Improving Generalization of Deepfake Detection with Data Farming and Few-Shot Learning

Pavel Korshunov and Sébastien Marcel

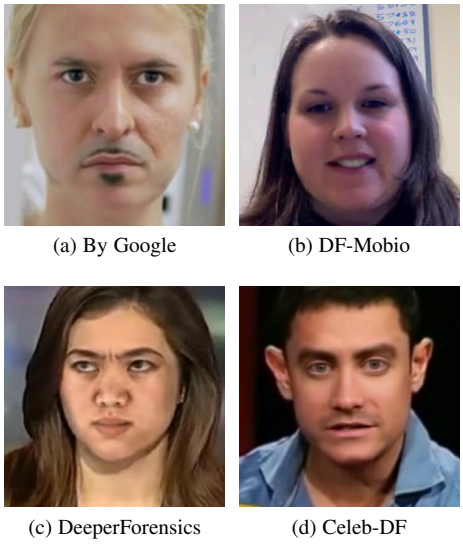
Abstract—Recent advances in automated video and audio editing tools, generative adversarial networks (GANs), and social media allow creation and fast dissemination of high quality tampered videos, which are generally called deepfakes. Typically, in these videos, a face is swapped with someone else's using GANs. Accessible open source software and apps for the face swapping led to a wide and rapid dissemination of the generated deepfakes, posing a significant technical challenge for their detection and filtering. In response to the threat, which deepfake videos can pose to our trust in video evidence, several large datasets of deepfake videos and several methods to detect them were proposed recently. However, the proposed methods suffer from a problem of overfitting on the training data and the lack of the generalization across different databases and the generative models. Therefore, in this paper, we investigate the techniques for improving the generalization of deepfake detection methods that can be employed in practical settings. We have selected two popular state of the art deepfake detectors: based on Xception and EfficientNet models, and we use five databases: from Google and Jigsaw, FaceForensics++, DeeperForensics, Celeb-DF, and our own publicly available large dataset DF-Mobio. To improve generalization, we apply different augmentation strategies used during training, including a proposed aggressive 'data farming' technique based on random patches. We also tested two few-shot tuning methods, when either a first convolutional layer or a last layer of a pre-trained model is tuned on 100 seconds from a training set of the test database. The experimental results clearly expose the generalization problem of deepfake detection methods, since the accuracy drops significantly when a model is trained on one dataset and evaluated on another. However, the silver lining is that an aggressive augmentation during training and a few-shot tuning on the test database can improve the accuracy of the detection methods in a cross-database scenario. As a side observation, we show the importance of database selection for training and evaluation, as FaceForensics++ is found to be better to use for training, while DeeperForensics is found to be significantly more challenging as a test database.

Index Terms—Deepfakes detection, generalization, evaluation, deepfake dataset.

1 Introduction

Autoencoders and generative adversarial networks (GANs) significantly improved the quality and realism of the automated image generation and face swapping, leading to the deepfake phenomena. The proverb 'seeing is believing' is starting to loose its meaning when it comes to digital video[1]. The concern for the impact of the widespread deepfake videos on the societal trust in video recording is growing. This public unease prompted researchers to propose various datasets of deepfakes and methods to detect them. Some of the latest approaches demonstrate encouraging accuracy, especially, if they are trained and evaluated on the same datasets [1], [2].

Many databases with deepfake videos were created to help develop and train deepfake detection methods. One of the first freely available database is based on VidTIMIT [3], followed by the FaceForensics [4] database and its extension FaceForensics++ [1], which contains several different types of deepfakes generated from $1'000$ Youtube videos. It also contains a subset provided by Google and Jigsaw, which can be considered as a separate database. A more recent DeeperForensics [5] database, using Youtube videos from FaceForensics, provides several types of custom deepfakes videos. Another but larger ($5'000$ videos) database based on Youtube videos (different from FaceForensics) is Celeb-Df [6]. But the most extensive and the largest database to date with more than $100K$ videos (80% of which are deepfakes) is the dataset from Facebook [7], which was provided in the Deepfake Detection Challenge 2020[2].



(a) By Google (b) DF-Mobio

(c) DeeperForensics (d) Celeb-DF

Fig. 1: Examples of deepfakes (faces cropped from videos) in different databases.

• *P. Korshunov and S. Marcel are with Idiap Research Institute, Martigny, Switzerland.*
E-mails: pavel.korshunov@idiap.ch, sebastien.marcel@idiap.ch

Manuscript received May 15, 2021;

[1]https://edition.cnn.com/interactive/2019/01/business/pentagons-race-against-deepfakes/

[2]https://www.kaggle.com/c/deepfake-detection-challenge

These datasets were generated using either the popular open source code[3], e.g., DeepfakeTIMIT [3], FaceForensics [4], DeeperForensics [5], and Celeb-Df [6], or the latest methods implemented by Google and Facebook for creating deepfakes (see Figure 1 for the examples of different deepfakes). The fact that even Google and Facebook, private companies which are typically very mindful about making large datasets publicly available, opened up some of the most extensive datasets, shows how important and challenging the deepfake detection is for both the scientific and industrial communities. The recent availability of large deepfake video databases allowed researchers to train and test detection approaches based on very deep neural networks, such as Xception [1], capsules networks [8], EfficientNet [2], and discriminators of GANs used to generate synthetic images [9], which were shown to outperform the methods based on shallow CNNs, facial physical characteristics [10], [11], [12], [13], or distortion features [14], [15].

However, as it is typical for deep learning based approaches, the latest methods for deepfake detection suffer from the lack of generalization [16], [17] on an unseen data and different types generative models used to create deepfakes. This problem was clearly shown during the Facebook's Deepfake Detection Challenge[2], when the top approaches of the competition have consistently shown a much lower error on the public validation set compared to the error on the secret test set, which contained unseen data and deepfakes generated using undisclosed methods. The lack of generalization is impeding the advances in the deepfake detection and their wide employment.

In this paper, using an extensive set of experiments, we investigate several techniques for improving the generalization of state of the art deepfake detection algorithms and evaluating them on five publicly available deepfake databases. It is important to note that we are not focusing on creating new algorithms or models for deepfake detection but focus on the techniques, which are independent of the algorithms themselves and can be applied to any underlying model to improve its generalization performance. For that reason, we have selected two state of the art algorithms, based on Xception model [18] and EfficientNet variant B4 [2], both of which showed a great performance on several databases [1] and were also successfully used in many top systems from Deepfake Detection Challenge [7].

We evaluate the algorithms using five large databases: FaceForensics++ [1], its separate subset from Google and Jigsaw, DeeperForensics [5], Celeb-DF [6], and our own database DFMobio[6] with almost $15'000$ deepfake videos that we generated from publicly available Mobio [19] video dataset. We have chosen these databases based on the following criteria: (i) a dataset need to have enough variability in terms of original videos used to generate deepfakes, the amount of data, the imbalance between real and fake subsets, and the methods used to generate deepfakes, and (ii) the database needs to be publicly available and accessible for research, so that the evaluation results could be reproduced and verified. The largest to-date database that was generated by Facebook [7] for the Deepfake Detection Challenge has a strict license and was not made publicly available outside of the challenge, so, we had to exclude it from this study.

