Autoencoder-based Data Augmentation for Deepfake Detection

Dan-Cristian Stanciu
AI Multimedia Lab
Politehnica University of Bucharest, Romania
dan.stanciu1203@upb.ro

Bogdan Ionescu
AI Multimedia Lab
Politehnica University of Bucharest, Romania
bogdan.ionescu@upb.ro

ABSTRACT

Image generation has seen huge leaps in the last few years. Less than 10 years ago we could not generate accurate images using deep learning at all, and now it is almost impossible for the average person to distinguish a real image from a generated one. In spite of the fact that image generation has some amazing use cases, it can also be used with ill intent. As an example, deepfakes have become more and more indistinguishable from real pictures and that poses a real threat to society. It is important for us to be vigilant and active against deepfakes, to ensure that the false information spread is kept under control. In this context, the need for good deepfake detectors feels more and more urgent. There is a constant battle between deepfake generators and deepfake detection algorithms, each one evolving at a rapid pace. But, there is a big problem with deepfake detectors: they can only be trained on so many data points and images generated by specific architectures. Therefore, while we can detect deepfakes on certain datasets with near 100% accuracy, it is sometimes very hard to generalize and catch all real-world instances. Our proposed solution is a way to augment deepfake detection datasets using deep learning architectures, such as Autoencoders or U-Net. We show that augmenting deepfake detection datasets using deep learning improves generalization to other datasets. We test our algorithm using multiple architectures, with experimental validation being carried out on state-of-the-art datasets like CelebDF and DFDC Preview. The framework we propose can give flexibility to any model, helping to generalize to unseen datasets and manipulations.

CCS CONCEPTS

• **General and reference** → **General conference proceedings**; • **Computing methodologies** → **Neural networks**; *Computer vision*; *Supervised learning by classification.*

KEYWORDS

deepfake, deep learning, digital video forensics, face manipulation, data augmentation, generalization, autoencoder

ACM Reference Format:
Dan-Cristian Stanciu and Bogdan Ionescu. 2023. Autoencoder-based Data Augmentation for Deepfake Detection. In *2nd ACM International Workshop on Multimedia AI against Disinformation (MAD '23), June 12, 2023, Thessaloniki, Greece.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3592572.3592840

MAD '23, June 12, 2023, Thessaloniki, Greece

ACM ISBN 979-8-4007-0187-0/23/06…$15.00
https://doi.org/10.1145/3592572.3592840

Figure 1: Four frames from the dataset CelebDF[23], from the same moment in time, containing one real person (upper left) and 3 deepfakes, resulted from changing the identity of the person.

1 INTRODUCTION

In the era of technology, we can find information everywhere, as we have access to almost every bit of data we desire, using the internet. And while that is a good thing, we must not forget that not all the data we see is curated to make sure it is accurate, real or it does not cause any harm. In fact, while we can have a few credible sources of information, the majority of information is posted without any kind of verification of approval process. Therefore, it is only natural that some of the information will eventually turn out to be fake - by mistake or not. While the fight against fake information is ongoing for a long time, there is a new kind of falsified information that we must fight against: Deepfakes.

Deepfakes are images and videos, usually portraying people, that have been manipulated in some way, using deep neural networks. The introduction of Generative Adversarial Networks (GAN) [16]

has sparked the evolution in image and video generation. GAN generators can create deepfakes by using a loss function based on an associated discriminator, that is specialised in detecting if a generated image is real or not. Therefore, images from the generator need to be extremely realistic to pass this test. Other very popular deepfake generation methods require switching identities using two autoencoders: one encoder which learns to represent a first class, and one for the second class. The creation of a deepfake involves using the encoder for the first class and the decoder for the second. These ideas inspired the creation of some open-source deepfake generators, such as FaceApp [4] or FaceSwap-GAN [2].

At this time, generated images are so hyper-realistic that people can no longer distinguish them with ease. As an example, as Figure 1 presents, we can compare 3 deepfake frames in which the identity has been changed to a real image (upper left). If is almost impossible to see which frame is authentic. Because of this, we can no longer rely on a human's attention to detect a deepfake.

