samples and cannot improve the representation of good characteristics initially classified as splicing. The network is trained only authentic inputs so as to reinforce learning for pristine class only, and increase the separation between errors of the two classes.

Wu *et al.*, 2019 treat the forgery localization problem as a local anomaly detection problem. They propose an architecture, which they call ManTra-Net, composed of two subnetworks: and image-manipulation–trace feature extractor, which creates a unified feature representation,

and a local anomaly detection network, which directly localizes forgery regions without postprocessing. The first subnetwork begins with a stack of three layers in parallel: (1) a simple convolutional layer, (2) an SRM convolutional layer (as done by Zhou *et al.*, 2018a, see Section 3.2), and (3) a constrained convolutional layer (as introduced by Bayar and Stamm, 2018, see Section 3.1.1). The features extracted from the previous layers are concatenated and passed through a VGG network (Simonyan and Zisserman, 2014). The output features (also known as feature maps) are then passed through the local anomaly detection network, which is characterized by two novel designs: a ZPool2D DNN layer, which standardizes the difference between a local feature and its reference in the Z-score manner, and a far-to-near analysis, which performs the Conv2DLSTM sequential analysis on ZPool2D feature maps pooled from different resolutions. The network performed well on four benchmark datasets, namely NIST 2016, CASIA, Cover, and Columbia (see Section 7.1 for detailed description of these datasets), reaching 79.5% AUC on NIST.

Yarlagadda *et al.*, 2018 propose an autoencoder architecture composed of five convolutional layers and five symmetric deconvolutional layers, which are trained on pristine satellite image data. The output goes thorough a one-class SVM classifier that detects forged patches as anomalies with respect to feature distribution learned from the autoencoder. Next, the authors compare the performance of their classifier by introducing a GAN model where the discriminator is aimed at accurately distinguishing between patches from real satellite images and patches created by the generator.

Regarding videos, some researchers have proposed architectures based on deep learning for video forgery detection and localization techniques. However, a lot of works concentrate their attention on specific tampering artifacts. Therefore, it becomes important to develop new solutions that are capable to detect different types of video forgery attacks. In this direction, D'Avino *et al.*, 2017 use an anomaly-detection tool that makes use of an architecture based on autoencoders and RNNs; these allow to take into consideration the temporal dependencies of the different frames of the video. By passing a 2D sliding window over the input, using image patches of 128×128 pixels taken with stride 8 (i.e.

each 128×128 patch volume is shifted across the input 8 units at a time), the network computes residuals using a third-order derivative, quantization, and truncation, following previous works, such as those of Verdoliva *et al.*, 2014; Cozzolino *et al.*, 2015a; Cozzolino and Verdoliva, 2016. A recurrent autoencoder anomaly detector receives each residual and computes an anomaly score. All the scores calculated are then projected back onto the image domain and aggregated to produce a global heat map, which can be used as the basis to detect spliced regions. Therefore, at test time, whenever a pristine feature vector is presented in input, the network succeeds in reproducing it with a small error, as opposed to the feature vector of spliced content, which is not well captured by the learned network parameters, and is reproduced with a larger error. The proposed method performs quite well on both compressed and uncompressed videos (the ROC curve has just 10% false positive rate when the true positive rate is around 90%) and it provides a large improvement with respect to PRNU-based methods.

The anomaly-based detectors have been applied with excellent results on several forgery detection tasks. These models can be strongly robust to new manipulations, supporting the design of general-purpose forgery detectors. Autoencoders can be designed to learn a forensic embedding space that correctly separates authentic contents from fake ones. RNNs can be similarly used to distinguish between authentic and anomalous content as well, due to their ability to notice the difference between local features in image patches or video sequences. Overall, anomaly detectors could be the secret to detect any kind of manipulations and has been an increasing trend in the last few years.

3.4 Challenges and Best Practices for Forgery Detection

The methodologies examined in this section highlight three main approaches: (1) manipulation specific, (2) multimanipulation, and (3) anomaly-based methods. There are some pros and cons to consider in all of these cases, as well as many similarities and common choices. In the next paragraphs we comment on them.

A common thread that unites many of the methods described concerns the backbones chosen to design the different deep-learning so-

lutions. Variations of the ResNet, Xception, and VGG networks are very common. Also, a common choice is to preprocess the input with some form of high-pass filter or some other noise fingerprint extraction such as Noiseprint. Another common choice is the use of hyperbolic tangent activation function and average pooling layers. Finally, it is very common to measure the performance of these models against the AUC, AP, or F1-score. Some researchers prefer the accuracy; however, this metric is not very robust with respect to unbalanced datasets.

Manipulation-specific solution approaches are typically based on single-task or multi-task architectures. These often use an attention mechanism to detect manipulated regions by discarding unaltered regions of the input. The constrained layer introduced by Bayar *et al.*, 2017 has been very effective, as well as the use of 1×1 convolutional filters after the highest-level feature maps followed by an average-pooling layer to retain the most representative features from the deepest convolutional layers in the network. Scaling, is also used quite commonly, which risks destroying valuable high-level information that is very useful for identifying manipulation. We find two possible solutions for addressing this issue: the addition of skip connections (Bi *et al.*, 2019; Salloum *et al.*, 2018; Mazaheri *et al.*, 2019) within the architecture and the use of a patch-level feature extractor followed by an aggregation layer, as suggested by Marra *et al.*, 2019b. The first approach, similar the one introduced by the ResNet architecture, is very fast, and it is designed to propagate low-level features extracted by shallow layers to deeper layers of a convolutional neural network. The second one was explicitly designed to cope with manipulation detection. In fact, the network is trained with no information on what to look for to find a manipulation. In particular, the model is trained to classify all patches of an image as manipulated if at least one of them has been tampered. In this way the network receives conflicting information between authentic patches of an entirely authentic image and authentic patches that are part of an image that has been manipulated. Consequently, the network must learn to make a high-level decision on the whole image to decide if it has been forged.

Because the traces left by a manipulation can be very faint and difficult to recognize, a more robust approach is to analyze the input in

different ways. This is an approach that we find in both manipulation-specific and multimanipulation architectures. A common choice is to analyze spatial and frequency features across multiple streams. These multiple streams can analyze RGB, noise or DCT features (Amerini *et al.*, 2017b; Zhou *et al.*, 2018a; Bappy *et al.*, 2017; Zhou *et al.*, 2018b; Mazaheri *et al.*, 2019). Moreover, multiple streams can be used to analyze different tasks in parallel such as the case of copy and move in which very often one stream deals with looking for the manipulated areas and another for the cloned areas within the image (Wu *et al.*, 2018).

Over time, manipulations have become more sophisticated and the emergence of new techniques makes increasingly difficult the development of detection solutions for specific manipulation techniques. This is particularly important in practical applications where the variety of manipulations cannot be controlled and in cases that are not limited to a specific attack. Although multimanipulation techniques offer a useful first defense against these cases, the approaches based on anomaly detection seek to offer a robust solution to any type of manipulation. This is possible through two approaches: Siamese networks and autoencoders. These approaches are very recent but have shown great potential: they can be trained on unmanipulated image datasets. Inconsistencies can be found in the noise of the image, on the spatial domain, or in metadata. Thanks to these characteristics, methods based on anomaly detection are attracting a lot of interest from researchers and could become the key to bringing forgery detection into practical applications.

Tables B.1 and B.2 summarize some of the most interesting architectures discussed in this section.

4

Assessing the Origin of Multimedia Content

Source identification is another fundamental forensic problem. Sometimes, we are not only interested in addressing the authenticity of multimedia content, but we also need to reconstruct the source that originated that image or video. Source identification can tell us whether two pictures or videos come from the same camera, or even from the same smartphone device. This could be important to reconstruct facts from crime scenes, to check the truthfulness of news, or demonstrate that a multimedia file has been downloaded from a social-media platform. Figure 4.1 shows an example of the typical image acquisition pipeline that is common for most of the commercially available devices.

There are three main categories of source-identification problems:

- The identification of the camera model or brand identification, which can tell us if an image or video has been compared with a specific camera set
- The identification of the specific device that has captured the content.
- The identification of the social network and messaging app, which tells us whether a picture or video comes from a social media platform.

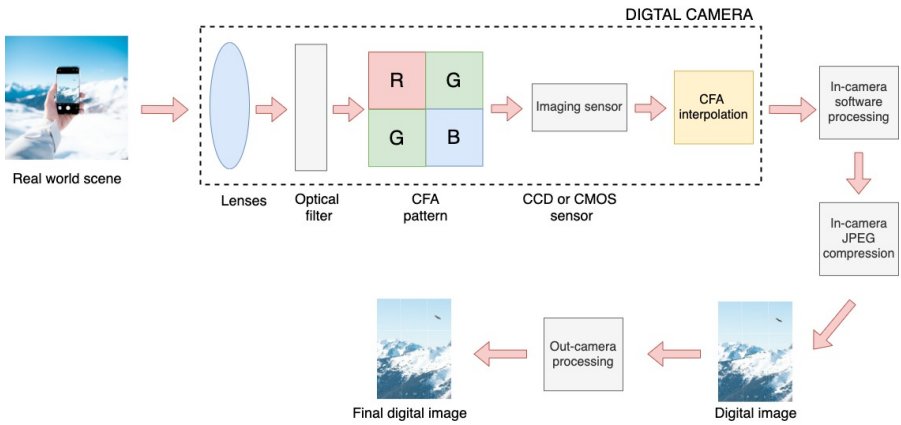


Figure 4.1: The image acquisition pipeline is common for most of the commercially available devices. Each phase leaves distinctive traces. When light passes the camera lens and the optical filter, the scene captured by the camera can be distorted. The color filter array (CFA) is a thin film that selectively permits a certain component of light to pass towards the sensor. In practice, for each pixel only one primary color is gathered. The sensor output is successively interpolated to obtain all the 3 colors for each pixel (demosaicking). Different CFA patterns and demosaicking algorithms can be used in this process. The CCD or CMOS sensor, that is the image sensor where each pixel sensor unit cell has a photodetector, captures photons until a certain level (pixel intensity) is reached. Practically, small variations in cell size and substrate material result in slightly different output values, introducing some noise into the image. Finally, the image is encoded in digital files according to a file format defined by a compression standard. Sometimes, out-camera processing steps can be carried out, like, for example, image re-compression when it is shared across social media platforms.

Designing deep-learning solutions to answer these challenges is not trivial, and requires new techniques to be developed. The next parts of this section discuss the design choices and works that have been proposed to address these challenges with deep learning. Table B.3 summarizes the main architectures. Figure 4.2 summarizes the approaches described in this section.

4.1 Camera-Model Identification

Digital cameras carry out a sequence of processes to convert an input light coming into the lens into an output image. While some of these

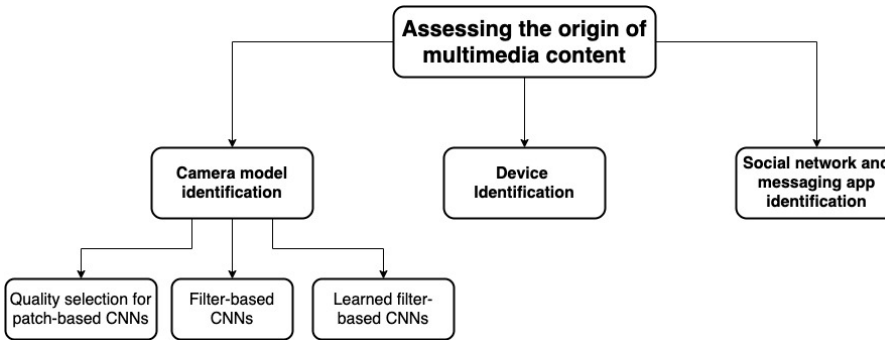


Figure 4.2: An overview of source identification approaches.

processes like interpolation, gamma correction or data compression are common to every camera model, all of these functionalities have different implementations that can vary from model to model. Due to all these internal processing steps, each camera model leaves important traces on acquired pictures that can be used for forensic purposes.

We open this section by presenting the works that proposed methodological approaches for camera model identification in general, tracing some of the key guidelines that one could keep for the designing of camera-model-identification systems. These papers can be very useful in recognizing the most common mistakes and most valuable choices to reinforce the subtle traces left by cameras.

4.1.1 Quality-Selection Procedure for Patch-Based CNNs for Camera-Model Identification

Deep-learning models that are successful for other computer-vision tasks, can be used to spot camera model fingerprint traces. However, they often need to be adapted to embrace to satisfy forensic needs.

As suggested by Bondi *et al.*, 2017b, particular attention must be paid to the process of selecting the patches that are provided as input to the models: indeed, not all patches contain enough statistical information about the used camera model. As an example, saturated and homogeneous areas with low-texture and low-frequency components do not contain significant statistical information about the used camera model. Therefore, Bondi *et al.*, 2017b introduce a patch selection

procedure based on a quality value selection. This quality measure gives lower scores for overly saturated or flat patches, and higher for textured patches showing some statistical variance. Then, following to the architecture design guidelines proposed by Szegedy *et al.*, 2015, the authors propose a base-model CNN (Bondi *et al.*, 2017a) made of a stack of four convolutional layers and a final inner product layer. Each of the first three convolutional blocks are made of a convolutional layer followed by a max-pooling layer. The last convolutional block gives as outputs a 128-dimensional vector and is followed by an inner product layer, a ReLU layer, and a soft-max operation. As shown by Simonyan and Zisserman, 2014 and by Szegedy *et al.*, 2014, the representational capacity of a network is largely determined by its depth. Thus, the base model is compared with four network variations in which varies only the depth of the three convolutional blocks. The results on the Dresden dataset (described in Section 7.2) suggest that higher classification accuracy (up to 94.93%) can be obtained as the network depth increases until an optimal value. Therefore, deeper CNNs might be beneficial for larger datasets with a higher number of classes and training images.

The process described by Bondi *et al.*, 2017b has inspired several researchers, who have proposed deep-learning-based camera-model identification pipelines using the same patch selection process. Even though the works in this section are not the only ones to adopt patches as input for their models, they are all based on the selection process by Bondi *et al.*, 2017b.

A lot of papers incorporate the process of selecting patches from the same or similar patch quality, by Bondi *et al.*, 2017b, into their pipeline. The selected patches are analyzed by a CNN that can be made of any number of streams of well-known architectures.

Ferreira *et al.*, 2018 propose a solution that consists in the processing of the output of two CNNs, based on the Inception networks (Szegedy *et al.*, 2014). First, patches are extracted and selected based on quality measures similarly to the approach of Bondi *et al.*, 2017b; then, the image patches are preprocessed by two pretrained streams based on Inception v4 (Szegedy *et al.*, 2016) and Xception (Chollet, 2016). The 256-dimensional feature vector outputs of the inception-based CNNs are merged creating a 512-dimensional input to be processed by a final

CNN. This CNN is composed of a flattening layer, a batch normalization layer, a fully connected layer with 256 neurons, a 10% dropout layer, and a final fully connected layer. Test results on original full-resolution images and 512×512 blocks from the IEEE Signal Processing Challenge on camera model identification (see Section 7.2) show that the model achieves an accuracy of 89.83% for the classification of blocks and 92.51% for the classification of images.

Still using the same patch selection procedure presented in Bondi *et al.*, 2017b, Rafi *et al.*, 2018 propose a solution based on the dense convolutional networks (DenseNets) of Huang *et al.*, 2016 and on squeeze-and-excitation (SE) as introduced by Hu *et al.*, 2017. In the DenseNet, the output of a certain layer is propagated to all the layers in front of it. As a result, if any of the input patches features are lost during forward propagation, they are spread at the input of later layers through the dense connections, making this architecture suitable for detecting weak statistical features such as those related to camera identification. First, patches of size 256×256 are selected to train the DenseNet-201 (Huang *et al.*, 2016). Then, using this model trained on 256×256 patches only, they extract features from the second to the last layer for patches of size 256×256 and nonoverlapping patches of size 128×128 and 64×64 from each training sample, thus producing three feature vectors for three different patch sizes. Next, the feature vectors extracted by the network on those inputs are concatenated and used to train an SE block and a classification block. The output of the SE block is passed through a dropout layer and a global average pooling layer, a layer that reduce each $h \times w \times c$ feature map to a single vector of size $1 \times 1 \times c$ by simply taking the average of all hw values. Finally, a softmax layer generates a probability distribution on the output classes. The authors evaluate their solution on a dataset of 2640 images of size 512×512 from the IEEE Signal Processing Challenge on camera model identification dataset (see Section 7.2), achieving 98.37% accuracy, and on the Dresden database with an overall accuracy of over 99%. Similarly, the DenseNet-161 (Huang *et al.*, 2016) architecture is used by Kuzin *et al.*, 2018, where the input images are randomly cropped to generate patches and augmented with a set of transformations (Dihedral Group D4, gamma, JPEG compression, and scale transformations).

As shown by Bondi *et al.*, 2017b, not all input patches contribute equally to the camera model classification. A preprocessing quality-selection step can bring about important performance enhancement, and it is extremely easy to introduce in every camera-model identification pipeline.

4.1.2 Filter-Based CNNs for Camera-Model Identification

Filters have been largely adopted to extract latent camera model fingerprints. Traditionally, these transformations have been used in combination with forensic techniques mostly related to machine learning. However, deep-learning models can be successfully applied to analyze these preprocessed inputs and extract the most relevant characteristics that can be used to classify different camera models. In general, the overall architecture that uses static filters provides a preprocessing module that applies static filters on an image or patch. Then, this input is analyzed through a CNN module and a classifier.

Tuama *et al.*, 2016 use a high-pass filter layer followed by three convolutional layers and three fully connected layers. For each input image, the residual noise on each color channel is extracted by subtracting the denoised version of the image from the image itself. Then, a static denoising high-pass filter (Qian *et al.*, 2015) is used on the input image to suppress the interference caused by image edges and textures so as to obtain the image residual. AlexNet CNN (Krizhevsky *et al.*, 2012) is adapted and modified to fit the model requirements. First, three convolutional layers, each one followed by ReLUs, extract feature maps. Then, a max-pooling operation is applied to decrease the spatial resolution. At the end there are two fully connected layers followed by ReLU activations and a final fully connected layer followed by a softmax function. For the evaluation, the authors used a dataset made of 27 camera models from the Dresden dataset, and 6 personal camera models, reaching 98.0% accuracy.

Bayar and Stamm, 2017a propose a novel approach for low-level residual feature extraction called *augmented convolutional feature maps*. The input layer of the network is a two-channel image obtained combining the output of two parallel operations. In the first operation, a

256×256 green layer central patch of the input image gets processed by a *constrained convolutional layer* where the central weight of the convolutional filter is constrained to be -1 and the rest of the weights sum to 1. The second operation computes the median filter residual (MFR, Kang *et al.*, 2013) features of the input image. In order to concatenate the features extracted with both operations and to obtain the augmented feature maps, the outputs of the constrained convolutional layer and the MFR features should have same dimension, thus, the MFR channel of the input layer is first convolved with a fixed 5×5 identity filter with a stride of 1. To learn higher-level classification features, the low-level augmented feature maps obtained from the previous steps are directly passed to a sequence of three regular convolutional layers followed by one 1×1 convolutional layer. Every convolutional layer is followed by a batch normalization layer, a hyperbolic tangent activation function and a pooling layer. The pooling layer is a max-pooling layer after all the regular convolutional layers, whereas, an average-pooling layer is used after the 1×1 convolutional layer to preserve the most representative features. The output of the hierarchical convolutional layers is fed to a regular neural network which consists of three fully-connected layers to perform classification. The two first layers are followed by a hyperbolic tangent activation function, whereas the output layer is followed by a softmax activation function. The experimental results on the Dresden dataset show that augmenting feature maps produced by the constrained convolutional layer with MFR features improve with 50% downscaling; more in detail, this can improve the camera model identification rate over an architecture which did not use nonlinear MFR features by 3.69% with JPEG post-compression and by 2.59% without JPEG post-compression.

Pengpeng *et al.*, 2017 propose a generalized model called Laplacian CNNs. The architecture is composed of two parts: the signal enhancement layer and a general CNN structure. Initially, the Laplacian filter is used to amplify the difference between the original images and the recapture images highlighting the regions of rapid intensity change. Then, convolutional blocks extract relevant features that are then classified by a fully connected layer followed by a softmax layer. Inspired by the Laplacian CNN, the same research group (Yang *et al.*, 2017) introduce

a content-adaptive fusion network. The network is built by applying in parallel three convolutional neural networks to capture more comprehensive information. The input image is divided in 64×64 patches and processed in the preprocessing layer by three kinds of convolutional kernel sizes 3×3, 5×5, and 7×7. Tested on the Dresden dataset, the method achieves average 94.17% accuracy for camera brand detection and 84.7% accuracy for camera model identification, and up to 70.19% for device identification.

Filters have long been used to suppress the content of an image or video and extract the latent features left by the camera model. Multimedia forensic researchers have developed or adopted several filters, proven to be effective in extracting camera fingerprints. The power of deep-learning models, however, comes from the ability of deep neural networks to learn both low- and high-level latent features without any guidance. This capability can be exploited to develop new models that learn the best filter extractors for every training dataset. The next section discusses this new trend.

4.1.3 Learned-Filter-Based CNNs for Camera-Model Identification

As we mentioned in the previous section, static filters have been successfully used to extract camera model fingerprints. However, the most interesting feature that distinguishes deep neural networks is the ability to automatically learn nonlinear transformations that effectively map input images or videos to latent feature maps that can be used for classification. Thus, deep neural networks can be used to automatically learn how to obtain a camera model representation.

The common idea in implementing dynamic filters is to train a network on different input patches obtained from images or videos captured by the same or different camera models. The best model to use in such cases is the Siamese architecture, which could be trained as a camera-model fingerprint-extractor network to identify a set of camera models. Then, the extracted representation can be used in any way for classification tasks or similarity measures.

The first step in this direction comes by Bayar *et al.*, 2017. They present a sequence of experiments to define some of the design choices

that one required to develop a camera-model identification CNN-based system. First, the authors show the benefits of using dynamic-learned layers instead of fixed filters to extract lowest-level features. The performance comparison of a high-pass filter and constrained convolutional layer (Bayar and Stamm, 2016; Bayar and Stamm, 2017b) suggests a possible improvement brought by learned descriptors. Then, the introduction of 1×1 convolutional filters after the highest-level feature maps (Xu *et al.*, 2016) force the network to learn the association between the highest-level feature maps in the network before the fully connected layers perform classification. Finally, an average-pooling layer after the 1×1 filter should be preferred over max pooling, because the most-representative features from the deepest convolutional feature map in the network could be destroyed by means of max pooling. Bayar *et al.*, 2017 built an experimental dataset by manually capturing images using 34 different camera models containing at least 300 images captured by each camera with its default settings. The proposed method achieves an accuracy higher than 97% for each model. Similarly, Freire-Obregón *et al.*, 2019 argue about the different elements that support the selection of the proposed CNN architecture comparing the performance of the network varying the number of convolutional layers, dropout rate, and activation function.

Cozzolino and Verdoliva, 2020 train a Siamese network with pairs of image patches coming from the same or different cameras. They train the to reduce output distance of input patches coming from the same camera model and to maximize this distance for inputs coming from different models. Training the architecture with both positives and negatives forces the network to discard irrelevant information common to all models and keep only the most relevant features. More importantly, during the training of their system, they consider two patches as a positive example only if the patches come from the same camera model and from the same position in the image. Generally, artifacts generated by in-camera processes are not spatially stationary and depend strictly on the acquisition and processing phase. As a result of the training choices mentioned above, the network suppresses the scene contents and improve the model-related artifacts that are useful for extracting a camera model fingerprint. The authors call this model "Noiseprint,". The

authors tested their approach on various datasets (the Dresden dataset, the Socrates dataset (Galdi *et al.*, 2018), the Vision dataset (introduced in Section 7.2), and a private dataset with 17 devices), and various forensics tasks (camera model and device identification, identification of JPEG quality factor, identification of demosaicing algorithm, image manipulation detection and splicing in remote sensing images). The model identification accuracy is 100%, both with 1024×1024 and 128×128 crops. Moreover, the method also ensures a modest device identification, with accuracy between 61.7% and 75.3%, even though the PRNU-based (see Section 4.3) method provides much higher accuracy, 91.3% and 69.7%, respectively.