Using the selected two deepfake detection methods and the five databases with deepfake videos, we demonstrate, experimentally, how well the methods perform when trained and evaluated on the same database and how their performance degrades when they are trained on one database but evaluated on another. We investigate augmentation strategies (illustrated in Figure 2) that increase the variability in the training data to confirm whether augmentation has a significant enough impact on deepfake detection as some of the related studies have suggested [9]. We propose to go beyond the traditional image augmentation, such as random image blurring, mirroring, JPEG compression, etc., and to mix together during training the background patches and faces for the class of genuine (real) samples to force the detector to learn the artifacts related to deepfakes instead of facial features. We call this strategy *data farming* (Figure 2c shows examples of input farmed during training), since an additional trove of data, which effectively doubles the often under-represented genuine class, is farmed from the existing videos. We also analyze and compare several few-shot tuning methods when the models pre-trained on one dataset are tuned on a few samples from the training set of another dataset with the aim to improve the generalization during testing.

The reasoning for using the drastic *data farming* strategy to increase variability of the input is that using patches cropped from the background in equal proportion with the faces will force the neural networks to learn the actual visual artifacts of deepfakes and reduce their overfitting on faces themselves. Adding more genuine data to the training will also help to balance out the genuine data portion (we are effectively doubling it by adding background patches), which is often significantly smaller compared to the amount of the deepfake data (Table 1 shows how significant the imbalance can be).

To summarize, we pose the following important research questions:

1) Do data augmentation and more aggressive data farming techniques improve the generalization of a given detection method to unseen real and deepfake data and by how much?
2) Considering the practical constraints of a given deepfake detector, which tuning technique (using a few samples from unseen data) maximize the generalization to the new data?

Additionally, we analyze the databases we used in the experiments in terms of their usefulness as either being a benchmark database (which database is the most challenging when it is tested on?) or as being the best for training the deepfake detectors (which database leads to the lowest detection errors across the whole test spectrum when it is trained on?).

To allow researchers to verify, reproduce, and extend our work, we provide pre-trained models and the code-base for running the experiments as an open source package[4]. The package also contains a Jupyter notebook with more detailed analysis of the results presented only as a condensed summary in this paper.

2 Related work

One of the first works on face swapping is by Bitouk *et al.* [20], where the authors searched in a database for a face similar in appearance to the input face and then focused on perfecting the blending of the found face into the input image. The main motivation for this work was de-identification of an input face and its privacy preservation. Hence, the approach did not allow for a

[3]https://www.kaggle.com/c/deepfake-detection-challenge/discussion/121313

[4]Code: https://gitlab.idiap.ch/bob/bob.paper.deepfakes_generalization

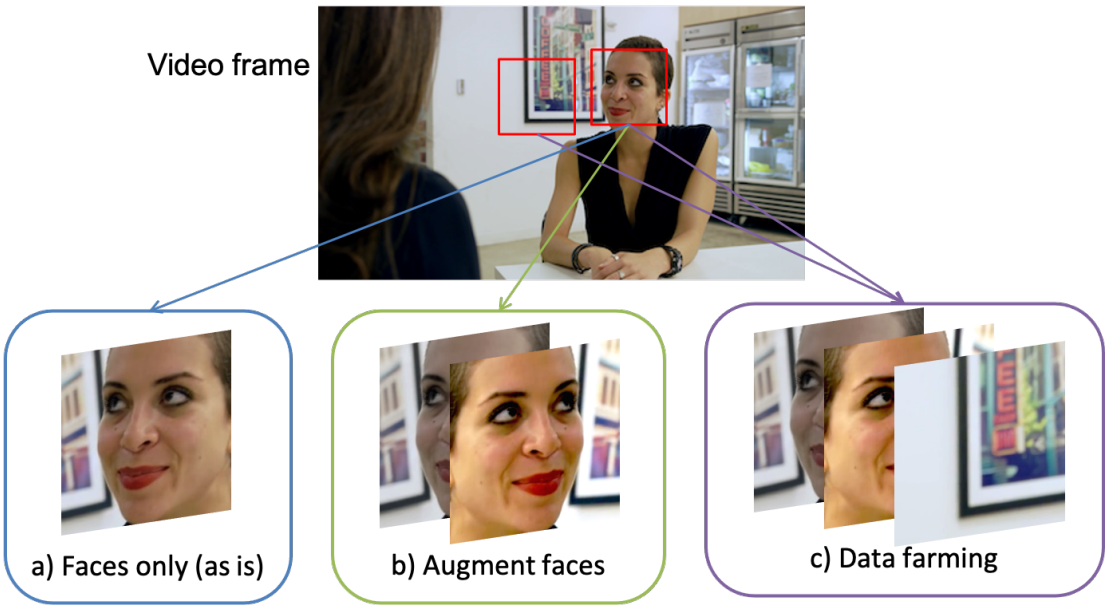


Fig. 2: Different augmentation techniques used in the experiments for increasing the variability of the training data: a) use detected faces as is with no augmentation, b) apply typical augmentation techniques to faces c) use a more drastic *data farming* that combines faces, augmentation, and background patches.

seamless swapping of any two given faces. Until the latest era of neural networks, most of the techniques for face swapping or facial reenactment were based on similarity searchers between faces or face patches in target and source video and various blending techniques [21], [22], [23], [24], [25].

The first approach that used a generative adversarial network to train a model between pre-selected two faces was proposed by Korshunova *et al.* in 2017 [26]. Another related work with even a more ambitious idea was to use long short term memory (LSTM) based architecture to synthesize a mouth feature solely from an audio speech [27]. Right after these publication became public, they attracted a lot of publicity. Open source approaches replicating these techniques started to appear, which resulted in the Deepfake phenomena.

The rapid spread of deepfakes and the ease of generating such videos motivated researchers to create large datasets of deepfakes and propose methods for their detection. Recently available deepfake video databases (see Figure 1 for some examples) include DeepfakeTIMIT [3], FaceForensics++ [1], with a subset provided by Google and Jigsaw [28], Celeb-Df [6], and the dataset by Facebook used in Kaggle Deepfake Detection Challenge. The large enough amount of video deepfakes allow researchers to train and test detection approaches based on very deep neural networks, such as Xception [1], capsules networks [8], and EfficientNet [2], which were shown to outperform the methods based on shallow CNNs, facial physical characteristics [11], [13], or distortion features [14], [15], [10].

Some of the most recent work focuses mainly on two problems: (i) detecting synthetic standalone faces generated by different GAN variants and (ii) detecting videos where faces are swapped using a GAN-generated face, which is blended in the original video with an image blending technique (either automated or manual). The first problem of detecting pure synthetic images and generalization for different types of GANs is intrinsically simpler, since GANs leave specific patterns and artifacts in an image and if it does not undergo a lot of post-processing, these patterns are relatively easy to detect, which was already shown by Wang *et al.* [9] and confirmed by several other works [29], [30].