The detection of deepfakes has been an increasingly important subject in the last few years. Several deepfake detection methods have been introduced, creating a great variety of solutions, such as: finding localized artifacts in images, finding inconsistencies in frequency or in physiological signals, deep convolutional approaches, detecting temporal changes using time domain deep learning such as LSTM [36] or Transformers [39] etc. While these methods are capable of an almost perfect differentiation of deepfakes versus real images, they all suffer from a common problem: it is extremely hard to generalize. Therefore, while many deepfake detection algorithms succeed on certain datasets, they do not yield good results when evaluating on samples coming from other distributions.

In this paper, we will be presenting a way to augment the training dataset to increase generalization: using autoencoders to replicate the image with a slight error. This way, we can add some noise or frequency components to the image while maintaining its quality. Furthermore, by doing this, we hope that the models would not be learning to identify the deepfake generator's "fingerprint" patterns, as they would be changed by running the image through the autoencoder models and therefore changing the image's signature.

The remainder of this article is structured as following: Section 2 contains an overview of deepfake detection methods in the context of generalization, Section 3 describes our augmentation approach and the models used, Section 4 contains our training methodology, architectures, attempts at data augmentation, results on multiple datasets and a comparison with the state-of-the-art models. We present our conclusions in Section 5.

2 RELATED WORK

The idea of multimedia manipulation is not a new thing. In 1860, one the first documented instances of photo manipulation took place: the head of president Abraham Lincoln was manipulated to appear on the body of politician John Calhoun. Over 100 years later, with the introduction of digital photography and photo editing tools like Adobe Photoshop, manipulating photos and faking content became incredibly easy and accessible. While "photoshopped" images are sometimes hard to discern from real ones, there are some techniques like Error Level Analysis [41] that can detect objects inserted into an image.

In the past, realistic media generation has been a challenge. Now, with the help of deep learning, convolutional neural networks and some state-of-the-art architectures like Generative Adversarial Networks, this challenge could be considered solved. Neural networks are not only capable of generating realistic images and videos, but also to determine if the image looks realistic enough or not. Perhaps the most popular example of the realism in deepfakes is the video of former President Barack Obama, created by Suwajanakorn *et al* [34], in a paper which demonstrates the generation of mouth movement in video frames to perfectly sync with an audio file.

Deepfake detection algorithms are essential to prevent the spread of disinformation or blackmail. The state-of-the-art deepfake detection algorithms are very diverse, each focusing on another aspect of this complicated problem. Furthermore, there are many open-source datasets that aim to teach the models to detect deepfakes in the worst and most diverse conditions. Lastly, perhaps the most important thing is generalization: it is very important to explain how we can make our models generalize better, against all kinds of conditions.

The following topics are presented in this section: (i) the deepfake detection datasets in the context of generalization, (ii) the state-of-the-art deepfake detection approaches, (iii) approaches to improve generalization.

2.1 Deepfake detection datasets

Today, deepfakes are more unrecognisable than ever before, and most humans have a hard time differentiating between real and fake videos. Because of that, the interest in detecting them has grown significantly. There is an increasing need of data depicting deepfakes, but as more and more deepfake generators are present, the data suddenly is not the most important thing. Instead, the algorithms are at least as important, as they can prevent overfitting the data.

There are multiple deepfake detection datasets, the majority of which being open-source and depicting real videos as well as falsified content. They usually involve people, have a short length of under 1 minute and depict deepfakes generated by identity change (using one person's face on another person's body), replicating the facial expressions, movements, skin tones of the original frames. The most commonly used datasets for deepfake detection are FaceForensics++ [31], CelebDF [23] and DFDC (from Facebook's Deepfake Detection Challenge) [13] and its preview, DFDC Preview [14].