In another work, Cozzolino and Verdoliva, 2018b consider a supervised setting, assuming the reference Noiseprint is available. The residuals of pristine images taken by the same camera of the image under analysis are averaged to obtain an estimate of the camera Noiseprint where high-level scene leakages, as well as traces of the PRNU, are mostly removed. Next, this reference is compared using a sliding window with the residual of the test image using the Euclidean distance as a measure of similarity. The idea to use Noiseprint-based model-related information to support PRNU-based device identification is also applied in a later work by Cozzolino *et al.*, 2020.

Cozzolino Giovanni Poggi Luisa Verdoliva, 2019 show the effectiveness of Noiseprint for video-source identification. They consider two different training ways for extracting the Noiseprint model: (1) Using only the I-frames of the videos and (2) using both I-frames and P-frames. Initially, for each camera model, their system extracts a reference fingerprint averaging the Noiseprints distilled from several frames. Subsequently it extracts a new fingerprint from the video under test; this fingerprint is compared with the reference fingerprint. To train the model, they treat all frames as independent images and they split them in patches as in the previous work by Cozzolino and Verdoliva, 2020. They evaluate the similarity of two Noiseprint fingerprints with normalized correlation coefficient (NCC) and mean squared error (MSE – see Section 6). They measure the performance considering as a base model the PRNU-based procedure with peak-to-correlation energy (PCE) Goljan *et al.*, 2009 as a similarity measure. The method using

only I-frames on the Socrates dataset (Galdi *et al.*, 2018) achieves 83.2% AUC for source identification and an average accuracy of 92.14% on FaceForensics++ (see Section 7.3) for video manipulation detection.

Mayer and Stamm, 2018 define a system that determines whether two input image patches are captured by different camera models. Initially, the feature-extractor network is trained to identify a set of camera models; Next, a similarity network is trained to learn similarity between pairs of high-level features extracted from a second set of camera models. The model is evaluated on the Dresden dataset (Gloe and Böhme, 2010), where the proposed approach detects correctly the camera models as different at a rate of 98%. Similarly, in another work by Mayer and Stamm, 2020, the authors propose a CNN feature extractor grasping low-level features from input image patches and a 3-layer neural network that estimates the similarity of two input patches. Still in the same direction, Mayer and Stamm, 2019 introduce a similarity-graph structure, in which communities in the graph correspond to separate camera models. Building on the same similarity network, Mayer *et al.*, 2020 propose a technique for open set camera model verification of videos.

Taking advantage of deep-learning architectures in all their ability to learn latent camera fingerprints has been the most relevant innovation with respect to previous techniques. In this way, the network is forced to automatically discard irrelevant information and reinforce the precious characteristics left by cameras. This direction could bring new interesting solutions with respect to preprocessing filtering operations usually adopted until know.

4.2 Device Identification

Device identification is an emerging research area from a deep-learning perspective. Through the analysis of multiple fingerprints left by device sensors, device identification systems try to identify every single device for both identification and authentication tasks. Compared to the camera-model identification task, it becomes more meaningful to identify the camera device as it can provide more accurate traceability. However, it becomes more challenging to identify the cameras at the de-

vice level, as device fingerprints are much weaker and difficult to collect than the camera ones. These problems have been widely addressed by the forensic community (Lukas *et al.*, 2006; Chen *et al.*, 2008; Fridrich, 2009; Li, 2010; Li and Li, 2012), but recent works propose new solutions based on the most recent advancements of deep neural networks.

Pipelines can vary a lot depending on the feature set they consider. Indeed, particular attention is given to feature selection, which most of the time includes a preprocessing step in which a static filter is applied to the input content. Sometimes, more than one source of information is used to increase the number of subtle traces left from the devices. As an example, the PRNU and Noiseprint or median filter residual (MFR) and constrained convolutional layers can be combined to provide the model with a richer representation. These features can then be analyzed by a deep neural network. A common choice is to test the system on a closed-set and an open-set scenario. In the closed-set scenario a set of candidate sources is predefined and the system is required to associate the test image with one of the devices in the set; thus, the task can be regarded as a multi-class classification problem. In the open-set scenario, the set of candidate sources is not given a priori and the task is treated as a binary classification problem.

Ding *et al.* (2019) propose a solution that takes advantage of both hand-crafted and learned features, which consists of two main parts, i.e., a domain knowledge-driven preprocessing module and a hierarchical multi-task learning method based on ResNet. The input image is initially fed into a preprocessing module to generate intermediate features. Inspired by the widely adopted multi-scale idea (as proposed in Bayar and Stamm, 2017a; He *et al.*, 2014; Burt and Adelson, 1983), the authors use a sequential multi-scale high-pass filter (HPF) for extracting the residual image. Three HPFs are obtained by subtracting the output of three Gaussian filters of sizes 3×3, 5×5, and 7×7. Next, the HPF outputs and the original image are fed to a ResNet (He *et al.*, 2015) to extract their feature maps. Considering that the correlation among camera brand, model, and device forms a hierarchical structure, the authors propose a cascaded multi-task learning method for camera identification. The proposed method first addresses the brand classification and model classification, and then identifies the individual device as the final

classification. Before sending feature maps to the final fully connected layer and softmax layer to compute the loss function, a global average pooling layer is used over all feature maps instead of one more fully connected layer. On the Dresden dataset, the proposed method achieves 99.6% accuracy on camera-brand identification, 97.1% on camera-model classification, and 52.4% accuracy on device identification. The authors also collect a second database for cell-phone identification, containing 51 cell-phone devices from 10 brands. On this dataset, the model achieves 92.1% accuracy on camera-brand identification, 89.3% on camera-model classification, and 84.3% accuracy on device identification.

Mandelli *et al.* (2020b) propose a solution for fast source-device identification. As the authors outline, the typical workflow depends on the PRNU estimation obtained from several images coming from the same device; then, at test time, an image can be compared with the sensor fingerprint through denoising and peak-to-correlation energy (PCE) computation (Chen *et al.*, 2007), which might be computationally expensive. Thus, inspired by the work of Zagoruyko and Komodakis, 2015, Mandelli et al. introduce a 2-channel-based CNN that learns to compare patches for device identification. Initially, the device PRNU estimate is obtained from a set of images, and then, for every query image, the noise residual can be extracted using a denoising procedure (Chen *et al.*, 2008). The pair constituted by the noise residual and the PRNU estimator is used as input for the network. The network is structured in three layers: a 2D convolution layer, a leaky ReLU layer, and a max-pooling layer. Then, the network ends with a pairwise pooling layer and a fully connected layer. The pairwise pooling layer ensures faster computations compared to the evaluation of the complete cross correlation between all the characteristic maps. The network is trained by concurrently feeding it with both PRNU and residual extracted using the denoising procedure introduced in Chen *et al.*, 2008 of the same or different devices. Experimental results show that this method is faster than PCE on Dresden and Vision datasets, requiring much less query image content to obtain enhanced attribution accuracy.

Cozzolino *et al.* (2020) use Noiseprint-based model-related information (according to the approach proposed by Cozzolino and Verdoliva (2020)) to support PRNU-based device identification. As the authors

suggest, if we could know the camera model of the device that we want to identify, the search for the source device could be restricted only to devices of the same model, thereby reducing the risk of wrong identification. However, most of the time the camera models are not known in advance, thus requiring a preliminary model identification phase, which is itself prone to errors. Therefore, they suggest combining these two pieces of information. The PRNU and the Noiseprint are independently estimated from the available images of a given camera and the testing image. Then, the PRNU-based and the Noiseprint-based pseudo-distances between the test sample and the references are separately computed and eventually combined to obtain the final score. Finally, the authors propose three different strategies to combine the two distances (SVM, likelihood-ratio test, and Fisher's linear discriminant) and evaluate the performance of all such variants. The evaluation on Dresden and Vision (Shullani *et al.*, 2017) datasets for the closed-set scenario suggest that the Noiseprint-based model classification remains largely successful in most conditions and helps to improve the overall performance, whereas for the various versions of the proposed approach, there seems to be no absolute winner. This is actually the case for the proposed method. Considering again the FLD column, the AUC is never less than 93.5%.

Camera sensor fingerprints may not be the only ones used to identify the device. Other works take into account information from other electronic sensors built in the device, such as an accelerometer, gyroscopes, magnetometer (Baldini *et al.*, 2017), or microphone. Consequently, it is possible to identify the device by exploiting any of the small but significant differences in the physical components of the sensors. As with cameras, these physical differences are mainly generated during the manufacturing process, which leaves small but reproducible variations in the digital output generated by the integrated sensors.

Baldini and Amerini (2019) developed a methodology to identify mobile phones through the analysis of the microphone. First, they prepared a dataset by collecting recordings from 32 different devices and then preprocessing them. They stored audio recordings for each smartphone in pulse-code-modulation format (PCM) at 44100 Hz. Next, the audio records are power normalized and synchronized to avoid the presence

of bias related to test bed configuration (e.g., the distance from the loudspeaker or the time shift among sound recordings). They added different types of noise to the sound recordings to simulate the presence of background noise or attenuation in practical environments. Finally, they applied the fast Fourier transform (FFT) to the digital representation of the sound recordings to obtain a frequency representation, which then used later as input for the classification model. They used a CNN for the classification task and they compared it with an SVM and an k-nearest neighbor (as introduced by Altman, 1992) baselines classifiers. The CNN is made by two convolutional layers each followed by a max-pooling layer and a third and final convolutional layer extracting feature maps, which are then classified by a fully connected network. The evaluation of the model demonstrated that the introduced CNN achieves a good accuracy on both identification and authentication.

Device identification is clearly a very complicated task compared to camera model identification. The difficulty comes from the fact that the traces to be identified are much more subtle and difficult to find. Even more, data suitable for device identification are much more difficult to find compared to camera-model identification. Hence, most of the time, camera fingerprints do not suffice, and researchers need to integrate the information coming from different built-in sensors. These data, however, are difficult and sometimes impossible to find. In this respect, a lot of work has to be done to introduce stronger and easier methods to solve these problems.

4.3 Social Network and Messaging App Identification

Social network and messaging app identification is an emerging task in multimedia forensics. Recently, a high number of researchers have focused their attention on applying forensics techniques to detect the fingerprints left by these platforms. Solutions can be categorized into two representations: DCT-based and PRNU-based.

A discrete cosine transform (DCT) represents a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCT is surely one of the most widely used transformation techniques in signal processing, especially for lossy compression, and

is successfully used in many forensic applications. This feature can be particularly useful for social-media identification and is usually adopted in combination with other information for achieving better classification accuracies.

Amerini *et al.* (2019a) use DCT-based features and image noise residuals to capture the signatures of different traces left by social network operations during downloading and uploading. They introduce a novel CNN architecture, which they call FusionNET, to combine the learning power of the inter-layer activation of two single-feature–based CNNs. The first stream, called 1D-CNN (Amerini *et al.*, 2017c), consists of two convolutional and max-pooling layers followed by a fully connected layer. The second one, called 2D-CNN Caldelli *et al.*, 2018a, is composed of two blocks, each one made of two convolutional layers, a max-pooling layer, and a dropout layer. The last convolutional block is connected to a dense layer. The two 256-unit vectors coming from the last layers of the 1D-CNN and the 2D-CNN are concatenated and fed to two fully connected layers. Their output enters a softmax layer with as many units as the number of classes k, a value to be identified. It is desirable for the output to be independent of the input image resolution. However, The DCT is highly affected by the content and size of the considered image. For this reason, the authors divide each picture in nonoverlapping patches. For each 8×8 block in a patch, the system selects the first 9 spatial frequencies in zig-zag scan order besides the DC coefficient. Then, for each spatial frequency it constructs the histogram representing the occurrences of the quantized DCT coefficients. The input to the first CNN is the resulting vector of 909 elements (101 histogram bins for each of the 9 DCT frequencies). the second network takes as input a 64×64 bidimensional matrix. The proposed system reaches a high accuracy both at the patch and image levels on several datasets, namely the UCID dataset (introduced in Section 7.2), the Image Ballistic and Social Network dataset (see Section 7.2), and the Vision one.

Phan *et al.* (2019) exploit traces left in DCT coefficient maps, and information from JPEG images metadata. They propose two CNN architectures. A patch-based CNN composed by 3 consecutive convolutional layers takes histograms of DCT coefficients and incorporate a statistical-moments layer introduced in Fuji Tsang and Fridrich, 2018 to extract 4

statistical values (min, max, mean and variance) of each feature map. These values are concatenated in a vector and fed to multilayer perceptron (MLP) classifier. A variant of this architecture concatenates DCT and metadata features used in Phan *et al.*, 2018. The obtained results show the improvements introduced from the features fusion to distinguish among different social platforms and the exact sequence of social-media platforms in which the image was posted. The authors test their solution on RAISE (see Section 7.2) and Vision datasets. They evaluate the accuracy of their method on the following three scenarios:

- Input images have have been shared, once, via Facebook, Flicker, or Twitter, and the goal is to identify the platform. This problem gives rise to a 3-class classification problem. The accuracy of their approach is 99.87%.
- Input images have been shared either in one platform, as in the first case, or in two platforms. This problem gives rise to a 12-class classification problem The accuracy of their approach is 65.91%.
- Input images have been share up to three times. This problem gives rise to a 12-class classification problem The accuracy of their approach is 36.18%.

Photo response non-uniformity (PRNU) is another common feature used to train convolutional neural networks. When uniform light falls on a camera sensor, each pixel should output the same value. However, small variations in cell size and substrate material may result in slightly different output values. These imperfections are intrinsically related to the physical properties of the sensor itself; as a result, it is practically impossible to eliminate them and so they are typically considered to be a normal characteristic of the sensor. As such, the processing operations led by social network platforms can leave precious traces in the PRNU, which can be used for social-media identification.

Caldelli *et al.* (2018b) exploit the alterations introduced by social networks on the PRNU noise embedded in images by source devices during the acquisition process. For each image, they compute the noise residual, representing the image from which its denoised version has been subtracted, and extract non-overlapping $N \times N$ patches. The

patches extracted from each image are fed to a convolutional neural network. The network is made of two blocks, each one composed of two convolutional layers followed by a pooling layer. Then, two fully connected layers classify the features extracted from the second block. The method achieves high accuracy (above 87%) on IPLAB (see Section 7.2), UCID, and Vision. Similarly, Amerini *et al.* (2017a) use a PRNU-based methodology to identify videos coming from social networks. Typically, a video is split in individual frames and then a wavelet denoising filter filters out the scene content for each RGB color channel. Then, by averaging on a specified number of filtered frames for each color channel and converting them to gray levels it is possible to obtain a *fingerprint* of the source of the video. Finally, by estimating the fingerprint for each class (source device, Twitter and Facebook), a new video can be classified by comparing its denoised version with the estimated fingerprints. However, Amerini et al. introduce an approach called *composite fingerprint*, obtained extracting and concatenating the PRNU noise from chunks of frames respectively taken from the original video and the videos downloaded from Facebook and Twitter. A composite fingerprint permits to take into account some changes on the PRNU noise introduced by the processing performed by the social network onto the video. At test time, only the original video need to be available to compute the composite fingerprint because it can be uploaded on the various social networks and then downloaded to compute the fingerprint.

Quan *et al.* (2019) investigate the effects of the predefined Instagram image filters on the sensor pattern noise (SPN, Lukas *et al.*, 2006) based image provenance analysis, whose dominant component is PRNU. Initially, the authors study how different filters affect the SPN pattern and whether it would be possible to identify the applied filter such that it would allow them to conduct a more reliable provenance analysis. Next, they propose a modified VGG network. Taking inspiration from Gatys *et al.*, 2015, Quan et al. claim that the visual clues added by Instagram filters can be mostly represented by the lower layers in a neural network, and use a network with seven convolutional layers followed by three fully connected layers, which is more compact than the shallowest VGG architecture (11 layer VGG-net) in Simonyan and Zisserman, 2014. They train their system using three different inputs:

the unprocessed images (I-net), the denoised images (Î-net), and the image noise residuals (n-net). On average, the I-net, Î-net, and n-net achieve an accuracy of 79.91%, 86.93% and 88.38%, respectively on the Vision dataset (Shullani *et al.*, 2017).

The identification of the fingerprint left on videos by social media platforms has been quite limited for now. In fact, the lack of public datasets on this problem makes hard to propose new methods. However, Amerini *et al.*, 2021 have recently proposed a method that is capable to distinguish native videos from Facebook and WhatsApp videos on the Vision dataset. The authors propose a multistream network called MultiFrame-Net. The first stream, is a convolutional network called Ind-Net, which analyzes input I-frames extracted from an input video. This network is made of six convolutional blocks. The first three blocks are three stacks of a convolutional layer, a batch normalization layer, and a ReLU layer, followed by a final max pooling layer; the last three blocks add one more stack of convolutional, batch normalization, and ReLU layers. The second stream is based on another convolutional network, called Pred-Net, which analyzes a stack of three consecutive P-frames. This network is made of five convolutional blocks. The first two blocks are two stacks of a convolutional layer, a batch normalization layer, and a ReLU layer, followed by a final average-pooling layer. The last three blocks are similar to the first two blocks but add one more block of a convolutional, a batch normalization, and a ReLU layers. In the last block the average pooling layer is substituted with a global average pooling layer to preserve the statistical properties of feature maps. The output vectors of the two streams are concatenated and fed to a two convolutional layers. The input frames are preprocessed as done in Nam *et al.*, 2019 before being fed to the network. The experiments on the Vision dataset show that the proposed method classifies the frames of a video with 95.51% accuracy. Moreover, Amerini *et al.*, 2021 report the results of a preliminary experiment, where they measure the ability of the Ind-Net to transfer features from the video domain to the image domain. By freezing the CNN parameters learned on the video domain and retraining the fully connected network on images from WhatsApp, the network classifies the images with 86.63% accuracy.

Being able to detect whether an image or video comes from a social media platform or a messaging app is a hot problem nowadays. The solutions reviewed in this section are mainly based on PRNU or DCT coefficients and lead to satisfactory results. In this respect, a possible future direction to explore could be the analysis of learned based models that learn to automatically extract the most relevant fingerprints, thus avoiding the preprocessing steps of extracting PRNU or DCT.

4.4 Best Practices for Source Identification

Source identification is one of the fundamental problems of media forensics. It contains multiple challenges: camera model, device and social-media–platform identification. Most of the approaches to address these challenges are commonly evaluated for accuracy and can be addressed in a closed-set (all possible classes of provenance are known in advance) or open-set context.

Many works highlight the importance of choosing patches of sufficient quality to rebuild the source. In fact, not all patches contribute in the same way with information that is sufficiently complete and rich to be used for this purpose. Saturated patches typically contain less statistical information than textured and mixed patches. Thus, applying a quality selection procedure for patch-based CNN can improve detection performances in all the problems described in this chapter (see Section 4.1.1). A good selection process (reused in many works) is the one described by Bondi *et al.*, 2017b.

Another very important design choice is that of filters. The choice can range from static filters (high pass-filters, MFR, and Laplacian are the most common) to filters learned dynamically from a neural network. The latter category has proven to be very effective in source-identification applications, but as with most deep-learning solutions they are dependent on training data. An open challenge is how to apply these techniques on unknown datasets. The most promising approaches are those based on similarity networks such as the methods described in Cozzolino and Verdoliva, 2020, Mayer and Stamm, 2018, and Mayer *et al.*, 2020.

Even though almost all of the solutions described in this section do not require deep-learning architectures that are too deep or complex, the complexity of networks nevertheless increases for social-media platform identification solutions. In this case, the network is designed to detect features left by social networks and messaging apps during the upload phase, which typically can involve recompression, resizing, and other operations aimed at improving the display quality of these contents on the platform. To capture all these subtle traces, very commonly the proposed solutions are based on multistream networks that analyze information such as PRNU, DCT, or noise residual features in parallel and then make a common decision based on all the extracted information.

Table B.3 summarizes some of the most interesting architectures described in this section.

5

Deepfakes: Strategies to Detect Artificially Generated Content

The progress of artificial intelligence has brought new threats related to manipulation or generation of visual and audio content with a high potential to deceive, making the creation of fake content very easy. Deepfakes have attracted widespread attention for their potential abuse in fake celebrity videos, fake news, hoaxes, and financial fraud. Most of the techniques applied to create deepfakes are based on deep learning and involve training generative neural network architectures, such as autoencoders, or generative adversarial networks (GANs, Goodfellow *et al.*, 2014). Even though these emerging methods have shown some incredible realistic achievements, artificially generated content has been used for a long time now, so many of the issues we discuss in this chapters are not new. Computer-generated imagery (CGI) and deep-learning generated fakes have something in common: the lack of some characteristic features that are present in images and videos captured through real cameras. These, for the time being, give the possibility to differentiate between real and fake images.

All such technologies are mostly used to generate fake human faces. In fact, it is very common in forensics to use the term *deepfake* to refer to fake (not just deep-learning–based) human faces. Deepfakes have

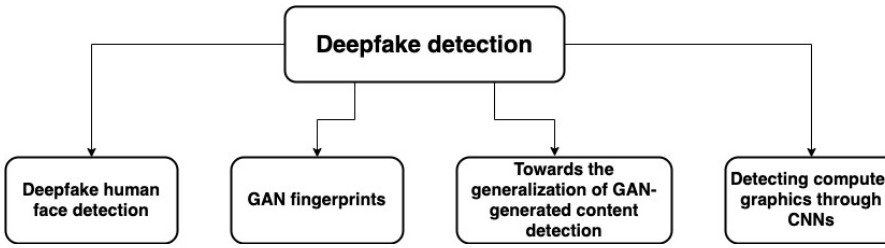


Figure 5.1: An overview of Deepfake detection approaches.

proven really good at creating synthetic faces of persons that do not exist, or to realistically reproduce faces of public figures. Therefore, in the next section, we discuss some of the recent works involving deep learning to spot fake human faces. after that, we review the works that use all the forensic tools developed to detect such contents in a general manner. Figure 5.1 summarizes the research problems reviewed in this section.

(a) Deepfake example. **(b)** Deepfake example. **(c)** Deepfake example.

Figure 5.2: On top we show some real actors and on the bottom the corresponding deepfake (Nick Dufour, 2019).

5.1 Deepfake Human Face Detection with Deep Learning

The most widely adopted application of deepfakes has probably been in the production of fake human faces. The impressive level of realism reached by deepfakes exposes public figures and society in general to new threats (see Figure 5.2). With the spread of social media, fake

images or videos can easily go viral, facilitating the spread of fake news. As such, being able to distinguish whether a human face is real or fake becomes a need to which the forensic community is responding very quickly. Even though, as said before, computer-generated content has been used for a long time now, the recent development of deep-learning techniques has attracted a lot of attention from the forensic community. Hence, in this section, we review all recent methods embracing this technology. Matern *et al.* (2019) reviewed current facial editing methods and several characteristic artifacts. Similarly, Nguyen *et al.* (2019b) presented a survey of algorithms used to create and detect deepfakes.

A common approach is to try to identify some anomalies in the typical movements of human faces, or inconsistent configurations of facial parts caused by weak global constraints. Hence, as a general pipeline, the works in this section usually work by detecting a specific inconsistency in the human face. Most of the time, this procedure could include a preprocessing step of landmarks extraction. A CNN can then be used to extract latent feature maps. Finally, the selected features can be classified by a fully connected network or other classifiers such as SVM.

Li *et al.* (2018) expose deepfakes by detecting anomalies in eye blinking in fake videos, taking advantage of the fact that generally they tend appear unnatural. The detector is designed by relying on a long-term recurrent convolutional network (LRCN, Donahue *et al.*, 2016) to capture the temporal regularities that characterize the eye blinking process, accurately distinguishing between the open and closed eye state for each video frame. Initially, a face detector extracts face areas and facial landmarks from each frame. The detected faces are then aligned into the same coordinate system to discount head movements and orientation changes. From the aligned face areas, bounding boxes of each eye's landmark points are extracted and the cropped eye area sequences are passed into the LRCN for dynamic state prediction. The evaluation of the AUC suggest that LRCN show improved performance (99%) compared to CNN (98%).