However, the second problem is very different. To make a swapped face in a video appear as realistic as possible, heavy post-processing techniques are applied, which 'in the wild' is often done manually. This pos-processing makes it much harder to develop an approach that would generalize well to unseen attacks, since it must not only detect the artifacts from the underlying GAN but also from an unspecified number of post-processing techniques. Several work demonstrate this problem as a much harder one, including ForensicTransfer method proposed by Cozzolino *et al.* [30], which showed that generalization for pure synthetic images is an easier one compared to face swapping. The authors of [31] evaluated ForensicTransfer comparing it to a simple Xception model (which we also employ in our paper) in cross-database scenario and stated that both are "far from a good performance" and that "there is no clear model or pre-processing method that stands out as best." Tolosana *et al.* [32] also noted that videos from Celeb-DF and the dataset by Facebook (the set by Google and Jigsaw is similar in quality) are much harder to detect as the quality of deepfakes is higher compared to those video that are in FaceForensics++. A good comparison of different state of the art deepfake detection methods can be found in several recent reviews [33], [34], [35]

An interesting work from the same team that proposed ForensicTransfer and created FaceForensics++ proposed a method that is specifically designed to generalize to different facial forgeries by learning an embedding space with a certain distribution [36]. The approach demonstrated that a few-short learning approach can improve the accuracy of detecting deepfakes in a new datasets, however, what the authors failed to show is how the performance of tuned model degraded on the original dataset (that was used for training) and what is the accuracy of detecting real videos from the new dataset. It is worth noting that some proposed solutions extensively evaluate the accuracy of detecting deepfakes,

(a) Original (b) FF++ Deepfakes (c) FF++ Face Shifter (d) FF++ Face Swap (e) DF End-to-end (f) DF Reenact

Fig. 3: Examples of cropped faces from videos of original and deefpake subsets of FaceForensics++ and DeeperForensics databases.

TABLE 1: Databases of deepfakes used in the experiments.

Database	Number of swapped identities	Original videos	Deepfakes
FaceForensics++	1000	1000	3000
DeeperForensics	1000	1000	2000
from Google and Jiqsaw	approx. 150	360	3068
Celeb-DF	59	590	5639
DF-Mobio[6]	72	31 950	14 546

while ignoring the degradation in detecting the real videos, but in that case, classifying every video as fake would result in 100% accuracy of deepfake detection without the need to create complex models.

What sets our work apart from the many previous approaches is that we propose the augmentation strategies and several few-shot learning techniques that can be used for any of the existing deep learning architectures or method to improve their generalization characteristics. We also present error rates for both classes: deepfakes and real videos, and we evaluate the proposed approaches on the same dataset that was used for training, across different datasets and the unseen attacks, and in the realistic settings, when the hyper-parameters of the system (i.e., the evaluation threshold) are chosen before the evaluation.

3 Databases and Methods

Table 1 summarizes the databases of deepfake videos that we have used in the experiments. We have used three subsets from FaceForensics++ that we can call as actually having deepfake videos (generated using GANs), including, 'Face Swap', 'Deepfakes', 'Face Shifter' (see The database by Google and Jigsaw and DF-Mobio database were split into three approximately equal in size subsets, for training, validation, and testing. The authors of Celeb-DF dataset predefined file lists for training and testing subsets but there was no validation set provided. Hence, in the experiments with Celeb-DF database, we used the test set for validation as well.

3.1 FaceForensics++

FaceForensics++[5] [1] is a deepfake dataset consisting of 1000 original video sequences and the corresponding fakes generated using several face manipulation and swapping methods, including the deepfakes (see Figure 3 for examples). Videos in the datasets have mostly frontal faces without occlusions which enables automated tampering methods to generate realistic forgeries. It also provides binary masks that can be used for image and video classification as well as segmentation.

[5]https://github.com/ondyari/FaceForensics

The dataset contains five different types of deepfake attacks (1000 videos each) however, since in this paper, we are focusing on deepfakes and their detection, we have excluded two subsets from our experiments, for which GANs were not used to generate the videos: 'Neural Textures' and 'Face2Face'. Hence the deefpakes of FaceForensics++ were represented (in 'c23' compression) by three sets that are called, in the database's terminology, 'Deepfakes', 'Face Shifter', and 'Face Swap' (see Figure 3 for examples).

As per provided protocol [1], we used 720 videos for training, 140 each for validation and testing for each set of deepfakes. The original videos subset from youtube is also split in a similar manner.

3.2 DeeperForensics

DeeperForensics [5] is one of the latest databases of deepfakes and uses the same original videos as in FaceForensics++, so it can be considered as its extension. DeeperForensics adds two types of deepfakes, generated in end-to-end manner and using reenacted faces (see Figure 3 for examples). The dataset also contains other manipulated videos that have various random-type distortions applied to them. In this paper, we focus on the deepfakes themselves and therefore use the two sets without any extra distortions.

We use the splits for training (703 videos), validation (96 videos), and testing (201 videos) provided by the authors of DeeperForensics which are different from those given in the original FaceForensics++ database.

3.3 Google and Jigsaw database

Strictly speaking this dataset is part of the FaceForensics++ [1] database, however, it is so different and is done by a different group of authors, hence we consider it as a separate dataset. To make this dataset, Google and Jigsaw [28] (see Table 1 for the comparison with other databases) worked with paid and consenting actors to record hundreds of videos. Using publicly available deepfake generation methods, Google then generated about $3K$ of deepfakes from these videos. The resulting videos, real and fake, comprise the contribution, which was created to

(a) Original 1 (b) Original 2 (c) Swap 1 → 2

Fig. 4: Screenshots of the original videos and a deepfake swap from DF-Mobio database.

directly support deepfake detection efforts. This dataset can be downloaded through FaceForensics++ GitHub account.

3.4 Celeb-DF

Celeb-DF (v2) [6] dataset contains real and deepfake synthesized videos having similar visual quality on par with those circulated online. The Celeb-DF (v2) dataset is greatly extended from the previous Celeb-DF (v1), which only contained 795 deepfake videos.

The v2 of the database contains more than $5K$ deepfakes and nearly $2K$ real videos, which are based on publicly available YouTube video clips of 59 celebrities of diverse genders, ages, and ethic groups. The deepfake videos are generated using an improved DeepFake synthesis method [6], which essentially is an extension of methods available online[7], similar to the one used to generate both FaceForensics and DF-Mobio database. The authors of the Celeb-DF database claim that their modified algorithm improves the overall visual quality of the synthesized deepfakes when compared to existing datasets. The authors also state that Celeb-DF is challenging to most of the existing detection methods, even though many deepfake detection methods are shown to achieve high, sometimes near perfect, accuracy on previous datasets. No consent was obtained for the videos, because the data is from the celebrities of the Youtube videos.

3.5 DF-Mobio

DF-Mobio[6] dataset is generated by us and is one of the largest databases available with almost $15K$ deepfake and $31K$ real videos (see Table 1 for the comparison with other databases). Original videos are taken from Mobio database [19], which contains videos of a single person talking to the camera recorded with a phone or a laptop. The scenario simulates the participation in a virtual meeting over Zoom or Skype.

The original Mobio database contains $31K$ videos from 152 subjects but deepfakes were generated only for manually preselected 72 pairs of people with similar hairstyles, facial features, facial hair, and eyewear. Using GAN-based face-swapping algorithm based on the available code[7], for each pair, we generated videos with swapped faces from subject one to subject two and visa versa (see Figure 4 for video screenshot examples).

The GAN model for face swapping was trained on face size input of 256×256 pixels. The training images were generated from laptop-recorded videos at 8 fps, resulting in more than $2K$ faces for each subject, the training was done for $40K$ iterations (about 24 hours on Tesla P80 GPU). The availability of this database to public is pending a journal publication.