There are 3 generations of deepfake detection datasets, categorised by the realism of the deepfakes, number of videos, quality of videos, number of different people in the videos, whether the people participating were actors or non-consenting subjects or whether the videos were staged or in the wild. The 1st generation of deepfake datasets contain a small number of videos, of low quality, depicting non-consenting subjects and uses fairly weak deepfake creation algorithms. The 2nd generation of datasets provides an improvement in the realism of the deepfakes, size of a few thousand videos and, sometimes, consenting actors. Lastly, the 3rd generation of datasets brings huge datasets, with over 100,000 videos, a bigger diversity in the deepfake generation methods and consenting actors, sometimes filmed in the wild. For the purposes of this paper, we will focus on FaceForensics++ [31], a 1st generation dataset that

contains 1000 real videos from YouTube and 4000 falsified videos, created with 4 distinct algorithms: 1000 deep learning generated identity swap videos [3] and 1000 videos generated with a public FaceSwap software [1], 1000 videos generated with Face2Face [38] and 1000 videos generated with Neural Textures [37]. We will also evaluate our models on datasets like (i) CelebDF [23], a 2nd generation dataset, containing 890 real videos and 5639 deepfake videos obtained from identity-swapping between 59 celebrities, with 10 different video scenarios for each celebrity and (ii) DFDC Preview [13], a challenging dataset with a variety of scenarios.

2.2 Deepfake Detection Approaches

Deepfake detection methods are very diverse, with each one using novel ways to highlight images and videos generated by neural networks. They can, however, be categorized into the following:

- Methods focusing on convolutional networks, like [10, 20, 21, 31]. Many of these approaches do not employ a time-level algorithm and focus on images.
- Methods focusing on low-level features. The most important feature may be frequency [15, 24, 25]. It has been proven that deepfake generators usually leave some kind of frequency "fingerprint" [26, 42] on the images, and they can be leveraged from a frequency spectrum. This detail may be important for this paper, as we try to alter those fingerprints using recreations of the image.
- Methods using image information, as well as a temporal network to leverage inconsistencies in time. Papers like [11, 17, 44] use 3DCNN, LSTM or Transformers, as they can look at video-level artifacts. Those approaches usually generalize best, because deepfakes are most often generated at image-level, and therefore are not always consistent in time.

2.3 Generalization in deepfake detection

The ultimate goal for deepfake detection algorithms is for them to be used in the wild, with a reasonable accuracy. The problem is that many of the algorithms mentioned above can learn to detect deepfakes from one dataset very well, but fail to detect them on other datasets. Therefore, the focus of the deepfake detection community recently has been the models' ability to generalize to unseen manipulations.

As many of the current state-of-the-art detectors fail when confronted with unseen manipulations, it is a good idea to look at some of the ones that perform better.

One of the most recent and best performers when it comes to generalization is LipForensics [18]. The model architecture starts from a pretrained lip reading neural network for a feature extractor and temporal network, also trained on lip reading. LipForensics looks at only the lips in a deepfake manipulation, as if was originally trained to read lips. In spite of that, using temporal models to detect inconsistencies in the lip movements is one of the best methods to detect deepfakes. LipForensics is also very good at dealing with many unseen corruptions, like Saturation, Contrast, Noise, Compression etc.

Another approach that generalizes is Face X-ray [22]. It is based on the presumption that every generated face is blended on the original face, therefore the image must must contain blending artifacts. Face X-ray reveals the boundaries between the original image and the pasted blended image.

In [32] we also see an approach that focuses on the image blending: by repeating the blending process, you can recreate statistical inconsistencies or color artifacts. Basically, it simulates the image blending in all situations. This approach also has a very high level of generalization, as it is based on the weakest point of the current deepfake detection datasets: the blending.

A method that explores the temporal coherence is presented in [44]. By using a 3DCNN architecture for groups of frames, along with a temporal Transformer, this paper achieves a very high generalization on multiple datasets. The reason for this is that deepfakes are not usually generated with time consistency in mind. More than that, even when they are, deepfake generators have a very hard time maintaining that image consistency across hundreds or thousands of frames.

As a conclusion, the methods that generalize most do so by using either the temporal domain artefacts and inconsistencies or by exploiting the artefacts resulting from the one common element that most deepfakes share: the blending of the generated image.