Yang *et al.* (2019b) expose GAN-generated faces using the vector of landmark locations as a feature vector to build a classification system for differentiating GAN-synthesized and real faces. An SVM or a neural

network classifier is trained to compare facial landmark locations detected on GAN-synthesized and real faces. The proposed method takes advantage of the following observation: current GAN-based algorithms use random noises as input; this approach works well in depicting the details of different face parts, but it lacks constraints on the configuration of different face components, leading to the introduction of errors in the locations of facial parts. The SVM classifier achieves an AUC of 94.13% and outperforms several methods based on deep neural networks (e.g., VGG19 and XceptionNet) with respect to the CelebFaces Attributes Dataset (CelebA, Liu *et al.*, 2014) and PGGAN datasets (Karras *et al.*, 2017). In another work, Yang *et al.* (2019a) propose an SVM-based classifier to detect deepfakes against real face images. This is done by detecting relevant errors that arise when comparing deepfake with real 3D head poses estimated from the face images. Initially, a deepfake generation pipeline is introduced. A real cropped face is warped to a standardized face using an affine transformation, and it is fed to a GAN deep neural network to produce a synthetic face. Then, the authors compare the head poses estimated using 68 facial landmarks from the whole face or only from the central face region. The alignment error between real and fake face is revealed as the differences in the head poses shown as their projections on the image plane. The difference of the head poses is then flattened into a vector and fed into an SVM classifier to differentiate the original image from the deepfake.

Do Nhu *et al.* (2018) use the feature extractor in the VGGFace (Parkhi *et al.*, 2015) to distinguish between input fake and real images or video frames.

Agarwal *et al.* (2019) use 16 facial action units (AU, according to the definition given by Ekman and Friesen, 1976) to represent the person's face. They notice that facial expressions and head movements are strongly correlated and that changing the former without modifying the latter may expose a synthetic face. These AUs are augmented, that is randomly transformed, with the following four features: (1) head rotation about the x-axis (horizontal); (2) head rotation about the depth-axis (depth); (3) the 3-D horizontal distance between the corners of the mouth; and (4) the 3-D vertical distance between the lower and upper lip. The first pair of features captures general head motion and the

second pair of these features captures mouth stretch and lip suck. Next, they use the Pearson correlation to measure the linearity between these features in order to characterize an individual's motion signature. Each 10-second video clip is then reduced to a feature vector of dimension 190 and used to classify a video as real or fake. A SVM is used to distinguish between real and fake contents. The authors evaluated the robustness of this technique against compression, variation of video clip length, and the context in which the person is talking (e.g., formal prepared remarks looking directly into the camera versus a live interview looking off-camera). They show that their method is robust against compression, but vulnerable to different contexts in which the person is talking, and accuracy drops for shorter clips, but is largely unaffected by clip lengths between 10 and 20 seconds. Thus, this technique can be used just in limited contexts, requiring a large number of videos for every single person.

Similar to external face characteristics, biological signals (e.g.,heart rate or eye blinking) can be used to identify synthetic faces. Ciftci and Demir (2019) extract biological signals from facial regions on authentic and fake portrait video pairs and analyze the spatial coherence and temporal consistency of such features. The authors compare the performances with various neural networks. The method outperforms the best baseline architecture (i.e., the one proposed by Tariq *et al.*, 2018) by 8.85%. In the same direction, other works, such as Fernandes *et al.*, 2019; Maras and Alexandrou, 2018 extract the heart rate from facial videos and use this information to train a classifier.

Although deep-learning models are usually characterized by a high number of deep layers, Afchar *et al.* (2018) propose two simple architectures with a small number of layers and parameters that make use of mesoscopic features. The networks are trained on 256×256 images or I-frames. The first network (Meso-4) is made of a sequence of four convolutional and pooling layers, and is followed by a dense network with one hidden layer. The second (MesoInception-4), is based on replacing the first two convolutional layers of Meso4 by a variant of the inception module introduced by Szegedy *et al.*, 2016. As for the original Inception network, each layer is made of a stack of several convolutional layers with different kernel shapes. However, instead of 5×5 convolutions,

Afchar et al. use 3×3 dilated convolutions (Yu and Koltun, 2015) to avoid high semantic. In order to reduce the dimension of the feature maps extracted by the convolutional layers, the authors add 1×1 convolutions before and in parallel to the dilated convolutions. Replacing more than two layers with inception modules did not offer better results for the classification. The authors test the two architectures and the show that they obtain an accurate detection rate with more than 98% on a deepfake dataset and 95% on the FaceForensics dataset (Rössler *et al.*, 2018a). Another simple lightweight approach is introduced by He *et al.* (2019), who employed residual signals of chrominance components from multi color spaces to train shallow CNNs: First the image is converted in YCbCr, HSV and Labs color spaces and preprocessed with a high-pass filter to reduce the influence of different image contents. The shallow networks are trained as feature extractors on the preprocessed images. Finally, the feature maps extracted by the shallow networks are classified by means of a random-forest classifier.

Looking at face inconsistencies is not the only way to reveal fake human faces. Li and Lyu (2018) made the following observation: current deepfake algorithms can only generate images of limited resolutions, so the images need to be further warped to match the original faces in the source video. Taking advantage of it, they trained a deep CNN classifier to capture distinctive artifacts in deepfake videos, relying on several alternative backbones (e.g., VGG-16, ResNet50, ResNet101 and ResNet152). To generate negative examples for training the model, they proposed a simple preprocessing pipeline: a detector extracts the face regions, then faces are aligned into multiple scales, and they are randomly selected and smoothed by a Gaussian blur with a 5×5 kernel. This process aims to recreate more resolution cases in warped faces that imitate better different kinds of resolution inconsistencies introduced in face warping. The smoothed face undergoes a warp back to the same sizes of original face to simulate the artifacts in a deepfake production pipeline. Finally, faces are resized to 224×224 and fed to the CNN models for training. The authors validate their method on the UADFV Yang *et al.*, 2019a and DeepfakeTIMIT (Korshunov and Marcel, 2018) datasets. The VGG-16, ResNet50, ResNet101, and ResNet152 models achieve an AUC performance of 84.5%, 98.7%, 99.1%, 97.8%, on UADFV

respectively, and of 57.4%, 93.2%, 86.9%, 91.2% on DeepfakeTIMIT. The method also outperforms some state-of-the-art methods the face tampering detection method, such as the Two-stream NN (Zhou *et al.*, 2018b), and the two deepfake detection methods MesoNet (Afchar *et al.*, 2018) and HeadPose (Yang *et al.*, 2019a). However, although Li et al. designed their method for images of limited resolutions, Karras *et al.* (2019) proved that it is possible to detect that an image was generated artificially even if it is of very high quality. The authors identified and fixed several image quality issues in StyleGAN, improving the quality further with the additional benefit that the generator becomes significantly easier to invert.

Li *et al.* (2019) analyze the problem from a pixel-level perspective by using segmentation methods and comparing the performance of several network architectures. As the authors suggest, most works cast the problem as a classification task that can only produce a global scalar value representing the confidence that the image is fake but cannot reflect the *extent* to which it has been manipulated. Initially, they show that any classification network can be easily converted to a fully convolutional network (Long *et al.*, 2014) by replacing the fully connected layers with convolutional layers. A Binary Cross Entropy loss can be used to measure the classification error on each pixel. Then, they compare several architectures (Xception, MesoInception-4, UNet, VGG-16 and a simple 3-layer convolutional network introduced by the authors) to evaluate their effectiveness on the problem of face forensics. First, they notice that a deep network, such as Xception with 36 layers, does not reach a high score, whereas the shallow models present better abilities, revealing that face forensics is supposed to be defined as a low-level vision problem than a high-level perception problem. Then, they notice that there is little difference between the pretrained model and the trained-from-scratch model. According to the experimental results, the features learned in a general vision recognition task such as ImageNet did not help quickly find better local optima. Finally, in order to have a better understanding of the features learned by the VGG-5 (i.e., a shallow versions of the VGG-16 containing the first 4 feature layers of VGG respectively and a classifier), they analyze the kernels by visualizing them using the technique in Springenberg *et al.*, 2015. The results show that apart from the features in the

first convolutional layer, which are mostly low-level edges and corners, the kernels in following layers do not make much sense to us humans.

Videos have recently gained more interest than still images have. Moreover, several state-of-the-art face recognition systems can be vulnerable to deepfake videos (Korshunov and Marcel, 2019). The recurring convolutional models adapt perfectly to the problem of detecting video manipulation, bringing a spatio–temporal dimension to the analysis.

Sabir *et al.* (2019) introduce a two-step method based on cropping and alignment of faces from video frames, followed by manipulation detection over the preprocessed facial region. For the preprocessing step, the authors experiment with two different techniques: (1) explicit cropping and alignment through landmarks and (2) an implicit alignment that uses a spatial transformer network (Jaderberg *et al.*, 2015), which is trained to predict the alignment parameters on each input image, thus learning to zoom on particular parts of the face. Then they incorporate the idea of using features at a mesoscopic level (Afchar *et al.*, 2018), attempting to learn multiple recurrent networks at different levels of the hierarchy. In this way, the model uses micro, meso, and macroscopic features for manipulation detection. The evaluation of the model on FaceForensics++ (Rössler *et al.*, 2019) using ResNet and DenseNet as the CNN component of the model show that DenseNet with alignment and bidirectional recurrent network is the best performing model compared to the baselines in Rössler *et al.*, 2019.

Guera and Delp (2018) adopt a CNN to extract frame-level features from a video sequence and an LSTM to produce a temporal sequence descriptor for image manipulation of the shot frame. The CNN is trained to extract 2048-dimensional feature vectors after the last pooling layer that are then fed in the sequential LSTM. The LSTM is followed by a 512 fully connected layer with 0.5 chance of dropout, and a softmax layer to compute the probabilities of the frame sequence being either authentic or deepfake. The authors collected 300 deepfake videos from multiple video-hosting websites for training, validation, and testing. The models achieve high-accuracy results (more than 96% accuracy) on their evaluation.

Amerini *et al.* (2019b) and Caldelli *et al.*, 2021 introduce a new technique able to detect deepfakes from original videos. They use a

sequence-based approach based on optical flow to analyze possible dissimilarities in the temporal structure of a video. The approach treats the optical flow is a vector-field computed on two consecutive frame to extract apparent motion between the observer and the scene itself. The forward flow is extracted using a PWC-Net (Sun *et al.*, 2017) and fed to a semi-trainable CNN named Flow-CNN, based on a pretrained network. The authors experiment with a VGG-16 and a ResNet50. Preliminary results on FaceForensic++ dataset (Rössler *et al.*, 2019) on that the ability of this approach to identify some existing dishomogeneities in a video, show a 81.61% and 75.46% binary detection accuracy for VGG-16 and ResNet50 respectively.

The architectures in this section are united by the choice of a specific feature to work on. However, along with the forensic tools, deepfake generation techniques improve as well, making these traces more and more difficult to recognize. As a result, we expect to see more complex and sophisticated detectors being designed in the future. As for forgery detection problems, the development of cross-manipulation detectors able to recognize any kind of attack remains an open challenge. The first steps in this direction have been done and will be discussed in Section 5.3. But first, we discuss the research on a more elementary topic.

5.2 Do GANs Leave Artificial Fingerprints?

Generative Adversarial Networks (GANs, Goodfellow *et al.*, 2014) have received a lot of attention in the last few years because of the high level of accuracy reached by these architectures. These results have led to a massive spread of deepfakes, which are now mainly generated by GANs. Being a recently emerged problem, it is important to understand how to face the spread of GAN generated content. So the first question that many researchers try to answer is: Do GANs leave artificial fingerprints? And if so, how can we detect them? In this section, we review recent publications working in this direction.

Marra *et al.* (2018b) show that each GAN leaves its specific fingerprint in the images it generates: this implies that source identification can be a relevant tool for image forensics purposes. Such fingerprints

depend on the GAN itself, both on its architecture (number and type of layers) and its specific parameters (filter weights). However, this also implies that the attacker can remove the fingerprint from the generated images as a counter-forensic measure. The authors consider three alternative GAN architectures: the Cycle-GAN (Zhu *et al.*, 2017), the ProGAN (Karras *et al.*, 2017), and the Star-GAN (Choi *et al.*, 2017) to perform image-to-image translation (e.g., the generator takes an input image of the source domain and transforms it into a new image of the target domain) and compare the related fingerprints with those of a set of images acquired by real cameras and belonging to the RAISE dataset. Eventually the authors used a deep neural network to differentiate between real and GAN-generated images. Initially, they show that, similarly to PRNU, an image residual can be calculated by processing the image with a denoising filter and then, by subtracting the filtered image from the original. Then, the fingerprint of the GAN is estimated by a simple average over the available residuals. Finally, they calculate the correlation between and fingerprints. They verified out in several models that the histogram of correlations are well separated across different GANs, allowing reliable discrimination. As an example, AUC score for Cycle-GAN and Pro-GAN are nearly perfect with values 99% and 99.8%, respectively. They carried out similar experiments for many other GANs. These results provide evidence that each GAN leaves a distinctive mark on each image generated by it, which can act as a *fingerprint*.

Different GAN models have been proposed, each one introducing its own training dataset distribution, loss, optimization strategy and hyperparameter settings. Both nonconvexity of the objective function and instability of adversarial equilibrium in GANs lead those models to be extremely sensitive to random initializations, therefore converging to different values during each training. As suggested by Yu *et al.* (2018), this indicates that, even though two different models may perform equivalently, they generate high-quality images differently, suggesting the uniqueness of GAN fingerprints. The authors replace the handcrafted fingerprints proposed by Marra *et al.* (2018b) with a learning-based formulation. They introduce three different attribution networks: (1) a pre-downsampling network that extracts low-frequency bands, (2) a

pre-downsampling residual network that extracts high-frequency bands between one resolution and its factor-2 sampled resolution, and (3) a post-pooling network that test whether fingerprints and attribution can be derived locally based on patch statistics. Experiments suggest that fingerprints persist across different image frequencies and patch sizes, and are not biased by GAN artifacts. Even more, the proposed model can be effectively tuned to overcome several perturbations attacks. Even though fingerprints can be deteriorated by several image perturbation attacks, they are effectively immunizable by simple fine-tuning. The fingerprints learned by the proposed method are consistently superior to the work by Marra *et al.* (2018b).

Nataraj *et al.* (2019) propose an approach to detect GAN-generated fake images using a combination of co-occurrence matrices and deep learning. They extract co-occurrence matrices on three color channels in the pixel domain and they use them to train a model using a deep CNN framework consisting of several suitably dimensioned alternate couples of convolutional and ReLu layers. Experimental results on two diverse and challenging GAN datasets comprising more than $56,000$ images based on unpaired image-to-image translations (cycleGAN) and facial attributes/expressions (StarGAN) show that the approach is promising and yields high classification accuracy on both datasets. The authors test the model accuracy on the cycleGAN and the StarGAN dataset, obtaining a testing accuracy of 99.71% on the cycleGAN dataset and of 99.37% on the StarGAN dataset. Next, they evaluate the capability of the method to generalize by training it on one dataset and testing it on the other. The model trained on cycleGAN dataset has a higher accuracy of 99.45%. In comparison, the model trained on StarGAN dataset achieve an accuracy of 93.42%.

Social networks can be the first diffusion medium for GAN-generated fake images. Marra *et al.* (2018a) show that high detection accuracy can be achieved by both conventional and deep-learning detectors; yet conventional detectors show dramatic impairments on Twitter-like compressed images. Instead, they show that deep CNNs (e.g., DenseNet, InceptionNet and XceptionNet) are robust, yielding high accuracy on compressed data and it the presence of training-test mismatching. According to the evaluation results, several detectors perform very well

on the original images, but some of them show dramatic impairments on Twitter-like compressed images. Moreover, deeper networks, and especially XceptionNet, turn out to be more robust even in the presence of training-test mismatching.

Even though various papers have promising results, it is still not clear how these systems will evolve in the future, thus it becomes important to keep studying the limits of deepfake detection. In this respect, Agarwal and Varshney (2019) pose the problem of deepfake detection—namely for classifying genuine against GAN-generated images—as a hypothesis-testing problem, and they derive statistical error bounds on the detection of deepfakes.

As we have seen in this section, GANs tend to leave their own fingerprint, therefore making it possible to detect if they have been used for the generation of the content under study. Even more, it different GAN architectures leave different fingerprints, thus enabling the design of fine grained detectors.

5.3 Towards the Generalization of GAN-Generated Content Detection

Most of the time, GANs are used to generate fake human faces; yet, these technologies can be used for many other applications. This means, even more robust forensic tools are needed to detect every kind of fake synthetic content. As new and more powerful GAN architectures develop rapidly, the generalization ability of forensics method to unseen types of generated fake images becomes increasingly important for the forensic analysis.

Cozzolino *et al.* (2018) propose an autoencoder-based neural network architecture that learns a latent space called *forensic embedding*. To prevent the net from discarding precious information during training, the latent space is constrained to preserve all the data necessary to reconstruct the image in compact form. The input images are high-pass filtered by means of third-order derivative so as to obtain the residual image. Next, the autoencoder detector is trained to disentangle encoding and decoding. Specifically, the latent vector is split into two disjoint parts, each associated with a class, "real" or "fake." Thus, the network is

trained to activate only the part of the latent vector that belongs to the input sample class. The learned embedding acts as a form of anomaly detector; that is, an image manipulated from an unseen method will be detected as fake provided it maps sufficiently far away from the cluster of real images. Compared to other reference methods, most of which provide accuracies close to 50% on unseen datasets, the proposed method maintains good performance, e.g., 90% accuracy from Cycle to Style-GAN. Similarly, Nguyen *et al.* (2019a) use the same approach with a deeper network adding a segmentation task, e.g. the reconstruction of an output mask. Again in this way, Du *et al.* (2019) introduce a locality-aware autoencoder that is enforced to capture discriminative representations from the manipulated regions by forcing the encoder to output a heatmap (or attention map) for each input and regularizing the attention map with pixelwise forgery ground truth (i.e. the segmentation mask). The goal of local interpretation is to identify the contributions of each pixel in the input image towards a specific model prediction. To obtain the attention map, the authors use global average pooling layer on the last conventional layer of the encoder and upsample the attention map to the same dimension as the input image using bilinear interpolation as in Zhou *et al.*, 2015. Then, to enforce the network to focus on the correct forgery region to make detection, the network is regularized by minimizing the distance between the predicted attention map and the ground truth. Compared to other methods (Cozzolino *et al.* (2018), Afchar *et al.* (2018) and Chollet (2016)) the proposed technique allows for better interpretation and stronger generalization.

Wang *et al.* (2019b) prove that a standard image classifier trained only on one specific CNN generator (Pro-GAN) is able to generalize well to unseen architectures, datasets, and training methods if trained with accurate data augmentation. At training time, the dataset is enriched with Gaussian blur, or JPEG compression, or combinations of them. Instead, at test time the input is not enriched. Xuan *et al.* (2019) propose a preprocessing step that involves random extent Gaussian blur (i.e. the result of blurring an image by a Gaussian function) and Gaussian noise; this preprocessing aims at destroying low-level clues. The preprocessed input enters a deep convolutional GAN with four convolutional layers (Radford *et al.*, 2015), which is trained to distinguish between real and fake images.

Marra *et al.* (2019a) address the detection of GAN-generated images by means of an incremental learning strategy. Specifically, they make use of of the incremental classifier and representation learning (iCaRL) method (Rebuffi *et al.*, 2016) to update the classifier as new GAN architectures appear over time. The authors compare two multitask versions of the original iCaRL: (1) a new separate binary classifier (called Multi-Task MultiClassifier) trained jointly with the original one, and (2) another (Multi-Task Single Classifier) in which the iCaRL detector structure is left unchanged and a binary cross entropy loss term is added to the original iCaRL loss. The proposed method ensures a detection accuracy over 90% even when more than one single GAN has to be detected.

As we have seen in the previous section, the artificial generation of human faces has attracted a lot of attention. Khodabakhsh *et al.* (2018) evaluate the generalizability of fake face detection methods through a series of experiments. This work, emphasizes the importance of validation of detectors across multiple datasets for wider applicability of these solutions to real-world scenarios.

Bondi *et al.*, 2020 analyze (1) how different training strategies and (2) data augmentation techniques affect CNN-based deepfake detectors when training and testing on the same dataset or across different datasets. They begin by comparing two training strategies to measure the intra-dataset and cross-dataset detection performances. Moving from their previous work (Bonettini *et al.*, 2020), they initially extract the faces in the video sequences through the BlazeFace (Bazarevsky *et al.*, 2019) network. Then, they train an EfficientNetB4 (Tan and Le, 2020) architecture on four datasets (FaceForensics, FaceForensics++, The Deepfake Detection Challenge—DFDC, and CelebDF—see Section 7.3) to predict whether a face is fake or real. Finally, they compare the performance of a baseline EfficientNetB4 trained using the binary cross entropy loss, and the same CNN trained with a triplet loss (Wang *et al.*, 2014). The results show that the triplet loss seems to offer a bit more separation between the two classes, despite the feature extractor being trained on a different dataset. The second batch of experiments evaluate the effect of different data augmentation techniques (horizontal flip, brightness and contrast changes, hue, saturation and value changes,

addition of ISO noise, Gaussian noise, downscaling and JPEG compression). They test all the augmentations independently one from each other, training with binary cross entropy loss on the DFDC dataset. The augmentations do not seem to help much increasing intra-dataset detection, maybe due to the cross-contamination between train, validation, and test set in terms of video settings and scenarios. However, some augmentations (horizontal flip, brightness and contrast changes, hue, saturation and value changes and JPEG compression) are beneficial in terms of cross-dataset generalization. The combination of augmentations brings up to +9% AUC improvements in terms of cross-dataset detection. Augmentations are not beneficial when applied to the CNN trained with triplet loss.

Designing a general-purpose classifier is one of them main goal of the forensic community. This would allow the user to detect any deepfake content regardless of how it has been generated. The development of models granting forensic transferability is surely the first step in this direction, making it possible to transfer the detection capability of a model to different but similar problems.

5.4 Detecting Other Computer Graphics Through CNNs

GANs and deepfakes are just the latest artificial content generation techniques. However, computer-generated imagery (CGI) has seen an enormous application of computer graphics in several industries, allowing for the creation of printed media, video games, movies, commercials, videos, and animated content. Realistic achievements of these technologies could still require forensic tools that allow to distinguish between real and fakes. In fact, CGI can be used to produce highly realistic content that can be sometimes hard to distinguish with respect to real ones, for this reason, it is important to pay attention to this type of content as well as done for deepfakes.

The general approach adopted in the recognition of CGI may involve preprocessing steps, such as the application of a filter, a feature extraction through a CNN architecture, and a final classification step.

Rahmouni *et al.* (2017) conducted a series of experiments with different CNN configurations and statistical feature extraction methods.

They used a boosting method to aggregate local estimations of the output class. Yao *et al.* (2018) use a high-pass filter layer made of three high-pass filters to remove low-frequency signals and to reveal the residual signal as well as sensor pattern noise introduced by the digital-camera device. The proposed method uses a five-layer CNN based on the work of Xu *et al.* (2016) to classify the input image patches. Then a majority vote scheme is used to obtain the classification results for the full-size images. Even though the compression with different quality factors has an effect on the classification accuracy of image patches, the classification accuracy for all different quality factors of the authentic images are 100%. Similarly, Qi Cui (2018) gray out each input image and they subject it to high-pass filtering to add high-frequency components.

de Rezende *et al.* (2017) propose a method that is able to classify computer generated images. Initially, the input images are preprocessed by subtracting the mean RGB value of each pixel computed on the ImageNet dataset as in Krizhevsky *et al.*, 2012. Then, the authors initialize a ResNet-50 pretrained on ImageNet dataset and freeze all the network parameters except for the last fully connected layer, which is replaced by a new classifier. The authors propose two distinct classifiers that are trained on the 2048 dimension feature vectors extracted from the pretrained ResNet-50: the first one with two fully connected layers followed by a softmax layer, and the second one with an SVM classifier. Experiments on a public dataset (Tokuda *et al.*, 2013) show an accuracy higher than 94% when using an SVM and 92% with a fully connected network. Similarly, Yu *et al.* (2017b) propose a simple neural-network architecture to train a binary CNN classifier that distinguishes between photorealistic computer graphics and real images. Randomly selected patches of an input image are fed to a VGG network with the dropout layer replaced by batch normalization and removed pooling layers.