4 Deepfake Detection Approaches

Several methods for detecting deepfakes were proposed recently but for our study, we consulted a public benchmark[8] of some of the latest CNN-based approaches for deepfake detection. Based on this benchmark, we selected two well-performing, published, and reproducible methods that were also used in the latest Deepfake Detection Challenge hosted by Kaggle[2]. Therefore, we selected the method based on the Xception model [18], which was proposed for deepfake detection by Röessler *et al.* [1], and the method based on EfficientNets [37], which was widely used by participants of in Deepfake Detection Challenge.

4.1 XceptionNet

XceptionNet [18] is a traditional CNN trained on ImageNet [38] based on separable convolutions with residual connections. For the purpose of deepfake detection the authors of FaceForensics++ database [1] proposed to replace the final layer of XceptionNet by a fully connected layer with one output and a sigmoid activation. In our experiments, we followed the same approach suggested in [1]. The network is trained for 20 epochs and the best performing model is chosen based on validation accuracy. The Xception-based approach is trained on the input images of 299×299 pixels, with a batch-size of 32, and a learning-rate of 0.0002, using Adam optimizer with the default values for the moments.

For brevity, we refer throughout the paper to the detection method that relies on XceptionNet simply as *Xception*.

4.2 EfficientNet variant B4

EfficientNets [37] are a family of image classification models, which achieve state-of-the-art accuracy, yet being an order-of-magnitude smaller and faster than previous models. These models were created via a so called *neural architecture* search with the aim to design a new type of network that can be easily scaled up to generate a set of different-size models suitable for solving different tasks. The resulted family of EfficientNets arguably achieve better accuracy on several image classification tasks compared to ConvNets [37].

[6]https://www.idiap.ch/dataset/df-mobio

[7]https://github.com/shaoanlu/faceswap-GAN

[8]http://kaldir.vc.in.tum.de/faceforensics_benchmark/

(a) Face, no augmentation (b) Face with augmentation applied (c) Patch of the background (d) Deepfake face

Fig. 5: Examples of preprocessing strategies applied to images from Google and Jigsaw database to increase data variability and improve generalization of trained models.

For our experiments, we have selected B4 variant of the EfficientNet family of networks, because it was one of the most popular models used in many top solutions [2] of Facebook Deepfake Detection Challenge[2]. Throughout the paper, we refer to this model simply as *EfficientNet*.

We train the EfficientNet model by following the same procedure and using the same parameters as we set for Xception model (see Section 4.1 for details). We present the rational for making these choices and describe the experimental process in detail in the next section.

5 Experiments Design

The experiments are designed to investigate the strategies for improving the generalization of deepfake detection methods, while minimizing the influence on the results from unrelated factors such as training parameters, image preprocessing, or the value of random seed.

We aim to investigate and answer two main questions:

1) To improve generalization of deepfake detection, what is the best input preprocessing strategy for increasing the variability of the images used for training?
2) Does tuning of a pre-trained model on a *small* subset of *unseen* data improve generalization and by how much?

5.1 Variability of input images

To answer the first question, we explore three preprocessing strategies (see the illustration summarizing the strategies in Figure 2 and the resulted image examples in Figure 5) that can be applied to input images during training:

1) Use detected and aligned faces, with no additional processing, as input to a neural network. This strategy is referred to as *face-only* in the paper.
2) Apply random image augmentation to input faces to increase input's variability. This strategy is referred to as *face-augm* in the paper. We used the following augmentation techniques implemented in Tensorflow framework[9]:
   - Changing hue of images
   - Changing brightness
   - Changing contrast
   - Changing saturation
   - Flipping a image left or right
   - Changing JPEG quality with quantizer value between 50 and 90.
3) In addition to using image augmentation, farm extra data from the existing video frames. For each genuine face, add a patch of the same size randomly cropped from the background (an area around the face). This approach doubles the amount of genuine input images in the training set, which now contains both faces and extra patches of the background. This strategy is referred to as *augm-patches* in the paper.

The reasoning for using the drastic *data farming* strategy to increase variability of the input is that using background in equal proportion with faces will force the neural networks to learn the actual visual artifacts of deepfakes and reduce their overfitting on faces themselves. Adding more genuine data to the training will also help to balance out the genuine data portion, which is often significantly smaller compared to the amount of the deepfake data (see Table 1 for examples of data imbalance).

5.2 Few-shot model tuning on new data

We consider a practical scenario when a deepfake detection model is trained on a large database of deepfake and genuine videos, the operational detection threshold is established on a validation set, and then the system is deployed in some other environment to detect the deepfakes. In practice, however, the deepfakes in the test environment can differ significantly from the deepfakes used to train and validate the system. Very different GAN architectures and blending techniques can be used to generate these *unseen* deepfakes, so it can be unreasonable to expect the pre-trained system to generalize well on such new data. However, in other research domains, it was shown [39] that a few-shot learning can be used to tune a pre-trained neural network and it will improve its generalization. Therefore, we investigate whether such model tuning can also significantly improve the generalization of deepfake systems.

We explore the following few-shot tuning approaches:

1) No model tuning, which is labeled in the paper as *none*.
2) Tune the first convolutional layer of the neural network, i.e., freeze all weights except for the first layer. This tuning method is labeled in the paper as *first layer*.

[9]https://www.tensorflow.org/tutorials/images/data_augmentation

3) Tune the last fully connected layer, i.e., freeze all weights except for the last layer. This tuning method is labeled as *last layer*.

Practically speaking, we want to check how the performance of a given neural network model, pre-trained on one database (training set), designated as A, will change when it is tuned using only a few samples taken from another database (training set), designated as B. Such tuning would follow a typical simple few-shot learning approach. Once the model is trained on database A and tuned on database B, we need to evaluate the effect of tuning in two scenarios: i) whether the performance on the original database A is degraded and ii) whether the performance on database B has improved. Therefore, the tuned model is evaluated on both, the test set of the original database A, and the test set of the database B.

5.3 Experimental process and parameters

To investigate the two approaches (one is data-driven and another is based on tuning) for improving the generalization of the deepfake detection systems, we conducted an extensive set of experiments. Since we have five databases, two neural networks, and three different strategies for increasing variability of the input data, we have trained 30 different original non-tuned models. Then, we tune these models on a small subset of data, which we take from the training set of another database, on which we did not train the original model. Therefore, for each database and each of the two tuning strategies (the *first layer* or the *last layer*), we would have 48 tuned models (tune on four other databases for two architectures and three strategies). In total, we have 30 trained models with no tuning and 240 tuned models, which amounts to 270 models in total that need to be evaluated. We evaluate the untuned models on the *test* sets of each of the five databases but tuned models only on those databases that they were trained (originally) and tuned on. As a result, we obtain 630 sets of evaluation scores in total, which we analyze in Section 6.

We have taken a special care to minimize all factors unrelated to the focus of our evaluation, so that we minimize all ambiguities pertained to our experimental results. Therefore, we have fixed all other parameters except for the generalization approaches we are actually evaluating. We have fixed the size of input images to 299×299 for both Xception and EfficientNet neural networks. For each database, we fixed the number of iterations per epoch and we have trained for the same number of iterations and epochs regardless of the strategy for increasing the input variability. We have trained for 20 epochs with an early stopping criteria based on the loss of the validation set. A cross entropy loss was used. The batch size for all training and tuning was fixed to 32. The Adam optimizer was used with learning rate of 0.0002.

