– Supplemental Material –
FaceForensics++: Learning to Detect Manipulated Facial Images

Andreas Rössler[1] Davide Cozzolino[2] Luisa Verdoliva[2] Christian Riess[3]
Justus Thies[1] Matthias Nießner[1]

[1]Technical University of Munich [2]University Federico II of Naples [3]University of Erlangen-Nuremberg

In *FaceForensics++*, we evaluate the performance of state-of-the-art facial manipulation detection approaches using a large-scale dataset that we generated with four different facial manipulation methods. In addition, we proposed an automated benchmark to compare future detection approaches as well as their robustness against unknown post-processing operations such as compression.

This supplemental document reports details on our pristine data acquisition (Section 1), ensuring suited input sequences. Section 2 lists the exact numbers of our binary classification experiments presented in the main paper. Besides binary classification, the database is also interesting for evaluating manipulation classification (Section 3). In Section 4, we list all chosen hyperparameters of both the manipulation methods as well as the detection techniques.

1. Pristine Data Acquisition

For a realistic scenario, we chose to collect videos in the wild, more specifically from YouTube. Early experiments with all manipulation methods showed that the pristine videos have to fulfill certain criteria. The target face has to be nearly front-facing and without occlusions, to prevent the methods from failing or producing strong artifacts (see Fig. 1). We use the YouTube-8m dataset [2] to collect videos with the tags “face”, “newscaster” or “newsprogram” and also included videos which we obtained from the YouTube search interface with the same tags and additional tags like “interview”, “blog”, or “video blog”. To ensure adequate video quality, we only downloaded videos that offer a resolution of 480p or higher. For every video, we save its metadata to sort them by properties later on. In order to match the above requirements, we first process all downloaded videos with the Dlib face detector [9], which is based on Histograms of Oriented Gradients (HOG). During this step, we track the largest detected face by ensuring that the centers of two detections of consecutive frames are pixel-wise close. The histogram-based face tracker was chosen to ensure that the resulting video sequences contain little occlusions and, thus, contain easy-to-manipulate faces.

Except FaceSwap, all methods need a sufficiently large set of image in a target sequence to train on. We select sequences with at least 280 frames. To ensure a high quality video selection and to avoid videos with face occlusions, we perform a manual screening of the clips which resulted in 1,000 video sequences containing $509,914$ images.

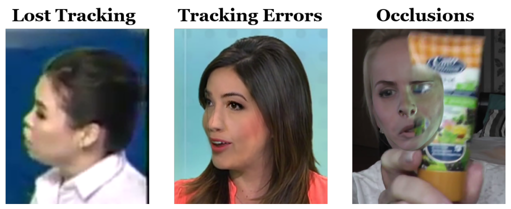

Figure 1: Automatic face editing tools rely on the ability to track the face in the target video. State-of-the-art tracking methods like Thies et al. [14] fail in cases of profile imagery of a face (left). Rotations larger than $45°$ (middle) and occlusions (right) lead to tracking errors.

Methods	Train	Validation	Test
Pristine	367,282	68,862	73,770
DeepFakes	367,282	68,862	73,770
Face2Face	367,282	68,862	73,770
FaceSwap	292,376	54,630	59,672
NeuralTextures	292,376	54,630	59,672

Table 1: Number of images per manipulation method.

All examined manipulation methods need a source and a target video. In case of facial reenactment, the expressions of the source video are transferred to the target video while retaining the identity of the target person. In contrast, face swapping methods replace the face in the target video with the face in the source video. To ensure high quality face swapping, we select video pairs with similar large faces (considering the bounding box sizes detected by DLib), the same gender of the persons and similar video frame rates.

Table 1 lists the final numbers of our dataset for all manipulation methods and the pristine data.

2. Forgery Detection

In this section, we list all numbers from the graphs of the main paper. Table 2 shows the accuracies of the manipulation-specific forgery detectors (i.e., the detectors are trained on the respective manipulation method). In contrast, Table 3 shows the accuracies of the forgery detectors trained on the whole *FaceForensics++* dataset. In Table 4, we show the importance of a large-scale database. The numbers of our user study are listed in Table 5 including the modality which is used to inspect the images.

3. Classification of Manipulation Method

To train the XceptionNet classification network to distinguish between all four manipulation methods and the pristine images, we adapted the final output layer to return five class probabilities. The network is trained on the full dataset containing all pristine and manipulated images. On raw data the network is able to achieve a 99.03% accuracy, which slightly decreases for the high quality compression to 95.42% and to 80.49% on low quality images.

4. Hyperparameters

For reproducibility, we detail the hyperparameters used for the methods in the main paper. We structured this section into two parts, one for the manipulation methods and the second part for the classification approaches used for forgery detection.

4.1. Manipulation Methods

DeepFakes and *NeuralTextures* are learning-based, for the other manipulation methods we used the default parameters of the approaches.