He *et al.* (2018) introduce a methodology based on the combination of a CNN and an RNN architecture. Initially, they preprocess each input image by transforming it from the RGB color space to the YCbCr color space and then applying the Schmid filter (Schmid, 2001) to the luminance component (Y), so as to suppress the irrelevant information. Next, they split the preprocessed image into nonoverlapping patches and they analyze color and texture characteristics by using two parallel

streams composed of a stack of four convolutional layers and average-pooling layers. They apply batch normalization after each convolutional layer. The feature maps coming from these two streams are concatenated and fed into a two-layers fully connected network. The vector represented by the first fully connected layer from the trained CNN is used as a representation for local image patches. Finally, an RNN is applied on the representation vector to model the spatial dependency of local patterns and the output score of RNN is used to classify the input patch. The proposed pipeline achieves 93.87% accuracy.

The generation of content through computer graphics techniques has been refined for many years thanks to the adoption of these technologies in large industries such as computer games and cinema. These technologies have reached a high level of realism, thus requiring the development of forensic tools if the goal is to distinguish the photos or videos created with these technologies from real-life content. The forensic community has responded very quickly to this need, and most recently it has developed new deep-learning methods to address this challenge. We expect of course this arms race to continue in the future.

5.5 Challenges and Best Practices for Deepfake Detection

Recognizing deepfakes is the latest big challenge in forensics. Despite being a new application area, it has generated multiple research directions and multiple questions to be answered. Research activities have focused not only on trying to recognize fake content, but also on proving the existence of fingerprints left by GANs and trying to find a solution for the generalization of recognition techniques to new deepfake techniques never seen before.

As generation techniques become more and more realistic, deepfakes become more and more difficult to recognize. From the research reviewed in this section we can find two main approaches: (1) methods based on video or image semantics, such as those based on biological signals or facial landmarks and (2) methods based on statistical inconsistencies. Both approaches have proved successful, however, both have limitations.

The problem of generalization has attracted a lot of attention in this area. The advancement in the generation of very realistic images and

videos is proceeding at a very rapid pace, which makes it impossible to rely solely on methods for recognizing the fingerprints left by the various GANs or other generative methods. The methods that seem to be the most promising today are based on autoencoder or incremental learning. For autoencoders, the key idea is to disentangle the latent space vectors learned by the network to split into real and fake components (Cozzolino *et al.* (2018), Nguyen *et al.* (2019a), and Du *et al.* (2019)). The incremental learning strategy could be useful to retrain a network on a new type of technology without having to retrain it also on the previously seen classes (Marra *et al.*, 2019a). As the accuracy of generalization techniques also increases, many techniques based on face inconsistencies or facial movement in video sequences may soon become ineffective.

Most current research, both on creation and detection has focused on human faces; in the future generation may no longer be limited to just faces but to the entire human body.

6

Evaluation Metrics for Multimedia Forensics

There are some standard evaluation measures used by the designers of forensic systems when they evaluate them on validation and testing datasets. In the main areas that we consider in this survey, forgery detection source identification, and deepfake detection, the main learning task can be cast as a classification problem. As such, we present the most common measures for assessing the performance of classifiers, as well as some less known ones, used in the works that we describe in this survey.

- **Confusion matrix**. This matrix is the well-known table that shows the number of detection instances divided among four categories: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN); see Table 6.1.

Table 6.1: An example confusion matrix.

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

- **Accuracy**. This metric is defined as the ratio between the correct predictions and all the predictions made:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}.$$

  It is the most classical measure of how much the classifier is able to correctly classify the input data. However, when the positive and the negative classes are heavily unbalanced it is usually better to use other measures.
- **Precision and recall**. Based on the information provided by the confusion matrix, it is also possible to measure the degree of precision and recall, which provide further insight into the flexibility of the adopted detector for the considered application scenario. Namely, precision indicates what proportion of the positive identifications is actually correct and it is computed as the number of correct detections out of all positive predictions. Recall, instead, also known as *sensitivity* or *true positive rate*, indicates what proportion of the actual positives is correctly detected with respect to the ground truth, and it is computed as the number of items that are correctly detected out of all the ground truth ones:

$$precision = \frac{TP}{TP + FP}$$
$$recall = \frac{TP}{TP + FN}.$$

- **F1 Score**. This evaluation metric is frequently used for comparing different classifiers, as it conveys in a single score the information coming from both precision and recall for a specific forgery or deepfake detection task: it is computed as the harmonic mean between precision and recall for the chosen detection task:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}.$$

- **Precision–recall curve**. Many families of classifiers offer the capability to control the recall thus affecting the precision; typically, increasing the recall will lead to a decrease in the precision, and vice versa. This allows the user to make the appropriate tradeoff

for her application. The precision–recall curve shows this tradeoff: it is a plot in which the the x axis corresponds to the recall, ranging from 0 to 1, and the y axis to the precision, ranging from 0 to 1. The graph displays the precision of the classifier for each value of recall.

- **Intersection over Union (IoU)**. This metric is the allows to assess the effectiveness of any segmentation or object detection task carried out in the considered problem. Given two, typically orthogonally shaped, segments (bounding boxes or segmentation mask) of a picture, it measures the extent to which they match. More precisely, given the ground truth bounding box and the predicted one, the IoU is defined as:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union}.$$

  The numerator is the area of overlap between the predicted and the ground-truth bounding box. The denominator is the area of the union, that is, the area comprised of both the predicted and the ground-truth bounding box/segmentation mask. It is, therefore, a score between 0 and 1, with 0 meaning no overlap and 1 meaning complete match. A common way to evaluate the IoU score, that is, to identify whether the detection is correct or not, is to fix a *threshold*. Typically, if the IoU is higher than 0.5, it is considered a positive, else it is considered a negative.
- **Average Precision (AP)**. *Average precision* is a measure that attempts to summarize the precision–recall curve. It is defined as the mean precision over a set of eleven equally spaced recall levels $L = \{0, 0.1, \ldots, 1\}$ (Everingham *et al.*, 2010; Davis and Goadrich, 2006):

$$AP = \frac{1}{11} \sum_{r \in L} \tilde{p}(r).$$

  The *interpolated precision* $\tilde{p}(r)$ is defined as the maximum precision over all recall levels greater than or equal to r:

$$\tilde{p}(r) = \max_{r' \geq r} \{p(r)\},$$

  where $p(r)$ is the precision at recall level r. AP is typically calculated for each class and averaged to get the *mean average precision*

(MAP). The mean MAP over a set of C classes is the mean of the AP scores for each class:

$$MAP = \frac{1}{C}\sum_{i=1}^{C} AP_i.$$

The AP can also be computed and averaged over multiple IoU values. Moreover, in some computer vision applications, as well as multimedia forensic applications, the MAP could be referred to as AP. For example, for COCO challenge evaluation (Davis and Goadrich, 2006), there is no difference between AP and MAP.

- **ROC curve and area under the curve**. Similarly to the precision–recall curve, the *receiving operating characteristic (ROC) curve* summarizes the tradeoff between the true positive rate and false positive rate for a predictive model, using different probability thresholds. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0 and 1. Given a ROC curve, the *area under the curve* (AUC), provides a summary of the effectiveness of the classifier over the entire range of the classifier. It is a value that ranges from 0 to 1, and can be interpreted as the the probability that the result of applying the detector to some content randomly extracted from the set of pristine ones is higher than the result obtained by applying the detector to some content randomly extracted from the set of fake ones (Powers, 2008). The trivial classifier that just selects randomly with probability 0.5 that the image is forged and 0.5 that it is not has an expected AUC score of 0.5.
- **Mean square Error (MSE)**. This measure is used in statistics to evaluate the performance of an estimator of some unknown values, and it measures the average of the squares of the errors of the test set. Assume that the value estimated for the ith input is $\hat{y}_i$ and the actual value is y_i. Then the MSE is the average squared difference between $\hat{y}_i$ and y_i. Assume that we the test set contains n values y_i to be predicted:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

- **Cross correlation**. *Cross correlation* measures the correlation between the entries of two random vectors. For image processing applications it is used to compare the similarity between local parts (like for example, a vertical edge or a color variation) of an image (I) and a kernel (k), as follows:

$$I(x,y) \circ k(x,y) = \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} I(i,j)k(x+i,y+j),$$

where W and H are the the width and height of the image. If $I = k$, then it is called *auto-correlation*.
As image characteristic like brightness may vary because of lighting and exposure conditions, images are often normalized first. This is typically done at each stage by subtracting the average value among all the pixels of the images of the dataset and dividing by the standard deviation of the, that is, for each of the N pixels of an image:

$$\overline{I}(i,j) = \frac{I(i,j) - \mu}{\sigma},$$

where $i \in [0,\ldots,W-1], j \in [0,\ldots,H-1]$,

$$\mu = \frac{1}{N} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} I(i,j),$$

and

$$\sigma = \sqrt{\frac{\sum_{i=0}^{W-1} \sum_{j=0}^{H-1} (I(i,j) - \mu)^2}{N}}.$$

Sometimes the *normalized cross-correlation* (NCC) can be used to compare the cross correlation between an image and a filter k. It can be formally defined as:

$$NCC(I,k) = \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \overline{I}(i,j)\overline{k}(x+i,y+j),$$

where $\overline{I}(i,j)$ and $\overline{k}(i,j)$ are the normalized image and filter, respectively.

7

Multimedia Forensic Datasets

As we saw in the previous chapters, multimedia forensics has an important number of open challenges to deal with, each one requiring to deal with different data samples. Despite the fact that computer forensic involves the performance of computer-vision tasks, most of the time, open datasets that are used in other computer-vision applications cannot be (directly) used without the effort of introducing diverse forensic traces. In particular, training for forgery detection often involves the active introduction of image or video manipulations that, depending on what kind of manipulation one is interested to reproduce, can be extremely time consuming and difficult to perform. Nevertheless, there are no shortcuts and thus the forensic community has been quite active during the last years, by creating a reasonable number of datasets that can be used for every single task. In this chapter, we review the most adopted datasets from researchers, exposing the advantages and limitations of these resources. In Appendix B we present some statistics about each of these datasets, as well as from where it can be downloaded.

7.1 Forgery Detection

This is probably the most difficult family of forensic problems to reproduce, both because of the large range and number of possible manipulations and of the time necessary to generate a satisfactory size data set.

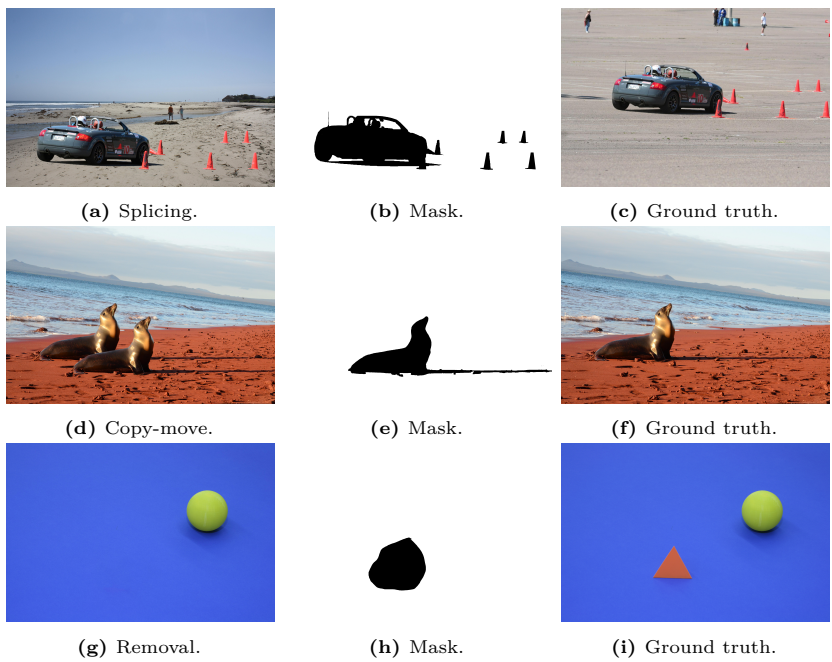
A first step in the direction of image-forgery detection is the development of datasets containing specific types of manipulations. Splicing is probably the most widespread image forgery, with a portion of an image copied and pasted into another image. Columbia gray (Ng and Chang, 2004) and Columbia color (Hsu and Chang, 2006) contain both authentic and spliced images of different size and format. Some of the works we present in this survey report results on the Columbia color dataset (see Table 7.1). Among these, the RR-Unet from Bi *et al.*, 2019 is the best performing model followed by the siamese network from Huh *et al.*, 2018.

Copy-move is another common manipulation in which a portion of an image is copied and reproduced into the same image. For this specific manipulation there are three different datasets: MICC F2000 (Amerini *et al.*, 2011), Erlangen (Christlein *et al.*, 2012), and COVERAGE (Wen *et al.*, 2016), with both authentic and manipulated samples. On this last dataset, the two-stream architecture from Zhou *et al.*, 2018a and ManTra-Net from Wu *et al.*, 2019 report comparable performance of 81.7% and 81.9% AUC respectively.

As we have seen in the previous chapters, face manipulation is another remarkable type of attack, which has attracted a lot of attention from the forensic community. Zhou *et al.*, 2018b introduced the FaceSwap dataset containing 1,927 face swapping examples. Their method reaches 92.7% AUC on this dataset.

Collecting heterogeneous manipulation classes, however, is much more valuable for the design of real-world forgery detectors; it includes a possibly infinite number of cases. CASIA 1 (Dong *et al.*, 2013) and CASIA 2 (Dong *et al.*, 2013) gather copy-move and splicing samples having more than 900 and 5,000 manipulated images, respectively. On thsee datasets, the RR-Unet by Bi *et al.*, 2019 reaches 84.1% F1-score, whereas the encoder-decoder architecture from Mazaheri *et al.*, 2019 reaches the highest score in terms of AUC (85.7%); see Table 7.1. Bappy

et al., 2019 introduce a synthetic dataset of 170,000 automatically generated samples of splicing, removal, and copy-move. Although large datasets are extremely important for deep-learning applications, the quality of counterfeiting cannot be underestimated. Therefore, the annual NIST-NC challenge, has proposed an increasing number of various manipulations of different quality levels, format, and sizes (Guan *et al.*, 2019) over time. Figure 7.1 shows some examples. Similarly, the PS Battles dataset (Heller *et al.*, 2018) stores 102,028 manipulated images of various sizes and formats. On these challenging datasets, multistream architectures typically perform better than other architectures (see Table 7.1).



(a) Splicing. **(b)** Mask. **(c)** Ground truth.

(d) Copy-move. **(e)** Mask. **(f)** Ground truth.

(g) Removal. **(h)** Mask. **(i)** Ground truth.

Figure 7.1: Image samples with corresponding ground truth masks from NIST 2016 Guan *et al.*, 2019.

Table 7.1 summarizes the performance of several methods on CASIA, Columbia color, and NIST datasets.

Table 7.1: Forgery detection experiments on Casia, Columbia color, and NIST datasets.

CASIA				
Citation	**Methodology**	**Acc.**	**AUC**	**F1**
Bi *et al.*, 2019	U-Net + residuals	—	—	84.1%
Wu *et al.*, 2018	Similarity net + Manipulation det. net	—	—	75.98%
Barni *et al.*, 2019	Siamese nets	75.86%	—	—
Zhou *et al.*, 2018a	RGB + noise streams	—	79.5%	40.8%
Mazaheri *et al.*, 2019	Encoder–decoder + LSTM	—	85.7%	—
Wu *et al.*, 2019	Feature extraction + anomaly detection	—	81.7%	—
Columbia color				
Citation	**Methodology**	**Acc.**	**AUC**	**F1**
Bi *et al.*, 2019	U-Net + residuals	—	—	91.5%
Zhou *et al.*, 2018a	RGB + noise streams	—	69.7%	69.7%
Wu *et al.*, 2019	Feature extraction + anomaly detection	—	82.4%	—
Mayer and Stamm, 2019	Siamese network	—	—	86.0%
Huh *et al.*, 2018	Siamese network + metadata	—	—	88.0%
NIST 2016				
Citation	**Methodology**	**Acc.**	**AUC**	**F1**
Zhou *et al.*, 2018a	RGB + noise streams	—	93.7% (NC16)	72.2% (NC16)

Citation	Methodology	Acc.	AUC	F1
Mazaheri *et al.*, 2019	Encoder–decoder + LSTM	—	81.4% (NC16)	-
Wu *et al.*, 2019	Feature extraction + anomaly detection	—	79.5% (NC16)	—
NIST 2017/2018				
Citation	**Methodology**	**Acc.**	**AUC**	**F1**
Marra *et al.*, 2019b	Patch features extraction + aggregation	—	93.2% (NC17) 90.2% (NC18)	—
Cozzolino and Verdoliva, 2020	Siamese net	—	—	39.5% (NC16) 38.0% (NC17) 38.0% (NC18)

7.2 Source Identification

Source identification opens the doors to two main challenges: social network and messaging app identification, and camera or device identification.

A well-made social-network–identification dataset should have a sufficiently high number of multimedia files, both directly from the camera and from social platforms. The larger the number of social platforms from which photos are taken the better for the design of a robust classifier. The Image Ballistic and Social Networks dataset (Giudice *et al.*, 2016) is surely the most heterogeneous dataset with respect to the number of platforms, with samples coming from Facebook, Google+, Twitter, Flickr, Instagram, Tumblr, Imgur, Tinypic, WhatsApp, and Telegram. The UCID (Caldelli *et al.*, 2017) and PUBLIC (Shullani *et al.*, 2017) datasets contain 30,000 and 3,000 JPEG images, respectively, from Flickr, Facebook, and Twitter. On the UCID dataset, the PRNU-based method from Caldelli *et al.*, 2018b reaches 90.83%

average precision, whereas the FusionNET from Amerini *et al.*, 2019a reaches 98.94% accuracy. Amerini *et al.*, 2019a also performed another experiment in which each image in the UCID dataset was uploaded and downloaded twice from Facebook, Flickr and Twitter, obtaining 86.49% accuracy on the classification of images shared across multiple social media. Hadwiger and Riess, 2020 have recently introduced a new dataset for camera identification with 27 smartphone devices, consisting of 25 different models from 9 brands. The dataset contains 23,000 images capturing 143 different scenes. Each image has exists in its original format and after it has been posted and processed by each of the Facebook, Instagram, Telegram, Twitter and WhatsApp.

During the years, researchers have created datasets containing addressing the source-identification problems on video content as well. The VISION dataset (Society, 2018) collects 34,427 JPEG images, and 1,914 mp4 videos from Facebook, YouTube and WhatsApp. The images and videos in this dataset were captured by 35 unique camera and device models. Starting from the VISION dataset, Yang *et al.*, 2020a introduced a new dataset for video-source identification called EVA-7k. The dataset contains 7,000 videos from 35 smartphones of 10 different brands. In particular, for each of the 35 smartphones it contains 4 videos from the VISION dataset, for a total of 140 videos. These 140 videos went under manipulation, both automatic using ffmpeg and Exiftool, and manual using Kdenlive, Avidemux, and Exiftool, resulting 1,260 additional videos. Each of these 1,400 videos is included in the dataset both intact as well as after having been shared through each of four social platforms—YouTube, Facebook, Tiktok and Weibo—for a total of 7,000 videos. Because of to the availability of images and videos captured by different devices and then uploaded to social networks, the VISION has been used by many works. Among these, Amerini *et al.*, 2019a; Phan *et al.*, 2019, and Quan *et al.*, 2019 have provided methods arriving at and accuracy above 77% for social media platform identification of images, whereas Amerini *et al.*, 2021 report 95.51% accuracy on the classification of videos shared across WhatsApp and YouTube. Cozzolino and Verdoliva, 2020 test the accuracy of their Noiseprint method on various forensics tasks (camera model and device identification, identification of JPEG quality factor, identification of

demosaicing algorithm, image manipulation detection and, splicing in remote sensing images). The model identification accuracy is 100%, both with 1024×1024 and 128×128 crops. Moreover, the method also ensures a modest device identification, with accuracy between 61.7% and 75.3%, even though the PRNU-based (see Section 4.3) method provides much higher accuracy, 91.3% and 69.7%, respectively.

A dataset for camera identification requires the collection of images and videos captures from different cameras and devices. To meet this need, several datasets have been released. Dresden (Gloe and Böhme, 2010) is the largest one, with 14,000 JPEG images captured by 25 different cameras and 73 unique devices. RAISE (Dang-Nguyen *et al.*, 2015) contains more than 8,000 raw images collected by 4 different cameras. In 2018, Kaggle launched the IEEE SPS Camera Model Identification challenge (Society, 2018), Bestagini, 2018) with 3,025 JPEG images captured by 10 different cameras and 20 unique devices.

Table 7.2 summarizes the performance of several methods on the Dresden and the VISION datasets.

Table 7.2: Source-identification experiments on Dresden and the VISION datasets.

Dresden		
Citation	**Methodology**	**Acc.**
Bondi *et al.*, 2017b	Quality-based patch selection + CNN	94.93%
Rafi *et al.*, 2018	DenseNet + SE	99.00%
Tuama *et al.*, 2016	HPF + CNN	91.9% 98.0%
Bayar and Stamm, 2017a	Constrained layer + MFR	98.52%
Pengpeng *et al.*, 2017	Laplacian filter + multi-steam CNN	70.19% 84.7% 94.17%

Citation	Methodology	Acc.
Mayer and Stamm, 2018	Siamese net	98.0%
Ding *et al.* (2019)	Hierarchical multi-task learning + multi-scale HPF	52.4% 97.1%
Mandelli *et al.* (2020b)	Noise residual and PRNU-based CNN	52.4% 97.1%
VISION		
Citation	**Methodology**	**Acc.**
Amerini *et al.* (2019a)	DCT + sensor-related noise residual CNN	99.47%
Phan *et al.* (2019)	DCT + metadata	100.00% 77.12%
Quan *et al.* (2019)	PRNU + CNN	79.91%
Amerini *et al.*, 2021	I-frames + P-frames CNN	95.51%

7.3 Deepfake Detection

In recent times, artificially generated content has attracted a lot of interest from researchers thanks to the incredible achievements of generative adversarial models. As a consequence, there has been an active production of new deepfakes datasets in the last few years.

The Fake Video Corpus dataset (Papadopoulou *et al.*, 2018) contains various types of forgery; it consists of 2,458 authentic videos and 3,957 forged videos.

Fake Faces in the Wild (FFW) by Khodabakhsh *et al.*, 2018 and FaceForensics++ by Rössler *et al.*, 2019 have been explicitly designed to allow for the evaluation of face-manipulation detectors. Both of them contain deepfakes and CGI, and FFW also has splicing samples. Similarly, the CelebDF dataset (Li *et al.*, 2020) includes 590 original videos collected from YouTube depicting people of different ages, ethic groups, and genders, and 5,639 corresponding deepfake videos. Bondi

et al., 2020 report 99.5% AUC when training and testing their model on the CelebDF dataset, and 71.71% when they train it on FaceForensics dataset and test it on CelebDF.