Since the databases are significantly unbalanced in terms of the amount of genuine and deepfake data, to ensure that each training input batch contains an equal amount of data from both classes, we have employed an oversampling strategy, specifically, we used *sample_from_datasets()* function implemented in Tensorflow framework. When testing the few-shot learning strategies, we tune a given model (pre-trained on a database A) on a small amount of data from the training set of database B. For tuning, we used the amount of data, randomly and equally drawn from both genuine and fake subsets, that roughly corresponds to 100 seconds of video or to 5 clips of 10 seconds video from each class. We assume it is a reasonable enough amount of data for tuning, given a practical scenario when a deepfake detection system is employed in a new operational environment.

Please note that most of the parameters were fixed to the values used by the pioneering work of the authors of FaceForensics++ database [1], who also suggested to use Xception network for deepfake detection. We have followed their guidelines on training and the parameters to make our results as comparable as possible with the existing work.

5.4 Evaluation protocols

We consider deepfake detection as a binary classification problem and evaluate the ability of detection approaches to distinguish original videos from deepfake videos. All videos in a given dataset were proportionally split into training, validation, and test subsets. For Celeb-DF database, only training and test subset were provided, so the test set was used in place of validation when necessary.

The result of a evaluation is a set of probabilistic scores where the values close to zero correspond to deepfakes and those that are close to one correspond to genuine videos.

Using the scores, for each possible threshold θ, we compute commonly used metrics for evaluation of classification systems: false accept rate (FAR), which is the same as false match rate (FMR), and false reject rate (FRR), which is the same as false non-match rate (FNMR). These rates are generally defined as follows:

$$\begin{aligned} \mathrm{FAR}(\theta) &= \frac{|\{h_{neg} \mid h_{neg} \geq \theta\}|}{|\{h_{neg}\}|} \\ \mathrm{FRR}(\theta) &= \frac{|\{h_{pos} \mid h_{pos} < \theta\}|}{|\{h_{pos}\}|}, \end{aligned} \tag{1}$$

where h_{pos} is a score for original genuine samples and h_{neg} is a score for the tampered samples.

We define the threshold θ_{far} on the validation set to correspond to the FAR value of 10%, which means 10% of fake videos are allowed to be misclassified as genuine. Using this threshold θ_{far} on the scores of the test set will result in test FAR and FRR values. As a single value metric, we can then use the half total error rate (HTER) defined as:

$$\mathrm{HTER}(\theta_{far}) = \frac{FAR_{test} + FRR_{test}}{2} \tag{2}$$

Please note that when computing the threshold θ^A_{far}, we use the validation set of the original database A that was used for training a given model. It means that even when we evaluate this model that was tuned on another database B, we use the same threshold θ^A_{far} to compute FAR, FRR, and HTER values on the test set of database B. With this approach on threshold computation, we are trying to simulate a practical scenario when a deepfake detection system is trained and built by a company that pre-sets all the thresholds and parameters of the system using its internal validation sets and ships the system to the client, which can either use the system *as-is* or may tune it on a small set of samples at its deployment site. But we assume that the client only has access to a small set of samples that would not warrant a large enough set to use for validation or training.

In addition to reporting FAR, FRR, and HTER values for the scores of the test set, we also report the area under the curve (AUC) metric, which is a popular metric for evaluation of classification system and is often used in the deepfake detection literature.

TABLE 2: Trained and evaluated on the same database. When tuned, a few samples were used from another database.

Network	Augm. type	Tune	AUC	FRR	FAR	HTER
EfficientNet	augm-patches	first layer	98.85	6.04	7.17	6.60
		last layer	99.24	3.62	10.63	7.12
		none	99.28	1.25	9.16	5.20
	faces-augm	first layer	98.95	1.51	14.51	8.01
		last layer	99.46	0.02	19.33	9.67
		none	99.40	0.04	10.38	5.21
	faces-only	first layer	99.22	2.16	9.51	5.83
		last layer	99.66	0.00	29.76	14.88
		none	99.56	0.17	9.56	4.87
Xception	augm-patches	first layer	97.77	11.81	9.74	10.77
		last layer	99.03	1.63	7.10	4.37
		none	99.01	2.48	7.41	4.95
	faces-augm	first layer	98.70	3.75	24.48	14.12
		last layer	99.35	0.42	8.48	4.45
		none	99.38	0.65	10.13	5.39
	faces-only	first layer	98.34	6.51	13.19	9.85
		last layer	99.50	0.00	8.37	4.19
		none	99.53	0.00	9.14	4.57

TABLE 3: Different databases were used for training/validation and evaluation.

Network	Augm. type	Tune	AUC	FRR	FAR	HTER
EfficientNet	augm-patches	first layer	71.01	5.62	73.76	39.69
		last layer	70.88	3.40	80.07	41.74
		none	69.53	4.11	81.30	42.71
	faces-augm	first layer	72.83	2.52	78.51	40.51
		last layer	72.02	0.77	83.85	42.31
		none	70.67	2.53	79.76	41.15
	faces-only	first layer	72.65	5.40	78.86	42.13
		last layer	71.27	0.19	92.06	46.13
		none	70.67	1.24	87.73	44.48
Xception	augm-patches	first layer	70.42	7.93	68.34	38.14
		last layer	68.49	2.64	80.68	41.66
		none	68.01	5.03	78.03	41.53
	faces-augm	first layer	72.43	4.98	74.89	39.93
		last layer	68.60	3.40	81.39	42.39
		none	68.09	4.43	78.98	41.71
	faces-only	first layer	74.47	3.13	76.74	39.93
		last layer	70.07	0.99	85.71	43.35
		none	69.37	1.73	81.82	41.77

6 Experimental results

Given that our experiments resulted in essentially 630 sets of evaluation scores, we analyze and present the results aggregated in several tables that demonstrate the trends from different angles. But as an example, Figure 6 shows several receiver operating characteristic curves (ROC) each corresponding to a separate set of scores. The figure shows evaluation of nine Xception models trained using training sets of three databases and using three different strategies for increasing the data variability (*face-only* for faces with no augmentation, *face-augm* for augmented faces, *augm-patches* for the augmented faces and patches farmed from the background). These nine models are evaluated on the test set of Celeb-DF database resulting in nine ROC curves. Figure 6 clearly demonstrates the generalization problem of the deepfake detection, since when a network is trained on the same Celeb-DF database it is evaluated on, the AUC value is equal to 100% and all three corresponding curves are at the very top left corner of the plot. Once the test database changes, the accuracy plummets. However, it can be also noted that background patches together with image augmentation strategies have positive impact on the accuracy, especially for low FAR and FRR values, which are critical in practical scenarios.

Since we have a lot of experimental results, their detailed analysis is not feasible in the scope of this paper, hence, we focus on the general trends about training/tuning strategies and database performances. However, the reader is invited to check the provided open source package[4] that includes all the computed scores and a Jupyter notebook that allows to view ROC curves and statistical data about each training/testing configuration.

6.1 Strategies for improving generalization

More generic trends can be observed by looking at Table 2 and Table 3. The tables present average accuracy values for train-test database pairs but while Table 2 focuses on scores corresponding to when a network was trained and tested on the same database, Table 3 only shows the average metric values for cases when a network was trained (and the FAR@10% threshold was computed) on one database but tested on another.