3 PROPOSED METHOD

In this section, we present our approach aimed at increasing generalization: augmenting the training datasets with images generated by deep learning, like Autoencoders or U-Net. We will present our augmentation pipeline, architectures used and a few reasons why this method could improve deepfake detection algorithms.

The motivation for this paper started from papers like [26, 42], that analyse generated images from the frequency standpoint. Here, we can see that certain operations applied on images, like upsampling, leave some kind signature in the frequency spectrum. More than that, it has been shown that certain deep neural network architectures like GANs can leave some frequency fingerprints that are visible on the frequency spectrum of the image.

One of the biggest problems with some of the current deepfake detectors is that they learn certain representations from the training data, but fail to generalize to deepfakes generated by other models. It is very reasonable to conclude that they must learn certain "signatures" of the training data that does not apply in other kinds of deepfakes. Therefore, it would be beneficial if we could somehow hide these signature patterns in the images, so that the deepfake detector models could learn from other sources, in the hope that they would generalize better. The data augmentation technique presented in this paper aims to achieve exactly that.

The proposed idea of this work is trying to augment the dataset to try to eliminate model-specific patterns from deepfakes. We do that by augmenting the dataset with the help of a multitude of other other image generators. The biggest question here is: how can we generate new realistic deepfake pictures and be sure that they are realistic enough, while not actually using new data? Our idea is using deep neural networks like Autoencoders or U-Net to recreate the images in the train dataset. This way, if the autoencoders are good enough, they will create an image that is indistinguishable from the original, but the patterns in that image will be influenced by the

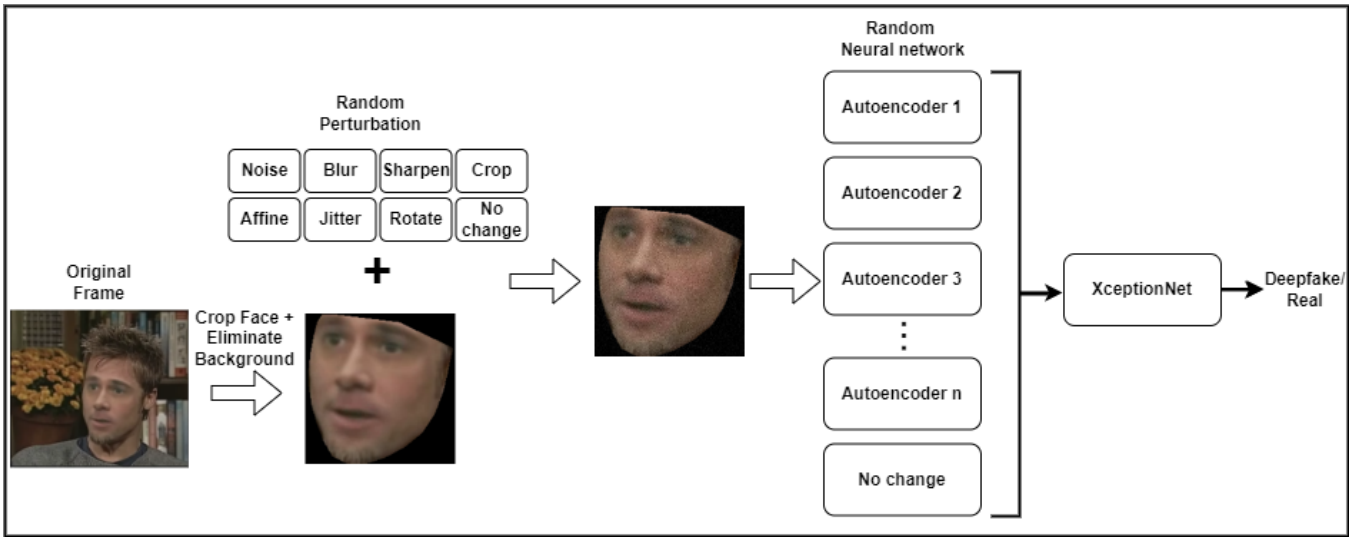

Figure 2: Our proposed training algorithm with autoencoder-based augmentation. It is composed of the following elements: (i) FaceForensics++ frames as an input, (ii) Cropping the facial region and eliminating the background using facial landmarks from OpenFace2 [8], (iii) Adding a random perturbation to the image, with each one having an equal chance of applying, including no change in the frame, (iv) Passing the image through a pretrained autoencoder network from a multitude of pretrained models (Autoencoders, U-Net), including an option to skip this step, (v) Passing the resulted image through a deepfake detection model, in this case XceptionNet, to train it, (vi) the model trains for multiple epochs on all the augmented training data and is able to output a decision