DeepfakeTIMIT (Korshunov and Marcel, 2018) has 620 JPEG fake images. The DeepFake Detection Dataset by Nick Dufour, 2019 collects more than 3,000 deepfakes in the H.264 encoding and different CRF values. Li and Lyu, 2018 measure the performance of their method with different backbones (namely VGG-16, ResNet50, ResNet101, and ResNet152) achieving up to 91.2% AUC on this dataset. DeeperForensics-1.0 (Jiang *et al.*, 2020) and the Deepfake Detection Challenge (AWS, Facebook, Microsoft, Partnership on AI's Media Integrity Steering Committee, 2020 - Dolhansky *et al.*, 2020) contain 10,000 and 100,000 artificial contents, respectively.

Table 7.3 summarizes the performance of several methods on the FaceForensics++ dataset.

Table 7.3: Forgery-detection experiments on the FaceForensics++ dataset.

FaceForensics++			
Method	**Methodology**	**Acc.**	**AUC**
Afchar *et al.*, 2018	Mesoscopic features + Inception	95.00%	—
Sabir *et al.*, 2019	Cropping + alignment of faces + manipulation detection	94.35% 96.9%	—
Bondi *et al.*, 2020	EfficientNetB4 + triplet loss	—	55.7% 96.0%
Caldelli *et al.*, 2021	RGB + optical flow	80.56% 95.75%	—

8

Discussion and Conclusions

With the significant diffusion of fake multimedia content, research in computer vision and its applications in multimedia forensics (especially the deep learning based ones) have become a hot topic and received a great deal of attention. Meanwhile, the enormous amount of data we daily have access to has allowed us to generate highly realistic forged multimedia contents as well as to devise successful methods for automatically spotting such fakes.

This survey provides a comprehensive outlook on the literature on forgery detection to anomaly-based architectures, from source identification to deepfake detection, especially with respect to GAN-generated content. It is clear that deep-learning methods are progressively bridging the long-standing semantic gap between computable low-level visual features and high-level image features. Despite recent progress on punctual tasks, investigating and modeling complex real-world problems still remains challenging.

Given the necessity to tackle these issues for forensic purposes as well as the enormous profit potential relative to such applications, the studies on multimedia forensic tasks will continue to grow and expand: in this respect, the survey highlights the most promising directions for future

research. First, as new and more complex generative manipulations and techniques emerge, simpler tools will become less effective. To address this problem, more complex multistream architectures have shown their potential. Therefore, more complex structures, tools, and data must be integrated to take advantage of all subtle information available to address multimedia-forensics problems. Along with the increasing complexity of media manipulation and generation techniques, the number of new tools and techniques being introduced makes it even more difficult to design deep-learning forgery-detection models that are robust to new attacks never seen before. In fact, despite the promising results, the main limitation of deep-neural networks originates from their high dependency on training data. The high number of operations (malevolent and innocent) that can be performed on an input, makes it practically impossible to reproduce all possible examples at training time. Consequently, higher robustness should be pursued by other means. Furthermore, to cope with rapid advances in manipulation technology, deep networks should be able to adapt to new manipulations, without complete retraining, which may simply be impossible because of lack of training data or lead to catastrophic forgetfulness. Still in this direction, the works reviewed in this survey, have been mostly applied in controlled settings. Thus, new techniques are needed to apply multimedia forensics in the wild. One attempt to cope with the complexity of the real world is to take into consideration multiple media at a time. For example, to decide on the authenticity of the news, we can rely not just on an image or video content, but also on the text or audio attached to it. In this direction, DARPA recently launched a new initiative on *semantic forensics*.[1] The challenge is not just to decide on the authenticity of an image or video, but to capture all semantic inconsistencies that can be discovered in a multimodal media asset. A multimodal approach can be particularly useful to detect deepfakes, where a video and an audio track are typically available. Also, semantic inconsistencies can be used in the future to detect anomalies on deepfakes of the entire human body, without examining only the human face.

[1] https://www.darpa.mil/program/semantic-forensics

One of the major current limitations of deep learning is their lack of interpretability. The complexity of deep learning-models makes it difficult to understand why they produce an output value. This problem is particularly relevant in multimedia forensics given the fact that they are often used for law-related applications. This means that it is often not sufficient that a classifier reports an image as fake or that a video is from a certain social network but to also report the features and the procedures that led to such an output. Furthermore, being able to interpret the logic of a deep neural network would allow to improve its design and training phase, and provide higher robustness with respect to malicious attacks. On a related issue, deep neural networks open up new vulnerabilities that can be exploited by an attacker. Despite the neural networks' ability to learn forensic features directly from data, intelligent attackers can use this to their advantage. Because the space of possible inputs to a neural network is substantially larger than the set of images used to train it, an attacker can create modified images that fall into an unseen space and force the neural network to misclassify. One method of accomplishing this involves introducing adversarial perturbations into an image (see Goodfellow *et al.*, 2015). With respect to this, GANs can become a new threat not just by generating very realistic images or videos, but also as counter forensics tools (see Barni *et al.*, 2018 for more details). They have already been used to remove forensic traces left by median filtering Kim *et al.*, 2018, and it is very likely that more GAN-based counter-forensic attacks will be developed in the near future.

Appendices

A

Computer Vision and Signal Processing for Media Forensics

Multimedia forensic is a research area that requires a basic understanding of computer-vision and signal-processing techniques. To facilitate the understanding of readers new to these two fields, in this section we want to introduce some basic background. Obviously, this section is not intended as an exhaustive treatment of these two disciplines, see the relevant books for more details (e.g., Goodfellow *et al.*, 2016). Specifically, in the next pages, we cover basic *deep-learning* topics for *computer-vision* applications and some basic *signal-processing* concepts that we refer to in the main text.

A.1 Deep-Learning Architectures for Computer Vision

Deep learning solves the fundamental problem in representation learning by learning representations that are expressed in terms of other, simpler forms. From a mathematical point of view, an artificial neural network is a mathematical function mapping some set of input values to output values. The function is constructed by composing many simpler functions. We can think of each application of a different mathematical function as providing a new representation of the input.

Feedforward neural networks are typically constructed by composing together many different functions, also informally called *neurons*. The model is associated with a directed acyclic graph describing how the functions are composed together. The network can be structured in several layers of neurons. The overall length of the chain gives the *depth* of the model. The first layer of a feedforward network is called the *input layer* and the last one the *output layer*. The layers in between the input and the output layers are called *hidden layers*. Each neuron in a layer typically performs two basic operations, a linear transformation and a nonlinear transformation. For example, given an input x, the output of a layer will be $\hat{y} = \sigma(W^T x + b)$, where $z = W^T x + b$ is a linear function and $\sigma(z)$ is a nonlinear function also called *activation function*.

In this section, we discuss different architecture choices and explain how each of these configurations can be most useful in solving a specific problem.

A.1.1 Fully Connected Networks

Fully connected networks (FCNs) are an essential method of deep learning. The main advantage of FCNs is that they are independent of the structure, that is, there is no need to make special assumptions about the input (for example, that the input consists of images or videos). They owe their name to the fact that each neuron in a certain layer is connected with all the neurons of the layer that precedes it and each neuron of the layer that follows it. As a result, these networks are fully connected. Figure A.1 shows an example of an FCN.

Although being independent of structure makes FCNs widely applicable, they tend to have lower performance than special networks tuned to the structure of a specific problem space. In fact, because of their structure, these networks are not robust to input data for which there is a two-dimensional or three-dimensional relationship such as images and videos. Furthermore, these networks do not take into account the dependence of input sequences such as text or video sequences. For these reasons, in computer-vision applications these networks are not commonly used to classify input features. Usually, these networks are used after a convolutional neural network or a recurrent neural network

that work as feature extractors, that is, they learn how to extract relevant features that are useful to classify the input. Then, the FCN takes the feature vector as input and predicts the corresponding class.

Even if the FNCs are very often used as classifiers, it is still possible to apply them for regression problems or to train a network to project inputs into a latent space as happens, for example, in some applications that use Siamese networks.

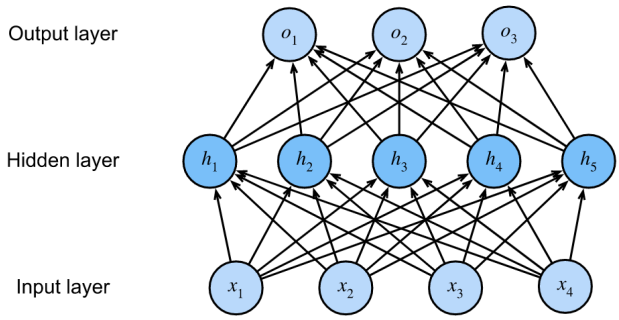


Figure A.1: An example of an FCN with a hidden layer of five hidden units (Zhang *et al.*, 2020).

A.1.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a specific kind of neural network for processing data that has a known grid-like structure. The most representative class of this family is image data, which can be thought of as a two-dimensional grid of pixels. These networks use a mathematical operation called *convolution* in place of a general matrix multiplication in at least one of their layers. Given a two-dimensional image I and a kernel K the convolution between I and K is defined as follows:

$$\begin{aligned}(I * K)(i,j) &= \sum_m \sum_n I(m,n)K(i-m, j-n) \\ &= \sum_m \sum_n I(i-m, j-n)K(m,n).\end{aligned}$$

Convolution leverages three important ideas that can help improve a computer-vision system: (1) sparse interactions, (2) parameter sharing,

and (3) equivariant representations. Traditional neural-network layers use matrix multiplication by a matrix of parameters with a separate parameter describing the interaction between each input unit and each output unit, meaning that every output unit interacts with every input unit. CNNs, however, typically have sparse interactions (also referred to as sparse connectivity or sparse weights), which is accomplished by making the kernel size smaller than the input size. Thanks to this strategy, we can use the same parameters for more than one input unit in a model (also referred as parameter sharing). In a traditional neural network, each element of the weight matrix is used exactly once when computing the output of a layer. It is multiplied by one element of the input and then never reused. For CNNs, the particular form of parameter sharing causes the layer to have a property called equivariance to translation. To say a function is equivariant means that if the input changes, the output changes in the same way. Figure A.2 shows an example of a CNN.

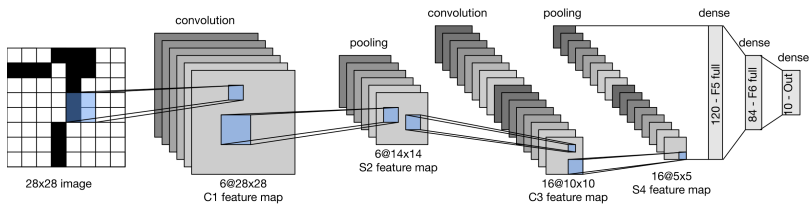


Figure A.2: Example of a CNN consisting of two convolutional layers; and a dense block consisting of three fully-connected layers (Zhang *et al.*, 2020; Lecun *et al.*, 1998).

A.1.3 Recurrent Neural Networks

Similarly to CNNs, *recurrent neural networks* (RNNs) are specialized neural networks for processing sequential data of the form $x^{(1)}, \ldots, x^{(t)}$. At each time step t, the state of a hidden unit h depends on its state at time $t-1$, that is:

$$\begin{aligned} h^{(t)} &= \sigma_h(W_{hh} \cdot h^{(t-1)} + W_{hx} \cdot x^{(t)} + b_h) \\ &= \sigma_h([W_{hh} W_{hx}] \cdot [h^{(t-1)} x^{(t)}] + b_h) \end{aligned}$$

where σ_h is a nonlinear (activation) function, $x^{(t)}$ represents the input at time t, W_{hh}, W_{hx} are the weight matrices associated to the actual hidden state $h^{(t-1)}$ and input $x^{(t)}$ respectively, and b_h a parameter vector. Forward propagation typically begins with a specification of the initial state $h^{(0)}$.

Depending on the problems on which they are applied, RNNs can be structured in different ways: (1) RNNs that generate an output at each time step and have recurrent connections between hidden units, (2) RNNs that produce an output at each time step and have recurrent connections only from the output at one time step to the hidden units at the next time step, and (3) RNNs with recurrent connections between hidden units, that read an entire sequence and then produce a single output. Figure A.3 shows an example of an RNN applied to character-level language processing.

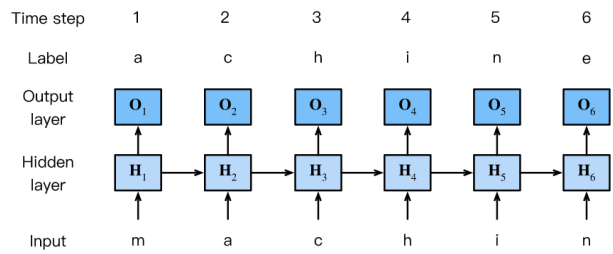


Figure A.3: Example character-level language RNN. The input and label sequences are *machin* and *achine*, respectively (Zhang *et al.*, 2020).

A.2 Common Deep Learning Backbones

Neural networks are often combined into complex design schemes that help them learn better the task they are solving. Every year, new architectures are published for solving new problems or achieving higher performance than previous models. In this section, we present some of the most common architectures used in the architectures of the survey. Obviously, our goal is not to provide an exhaustive discussion of all the backbones that can be used in computer vision or multimedia forensics, but to offer a quick guide to learn about the most used architectures in the works that we survey.

A.2.1 VGG

The VGG network (see Figure A.4) was designed by Simonyan and Zisserman, 2014. The input image passes through a stack of convolutional layers that use 3×3 filters, which is the smallest size to capture the notion of left/right, up/down, center. The convolution stride is fixed to 1 pixel and the padding is 1 pixel.[1] Each of the convolutional blocks is followed by a max-pooling layer which is performed over a 2×2 pixel window, with stride 2. The stack of convolutional layers (which can be constructed with different depths) is followed by three fully connected layers: the first two have 4096 channels each and the third has 1000 neurons corresponding to the output number of classes of the ImageNet dataset. The final layer is the softmax layer. In one of the configurations (VGG16), the network also uses 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by nonlinearity). All hidden layers are followed by ReLU activations. This network can be configured with different depths varying from 11 weight layers to 19 weight layers. The width of the convolutional layers (the number of channels) is rather small, starting from 64 in the first layer and then increasing by a factor of 2 after each max-pooling layer, until it reaches 512. Depending on the number of layers N, this network is typically referred to as VGGN. The most common configurations are VGG16 and VGG19.

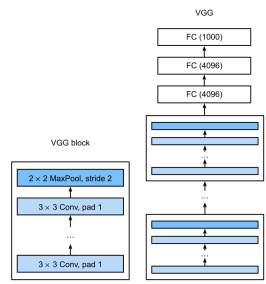


Figure A.4: Example of the VGG architecture from building blocks to the entire model (Zhang *et al.*, 2020).

[1] *Stride* and padding are parameters of CNNs; see Goodfellow *et al.*, 2016.

A.2.2 ResNet

He *et al.*, 2015 introduced *ResNets* (see Figure A.5) to solve the vanishing-gradient problem: When a neural network is too deep, the gradients are easily reduced to zero for the early layers of the network, with the result that the weights no longer update their values and, therefore, the model stops learning. The key idea is to use shortcut connections from early layers up to deeper (later) layers. Formally, denoting the desired underlying mapping as $H(x)$, we let the stacked nonlinear layers fit another mapping of $F(x) = H(x) - x$. The original mapping is recast into $F(x) + x$. The dimensions of x and $F(x)$ must be equal, thus the ResNet performs a linear projection W_s by the shortcut connections to match the dimensions:

$$y = F(x, \{W_i\}) + W_s \cdot x.$$

where $F(x, \{W_i\})$ represents the residual mapping to be learned. For example, it may represent two layers of the form $F = W_2 \cdot \sigma(W_1 \cdot x)$, in which σ denotes the ReLu function.

Skip connections between layers add the outputs from previous layers to the outputs of stacked layers. This allows information to be propagated to later levels without running into the problem of vanishing gradients thus allowing us to train deeper networks than was previously possible. He *et al.*, 2015 designed a plain network with 3×3 filters by following two simple design rules: (1) for the same output feature map size, the layers have the same number of filters and (2) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. The network performs downsampling directly by convolutional layers that have a stride of 2. The network ends with a global average pooling layer and a 1000-dimensional fully connected layer with softmax. Shortcut connections between layers increase the depth of the network. The ResNet network can be configured with different depths varying from 18 to 152 layers. Depending on the number of layers N, the network is typically referred to as ResNet-N. Very commonly, the network is used as ResNet-18, ResNet-50, or ResNet-100.

Figure A.5 shows two examples of residual blocks. ResNet follows VGG's convolutional layer design. The residual block has two 3×3

convolutional layers with the same number of output channels. Each convolutional layer is followed by a batch normalization layer and a ReLU activation function. Then, a residual connection propagates the input of these two convolution operations directly before the final ReLU activation function. This kind of design requires that the output of the two convolutional layers has to be of the same shape as the input, so that they can be added together. To change the number of output channels, an additional 1×1 convolutional layer can be used to transform the input into the desired shape for the addition operation.

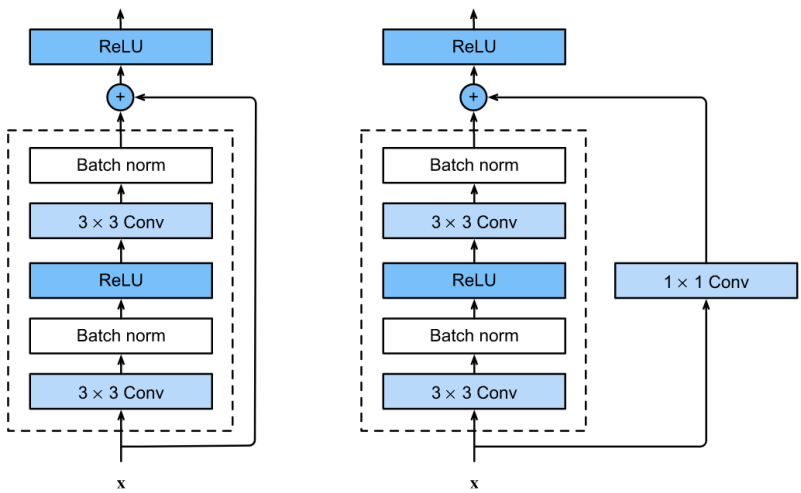


Figure A.5: Example ResNet blocks. A regular block (left) and a residual block (right) (Zhang *et al.*, 2020).

A.2.3 Inception

Parts of interest in an image can have extremely large variations in their size. This variety in the area of interest can make difficult the determination of the right kernel size for the convolution operation. A larger kernel is preferred for information that is distributed more globally, whereas a smaller kernel is preferred for information that is distributed more locally. The idea of the *inception network* (also known as *GoogLeNet*; see Szegedy *et al.*, 2014) is to have filters with multiple sizes operating on the same layer, called the *inception layer*. An inception

layer performs a convolution on the input with three different kernel sizes: 1×1, 3×3, and 5×5. Additionally, max pooling is also performed in parallel to the filters. However, CNNs are computationally expensive. In GoogLeNet, 1×1 convolution is used as a dimensionality-reduction module to reduce the computation. By reducing the computation bottleneck, depth and width can be increased. Thus, Szegedy *et al.*, 2014 limit the number of input channels by adding an extra 1×1 convolution before the 3×3 and 5×5 convolutions. The 1×1 convolutions require much less computation than 5×5 convolutions, and applying them before the other filters reduces the size of input channels. The 1×1 convolution is also applied after the max-pooling layer. After that, all feature maps at different paths are concatenated together as the input of the next module. Figure A.6 shows an example of the inception block.

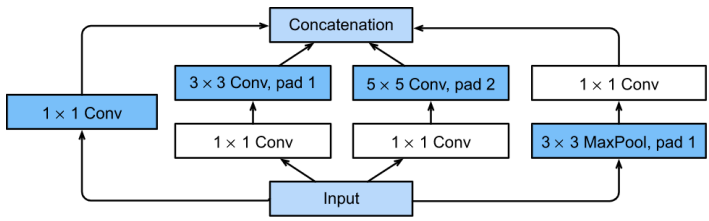


Figure A.6: Example of the structure of the inception block (Zhang *et al.*, 2020).

In GoogLeNet (Figure A.7), global average pooling is used at the end of network by averaging each feature map from 7×7 to 1×1.

The Inception network described so far is also known as Inception-v1. Subsequently, several enhancements of this version were introduced also known as Inception-v2 and Inception-v3 (Szegedy *et al.*, 2015), Inception-v4 and Inception-ResNet (Szegedy *et al.*, 2016).

A frequently used variation of Inception is called Xception (Chollet, 2016), which stands for *extreme inception*. In a traditional CNNs, convolutional layers seek out correlations across both space and depth. In Inception, 1×1 convolutions project the original input onto several separate, smaller input spaces, and from each of these input spaces some other type of filter transforms those smaller 3D blocks of data. Xception takes this one step further. Instead of partitioning input data into multiple compressed chunks, it maps the spatial correlations for

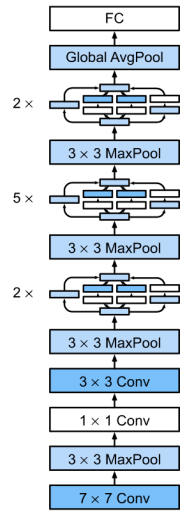


Figure A.7: The GoogLeNet architecture (Zhang *et al.*, 2020).

each output channel separately, and then performs a 1×1 depthwise convolution to capture cross-channel correlation. This is equivalent to an existing operation known as a *depthwise separable convolution*, which consists of a depthwise convolution (a spatial convolution performed independently for each channel) followed by a pointwise convolution (a 1×1 convolution across channels). See Chollet, 2016 for further information.

A.2.4 Long Short-Term Memory Networks

Long short-term memory networks (LSTMs) Hochreiter and Schmidhuber, 1997 are a special kind of RNN, capable of learning long-term dependencies. Typical RNNs suffer from short-term memory. If a sequence is long enough, they will have a hard time carrying information from earlier time steps to later ones. LSTMs are designed to avoid the long-term dependency problem. They have internal mechanisms called *gates* that can regulate the flow of information. These gates can learn what data in a sequence are important to keep or throw away. By doing that, they can pass relevant information down the long chain of sequences to make predictions.

The LSTM has four types of gates (see Figure A.8):

- *Forget gate* (F_t). This gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through a sigmoid function.
$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f)$$
- *Input gate* (I_t). It decides what values will be updated. The previous hidden state and current input are passed into a sigmoid function. This decides what values will be updated by transforming them to be between 0 and 1.
$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i)$$
- *Cell state* or *long-term memory* (C_t). The cell state is pointwise multiplied by the forget vector. This has the possibility of dropping values in the cell state if it is multiplied by values close to 0. Then it takes the output from the input gate and computes a pointwise addition with the candidate memory cell $\tilde{C}_t = \tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c)$, which updates the cell state to new values that the neural network finds relevant.
$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t.$$
- *Output gate* (O_t). It decides what the next hidden state should be. It passes the previous hidden state and the current input into a sigmoid function. Then it passes the newly modified cell state to the tanh function. The output of the tanh is multiplied with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden state are then carried over to the next time step.
$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o)$$

Finally, the output hidden state can be simply calculated as $H_t = O_t \odot \tanh(C_t)$. If the output gate approximates 1 then it passes all memory information through to the predictor, whereas if the output gate is close to 0, it retains all the information only within the memory cell and performs no further processing.

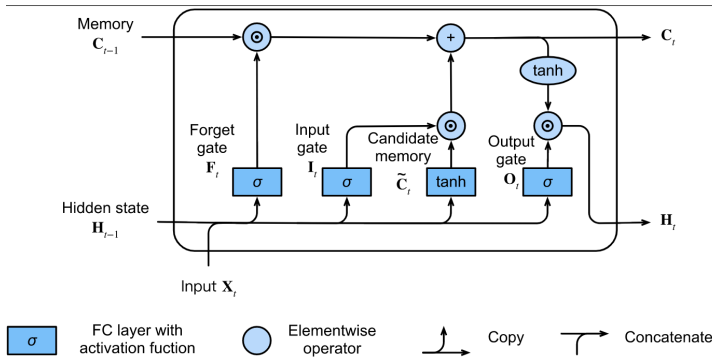


Figure A.8: Example of a hidden state in an LSTM model (Zhang *et al.*, 2020).