Table 2 and Table 3 demonstrate several important trends:

- When trained and tested on the same database, both Xception and EfficientNet based systems show consistently high accuracy. It is especially evident when comparing FAR and HTER values in Table 2 with the corresponding values in Table 3. A FAR value higher than 50% renders a system practically useless, and many of the FAR values in cross database scenario are even higher than 80%.
- The best performing system configuration for the same database scenario is to train on faces only and have no tuning or at the most tune the last layer, without data augmentation, leading to the lowest HTER values, as highlighted in green in Table 2. This is true for both types of network models. However, in cross database, these models lead to poor results, indicating that they overfit fast on the training data in the absence of any augmentation.
- The best strategy for improving deepfake detection generalization is to use the data farming strategy with the mix of face augmentation and patches cropped from background for increasing data variability and to tune first layer on the test set of another database. Together, these two strategies lead to the lowest HTER values in cross-database scenario (highlighted in green in Table 3), while keeping the error relatively low for the same database scenario (see values corresponding to 'augm-patches' and 'first layer' in Table 2 highlighted in blue).
- Although 'first layer' tuning is a clear winning strategy when it comes to cross-database evaluations, tuning the last layer is a mixed bag in terms of FAR or HTER results. Sometimes it leads to better performance compared to 'no tuning' strategy and sometimes to worse performance as evident from both Table 2 and Table 3. Hence, we do not recommend to perform this method of tuning in a practical system.

Looking at Table 3, one may note that despite the efforts with aggressive data farming strategy and first layer tuning, FAR values are close to 70%, which means a very high percentage of

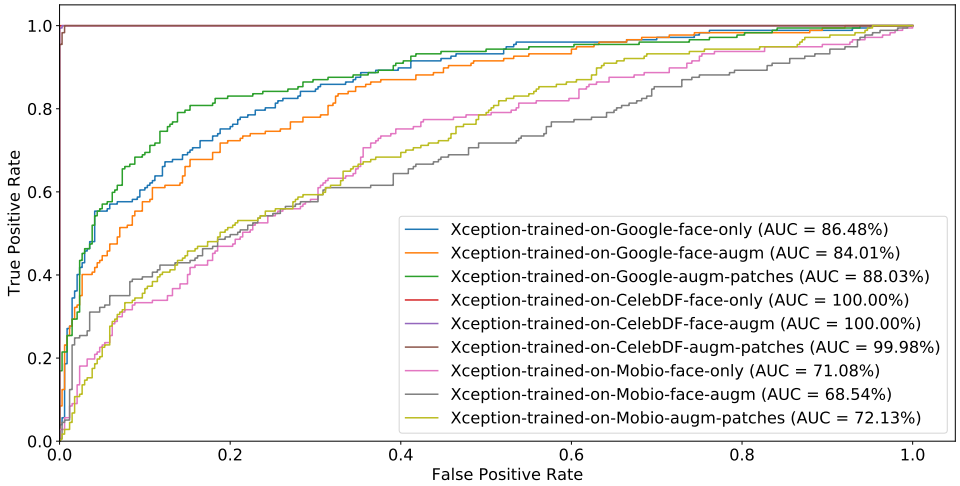


Fig. 6: ROC curves with the corresponding AUC value of Xception model trained on Google, Celeb-DF, and DF-Mobio databases and evaluated on the test set of Celeb-DF.

deepfakes are misclassified as real videos. These results reflect two main issues: (i) the generalization of deepfake detection is far from a solved problem and (ii) the choice of database and overfitting on the training data may be so devastating that no strategy for increasing data variability will solve it. We will look at the second issue in the next section.

6.2 Compare databases

The large amount of experiments and evaluation scores also allow us to take a look at the databases themselves. We compare databases in Table 4 and Table 5. In these tables, we still show the results for different strategies to improve generalization but the final scores are averaged for neural network models and averaged for both same and cross-database scenarios. These tables allow us to understand what we can expect overall when we use a particular database for training or testing.

From both Table 4 and Table 5, we can immediately notice that whether DeeperForensics (denoted as 'DF' in the tables) database was used for testing or to train models, the HTER values are averaging at 50% and FAR at nearly 99%, independent of the training or tuning strategies. It means that the deepfakes in this dataset are so different from every other four databases that none of the trained models can detect these deepfakes and even tuning strategy can bring FAR value down by a mere few percentage points. Hence, it is the most challenging dataset for detection as indicated by the orange highlight in Table 4. But the high error rates in Table 5 also mean that using DeeperForensics for training is not going to help in detecting deepfakes of other four datasets. Note that some of these databases, like FaceForensics++, are very popular in the scientific community. Given that DeeperForensics is a more recent database, it may be that its deepfakes are more realistic and of the next generation of deepfake datasets as per classification proposed by Tolosana *et al.* [40]. In addition to being different from other datasets, DeeperForensics is also on the smaller side among the evaluated, hence the network models may be overfitting more on this dataset.

If we remove DeeperForensics (the reader is invited to experiment with a Jupyter notebook[4]) from the overall results, the error rates drop significantly, with the HTER values at 31% and FAR at 50% for Xception with 'augm-patches' training and 'first layer' tuning from the current 38% and 68% for the same configuration in the cross-database experiments shown in Table 3. The overall trends stay the same but the end result is more hopeful if we evaluate on comparable databases.

The Table 5 demonstrates that the best database to use for training is FaceForensics++ (denoted as 'FF++' in the table). Regardless of the choice of network models or strategies, the error rates corresponding to FaceForensics++ are lower in Table 5 compared to other databases, which means systems trained using this database will generalize better. This finding is especially surprising given that the most challenging DeeperForensics database has the same real videos as FaceForensics++, which again emphasizes how critically a different kid of deepfakes can impact the generalization performance of a detection system. The next best performing database if used for training is DF-Mobio, which suggests that using DF-Mobio in combination with FaceForensics++ may make the detection system generalize even better. However, data and training fusion is out of the scope of this paper and we leave it for the future work.

To take a peek at the generalization power of the considered detection systems, we computed t-distributed stochastic neighbor embedding (t-SNE) plots on the output layer (the one before the final classification layer) of the Xception network model trained using data farming strategy ('augm-patches') on FaceForensics++ database, as the best performing in Table 5, and evaluated on all five databases as shown in Figure 7. The t-SNE plots further illustrate the difficulty of achieving generalization with real and deepfake videos from Celeb-DF and Google being the least separable. Figure 7 also shows that tuning on the first layer even for a small number of samples can improve the separability, at least it is visibly so for Celeb-DF and DF-Mobio databases, of the deepfake and real videos in the dataset. More t-SNE plots can be found in our open source package[4].

TABLE 4: Comparing databases when they are used for testing.