autoencoder's architecture. Therefore, the original image's fingerprint may be changed or altered. The idea is that the autoencoders should not recreate the image perfectly. Some error is expected, but it should not be too large, as they should not generate unrealistic images. The errors might actually benefit the learning process, similar to classic data augmentation like adding noise, blurring etc. More than that, the addition of new deep learning fingerprints to the image can help with generalization. In order to make sure that the deepfake detector network would not just learn the new image fingerprint from the autoencoder, we decided to use multiple models to augment the data. This way, if there are enough models, there would be enough variety in the images. Lastly, we want to make sure that we would also augment the training dataset the normal way, so we add noise, color, perspective changes, rotation, blur, sharpening to the images randomly.

Figure 2 presents our proposed data augmentation and training algorithm. It consists of the following elements:

- The training dataset in FaceForensics++. Random frames from each video are selected and used in training. To ensure that the neural network focuses on the face, we eliminate the background and crop the facial region using facial coordinates from the open-source software OpenFace2 [8]. The frames are resized to $3 \times 299 \times 299$.
- A random perturbation is added to the image to augment the training dataset. The possible perturbations are: Random noise, Blur, Sharpening, Random Crop, Random Affine Transform, Color Jitter (changes in Hue, Saturation, Brightness), Random Rotation. It is also possible for the photo to go through this step without being affected by a random perturbation. We tested adding these perturbations both before and after passing the photo through the autoencoders and the results were similar.
- A randomly selected pretrained autoencoder. These neural networks were trained to output the same image that was used as an input, but with a few small differences. The autoencoders were trained using random images from DFDC [13] (the complete dataset, not the Preview Dataset). They get an image as an input and are trained to output the same image, but with a Average Mean Absolute Error greater than 3%. This means that the output image would always be slightly different from the input, adding a new layer of noise. There are a lot of architectures used for the autoencoders and they will be described in detail in the next chapter. The architectures are a mix of convolutional autoencoders and U-Net [30] architectures. We use wide autoencoders and U-Net because it is our interest that the models output a similar image as the input without struggling to do so. Because U-Net architectures output the same image as the input, they can also be considered to be autoencoder architectures for this task. It is also possible for the image to skip this step, randomly.
- A convolutional neural network is trained for the deepfake detection algorithm. We selected 3 architectures for this test: XceptionNet [9], Capsule Networks [27, 28, 33] and EfficientNet B4 [35].

The benefits of adding the random autoencoding step are:

- Because the autoencoders are not perfect and the output images are at least 3% different, this step is in itself a noise source that augments the dataset. More than that, colors, shapes or even pixel structures can vary as a result of using these autoencoders.
- Because we use multiple autoencoder architectures, there will be a lot of variation in the outputs. Some architectures are very big and will output images that are not very changed, while others are small and will sometimes struggle to output a similar frame.
- Each autoencoder will leave a different "fingerprint" on the image, due to the diversity in architecture, size, training paradigm. Therefore, any deepfake fingerprints from the training dataset will be altered in a different way, resulting in the model struggling to learn from the fingerprint alone and overfit, as they will be very diverse.
- Although there can be some perturbations that can be seen with the naked eye in some of the worst-performing models, these can help the deepfake detectors learn as well.
- The real training data is also passed through these autoencoders. Therefore, the real data will also exhibit some fingerprint specific to those architectures, similar to the deepfakes. Deepfake detectors will not be able to learn the autoencoder-specific patterns because the patterns will also be present in the real data. More than that, the real data is usually from the internet, like Youtube. Therefore, it has been compressed in some way and it has some compression artifacts, which the detector can learn. This does not help with overfitting, as different compression algorithms will affect the image differently.