A.3 Signal Processing for Multimedia Forensics

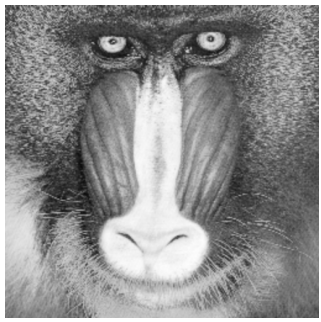
Signal processing is an important part of multimedia forensics. Indeed, image data can be represented as a signal that can be modeled by waves. For grayscale images, we can model them as a matrix of values, where the element at position (i, j) in the matrix corresponds to the pixel at position (i, j) in the image, and the value of that matrix element is the pixel's intensity. For example, 0 may correspond to black pixels, and 255 to white pixels. Pixel intensities between 0 and 255 are interpreted as colors between black and white. Figure A.9 shows an example applied on a grayscale image.

A similar approach can be used for color images modelling colors as separate signals or as a three-dimensional signal (one dimension for each color channel).

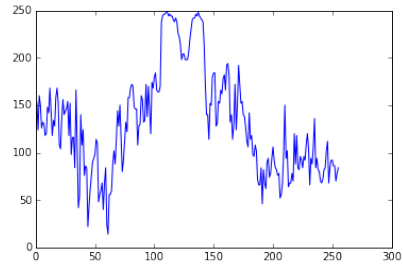
For most concepts (discrete Fourier transform, filters, etc.) consult textbooks on signal and image processing Vetterli *et al.*, 2018; Szeliski, 2011. Here we present some more specific concepts that may help in reading this survey.

A.3.1 Discrete Cosine Transform

Discrete cosine transform (DCT) is a signal-processing operation that expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. The DCT is a type of



(a) Greyscale image (Shanker, 2021).



(b) Signal representation.

(c) Row of pixels from image A.9a.

Figure A.9: An example (Shanker, 2021) grayscale image (A.9a). Given a pixel's row of pixels extracted from the image ((A.9c)), it can be represented as a signal (A.9b).

Fourier-related transformation and is commonly used as a lossy compression technique. A Fourier transform is the process of decomposing a digital signal into the sum of some trigonometric functions. A Fourier transform is called a transform because it transforms the data from one form (the amplitude or pixel intensity over time) into a list of frequency coefficients controlling their contribution. The DCT has the property that most of the visually significant information about the image is concentrated in just a few coefficients of the DCT. For this reason, in image processing applications, DCT is very often used as a form of lossy compression technique. As an example, the DCT is at the heart of the international standard JPEG and MPEG algorithms. In the frequency representation of an image, some of the higher frequency components, such as the smaller changes in amplitude leading up to peaks, are less important, and could be removed without losing visual components that are needed to understand the image content. Once that the image has been decomposed into a collection of trigonometric functions, it becomes easy to remove less important frequency functions that don't contribute as much to the core structures of the image.

The DCT is a linear transformation that transforms a vector of length n of pixel intensities (a row of pixels of an image), and returns

a different vector of length n containing the coefficients for n different cosine functions. Thus, the vector is encoded by an $n \times n$ matrix, in which each row corresponds to a cosine function of a different frequency. Using n cosine functions is the key to being able to get our data back in terms of amplitudes after converting it to cosine coefficients. To represent each cosine wave as a row in the $n \times n$ matrix X, we compute it as:

$$X_{i,j} = \cos\left(\frac{\pi}{n} i \left(j + \frac{1}{2}\right)\right)$$

where i and j indicate rows and columns of the matrix respectively. In the equation above, each row corresponds to a different cosine function and the higher values of i correspond to cosine waves of higher frequency.

The last step, after calculating the DCT matrix, is to calculate the decomposition and the correct coefficients for each of the component waves. The decomposition can be easily computed by taking the dot product of the input vector of pixel intensities and X_i. The dot product of these two components can be interpreted as a measure of similarity between the two vectors, that is, if the pixel data is coincident with the values in one particular wave, it will be 0. Therefore, by computing this dot product, we can figure out what coefficient to use for that particular wave. This technique, can be similarly applied on two-dimensional matrices (i.e., two-dimensional image signals) by performing the DCT twice, once along the rows, and once along the columns.

To compress the image, we take the K most significant cosine waves in X_i, and save the coefficients. To get the compressed image back, we pad the matrix with 0s to get an $n \times n$ matrix (the original image's size), and then apply on it the inverse DCT transform to obtain the compressed image.

A.3.2 PRNU

When an photograph is taken by a camera, it is processed through a sequence of operations illustrated in Section 4. These operations may introduce noise and various imperfections to the image. Even if the imaging sensor takes a picture of an absolutely evenly lit scene, the resulting digital image will typically still exhibit small changes in intensity between individual pixels. This is partly because of the *shot*

noise created by the electronic circuits, which is a random component, and partly because of the pattern noise created by the image sensors, a fixed component that remains approximately the same if multiple pictures of the exact same scene are taken. This implies that the pattern noise is impressed in every image the sensor takes and, thus, can be used for camera identification. Averaging over multiple images reduces the random components and enhances the pattern noise.

The two main ingredients of pattern noise are *fixed pattern noise* (FPN) and *photo-response nonuniformity* (PRNU). The FPN is caused by dark currents, that is, by pixel-to-pixel differences when the sensor array is not exposed to light. As it is an additive noise, very commonly, consumer cameras suppress it automatically by subtracting a dark frame from every image they take. Therefore, the dominant part of the pattern noise of an image is the PRNU. It is caused primarily by pixel nonuniformity (PNU), which is the different sensitivity of pixels to light caused by the inhomogeneity of silicon wafers and imperfections during the sensor manufacturing process. Because of its origin, it is unlikely that even sensors coming from the same wafer would exhibit correlated PNU patterns. So, the PNU noise is not affected by ambient temperature or humidity, but light refraction on dust particles and optical surfaces and zoom settings contribute to the PRNU noise. Since these low-frequency components are not a characteristic of the sensor, if we capture this noise pattern, we can create a distinctive link between a camera and its photos.

Formally, given a digital image I taken from camera a C, it can be modeled as:

$$I = I^{den} + I^{den}K + \theta$$

where I it the acquired image, I^{den} is the denoised image, K is the PRNU and θ represents other noise terms (e.g., shot noise). PRNU is usually estimated from N images captured with the same camera. The estimate can be computed with two simple steps: (1) the application of high-pass filtering $W_i = I_i - I_i^{den}$ on each image i, followed by (2) an estimate operation:

$$\hat{K} = \frac{\sum_{i=1}^{N} W_i \cdot I_i}{\sum_{i=1}^{N} (I_i)^2}.$$

The PRNU fingerprint $\hat{K}$ is obtained through a minimum variance estimator as indicated in the equation above, where N is the number of images used for the estimation.

A.3.3 JPEG Compression

JPEG is an acronym for *joint photographic experts group* and it refers to the *JPEG file interchange format* (JFIF). Usually, the files with the `.jpg` extension are JFIF files. It was created as a standard for digital image compression. JPEG is *lossy* compression technique, meaning that the image changes and loses some detail as a result of the compression. JPEG compression is actually composed of three different compression techniques, which are applied in successive layers: (1) chrominance subsampling, (2) DCT and quantization, and (3) delta, run-length, and Huffman encoding. Chrominance subsampling is the process of representing an image's color components at a lower resolution than its actual luminance components. This step is used to reduce the file size of colored images. For grayscale images, this step can be skipped. This step begins by converting the image from RGB to YUV color space. Because the human eye is more sensitive to luminance than to chrominance, typically JPEGs discard most of the chrominance information before any other compression takes place, so the image contains only half as much color information as it originally did. This first step already reduces the amount of information of the image to be stored. Next, the image is partitioned into 8×8 nonoverlapping pixel blocks and the DCT of each block is computed, resulting into a set of 64 subbands of DCT coefficients. The DCT coefficients are then quantized by dividing them by the entry in a quantization matrix that corresponds to the coefficient's subband and then rounding the resulting value to the nearest integer. Because the human visual system has different sensitivities to luminance and color distortions, different quantization tables are generally used to quantize the luminance and chrominance layers. Finally, each quantized DCT coefficient is converted to binary and then reordered into a single bit stream using the zigzag scan order [2]. The third and last compression

[2] See https://www.ece.ucdavis.edu/cerl/reliablejpeg/compression/ for further details.

layer is lossless. Initially, each DCT coefficient is converted from an absolute value to a relative value: Adjacent blocks in an image tend to have a high degree of correlation, so the protocol encodes the DCT term of a given block as a difference from the previous DCT term; the difference is typically a very a very small number and can be stored in a small number of bits—we call this encoding *delta encoding*. This process will typically create a lot of differences of value equal to zero. The next step encodes zeros into a *run-length encoding*, that is, it only stores the count of consecutive (differences of) zero values. Finally, the image is compressed with Huffman encoding, which is stored in the JPEG header.

MPEG (moving picture experts group) is a standard for video coding. It is used to compress video sequences and it is very similar to JPEG. The main difference with videos is that it also performs block-based motion compensation (see Sullivan *et al.*, 2012): it encodes the difference between each block and a predicted set of pixel values obtained from a shifted block in the previous frame. In fact, the encoder splits the video frame sequence into smaller segments called *group of pictures* (GOP). Each GOP starts with an *I-frame* which is an image independently encoded using a process similar to JPEG compression and continues with the predicted frames (*P-frames*) and bidirectional frames (*B-frames*). P-frames are predicted from preceding frames and B-frames can be predicted from I-frames or P-frames preceding or following them in the GOP. Check the MPEG official web page[3] of the MPEG group for further details.

In Section 2.1 you will find more details on how compression can be used in multimedia forensics applications.

[3] https://www.mpegstandards.org

B

Tables

B.1 Forgery Detection Methods

Table B.1: Deep learning architectures for image forgery detection.

Archit.	Convolutional layers					F.C. layers		DB	Ref.
	Preproc.	Input	Streams	Activ.	Pool.	Layers	Activ.		
RRU-Net	Gaussian noise, JPEG compr.	384 × 256 × 3	1 (27 layers)	ReLU, sigmoid	Max	—	—	Hsu and Chang 2006 Dong *et al.*, 2013	Bi *et al.*, 2019
DRN-C-26	Resizing, JPEG compr., brightness, contrast, saturation	400 or 700 in the shorter size	1 (26 layers)	ReLU	Max	—	—	Rössler *et al.*, 2019	Wang *et al.*, 2019a
BusterNet	—	256 × 256 × 3	2	—	Bilinear, per-centile	—	—	Tralic *et al.*, 2013 Dong *et al.*, 2013	Wu *et al.*, 2018
Multi-Task Fully Convolutional Network (MFCN)	—	—	2	ReLU	Max	—	—	Dong *et al.*, 2013 Hsu and Chang, 2006 Guan *et al.*, 2019 Carvalho *et al.*, 2013	Salloum *et al.*, 2018
Multi-domain CNN	DCT	64 × 64 × 3, 909 × 1	2	ReLU	Max	3	Softmax	Tralic *et al.*, 2013 Dong *et al.*, 2013	Amerini *et al.*, 2017b
Two-Stream Faster R-CNN	SRM filter	600 in the shorter size	2	ReLU	Bilinear, max	1	Softmax	Hsu and Chang, 2006 Dong *et al.*, 2013 Wen *et al.*, 2016 Guan *et al.*, 2019	Zhou *et al.*, 2018a
Two-Stream Neural Networks for Tampered Face Detection	patches, steganalysis	299 × 299 × 3, 3×128×128×3	2	ReLU	Global avg, max	—	—	Zhou *et al.*, 2018b	Zhou *et al.*, 2018b

Table B.1: (Continued) Deep learning architectures for image forgery detection

Archit.	Convolutional layers					F.C. layers		DB	Ref.
	Preproc.	**Input**	**Streams**	**Activ.**	**Pool.**	**Layers**	**Activ.**		
Hybrid LSTM and Encoder-Decoder	Laplacian filter, radon transform, FFT	$256 \times 256 \times 3$, $256 \times 256 \times 3$	2	ReLU	Max	—	—	Wen *et al.*, 2016 Guan *et al.*, 2019 Gloe and Böhme, 2010 Society, 2014	Bappy *et al.*, 2019
Forensic Similarity Network	Patches	$256 \times 256 \times 3$ / $128 \times 128 \times 3$	2	ReLU	Max	2	Tanh, sigmoid	Gloe and Böhme, 2010	Mayer and Stamm, 2020
ForensicTransfer	Third-order derivative	$256 \times 256 \times 3$	1	ReLU, tanh	—	—	—	Rössler *et al.*, 2018b Cozzolino *et al.*, 2018	Cozzolino *et al.*, 2018
ManTra-Net	—	$256 \times 256 \times 3$ or $512 \times 512 \times 3$	1	ReLU, L2norm, sigmoid	Max, avg, ZPool2D	—	—	Gloe and Böhme, 2010 Society, 2018 Bestagini, 2018 Wu *et al.*, 2019	Wu *et al.*, 2019

Table B.2: Deep learning architectures for video forgery detection.

Archit.	Convolutional layers					F.C. layers		DB	Ref.
	Preproc.	Input	Streams	Activ.	Pool.	Layers	Activ.		
TS-N	I-frames, P-frames	256 × 256	2	ReLU	Max, avg, global avg	3	ReLU, softmax	Montgomery *et al.*, 1994 Lin *et al.*, 2015 Almohamedh *et al.*, 2015	Nam *et al.*, 2019
C3D-based Convolutional Neural Network for Frame Dropping Detection in a Single Video Shot	Convert frames to motion residual images by means of an absolute difference algorithm	16-frames	1	ReLU	3D pool	2	Softmax	Long *et al.*, 2017	Long *et al.*, 2017
C2F-DCNN	—	64-frames	2	ReLU	Max, avg	—	—	Guan *et al.*, 2019 Long *et al.*, 2018	Long *et al.*, 2018
Video Codec Forensics Based on Convolutional Neural Networks	Patches	64 × 64 × 3	2	ReLU, SeLU	Max	1	SeLU, softmax	Verde *et al.*, 2018	Verde *et al.*, 2018

B.2 Source Camera Model Identification Methods

Table B.3: Deep learning architectures for source camera model identification.

Archit.	Convolutional layers					F.C. layers		DB	Ref.
	Preproc.	Input	Streams	Activ.	Pool.	Layers	Activ.		
Noiseprint	Patches	$48 \times 48 \times 3$	1	ReLU	—	—	—	Guan *et al.*, 2019 Zhou *et al.*, 2018b Carvalho *et al.*, 2013 Bianchi and Piva, 2012 Shullani *et al.*, 2017 Gloe and Böhme, 2010	Cozzolino and Verdoliva, 2020
Forensic Similarity Network	Patches	$256 \times 256 \times 3$ or $128 \times 128 \times 3$	2	ReLU	Max	2	Tanh, sigmoid	Gloe and Böhme, 2010	Mayer and Stamm, 2020
ACFM-based CNN	Green channel, MFR	$256 \times 256 \times 2$	1	—	Max, avg	3	Tanh, softmax	Gloe and Böhme, 2010	Bayar and Stamm, 2017a
Inception-Based Data-Driven Ensemble Approach to Camera Model identification	Patches as Bondi *et al.*, 2017b	$64 \times 64 \times 3$, $71 \times 71 \times 3$, $224 \times 224 \times 3$, $256 \times 256 \times 3$ or $299 \times 299 \times 3$	2	ReLU	Max, global avg	2	Softmax	Society, 2018 Bestagini, 2018	Ferreira *et al.*, 2018
Augmented convolutional feature maps for robust CNN-based camera model identification	Green channel, MFR	$256 \times 256 \times 2$	1	Tanh	Max, avg	3	Tanh, softmax	Gloe and Böhme, 2010 Bayar and Stamm, 2017a	Bayar and Stamm, 2017a
Content-adaptive fusion network	Patches	$64 \times 64 \times 3$	3	ReLU	Avg, global avg	—	Softmax	Gloe and Böhme, 2010 Yang *et al.*, 2017	Yang *et al.*, 2017
CNN-based fast source device identification	PRNU, noise residual as Chen *et al.*, 2008	from 80×80 to 720×720	1	Leaky ReLU	Max, pair-wise correlation pooling	1	—	Gloe and Böhme, 2010 Shullani *et al.*, 2017 Mandelli *et al.*, 2020b	Mandelli *et al.*, 2020b

B.3 Datasets

Table B.4: Image forgery detection datasets.

Dataset	Forgery	Auth./Forg.	Size	Format	Avail.	Year	Ref.
Columbia gray	Splicing	933-912	128 × 128	BMP	online	2004	Ng and Chang, 2004
Columbia color	Splicing	182 / 180	757×568 – 1152×768	BMP, TIF	online	2006	Hsu and Chang, 2006
MICC F2000	Copy-move	110 / 110	2048 × 1536	JPEG	online	2011	Amerini *et al.*, 2011
Erlangen	Copy-move	—	3000 × 2300	JPEG	online	2013	Christlein *et al.*, 2012
CASIA 1	Splicing, copy-move	800 / 921	374 × 256	JPEG	—	2013	Dong *et al.*, 2013
CASIA 2	Splicing, copy-move	7,200 / 5,123	320×240 – 800×600	JPEG, BMP, TIF	-	2013	Dong *et al.*, 2013
CoMoFoD	Copy-move	— / 260	512 × 512 – 3000 × 2000	JPEG	-	2013	Tralic *et al.*, 2013
COVERAGE	Copy-move	100 / 100	400 × 486	TIF	online	2016	Wen *et al.*, 2016
NIST NC2016	Splicing, copy-move, removal	560 / 564	500 × 500 – 5,616 × 3,744	JPEG	upon request	2016	Guan *et al.*, 2019
NIST NC2017	Various	2667 / 1410	160 × 120 – 8000 × 5320	RAW, PNG, BMP, JPEG	upon request	2017	Guan *et al.*, 2019
FaceSwap	Face swapping	1,758 / 1,927	450×338-7360×4912	JPEG	-	2017	Zhou *et al.*, 2018b
NIST NC2018	Various	14,156 / 3,265	128 × 104 – 7952 × 5304	RAW, PNG, BMP, JPEG	upon request	2017	Guan *et al.*, 2019
PS Battles	Various	11,142 / 102,028	130 × 60 – 10,000 × 8558	PNG, JPEG	online	2018	Heller *et al.*, 2018
Synthetic Dataset	Splicing, copy-move	— / 170,000	—	PNG, BMP, JPEG, TIF	online	2019	Bappy *et al.*, 2019
NIST NC2019	Various	10,279 / 5,750	128 × 104 – 7952 × 5304	PNG	upon request	2019	Guan *et al.*, 2019

Table B.5: Source identification datasets.

Dataset	Cam.	Dev.	Size	Format	Social	Avail.	Year	Ref.
Dresden	25	73	14,000	JPEG	None	online	2010	Gloe and Böhme, 2010
RAISE	4	—	8,156	RAW	None	online	2015	Dang-Nguyen *et al.*, 2015
Image Ballistic and Social Networks	—	—	2,720	JPEG	Facebook, Google+, Twitter, Flickr, Instagram, Tumblr, Imgur, Tinypic, Whatsapp, Telegram	online	2016	Giudice *et al.*, 2016
UCID	—	—	30,000	JPEG	Flickr, Facebook, Twitter	online	2017	Caldelli *et al.*, 2017
PUBLIC	—	—	3,000	JPEG	Flickr, Facebook, Twitter	online	2017	Caldelli *et al.*, 2017
VISION	35	35	34,427 images / 1,914 videos	JPEG, mp4	Facebook, Youtube, Whatsapp	online	2017	Shullani *et al.*, 2017
IEEE SPS Camera Model identification	10	20	3,025	JPEG	None	online	2018	Society, 2018, Bestagini, 2018

Table B.6: Deepfakes datasets.

Dataset	Forgery	Size	Format	Availab.	Year	Ref.
DeepfakeTIMIT	Deepfake	— / 620	JPEG	upon request	2018	Korshunov and Marcel, 2018
Fake video corpus (FVC)	Various	2,458 / 3,957	—	online	2018	Papadopoulou *et al.*, 2018
Fake Faces in the Wild (FFW)	Deepfake, splicing, CGI	— / 150	H264, Youtube	online	2018	Khodabakhsh *et al.*, 2018
FaceForensics++	Deepfake, CGI	1,000 / 4,000	H264 crf 0/23/40	online	2019	Rössler *et al.*, 2019
DeepFake Detection Dataset	Deepfake	363 / 3,068	H264 crf 0/23/40	online	2019	Nick Dufour, 2019
Celeb-DF	Deepfake	590 / 5,639	M4PEG	online	2019	Li *et al.*, 2020
Deepfake Detection Challenge (DFDC)	Deepfake	19,154 / 100,000	H264	online	2019	AWS, Facebook, Microsoft, Partnership on AI's Media Integrity Steering Committee, 2020
DeeperForensics-1.0	Deepfake	50,000 / 10,000	—	online	2020	Jiang *et al.*, 2020

References

Afchar, D., V. Nozick, J. Yamagishi, and I. Echizen. (2018). "MesoNet: a Compact Facial Video Forgery Detection Network". *CoRR*. abs/1809.00888. arXiv: 1809.00888. URL: http://arxiv.org/abs/1809.00888.

Agarwal, S. and L. R. Varshney. (2019). "Limits of Deepfake Detection: A Robust Estimation Viewpoint". *CoRR*. abs/1905.03493. arXiv: 1905.03493. URL: http://arxiv.org/abs/1905.03493.

Agarwal, S., H. Farid, Y. Gu, M. He, K. Nagano, and H. Li. (2019). "Protecting World Leaders Against Deep Fakes". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Almohamedh, H., F. Qurashi, and I. Kostanic. (2015). "Mobile Video Quality Prediction (MVQP) for Long Term Evolution (LTE)". *IAENG International Journal of Computer Science*. 42(Feb.): 46–53.

Altman, N. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". English (US). *American Statistician*. 46(3): 175–185. DOI: 10.1080/00031305.1992.10475879.

Amerini, I., L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra. (2011). "A SIFT-Based Forensic Method for Copy–Move Attack Detection and Transformation Recovery". *IEEE Transactions on Information Forensics and Security*. 6(3): 1099–1110. DOI: 10.1109/TIFS.2011.2129512.

Amerini, I., C. Li, and R. Caldelli. (2019a). "Social Network Identification Through Image Classification With CNN". *IEEE Access*. 7: 35264–35273. DOI: 10.1109/ACCESS.2019.2903876.

Amerini, I., A. Anagnostopoulos, L. Maiano, and L. Ricciardi Celsi. (2021). "Learning Double-Compression Video Fingerprints Left from Social-Media Platforms".

Amerini, I., R. Caldelli, A. D. Mastio, A. D. Fuccia, C. Molinari, and A. P. Rizzo. (2017a). "Dealing with video source identification in social networks". *Signal Processing: Image Communication*. 57: 1–7. DOI: https://doi.org/10.1016/j.image.2017.04.009.

Amerini, I., L. Galteri, R. Caldelli, and A. Del Bimbo. (2019b). "Deepfake Video Detection through Optical Flow Based CNN". In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*.

Amerini, I., T. Uricchio, L. Ballan, and R. Caldelli. (2017b). "Localization of JPEG double compression through multi-domain convolutional neural networks". *CoRR*. abs/1706.01788. arXiv: 1706.01788. URL: http://arxiv.org/abs/1706.01788.

Amerini, I., T. Uricchio, and R. Caldelli. (2017c). "Tracing images back to their social network of origin: A CNN-based approach". *2017 IEEE Workshop on Information Forensics and Security (WIFS)*: 1–6.

AWS, Facebook, Microsoft, Partnership on AI's Media Integrity Steering Committee. (2020). "Deepfake Detection Challenge: Identify videos with facial or voice manipulations". URL: https://www.kaggle.com/c/deepfake-detection-challenge/.