Test DB	Augm. type	Tune	AUC	FRR	FAR	HTER
Celeb-DF	augm-patches	first layer	70.48	6.36	78.79	42.57
		last layer	67.43	2.90	88.12	45.51
		none	67.05	4.45	81.69	43.07
	faces-augm	first layer	72.72	2.97	86.29	44.63
		last layer	67.61	2.68	88.09	45.39
		none	67.50	5.79	77.17	41.48
	faces-only	first layer	72.85	1.06	97.21	49.13
		last layer	70.11	0.64	97.61	49.12
		none	70.05	2.47	87.61	45.04
DF	augm-patches	first layer	63.48	0.50	96.33	48.41
		last layer	60.30	0.00	99.72	49.86
		none	60.17	0.00	99.66	49.83
	faces-augm	first layer	70.16	0.31	97.85	49.08
		last layer	66.83	0.12	99.91	50.02
		none	66.45	0.19	99.63	49.91
	faces-only	first layer	70.61	1.43	94.68	48.06
		last layer	68.66	0.12	99.88	50.00
		none	68.45	0.12	99.91	50.02
Google	augm-patches	first layer	76.45	12.82	49.65	31.24
		last layer	76.06	6.36	66.14	36.25
		none	73.18	6.25	69.37	37.81
	faces-augm	first layer	77.49	3.12	65.01	34.07
		last layer	73.91	1.62	72.73	37.17
		none	70.96	2.05	72.86	37.45
	faces-only	first layer	81.27	5.93	58.79	32.36
		last layer	74.38	1.08	79.36	40.22
		none	72.15	0.65	79.46	40.05
FF++	augm-patches	first layer	58.57	8.30	82.65	45.48
		last layer	59.71	0.71	92.32	46.52
		none	59.08	1.43	90.83	46.13
	faces-augm	first layer	59.55	9.20	83.01	46.10
		last layer	61.01	4.29	89.23	46.76
		none	60.25	4.55	87.65	46.10
	faces-only	first layer	58.42	7.50	85.83	46.67
		last layer	58.61	0.27	96.43	48.35
		none	58.51	2.14	92.44	47.29
DF-Mobio	augm-patches	first layer	84.60	5.89	47.84	26.87
		last layer	84.93	5.14	55.56	30.35
		none	84.36	10.72	56.77	33.74
	faces-augm	first layer	83.22	3.15	51.34	27.24
		last layer	82.18	1.73	63.14	32.44
		none	81.76	4.84	59.55	32.19
	faces-only	first layer	84.64	5.41	52.48	28.95
		last layer	81.60	0.85	71.14	36.00
		none	80.92	2.04	64.44	33.24

TABLE 5: Comparing databases when they are used for training.

Train DB	Augm. type	Tune	AUC	FRR	FAR	HTER
Celeb-DF	augm-patches	first layer	75.96	6.11	69.09	37.60
		last layer	77.77	0.11	91.34	45.73
		none	76.52	0.09	93.55	46.82
	faces-augm	first layer	77.73	3.42	76.34	39.88
		last layer	76.96	0.18	94.01	47.09
		none	75.14	0.18	95.42	47.80
	faces-only	first layer	75.67	6.32	70.63	38.47
		last layer	75.83	0.09	97.50	48.80
		none	74.23	0.09	98.12	49.11
DF	augm-patches	first layer	53.27	2.83	96.94	49.88
		last layer	52.22	0.07	99.88	49.97
		none	50.08	0.07	99.92	49.99
	faces-augm	first layer	54.71	0.24	99.58	49.91
		last layer	48.20	0.22	99.92	50.07
		none	46.68	0.12	99.91	50.02
	faces-only	first layer	56.87	11.46	85.38	48.42
		last layer	48.39	0.29	99.60	49.94
		none	47.38	0.72	99.48	50.10
Google	augm-patches	first layer	77.10	4.03	71.98	38.00
		last layer	74.94	2.82	73.10	37.96
		none	74.56	8.92	65.95	37.44
	faces-augm	first layer	79.74	5.42	71.76	38.59
		last layer	78.96	3.08	70.77	36.92
		none	78.56	8.21	57.41	32.81
	faces-only	first layer	81.19	0.84	80.87	40.85
		last layer	78.67	0.22	83.27	41.74
		none	78.83	3.92	65.78	34.85
FF++	augm-patches	first layer	81.99	12.95	51.13	32.04
		last layer	79.74	10.41	56.87	33.64
		none	79.91	12.63	56.18	34.40
	faces-augm	first layer	83.27	2.89	61.53	32.21
		last layer	81.96	2.83	66.39	34.61
		none	81.58	3.69	65.36	34.53
	faces-only	first layer	86.91	0.81	68.28	34.55
		last layer	86.04	1.67	73.01	37.34
		none	85.60	1.62	71.99	36.80
DF-Mobio	augm-patches	first layer	65.26	7.95	66.13	37.04
		last layer	63.76	1.69	80.68	41.18
		none	62.78	1.14	82.72	41.93
	faces-augm	first layer	67.71	6.77	74.28	40.53
		last layer	65.47	4.13	82.00	43.06
		none	64.96	5.21	78.75	41.98
	faces-only	first layer	67.14	1.89	83.84	42.87
		last layer	64.44	0.69	91.03	45.86
		none	64.05	1.08	88.49	44.78

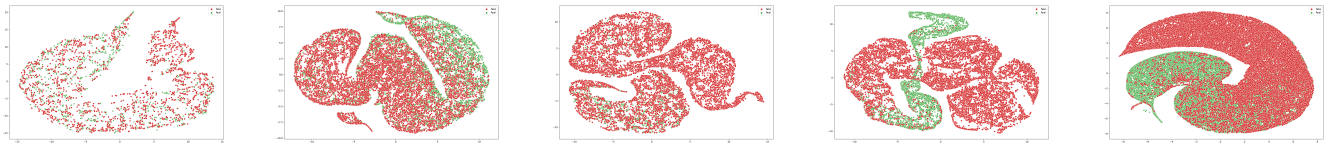
7 Conclusion

The evaluation of the state of the art deepfake detection algorithms in the cross database scenario, using Google, FaceForensics++, DeeperForensics, Celeb-DF, and DF-Mobio databases, demonstrated that the algorithms perform very well when trained and evaluated on the same database but struggle significantly when trained on one database and evaluated on another. The percentage of deepfake videos misclassified as real can go from a desirable 7% on average in a single database scenario to well above 80% in the cross-database scenario. It means the deepfake detection algorithms are struggling to generalize to unseen and unknown data and special care needs to be taken to improve their overall performance.

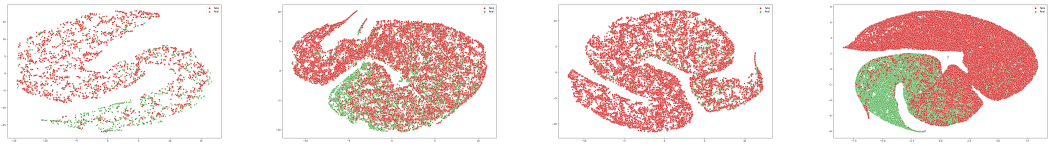
Thus, we have investigated two major strategies for improving the generalization: (i) increase variability of the input data and (ii) tune the pre-trained model on the few samples from the unseen dataset. The extensive experiments on five large publicly available datasets demonstrated that both a drastic strategy for increasing the data variability and tuning, specifically using the first layer of the model, can significantly improve the generalization in cross-database scenario, while suffering a mild set back when evaluated on the same database which it was originally trained on. The experiments also demonstrated that not all databases of deepfakes are made equal in terms of how useful they are for training or how challenging they are when tested on. Notably, DeeperForensics dataset stood out as having the most challenging deepfakes which systems that were trained on any other dataset struggled to detect.

Acknowledgements

This work was funded by Hasler Foundation's VERIFAKE project and Swiss Center for Biometrics Research and Testing.



(a) Celeb-DF (b) DeeperForensics (c) Google (d) FaceForensics++ (e) DF-Mobio



(f) tuned, Celeb-DF (g) tuned, DeeperForensics (h) tuned, Google (i) tuned, DF-Mobio

Fig. 7: t-SNE plots for Xception model trained with data farming strategy on training set of FaceForensics++ in the top row; and when its first layer is tuned on the same database it is tested on. Real videos are in green color and deepfakes are in red.