DeepFakes: Our *DeepFakes* implementation is based on the *deepfakes faceswap github project* [1]. MTCNN ([15]) is used to extract and align the images for each video. Specifically, the largest face in the first frame of a sequence is detected and tracked throughout the whole video. This tracking information is used to extract the training data for *DeepFakes*. The auto-encoder takes input images of 64 (default). It uses a shared encoder consisting of four convolutional layers which downsizes the image to a bottleneck of 4×4, where we flatten the input, apply a fully connected layer, reshape the dense layer and apply a single upscaling using a convolutional layer as well as a pixel shuffle layer (see [12]). The two decoders use three identical up-scaling layers to attain full input image resolution. All layers use Leaky ReLus as non-linearities. The network is trained using Adam with a learning rate of 10^{-5}, $\beta_1 = 0.5$ and $\beta_2 = 0.999$ as well as a batch size of 64. In our experiments, we run the training for 200000 iterations on a cloud platform. By exchanging the decoder of one person to another, we can generate an identity-swapped face region. To insert the face into the target image, we chose Poisson Image Editing [10] to achieve a seamless blending result.

NeuralTextures: *NeuralTextures* is based on a U-Net architecture. For data generation, we employ the original pipeline and network architecture (for details see [13]). In addition to the photo-metric consistency, we added an adversarial loss. This adversarial loss is based on the patch-based discriminator used in Pix2Pix [8]. During training we weight the photo-metric loss with 1 and the adversarial loss with 0.001. We train three models per manipulation for a fixed 45 epochs using the Adam optimizer (with default settings) and manually choose the best performing model based on visual quality. All manipulations are created at a resolution of 512×512 as in the original paper, with a texture resolution of 512×512 and 16 feature per texel. Instead of using the entire image, we only train and modify the cropped image containing the face bounding box ensuring high resolution outputs even on higher resolution images. To do so, we enlarge the bounding box obtained by the Face2Face tracker by a factor of 1.8.

4.2. Classification Methods

For our forgery detection pipeline proposed in the main paper, we conducted studies with five classification approaches based on convolutional neural networks. The networks are trained using the Adam optimizer with different parameters for learning-rate and batch-size. In particular, for the network proposed in *Cozzolino at al.* [6] the used learning-rate is 10^{-5} with batch-size 16. For the proposal of *Bayar and Stamm* [4], we use a learning-rate equal to 10^{-5} with a batch-size of 64. The network proposed by *Rahmouni* [11] is trained with a learning-rate of 10^{-4} and a batch-size equal to 64. *MesoNet* [3] uses a batch-size of 76 and the learning-rate is set to 10^{-3}. Our *XceptionNet* [5]-based approach is trained with a learning-rate of 0.0002 and a batch-size of 32. All detection methods are trained with the Adam optimizer using the default values for the moments ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$).

We compute validation accuracies ten times per epoch and stop the training process if the validation accuracy does not change for 10 consecutive checks. Validation and test accuracies are computed on 100 images per video, training is evaluated on 270 images per video to account for frame count imbalance in our videos. Finally, we solve the imbalance between real and fake images in the binary task (i.e., the number of fake images being roughly four times as large as the number of pristine images) by weighing the training images correspondingly.

	Raw				**Compressed 23**				**Compressed 40**			
	DF	**F2F**	**FS**	**NT**	**DF**	**F2F**	**FS**	**NT**	**DF**	**F2F**	**FS**	**NT**
Steg. Features + SVM [7]	99.03	99.13	98.27	**99.88**	77.12	74.68	79.51	76.94	65.58	57.55	60.58	60.69
Cozzolino *et al.* [6]	98.83	98.56	98.89	**99.88**	81.78	85.32	85.69	80.60	68.26	59.38	62.08	62.42
Bayar and Stamm [4]	99.28	98.79	98.98	98.78	90.18	94.93	93.14	86.04	80.95	77.30	76.83	72.38
Rahmouni *et al.* [11]	98.03	98.96	98.94	96.06	82.16	93.48	92.51	75.18	73.25	62.33	67.08	62.59
MesoNet [3]	98.41	97.96	96.07	97.05	95.26	95.84	93.43	85.96	89.52	84.44	83.56	75.74
XceptionNet [5]	**99.59**	**99.61**	**99.14**	99.36	**98.85**	**98.36**	**98.23**	**94.5**	**94.28**	**91.56**	**93.7**	**82.11**

Table 2: Accuracy of manipulation-specific forgery detectors. We show the results for raw and the compressed datasets of all four manipulation methods (DF: DeepFakes, F2F: Face2Face, FS: FaceSwap and NT: NeuralTextures).