Bakas, J. and R. Naskar. (2018). "A Digital Forensic Technique for Inter-Frame Video Forgery Detection Based on 3D CNN". DOI: 10.1007/978-3-030-05171-6-16.

Baldini, G. and I. Amerini. (2019). "Smartphones Identification Through the Built-In Microphones With Convolutional Neural Network". *IEEE Access*. 7: 158685–158696. DOI: 10.1109/ACCESS.2019.2950859.

Baldini, G., G. Steri, I. Amerini, and R. Caldelli. (2017). "The identification of mobile phones through the fingerprints of their built-in magnetometer: An analysis of the portability of the fingerprints". In: *2017 International Carnahan Conference on Security Technology (ICCST)*. 1–6. DOI: 10.1109/CCST.2017.8167855.

Bappy, J. H., A. K. Roy-Chowdhury, J. Bunk, L. Nataraj, and B. S. Manjunath. (2017). "Exploiting Spatial Structure for Localizing Manipulated Image Regions". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 4980–4989. DOI: 10.1109/ICCV.2017.532.

Bappy, J. H., C. Simons, L. Nataraj, B. S. Manjunath, and A. K. Roy-Chowdhury. (2019). "Hybrid LSTM and Encoder–Decoder Architecture for Detection of Image Forgeries". *IEEE Transactions on Image Processing*. 28(7): 3286–3300. DOI: 10.1109/TIP.2019.2895466.

Barni, M., Q.-T. Phan, and B. Tondi. (2019). "Copy Move Source-Target Disambiguation through Multi-Branch CNNs".

Barni, M., M. C. Stamm, and B. Tondi. (2018). "Adversarial Multimedia Forensics: Overview and Challenges Ahead". In: *2018 26th European Signal Processing Conference (EUSIPCO)*. 962–966. DOI: 10.23919/EUSIPCO.2018.8553305.

Bas, P., T. Filler, and T. Pevný. (2011). ""Break Our Steganographic System": The Ins and Outs of Organizing BOSS". In: *Information Hiding*. Ed. by T. Filler, T. Pevný, S. Craver, and A. Ker. Berlin, Heidelberg: Springer Berlin Heidelberg. 59–70.

Bayar, B. and M. C. Stamm. (2017a). "Augmented convolutional feature maps for robust CNN-based camera model identification". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 4098–4102.

Bayar, B. and M. C. Stamm. (2017b). "On the robustness of constrained convolutional neural networks to JPEG post-compression for image resampling detection". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2152–2156.

Bayar, B. and M. C. Stamm. (2018). "Constrained Convolutional Neural Networks: A New Approach Towards General Purpose Image Manipulation Detection". *IEEE Transactions on Information Forensics and Security*. 13(11): 2691–2706.

Bayar, Belhassen, Stamm, and M. C. (2017). "Design Principles of Convolutional Neural Networks for Multimedia Forensics". DOI: https://doi.org/10.2352/ISSN.2470-1173.2017.7.MWSF-328.

Bayar, B. and M. C. Stamm. (2016). "A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer". In: *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security. IH&MMSec '16*. Vigo, Galicia, Spain: Association for Computing Machinery. 5–10. DOI: 10.1145/2909827.2930786.

Bazarevsky, V., Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann. (2019). "BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs". arXiv: 1907.05047 [cs.CV].

BBC-News. (2020). "Instagram will overtake Twitter as a news source". URL: https://www.bbc.com/news/technology-53050959.

Bestagini, M. C. S. P. (2018). "IEEE Signal Processing Cup 2018 Database - Forensic Camera Model Identification". DOI: 10.21227/H2XM2P.

Bi, X., Y. Wei, B. Xiao, and W. Li. (2019). "RRU-Net: The Ringed Residual U-Net for Image Splicing Forgery Detection". In: *CVPR Workshops*.

Bianchi, T. and A. Piva. (2012). "Image Forgery Localization via Block-Grained Analysis of JPEG Artifacts". *IEEE Transactions on Information Forensics and Security*. 7(3): 1003–1017.

Bill Posters UK, Instagram page. (2019). URL: https://www.instagram.com / p / BypkGIvFfGZ / ?utm _ source = ig _ embed & utm _ campaign=embed_video_watch_again.

Bondi, L., L. Baroffio, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro. (2017a). "First Steps Toward Camera Model Identification With Convolutional Neural Networks". *IEEE Signal Processing Letters*. 24(3): 259–263. DOI: 10.1109/LSP.2016.2641006.

Bondi, L., E. D. Cannas, P. Bestagini, and S. Tubaro. (2020). "Training Strategies and Data Augmentations in CNN-based DeepFake Video Detection". arXiv: 2011.07792 [cs.CV].

Bondi, L., D. Güera, L. Baroffio, P. Bestagini, E. Delp, and S. Tubaro. (2017b). "A Preliminary Study on Convolutional Neural Networks for Camera Model Identification". DOI: 10.2352/ISSN.2470-1173.2017.7.MWSF-327.

Bonettini, N., E. D. Cannas, S. Mandelli, L. Bondi, P. Bestagini, and S. Tubaro. (2020). "Video Face Manipulation Detection Through Ensemble of CNNs". arXiv: 2004.07676 [cs.CV].

Bromley, J., J. Bentz, L. Bottou, I. Guyon, Y. Lecun, C. Moore, E. Sackinger, and R. Shah. (1993). "Signature Verification using a "Siamese" Time Delay Neural Network". *International Journal of Pattern Recognition and Artificial Intelligence*. 7(Aug.): 25. DOI: 10.1142/S0218001493000339.

Burt, P. and E. Adelson. (1983). "The Laplacian Pyramid as a Compact Image Code". *IEEE Transactions on Communications*. 31(4): 532–540.

Caldelli, R., I. Amerini, and C. T. Li. (2018a). "PRNU-based Image Classification of Origin Social Network with CNN". In: *2018 26th European Signal Processing Conference (EUSIPCO)*. 1357–1361.

Caldelli, R., R. Becarelli, and I. Amerini. (2017). "Image Origin Classification Based on Social Network Provenance". *IEEE Transactions on Information Forensics and Security*. 12(6): 1299–1308.

Caldelli, R., I. Amerini, and C.-T. Li. (2018b). "PRNU-based Image Classification of Origin Social Network with CNN". DOI: 10.23919/EUSIPCO.2018.8553160.

Caldelli, R., L. Galteri, I. Amerini, and A. Del Bimbo. (2021). "Optical Flow based CNN for detection of unlearnt deepfake manipulations". *Pattern Recognition Letters*. 146: 31–37. DOI: https://doi.org/10.1016/j.patrec.2021.03.005.

Carreira, J. and A. Zisserman. (2017). "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". *CoRR*. abs/1705.07750. arXiv: 1705.07750. URL: http://arxiv.org/abs/1705.07750.

Carvalho, T. J. d., C. Riess, E. Angelopoulou, H. Pedrini, and A. d. R. Rocha. (2013). "Exposing Digital Image Forgeries by Illumination Color Classification". *IEEE Transactions on Information Forensics and Security*. 8(7): 1182–1194.

Chen, M., J. Fridrich, M. Goljan, and J. Lukas. (2008). "Determining Image Origin and Integrity Using Sensor Noise". *IEEE Transactions on Information Forensics and Security*. 3(1): 74–90.

Chen, M., J. Fridrich, M. Goljan, and J. Lukás. (2007). "Source digital camcorder identification using sensor photo response non-uniformity - art. no. 65051G". *SPIE*. 6505(Feb.). DOI: 10.1117/12.696519.

Chen, S., S. Tan, B. Li, and J. Huang. (2016). "Automatic Detection of Object-Based Forgery in Advanced Video". *IEEE Transactions on Circuits and Systems for Video Technology*. 26: 2138–2151.

Cho, K., B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". *CoRR*. abs/1406.1078. arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

Choi, Y., M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. (2017). "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation". arXiv: 1711.09020 [cs.CV].

Chollet, F. (2016). "Xception: Deep Learning with Depthwise Separable Convolutions". *CoRR*. abs/1610.02357. arXiv: 1610.02357. URL: http://arxiv.org/abs/1610.02357.

Christlein, V., C. Riess, J. Jordan, C. Riess, and E. Angelopoulou. (2012). "An Evaluation of Popular Copy-Move Forgery Detection Approaches". *CoRR*. abs/1208.3665. arXiv: 1208.3665. URL: http://arxiv.org/abs/1208.3665.

Ciftci, U. A. and I. Demir. (2019). "FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals". *CoRR*. abs/1901.02212. arXiv: 1901.02212. URL: http://arxiv.org/abs/1901.02212.

Cortes, C. and V. Vapnik. (1995). "Support-Vector Networks." *Mach. Learn.* 20(3): 273–297. URL: http://dblp.uni-trier.de/db/journals/ml/ml20.html#CortesV95.

Cozzolino, D., D. Gragnaniello, and L. Verdoliva. (2014a). "Image forgery detection through residual-based local descriptors and block-matching". In: *2014 IEEE International Conference on Image Processing (ICIP)*. 5297–5301.

Cozzolino, D., D. Gragnaniello, and L. Verdoliva. (2014b). "Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques". In: *2014 IEEE International Conference on Image Processing (ICIP)*. 5302–5306.

Cozzolino, D., G. Poggi, and L. Verdoliva. (2015a). "Splicebuster: A new blind image splicing detector". In: *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*. 1–6. DOI: 10.1109/WIFS.2015.7368565.

Cozzolino, D. and L. Verdoliva. (2016). "Single-image splicing localization through autoencoder-based anomaly detection". In: *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*. 1–6.

Cozzolino, D. and L. Verdoliva. (2018a). "Camera-based Image Forgery Localization using Convolutional Neural Networks". In: *2018 26th European Signal Processing Conference (EUSIPCO)*. 1372–1376.

Cozzolino, D. and L. Verdoliva. (2020). "Noiseprint: A CNN-Based Camera Model Fingerprint". *IEEE Transactions on Information Forensics and Security*. 15: 144–159. DOI: 10.1109/TIFS.2019.2916364.

Cozzolino, D., F. Marra, D. Gragnaniello, G. Poggi, and L. Verdoliva. (2020). "Combining PRNU and noiseprint for robust and efficient device source identification". *EURASIP Journal on Information Security*. Jan. DOI: 10.1186/s13635-020-0101-7.

Cozzolino, D., G. Poggi, and L. Verdoliva. (2015b). "Copy-move forgery detection based on PatchMatch". *2014 IEEE International Conference on Image Processing, ICIP 2014*. Jan.: 5312–5316. DOI: 10.1109/ICIP.2014.7026075.

Cozzolino, D., G. Poggi, and L. Verdoliva. (2017). "Recasting Residual-based Local Descriptors as Convolutional Neural Networks: an Application to Image Forgery Detection". arXiv: 1703.04615 [cs.CV].

Cozzolino, D., J. Thies, A. Rössler, C. Riess, M. Nießner, and L. Verdoliva. (2018). "ForensicTransfer: Weakly-supervised Domain Adaptation for Forgery Detection". *CoRR*. abs/1812.02510. arXiv: 1812.02510. URL: http://arxiv.org/abs/1812.02510.

Cozzolino, D. and L. Verdoliva. (2018b). "Camera-based Image Forgery Localization using Convolutional Neural Networks". *CoRR*. abs/1808.09714. arXiv: 1808.09714. URL: http://arxiv.org/abs/1808.09714.

Cozzolino Giovanni Poggi Luisa Verdoliva, D. (2019). "Extracting camera-based fingerprints for video forensics". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

D'Avino, D., D. Cozzolino, G. Poggi, and L. Verdoliva. (2017). "Autoencoder with recurrent neural networks for video forgery detection". *Electronic Imaging*. 2017(Jan.): 92–99. DOI: 10.2352/ISSN.2470-1173.2017.7.MWSF-330.

Dang, H., F. Liu, J. Stehouwer, X. Liu, and A. Jain. (2019). "On the Detection of Digital Face Manipulation". arXiv: 1910.01717 [cs.CV].

Dang-Nguyen, D.-T., C. Pasquini, V. Conotter, and G. Boato. (2015). "RAISE: A Raw Images Dataset for Digital Image Forensics". In: *Proceedings of the 6th ACM Multimedia Systems Conference. MMSys '15*. Portland, Oregon: Association for Computing Machinery. 219–224. DOI: 10.1145/2713168.2713194.

Davis, J. and M. Goadrich. (2006). "The Relationship between Precision-Recall and ROC Curves". In: *Proceedings of the 23rd International Conference on Machine Learning. ICML '06*. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery. 233–240. DOI: 10.1145/1143844.1143874.

de Rezende, E. R. S., G. C. S. Ruppert, and T. Carvalho. (2017). "Detecting Computer Generated Images with Deep Convolutional Neural Networks". In: *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. 71–78.

"Deepfakes Faceswap". (2018). URL: https://github.com/deepfakes/faceswap/.

Ding, X., Y. Chen, Z. Tang, and Y. Huang. (2019). "Camera Identification Based on Domain Knowledge-Driven Deep Multi-Task Learning". *IEEE Access*. 7: 25878–25890.

Do Nhu, T., I. Na, and S. Kim. (2018). "Forensics Face Detection From GANs Using Convolutional Neural Network".

Dolhansky, B., J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, and C. C. Ferrer. (2020). "The DeepFake Detection Challenge (DFDC) Dataset". arXiv: 2006.07397 [cs.CV].

Donahue, J., L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. (2016). "Long-term Recurrent Convolutional Networks for Visual Recognition and Description". arXiv: 1411.4389 [cs.CV].

Dong, J., W. Wang, and T. Tan. (2013). "CASIA Image Tampering Detection Evaluation Database". In: *2013 IEEE China Summit and International Conference on Signal and Information Processing*. 422–426. DOI: 10.1109/ChinaSIP.2013.6625374.

Du, M., S. Pentyala, Y. Li, and X. Hu. (2019). "Towards Generalizable Forgery Detection with Locality-aware AutoEncoder". arXiv: 1909.05999 [cs.CV].

Ekman, P. and W. V. Friesen. (1976). "Measuring facial movement". *Environmental psychology and nonverbal behavior*. 1(1): 56–75. DOI: 10.1007/BF01115465.

Everingham, M., L. Gool, C. K. Williams, J. Winn, and A. Zisserman. (2010). "The Pascal Visual Object Classes (VOC) Challenge". *Int. J. Comput. Vision*. 88(2): 303–338. DOI: 10.1007/s11263-009-0275-4.

"Fake App". (2018). URL: https://www.fakeapp.com/.

Fernandes, S., S. Raj, E. Ortiz, I. Vintila, M. Salter, G. Urosevic, and S. Jha. (2019). "Predicting Heart Rate Variations of Deepfake Videos using Neural ODE". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 1721–1729.

Fernando, T., C. Fookes, S. Denman, and S. Sridharan. (2019). "Exploiting Human Social Cognition for the Detection of Fake and Fraudulent Faces via Memory Networks". arXiv: 1911.07844 [cs.CV].

Ferreira, A., H. Chen, B. Li, and J. Huang. (2018). "An Inception-Based Data-Driven Ensemble Approach to Camera Model Identification". In: *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. 1–7.

Freire-Obregón, D., F. Narducci, S. Barra, and M. Castrillón-Santana. (2019). "Deep learning for source camera identification on mobile devices". *Pattern Recognition Letters*. 126(Sept.): 86–91. DOI: 10.1016/j.patrec.2018.01.005.

Fridrich, J. and J. Kodovsky. (2012). "Rich Models for Steganalysis of Digital Images". *IEEE Transactions on Information Forensics and Security*. 7(3): 868–882.

Fridrich, J. J. (2009). "Digital Image Forensics Using Sensor Noise".

Fuji Tsang, C. and J. Fridrich. (2018). "Steganalyzing Images of Arbitrary Size with CNNs". In: vol. 2018. DOI: 10.2352/ISSN.2470-1173.2018.07.MWSF-121.

Galdi, C., M. Nappi, J.-L. Dugelay, and Y. Yu. (2018). "Exploring New Authentication Protocols for Sensitive Data Protection on Smartphones". *IEEE Communications Magazine*. 56(Jan.): 136–142. DOI: 10.1109/MCOM.2017.1700342.

Gatys, L. A., A. S. Ecker, and M. Bethge. (2015). "A Neural Algorithm of Artistic Style". *CoRR*. abs/1508.06576. arXiv: 1508.06576. URL: http://arxiv.org/abs/1508.06576.

Giudice, O., A. Paratore, M. Moltisanti, and S. Battiato. (2016). "A Classification Engine for Image Ballistics of Social Data". *CoRR*. abs/1610.06347. arXiv: 1610.06347. URL: http://arxiv.org/abs/1610.06347.

Gloe, T. and R. Böhme. (2010). "The Dresden Image Database for Benchmarking Digital Image Forensics". *Journal of Digital Forensic Practice*. 3(2-4): 150–159. DOI: 10.1080/15567281.2010.531500. eprint: https://doi.org/10.1080/15567281.2010.531500.

Goljan, M., J. Fridrich, and T. Filler. (2009). "Large scale test of sensor fingerprint camera identification". In: *Media Forensics and Security*. Ed. by E. J. D. III, J. Dittmann, N. D. Memon, and P. W. Wong. Vol. 7254. International Society for Optics and Photonics. SPIE. 170–181. DOI: 10.1117/12.805701.

Goodfellow, I., Y. Bengio, and A. Courville. (2016). *Deep Learning*. MIT Press.

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. (2014). "Generative Adversarial Networks". arXiv: 1406.2661 [stat.ML].

Goodfellow, I. J., J. Shlens, and C. Szegedy. (2015). "Explaining and Harnessing Adversarial Examples". arXiv: 1412.6572 [stat.ML].

Guan, H., M. Kozak, E. Robertson, Y. Lee, A. N. Yates, A. Delgado, D. Zhou, T. Kheyrkhah, J. Smith, and J. Fiscus. (2019). "MFC Datasets: Large-Scale Benchmark Datasets for Media Forensic Challenge Evaluation". In: *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*. 63–72. DOI: 10.1109/WACVW.2019.00018.

Guera, D. and E. J. Delp. (2018). "Deepfake Video Detection Using Recurrent Neural Networks". *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*: 1–6.

Hadwiger, B. and C. Riess. (2020). "The Forchheim Image Database for Camera Identification in the Wild". arXiv: 2011.02241 [cs.CV].

Hao, K. (2020). "Deepfake Putin is here to warn Americans about their self-inflicted doom". URL: https://www.technologyreview.com/2020/09/29/1009098/ai-deepfake-putin-kim-jong-un-us-election/.

He, K., X. Zhang, S. Ren, and J. Sun. (2014). "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". *CoRR*. abs/1406.4729. arXiv: 1406.4729. URL: http://arxiv.org/abs/1406.4729.

He, K., X. Zhang, S. Ren, and J. Sun. (2015). "Deep Residual Learning for Image Recognition". *CoRR*. abs/1512.03385. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

He, P., X. Jiang, T. Sun, and H. Li. (2018). "Computer Graphics Identification Combining Convolutional and Recurrent Neural Networks". *IEEE Signal Processing Letters*. 25(9): 1369–1373.

He, P., H. Li, and H. Wang. (2019). "Detection of Fake Images Via The Ensemble of Deep Representations from Multi Color Spaces". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2299–2303.

He, P., X. Jiang, T. Sun, S. Wang, B. Li, and Y. Dong. (2017). "Frame-wise detection of relocated I-frames in double compressed H.264 videos based on convolutional neural network". *Journal of Visual Communication and Image Representation*. 48: 149–158. DOI: https://doi.org/10.1016/j.jvcir.2017.06.010.

Heller, S., L. Rossetto, and H. Schuldt. (2018). "The PS-Battles Dataset - an Image Collection for Image Manipulation Detection". arXiv: 1804.04866 [cs.MM].

Hochreiter, S., Y. Bengio, P. Frasconi, and J. Schmidhuber. (2001). "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies". *A Field Guide to Dynamical Recurrent Neural Networks*.

Hochreiter, S. and J. Schmidhuber. (1997). "Long Short-term Memory". *Neural computation*. 9(Dec.): 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Hsu, Y.-F. and S.-F. Chang. (2006). "Detecting Image Splicing Using Geometry Invariants and Camera Characteristics Consistency". In: *International Conference on Multimedia and Expo*. Toronto, Canada.

Hu, J., L. Shen, and G. Sun. (2017). "Squeeze-and-Excitation Networks". *CoRR*. abs/1709.01507. arXiv: 1709.01507. URL: http://arxiv.org/abs/1709.01507.

Huang, G., Z. Liu, and K. Q. Weinberger. (2016). "Densely Connected Convolutional Networks". *CoRR*. abs/1608.06993. arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993.

Huh, M., A. Liu, A. Owens, and A. A. Efros. (2018). "Fighting Fake News: Image Splice Detection via Learned Self-Consistency". *CoRR*. abs/1805.04096. arXiv: 1805.04096. URL: http://arxiv.org/abs/1805.04096.

Jaderberg, M., K. Simonyan, A. Zisserman, and K. Kavukcuoglu. (2015). "Spatial Transformer Networks". *CoRR*. abs/1506.02025. arXiv: 1506.02025. URL: http://arxiv.org/abs/1506.02025.

Jiang, L., R. Li, W. Wu, C. Qian, and C. C. Loy. (2020). "DeeperForensics-1.0: A Large-Scale Dataset for Real-World Face Forgery Detection". In: *CVPR*.

Kalkowski, S., C. Schulze, A. Dengel, and D. Borth. (2015). "Real-Time Analysis and Visualization of the YFCC100m Dataset". In: *Proceedings of the 2015 Workshop on Community-Organized Multimodal Mining: Opportunities for Novel Solutions. MMCommons '15*. Brisbane, Australia: Association for Computing Machinery. 25–30. DOI: 10.1145/2814815.2814820.

Kang, X., M. C. Stamm, A. Peng, and K. J. R. Liu. (2013). "Robust Median Filtering Forensics Using an Autoregressive Model". *IEEE Transactions on Information Forensics and Security*. 8(9): 1456–1468.

Karras, T., T. Aila, S. Laine, and J. Lehtinen. (2017). "Progressive Growing of GANs for Improved Quality, Stability, and Variation". *CoRR*. abs/1710.10196. arXiv: 1710.10196. URL: http://arxiv.org/abs/1710.10196.

Karras, T., S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. (2019). "Analyzing and Improving the Image Quality of StyleGAN". arXiv: 1912.04958 [cs.CV].

Khodabakhsh, A., R. Ramachandra, K. Raja, P. Wasnik, and C. Busch. (2018). "Fake Face Detection Methods: Can They Be Generalized?" In: *2018 International Conference of the Biometrics Special Interest Group (BIOSIG)*. 1–6.

Kim, D., H.-U. Jang, S.-M. Mun, S. Choi, and H.-K. Lee. (2018). "Median Filtered Image Restoration and Anti-Forensics Using Adversarial Networks". *IEEE Signal Processing Letters*. 25(2): 278–282. DOI: 10.1109/LSP.2017.2782363.

Korshunov, P. and S. Marcel. (2019). "Vulnerability assessment and detection of Deepfake videos". In: *2019 International Conference on Biometrics (ICB)*. 1–6.

Korshunov, P. and S. Marcel. (2018). "DeepFakes: a New Threat to Face Recognition? Assessment and Detection". *CoRR*. abs/1812.08685. arXiv: 1812.08685. URL: http://arxiv.org/abs/1812.08685.

Korus, P. and J. Huang. (2016). "Evaluation of random field models in multi-modal unsupervised tampering localization". In: *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*. 1–6.

Korus, P. and J. Huang. (2017). "Multi-Scale Analysis Strategies in PRNU-Based Tampering Localization". *IEEE Transactions on Information Forensics and Security*. 12(4): 809–824.

Krizhevsky, A., I. Sutskever, and G. Hinton. (2012). "ImageNet Classification with Deep Convolutional Neural Networks". *Neural Information Processing Systems*. 25(Jan.). DOI: 10.1145/3065386.