REFERENCES

[1] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics++: Learning to detect manipulated facial images," in *International Conference on Computer Vision (ICCV)*, 2019.

[2] D. M. Montserrat, H. Hao, S. K. Yarlagadda, S. Baireddy, R. Shao, J. Horvath, E. Bartusiak, J. Yang, D. Güera, F. Zhu, and E. J. Delp, "Deepfakes detection with automatic face weighting," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 2851–2859.

[3] P. Korshunov and S. Marcel, "Vulnerability assessment and detection of Deepfake videos," in *International Conference on Biometrics (ICB 2019)*, Crete, Greece, Jun. 2019.

[4] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics: A large-scale video dataset for forgery detection in human faces," *arXiv.org*, 2018. [Online]. Available: http://arxiv.org/abs/1803.09179

[5] L. Jiang, W. Wu, R. Li, C. Qian, and C. C. Loy, "Deeperforensics-1.0: A large-scale dataset for real-world face forgery detection," *arXiv preprint*, 2020. [Online]. Available: http://arxiv.org/abs/2001.03024

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

[7] B. Dolhansky, J. Bitton, B. Pflaum, J. L. andRuss Howes, M. Wang, and C. C. Ferrer, "The deepfake detection challenge dataset," *arXiv preprint*, 2020. [Online]. Available: https://arxiv.org/abs/2006.07397

[8] H. Nguyen, J. Yamagishi, and I. Echizen, "Capsule-forensics: using Capsule networks to detect forged images and videos," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[9] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "Cnn-generated images are surprisingly easy to spot...for now," in *CVPR*, 2020.

[10] Y. Li, M.-C. Chang, and S. Lyu, "In ictu oculi: Exposing ai generated fake face videos by detecting eye blinking," *arXiv.org*, 2018. [Online]. Available: http://arxiv.org/abs/1806.02877

[11] X. Yang, Y. Li, H. Qi, and S. Lyu, "Exposing GAN-synthesized faces using landmark locations," in *ACM Workshop on Information Hiding and Multimedia Security*, June 2019, pp. 113–118.

[12] X. Yang, Y. Li, and S. Lyu, "Exposing deep fakes using inconsistent head pose," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[13] S. Agarwal, T. El-Gaaly, H. Farid, and S. Lim, "Detecting deep-fake videos from appearance and behavior," *arXiv preprint arXiv:2004.14491*, 2020.

[14] Y. Zhang, L. Zheng, and V. L. L. Thing, "Automated face swapping and its detection," in *IEEE International Conference on Signal and Image Processing (ICSIP)*, Aug 2017, pp. 15–19.

[15] A. Agarwal, R. Singh, M. Vatsa, and A. Noore, "Swapped! digital face presentation attack detection via weighted local magnitude pattern," in *IEEE International Joint Conference on Biometrics (IJCB)*, Oct 2017, pp. 659–665.

[16] R. Mama and S. Shi, "Towards deepfake detection that actually works," Dessa, Nov. 2019. [Online]. Available: https://www.dessa.com/post/deepfake-detection-that-actually-works

[17] N. Hulzebosch, S. Ibrahimi, and M. Worring, "Detecting cnn-generated facial images in real-world scenarios," *arXiv preprint*, 2020. [Online]. Available: https://arxiv.org/abs/2005.05632

[18] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800–1807.

[19] C. McCool, S. Marcel, A. Hadid, M. Pietikäinen, P. Matejka, J. Cernocký, N. Poh, J. Kittler, A. Larcher, C. Lévy, D. Matrouf, J. Bonastre, P. Tresadern, and T. Cootes, "Bi-modal person recognition on a mobile phone: Using mobile phone data," in *2012 IEEE International Conference on Multimedia and Expo Workshops*, 2012, pp. 635–640.

[20] D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur, and S. K. Nayar, "Face swapping: Automatically replacing faces in photographs," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 39:1–39:8, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360638

[21] N. M. Arar, N. K. Bekmezci, F. Gney, and H. K. Ekenel, "Real-time face swapping in video sequences: Magic mirror," in *IEEE Signal Processing and Communications Applications Conference (SIU)*, April 2011, pp. 825–828.

[22] Z. Xingjie, J. Song, and J. Park, "The image blending method for face swapping," in *IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, Sept 2014, pp. 95–98.

[23] T. M. den Uyl, H. E. Tasli, P. Ivan, and M. Snijdewind, "Who do you want to be? real-time face swap," in *IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, May 2015.

[24] S. Mahajan, L. Chen, and T. Tsai, "SwapItUp: A face swap application for privacy protection," in *IEEE International Conference on Advanced Information Networking and Applications (AINA)*, March 2017, pp. 46–50.

[25] Y. Nirkin, I. Masi, A. T. Tuan, T. Hassner, and G. Medioni, "On face segmentation, face swapping, and face perception," in *IEEE International Conference on Automatic Face Gesture Recognition (FG)*, May 2018, pp. 98–105.

[26] I. Korshunova, W. Shi, J. Dambre, and L. Theis, "Fast face-swap using convolutional neural networks," in *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 3697–3705.

[27] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing Obama: Learning lip sync from audio," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 95:1–95:13, Jul. 2017. [Online]. Available: http://doi.acm.org/10.1145/3072959.3073640

[28] D. Stanton, P. Karlsson, A. Vorobyov, T. Leung, J. Childs, A. Sud, and C. Bregler, "Contributing data to deepfake detection research," in *Blogpost*, 2019.

[29] X. Xuan, B. Peng, W. Wang, and J. Dong, "On the generalization of gan image forensics," in *Biometric Recognition*, Z. Sun, R. He, J. Feng, S. Shan, and Z. Guo, Eds. Cham: Springer International Publishing, 2019, pp. 134–141.

[30] D. Cozzolino, J. Thies, A. Rössler, C. Riess, and M. Nießner.

[31] N. Hulzebosch, S. Ibrahimi, and M. Worring, "Detecting cnn-generated facial images in real-world scenarios," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[32] R. Tolosana, S. Romero-Tapiador, J. Fierrez, and R. Vera-Rodriguez, "Deepfakes evolution: Analysis of facial regions and fake detection performance," *arXiv preprint*, 2020. [Online]. Available: https://arxiv.org/abs/2004.07532

[33] L. Verdoliva, "Media forensics and deepfakes: An overview," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 910–932, 2020.

[34] Y. Mirsky and W. Lee, "The creation and detection of deepfakes: A survey," *ACM Comput. Surv.*, vol. 54, no. 1, Jan. 2021. [Online]. Available: https://doi.org/10.1145/3425780

[35] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia, "Deepfakes and beyond: A survey of face manipulation and fake detection," *Information Fusion*, vol. 64, pp. 131–148, 2020.

[36] S. Aneja and M. Niener, "Generalized zero and few-shot transfer for facial forgery detection," *arXiv preprint*, 2020. [Online]. Available: https://arxiv.org/abs/2006.11863

[37] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019. [Online]. Available: http://arxiv.org/abs/1905.11946

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[39] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, Jun. 2020. [Online]. Available: https://doi.org/10.1145/3386252

[40] R. Tolosana, S. Romero-Tapiador, J. Fierrez, and R. Vera-Rodriguez, "Deepfakes evolution: Analysis of facial regions and fake detection performance," in *ICPR International Workshops and Challenges*, Feb. 2021.