	Raw					**Compressed 23**					**Compressed 40**				
	DF	**F2F**	**FS**	**NT**	**P**	**DF**	**F2F**	**FS**	**NT**	**P**	**DF**	**F2F**	**FS**	**NT**	**P**
Steg. Features + SVM [7]	97.96	98.40	91.35	98.56	98.70	68.80	67.69	70.12	69.21	72.98	67.07	48.55	48.68	55.84	56.94
Cozzolino *et al.* [6]	97.24	98.51	95.93	98.74	99.53	75.51	86.34	76.81	75.34	78.41	75.63	56.01	50.67	62.15	56.27
Bayar and Stamm [4]	99.25	99.04	96.80	**99.11**	98.92	90.25	93.96	87.74	83.69	77.02	86.93	83.66	74.28	74.36	53.87
Rahmouni *et al.* [11]	94.83	98.25	97.59	96.21	97.34	79.66	87.87	84.34	62.65	79.52	80.36	62.04	59.90	59.99	56.79
MesoNet [3]	99.24	98.35	98.15	97.96	92.04	89.55	88.60	81.24	76.62	82.19	80.43	69.06	59.16	44.81	**77.58**
XceptionNet [5]	**99.29**	**99.23**	**98.39**	98.64	**99.64**	**97.49**	**97.69**	**96.79**	**92.19**	**95.41**	**93.36**	**88.09**	**87.42**	**78.06**	75.27
Full Image Xception [5]	87.73	83.22	79.29	79.97	81.46	88.00	84.98	82.23	79.60	65.85	84.06	77.56	76.12	66.03	65.09

Table 3: Detection accuracies when trained on all manipulation methods at once and evaluated on specific manipulation methods or pristine data (DF: DeepFakes, F2F: Face2Face, FS: FaceSwap, NT: NeuralTextures, and P: Pristine). The average accuricies are listed in the main paper.

	Raw					**Compressed 23**					**Compressed 40**				
	DF	**F2F**	**FS**	**NT**	**All**	**DF**	**F2F**	**FS**	**NT**	**All**	**DF**	**F2F**	**FS**	**NT**	**All**
10 videos	89.18	76.6	90.89	93.53	92.81	76.06	59.84	81.15	76.73	67.71	64.55	53.99	60.04	65.14	60.55
50 videos	99.52	98.84	97.56	96.67	95.89	92.48	91.33	92.63	85.98	82.89	75.53	66.44	74.25	71.48	65.76
100 videos	99.51	99.09	98.64	98.23	97.54	95.39	95.8	95.56	90.09	87.19	83.68	72.69	79.56	73.72	66.81
300 videos	**99.59**	**99.53**	**98.78**	**98.73**	**98.88**	**97.30**	**97.41**	**97.51**	**92.4**	**92.65**	**91.57**	**86.38**	**88.35**	**79.65**	**76.01**

Table 4: Analysis of the training corpus size. Numbers reflect the accuracies of the XceptionNet detector trained on single and all manipulation methods (DF: DeepFakes, F2F: Face2Face, FS: FaceSwap, NT: NeuralTextures and All: all manipulation methods).

	Raw					**Compressed 23**					**Compressed 40**				
	DF	**F2F**	**FS**	**NT**	**P**	**DF**	**F2F**	**FS**	**NT**	**P**	**DF**	**F2F**	**FS**	**NT**	**P**
Average	77.60	49.60	76.12	32.28	78.19	78.17	50.19	74.80	30.75	75.41	73.18	43.86	64.26	39.07	62.06
Desktop PC	**80.41**	**53.73**	75.10	**34.36**	**80.12**	**81.17**	**51.57**	**79.51**	29.50	**78.10**	71.71	**44.09**	62.99	35.71	**64.32**
Mobile Phone	74.80	44.96	**77.11**	30.58	76.40	75.47	48.95	70.08	**31.97**	72.84	**74.62**	43.62	**65.50**	**41.85**	60.00

Table 5: User study result w.r.t. the used device to watch the images (DF: DeepFakes, F2F: Face2Face, FS: FaceSwap, NT: NeuralTextures and P: Pristine). 99 participants used a PC and 105 a mobile phone.

References

[1] Deepfakes github. https://github.com/deepfakes/faceswap. Accessed: 2018-10-29. 2

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. YouTube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 1

[3] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Mesonet: a compact facial video forgery detection network. *arXiv preprint arXiv:1809.00888*, 2018. 2, 3

[4] Belhassen Bayar and Matthew C. Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *ACM Workshop on Information Hiding and Multimedia Security*, pages 5–10, 2016. 2, 3

[5] Francois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 3

[6] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. In *ACM Workshop on Information Hiding and Multimedia Security*, pages 1–6, 2017. 2, 3

[7] Jessica Fridrich and Jan Kodovský. Rich Models for Steganalysis of Digital Images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, June 2012. 3

[8] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 2

[9] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009. 1

[10] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on graphics (TOG)*, 22(3):313–318, 2003. 2

[11] Nicolas Rahmouni, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Distinguishing computer graphics from natural images using convolution neural networks. In *IEEE Workshop on Information Forensics and Security*, pages 1–6, 2017. 2, 3

[12] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016. 2

[13] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics 2019 (TOG)*, 2019. 2

[14] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2Face: Real-Time Face Capture and Reenactment of RGB Videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2387–2395, June 2016. 1

[15] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016. 2