Kuzin, A., A. Fattakhov, I. Kibardin, V. I. Iglovikov, and R. Dautov. (2018). "Camera Model Identification Using Convolutional Neural Networks". In: *2018 IEEE International Conference on Big Data (Big Data)*. 3107–3110.

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner. (1998). "Gradient-based learning applied to document recognition". *Proceedings of the IEEE*. 86(11): 2278–2324. DOI: 10.1109/5.726791.

Li, C. (2010). "Source Camera Identification Using Enhanced Sensor Pattern Noise". *IEEE Transactions on Information Forensics and Security*. 5(2): 280–287.

Li, C. and Y. Li. (2012). "Color-Decoupled Photo Response Non-Uniformity for Digital Image Forensics". *IEEE Transactions on Circuits and Systems for Video Technology*. 22(2): 260–271.

Li, J., T. Shen, W. Zhang, H. Ren, D. Zeng, and T. Mei. (2019). "Zooming into Face Forensics: A Pixel-level Analysis".

Li, Y., M. Chang, and S. Lyu. (2018). "In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking". In: *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. 1–7. DOI: 10.1109/WIFS.2018.8630787.

Li, Y. and S. Lyu. (2018). "Exposing DeepFake Videos By Detecting Face Warping Artifacts". *CoRR*. abs/1811.00656. arXiv: 1811.00656. URL: http://arxiv.org/abs/1811.00656.

Li, Y., P. Sun, H. Qi, and S. Lyu. (2020). "Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics". In: *IEEE Conference on Computer Vision and Patten Recognition (CVPR)*. Seattle, WA, United States.

Lin, J. Y., R. Song, C.-H. Wu, T. Liu, H. Wang, and C.-C. J. Kuo. (2015). "MCL-V: A streaming video quality assessment database". *Journal of Visual Communication and Image Representation*. 30: 1–9. DOI: https://doi.org/10.1016/j.jvcir.2015.02.012.

Liu, Z., P. Luo, X. Wang, and X. Tang. (2014). "Deep Learning Face Attributes in the Wild". arXiv: 1411.7766 [cs.CV].

Long, C., A. Basharat, and A. Hoogs. (2018). "A Coarse-to-fine Deep Convolutional Neural Network Framework for Frame Duplication Detection and Localization in Video Forgery". *CoRR*. abs/1811.10762. arXiv: 1811.10762. URL: http://arxiv.org/abs/1811.10762.

Long, C., E. Smith, A. Basharat, and A. Hoogs. (2017). "A C3D-Based Convolutional Neural Network for Frame Dropping Detection in a Single Video Shot". DOI: 10.1109/CVPRW.2017.237.

Long, J., E. Shelhamer, and T. Darrell. (2014). "Fully Convolutional Networks for Semantic Segmentation". *CoRR*. abs/1411.4038. arXiv: 1411.4038. URL: http://arxiv.org/abs/1411.4038.

Lukas, J., J. Fridrich, and M. Goljan. (2006). "Digital camera identification from sensor pattern noise". *IEEE Transactions on Information Forensics and Security*. 1(2): 205–214.

Mandelli, S., N. Bonettini, P. Bestagini, and S. Tubaro. (2020a). "Training CNNs in Presence of JPEG Compression: Multimedia Forensics vs Computer Vision". arXiv: 2009.12088 [cs.CV].

Mandelli, S., D. Cozzolino, P. Bestagini, L. Verdoliva, and S. Tubaro. (2020b). "CNN-based fast source device identification". arXiv: 2001.11847 [cs.NE].

Maras, M.-H. and A. Alexandrou. (2018). "Determining Authenticity of Video Evidence in the Age of Artificial Intelligence and in the Wake of Deepfake Videos". *International Journal of Evidence and Proof*. 23(Oct.). DOI: 10.1177/1365712718807226.

Marra, F., D. Gragnaniello, D. Cozzolino, and L. Verdoliva. (2018a). "Detection of GAN-Generated Fake Images over Social Networks". In: *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. 384–389. DOI: 10.1109/MIPR.2018.00084.

Marra, F., C. Saltori, G. Boato, and L. Verdoliva. (2019a). "Incremental learning for the detection and classification of GAN-generated images". In: *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*. 1–6.

Marra, F., D. Gragnaniello, L. Verdoliva, and G. Poggi. (2018b). "Do GANs leave artificial fingerprints?" arXiv: 1812.11842 [cs.CV].

Marra, F., D. Gragnaniello, L. Verdoliva, and G. Poggi. (2019b). "A Full-Image Full-Resolution End-to-End-Trainable CNN Framework for Image Forgery Detection". arXiv: 1909.06751 [cs.CV].

Matern, F., C. Riess, and M. Stamminger. (2019). "Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations". In: *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*. 83–92. DOI: 10.1109/WACVW.2019.00020.

Mayer, O., B. Hosler, and M. C. Stamm. (2020). "Open Set Video Camera Model Verification". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2962–2966. DOI: 10.1109/ICASSP40776.2020.9054261.

Mayer, O. and M. C. Stamm. (2018). "Learned Forensic Source Similarity for Unknown Camera Models". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2012–2016.

Mayer, O. and M. C. Stamm. (2019). "Exposing Fake Images with Forensic Similarity Graphs". arXiv: 1912.02861 [eess.IV].

Mayer, O. and M. C. Stamm. (2020). "Forensic Similarity for Digital Images". *IEEE Transactions on Information Forensics and Security*. 15: 1331–1346. DOI: 10.1109/tifs.2019.2924552.

Mazaheri, G., N. Chowdhury Mithun, J. H. Bappy, and A. K. Roy-Chowdhury. (2019). "A Skip Connection Architecture for Localization of Image Manipulations". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Montgomery, C. *et al.* (1994). "Xiph. org video test media (derf's collection), the xiph open source community". URL: https://media.xiph.org/video/derf/.

Nam, S., J. Park, D. Kim, I. Yu, T. Kim, and H. Lee. (2019). "Two-Stream Network for Detecting Double Compression of H.264 Videos". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 111–115.

Nataraj, L., T. M. Mohammed, B. S. Manjunath, S. Chandrasekaran, A. Flenner, J. H. Bappy, and A. K. Roy-Chowdhury. (2019). "Detecting GAN generated Fake Images using Co-occurrence Matrices". *CoRR*. abs/1903.06836. arXiv: 1903.06836. URL: http://arxiv.org/abs/1903.06836.

Ng, T.-T. and S. Chang. (2004). "A data set of authentic and spliced image blocks".

Nguyen, H. H., F. Fang, J. Yamagishi, and I. Echizen. (2019a). "Multi-task Learning For Detecting and Segmenting Manipulated Facial Images and Videos". *CoRR*. abs/1906.06876. arXiv: 1906.06876. URL: http://arxiv.org/abs/1906.06876.

Nguyen, T. T., C. M. Nguyen, D. T. Nguyen, D. T. Nguyen, and S. Nahavandi. (2019b). "Deep Learning for Deepfakes Creation and Detection". arXiv: 1909.11573 [cs.CV].

Nick Dufour, A. G. (2019). "Contributing Data to Deepfake Detection Research". URL: https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html.

Niu, Y., B. Tondi, Y. Zhao, R. Ni, and M. Barni. (2021). "Image Splicing Detection, Localization and Attribution via JPEG Primary Quantization Matrix Estimation and Clustering". arXiv: 2102.01439 [eess.IV].

Noh, H., S. Hong, and B. Han. (2015). "Learning Deconvolution Network for Semantic Segmentation". In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). ICCV '15*. USA: IEEE Computer Society. 1520–1528. DOI: 10.1109/ICCV.2015.178.

Papadopoulou, O., M. Zampoglou, S. Papadopoulos, and I. Kompatsiaris. (2018). "A corpus of debunked and verified user-generated videos". *Online Information Review*. DOI: 10.1108/OIR-03-2018-0101.

Park, J., D. Cho, W. Ahn, and H.-K. Lee. (2018). "Double JPEG Detection in Mixed JPEG Quality Factors using Deep Convolutional Neural Network". In: *The European Conference on Computer Vision (ECCV)*.

Parkhi, O. M., A. Vedaldi, and A. Zisserman. (2015). "Deep Face Recognition". In: *British Machine Vision Conference*.

Pengpeng, Y., R. Ni, and Y. Zhao. (2017). "Recapture Image Forensics Based on Laplacian Convolutional Neural Networks". DOI: 10.1007/978-3-319-53465-7-9.

Pevný, T., P. Bas, and J. Fridrich. (2010). "Steganalysis by Subtractive Pixel Adjacency Matrix". *IEEE Transactions on Information Forensics and Security*. 5(July). DOI: 10.1145/1597817.1597831.

Phan, Q., G. Boato, R. Caldelli, and I. Amerini. (2019). "Tracking Multiple Image Sharing on Social Networks". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 8266–8270. DOI: 10.1109/ICASSP.2019.8683144.

Phan, Q.-T., C. Pasquini, G. Boato, and F. Natale. (2018). "Identifying Image Provenance: An Analysis of Mobile Instant Messaging Apps". DOI: 10.1109/MMSP.2018.8547050.

Powers, D. (2008). "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation".

Qi Cui Suzanne McIntosh, H. S. (2018). "Identifying Materials of Photographic Images and Photorealistic Computer Generated Graphics Based on Deep CNNs". *Computers, Materials & Continua*. 55(2): 229–241. DOI: 10.3970/cmc.2018.01693.

Qian, Y., J. Dong, W. Wang, and T. Tan. (2015). "Deep learning for steganalysis via convolutional neural networks". In: *Electronic Imaging*.

Quan, Y., X. Lin, and C.-T. Li. (2019). "Provenance Analysis for Instagram Photos". In: *Data Mining*. Singapore: Springer Singapore. 372–383.

Radford, A., L. Metz, and S. Chintala. (2015). "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". arXiv: 1511.06434 [cs.LG].

Rafi, A. M., U. Kamal, R. Hoque, A. Abrar, S. Das, R. Laganière, and M. K. Hasan. (2018). "Application of DenseNet in Camera Model Identification and Post-processing Detection". arXiv: 1809.00576 [eess.IV].

Rahmouni, N., V. Nozick, J. Yamagishi, and I. Echizen. (2017). "Distinguishing computer graphics from natural images using convolution neural networks". In: *2017 IEEE Workshop on Information Forensics and Security (WIFS)*. 1–6.

Rebuffi, S., A. Kolesnikov, and C. H. Lampert. (2016). "iCaRL: Incremental Classifier and Representation Learning". *CoRR*. abs/1611.07725. arXiv: 1611.07725. URL: http://arxiv.org/abs/1611.07725.

Ren, S., K. He, R. B. Girshick, and J. Sun. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". *CoRR*. abs/1506.01497. arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

Richard Fletcher Nic Newman, A. S. (2020). "A Mile Wide, an Inch Deep: Online News and Media Use in the 2019 UK General Election". URL: https://reutersinstitute.politics.ox.ac.uk/sites/default/files/2020-02/Fletcher_News_Use_During_the_Election_FINAL.pdf.

Ronneberger, O., P. Fischer, and T. Brox. (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". *CoRR*. abs/1505.04597. arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

Rössler, A., D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. (2018a). "FaceForensics: A Large-scale Video Dataset for Forgery Detection in Human Faces". *CoRR*. abs/1803.09179. arXiv: 1803.09179. URL: http://arxiv.org/abs/1803.09179.

Rössler, A., D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. (2018b). "FaceForensics: A Large-scale Video Dataset for Forgery Detection in Human Faces". *CoRR*. abs/1803.09179. arXiv: 1803.09179. URL: http://arxiv.org/abs/1803.09179.

Rössler, A., D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. (2019). "FaceForensics++: Learning to Detect Manipulated Facial Images". arXiv: 1901.08971 [cs.CV].

Sabir, E., J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan. (2019). "Recurrent Convolutional Strategies for Face Manipulation Detection in Videos". *CoRR*. abs/1905.00582. arXiv: 1905.00582. URL: http://arxiv.org/abs/1905.00582.

Salloum, R., Y. Ren, and C.-C. Jay Kuo. (2018). "Image Splicing Localization using a Multi-task Fully Convolutional Network (MFCN)". *Journal of Visual Communication and Image Representation*. 51(Feb.): 201–209. DOI: 10.1016/j.jvcir.2018.01.010.

Schmid, C. (2001). "Constructing models for content-based image retrieval". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 2. II–II. DOI: 10.1109/CVPR.2001.990922.

Shanker, S. (2021). *The Discrete Cosine Transform in Action*. URL: https://squidarth.com/rc/math/2018/06/24/fourier.html. (accessed: 05.05.2021).

Shullani, D., M. Fontani, M. Iuliani, O. Alshaya, and A. Piva. (2017). "VISION: a video and image dataset for source identification". *EURASIP Journal on Information Security*. 2017(Oct.): 15. DOI: 10.1186/s13635-017-0067-2.

Simonyan, K. and A. Zisserman. (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition". *arXiv 1409.1556*. Sept.

Society, I. S. P. (2014). "IEEE IFS-TC Image Forensics Challenge Dataset". URL: http://ifc.recod.ic.%20unicamp.br/fc.website/index.py.

Society, I. S. P. (2018). "Camera Model Identification". URL: https://www.kaggle.com/c/sp-society-camera-model-identification/.

Soomro, K., A. R. Zamir, and M. Shah. (2012). "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild". arXiv: 1212.0402 [cs.CV].

Springenberg, J. T., A. Dosovitskiy, T. Brox, and M. Riedmiller. (2015). "Striving for Simplicity: The All Convolutional Net". arXiv: 1412.6806 [cs.LG].

Staff, R. (2021). "Fact check: Donald Trump concession video not a confirmed deepfake". URL: https://www.reuters.com/article/uk-factcheck-trump-consession-video-deep-idUSKBN29G2NL.

Stamm, M. C., M. Wu, and K. J. R. Liu. (2013). "Information Forensics: An Overview of the First Decade". *IEEE Access*. 1: 167–200.

Sullivan, G. J., J.-R. Ohm, W.-J. Han, and T. Wiegand. (2012). "Overview of the High Efficiency Video Coding (HEVC) Standard". *IEEE Transactions on Circuits and Systems for Video Technology*. 22(12): 1649–1668. DOI: 10.1109/TCSVT.2012.2221191.

Sun, D., X. Yang, M. Liu, and J. Kautz. (2017). "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume". *CoRR*. abs/1709.02371. arXiv: 1709.02371. URL: http://arxiv.org/abs/1709.02371.

Szegedy, C., S. Ioffe, and V. Vanhoucke. (2016). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". *CoRR*. abs/1602.07261. arXiv: 1602.07261. URL: http://arxiv.org/abs/1602.07261.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. (2014). "Going Deeper with Convolutions". *CoRR*. abs/1409.4842. arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. (2015). "Rethinking the Inception Architecture for Computer Vision". *CoRR*. abs/1512.00567. arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.

Szeliski, R. (2011). *Computer vision algorithms and applications*. URL: http://dx.doi.org/10.1007/978-1-84882-935-0.

Tan, M. and Q. V. Le. (2020). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". arXiv: 1905.11946 [cs.LG].

Tariq, S., S. Lee, H. Kim, Y. Shin, and S. S. Woo. (2018). "Detecting Both Machine and Human Created Fake Face Images In the Wild". In: *Proceedings of the 2nd International Workshop on Multimedia Privacy and Security. MPS '18*. Toronto, Canada: Association for Computing Machinery. 81–87. DOI: 10.1145/3267357.3267367.

Thies, J., M. Zollhöfer, and M. Nießner. (2019). "Deferred Neural Rendering: Image Synthesis using Neural Textures". *CoRR*. abs/1904.12356. arXiv: 1904.12356. URL: http://arxiv.org/abs/1904.12356.

Thies, J., M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. (2020). "Face2Face: Real-time Face Capture and Reenactment of RGB Videos". arXiv: 2007.14808 [cs.CV].

Tokuda, E., H. Pedrini, and A. Rocha. (2013). "Computer generated images vs. digital photographs: A synergetic feature and classifier combination approach". *Journal of Visual Communication and Image Representation*. 24(8): 1276–1292. DOI: https://doi.org/10.1016/j.jvcir.2013.08.009.

Tralic, D., I. Zupancic, S. Grgic, and M. Grgic. (2013). "CoMoFoD — New database for copy-move forgery detection". In: *Proceedings ELMAR-2013*. 49–54.

Tran, D., L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. (2014). "C3D: Generic Features for Video Analysis". *CoRR*. abs/1412.0767. arXiv: 1412.0767. URL: http://arxiv.org/abs/1412.0767.

Trump: Deepfakes Replacement, Youtube video. (2018). URL: https://www.youtube.com/watch?v=hoc2RISoLWU&feature=emb_title.

Tuama, A., F. Comby, and M. Chaumont. (2016). "Camera model identification with the use of deep convolutional neural networks". In: *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*. 1–6.

Verde, S., L. Bondi, P. Bestagini, S. Milani, G. Calvagno, and S. Tubaro. (2018). "Video Codec Forensics Based on Convolutional Neural Networks". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. 530–534.

Verdoliva, L., D. Cozzolino, and G. Poggi. (2014). "A feature-based approach for image tampering detection and localization". In: *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*. 149–154.

Verdoliva, L. (2020). "Media Forensics and DeepFakes: an overview". arXiv: 2001.06564 [cs.CV].

Vetterli, M., J. Kovacevic, and V. Goyal. (2018). "Foundations of Signal Processing". May: xxv–xxviii. DOI: 10.1017/cbo9781139839099.001.

Wang, J., Y. song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. (2014). "Learning Fine-grained Image Similarity with Deep Ranking". arXiv: 1404.4661 [cs.CV].

Wang, Q. and R. Zhang. (2016). "Double JPEG compression forensics based on a convolutional neural network". *EURASIP Journal on Information Security*. 2016(Dec.). DOI: 10.1186/s13635-016-0047-y.

Wang, S., O. Wang, A. Owens, R. Zhang, and A. A. Efros. (2019a). "Detecting Photoshopped Faces by Scripting Photoshop". *CoRR*. abs/1906.05856. arXiv: 1906.05856. URL: http://arxiv.org/abs/1906.05856.

Wang, S.-Y., O. Wang, R. Zhang, A. Owens, and A. A. Efros. (2019b). "CNN-generated images are surprisingly easy to spot... for now". arXiv: 1912.11035 [cs.CV].

Wen, B., Y. Zhu, R. Subramanian, T.-T. Ng, X. Shen, and S. Winkler. (2016). "COVERAGE – A NOVEL DATABASE FOR COPY-MOVE FORGERY DETECTION". In: *IEEE International Conference on Image processing (ICIP)*. 161–165.

Wojna, Z., V. Ferrari, S. Guadarrama, N. Silberman, L. Chen, A. Fathi, and J. R. R. Uijlings. (2017). "The Devil is in the Decoder". *CoRR*. abs/1707.05847. arXiv: 1707.05847. URL: http://arxiv.org/abs/1707.05847.

Wu, Y., W. AbdAlmageed, and P. Natarajan. (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)*. 9535–9544.

Wu, Y., W. Abd-Almageed, and P. Natarajan. (2018). "BusterNet: Detecting Copy-Move Image Forgery with Source/Target Localization". In: *Computer Vision – ECCV 2018*. Cham: Springer International Publishing. 170–186.

Xia, Y., X. Cao, F. Wen, G. Hua, and J. Sun. (2015). "Learning Discriminative Reconstructions for Unsupervised Outlier Removal". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 1511–1519.

Xu, G., H. Wu, and Y. Shi. (2016). "Structural Design of Convolutional Neural Networks for Steganalysis". *IEEE Signal Processing Letters*. 23(5): 708–712.

Xuan, X., B. Peng, J. Dong, and W. Wang. (2019). "On the generalization of GAN image forensics". *CoRR*. abs/1902.11153. arXiv: 1902.11153. URL: http://arxiv.org/abs/1902.11153.

Yang, P., D. Baracchi, M. Iuliani, D. Shullani, R. Ni, Y. Zhao, and A. Piva. (2020a). "Efficient Video Integrity Analysis Through Container Characterization". *IEEE Journal of Selected Topics in Signal Processing*. 14(5): 947–954. DOI: 10.1109/JSTSP.2020.3008088.

Yang, P., D. Baracchi, R. Ni, Y. Zhao, F. Argenti, and A. Piva. (2020b). "A Survey of Deep Learning-Based Source Image Forensics". *Journal of Imaging*. 6(3): 9. DOI: 10.3390/jimaging6030009.

Yang, P., W. Zhao, R. Ni, and Y. Zhao. (2017). "Source Camera Identification Based On Content-Adaptive Fusion Network". *CoRR*. abs/1703.04856. arXiv: 1703.04856. URL: http://arxiv.org/abs/1703.04856.

Yang, X., Y. Li, and S. Lyu. (2019a). "Exposing Deep Fakes Using Inconsistent Head Poses". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 8261–8265. DOI: 10.1109/ICASSP.2019.8683164.

Yang, X., Y. Li, H. Qi, and S. Lyu. (2019b). "Exposing GAN-synthesized Faces Using Landmark Locations". *CoRR*. abs/1904.00167. arXiv: 1904.00167. URL: http://arxiv.org/abs/1904.00167.

Yao, Y., W. Hu, W. Zhang, T. Wu, and Y.-Q. Shi. (2018). "Distinguishing Computer-Generated Graphics from Natural Images Based on Sensor Pattern Noise and Deep Learning". *Sensors*. 18(4): 1296. DOI: 10.3390/s18041296.

Yao, Y., Y. Shi, S. Weng, and B. Guan. (2017). "Deep Learning for Detection of Object-Based Forgery in Advanced Video". *Symmetry*. 10(Dec.): 3. DOI: 10.3390/sym10010003.

Yarlagadda, S. K., D. Güera, P. Bestagini, F. M. Zhu, S. Tubaro, and E. J. Delp. (2018). "Satellite Image Forgery Detection and Localization Using GAN and One-Class Classifier". *CoRR*. abs/1802.04881. arXiv: 1802.04881. URL: http://arxiv.org/abs/1802.04881.

You Won't Believe What Obama Says in This Video!, Youtube video. (2018). URL: https://www.youtube.com/watch?v=cQ54GDm1eL0.

Yu, F. and V. Koltun. (2015). "Multi-Scale Context Aggregation by Dilated Convolutions". arXiv: 1511.07122 [cs.CV].

Yu, F., V. Koltun, and T. A. Funkhouser. (2017a). "Dilated Residual Networks". *CoRR*. abs/1705.09914. arXiv: 1705.09914. URL: http://arxiv.org/abs/1705.09914.

Yu, I., D. Kim, J. Park, J. Hou, S. Choi, and H. Lee. (2017b). "Identifying photorealistic computer graphics using convolutional neural networks". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 4093–4097.

Yu, N., L. Davis, and M. Fritz. (2018). "Attributing Fake Images to GANs: Analyzing Fingerprints in Generated Images". *CoRR*. abs/1811.08180. arXiv: 1811.08180. URL: http://arxiv.org/abs/1811.08180.

Zagoruyko, S. and N. Komodakis. (2015). "Learning to Compare Image Patches via Convolutional Neural Networks". *CoRR*. abs/1504.03641. arXiv: 1504.03641. URL: http://arxiv.org/abs/1504.03641.

Zhang, A., Z. C. Lipton, M. Li, and A. J. Smola. (2020). *Dive into Deep Learning*.

Zhou, B., A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. (2015). "Learning Deep Features for Discriminative Localization". arXiv: 1512.04150 [cs.CV].

Zhou, P., X. Han, V. I. Morariu, and L. S. Davis. (2018a). "Learning Rich Features for Image Manipulation Detection". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1053–1061. DOI: 10.1109/CVPR.2018.00116.

Zhou, P., X. Han, V. I. Morariu, and L. S. Davis. (2018b). "Two-Stream Neural Networks for Tampered Face Detection". *CoRR*. abs/1803.11276. arXiv: 1803.11276. URL: http://arxiv.org/abs/1803.11276.

Zhu, J.-Y., T. Park, P. Isola, and A. A. Efros. (2017). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". arXiv: 1703.10593 [cs.CV].