A potential weakness of this method is that it is frame-level and does not consider the temporal dimension. More on that will be elaborated in the following chapter.

4 EXPERIMENTAL RESULTS

This chapter contains details regarding the experimental results. We will outline the performance improvements of our algorithm. comparing it to the state of the art. More than that, we will draw conclusions regarding the effectiveness of the augmentation algorithm and also its weaknesses.

4.1 Datasets and evaluation

For training, we used FaceForensics++ [31]. This is a dataset containing 1000 real videos and 4000 fake videos, generated using 4 different generation algorithms: Face2Face (F2F), FaceSwap (FS), Deepfakes (DF) and NeuralTextures (NT). We will be using the slightly compressed version of this dataset (HQ). This dataset is one of the most easy in the state of the art, containing many videos that are very distinguishable as being deepfakes.

Contrasting to that, we will be evaluating on datasets that have videos that are much harder to identify for humans: CelebDF - a dataset with videos generated from Youtube clips of actors and DFDC Preview - a preview dataset for a competition that has a lot of harder videos, compression, different angles, noise or resolutions. We follow the train-test splits for the evaluation and training datasets.

For evaluation, we use the AUC-ROC score, as it is a binary classification problem. AUC measures how well the model can differentiate between real and fake samples, at different thresholds. The evaluation for one video clip will be made by averaging the outputs for all its frames.

We also implemented the same algorithms, but without the random autoencoder step, for comparison.

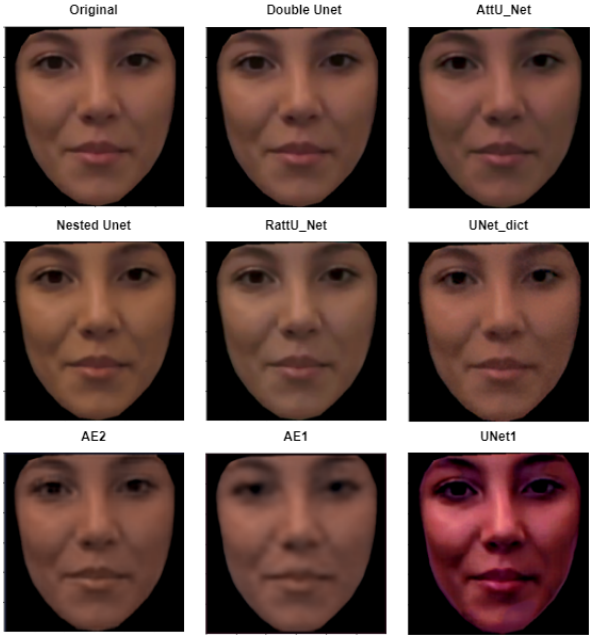

Figure 3: Frames produced by the Autoencoder and UNet models. The original frame is upper left.

4.2 Autoencoder architectures

We trained several autoencoder/U-Net architectures with the aim to augment the training dataset. Below are some details regarding every state-of-the-art architecture used:

- AE1 - basic autoencoder architecture. We used multiple variations with different convolution kernels(3x3, 5x5, 9x9), upsampling techniques (ConvTranspose2d, Bilinear upsampling, Bicubic upsampling, Nearest upsampling), and different number of image features for encoding ($1024 \times 9 \times 9$ features = less than 4 times the number of features than the original frame ($299 \times 299 \times 3$), or $1024 \times 18 \times 18$ features = more features than the original image). Absolute difference between input or output can be up to 20%.
- AE2 - basic autoencoder, using a smaller number of upsampling / downsampling layers and less filters. We used multiple variations with different convolution kernels(3x3, 5x5, 9x9), upsampling techniques (ConvTranspose2d, Bilinear upsampling, Bicubic upsampling, Nearest upsampling), and different number of image features for encoding ($512 \times 9 \times 9$ features = less than 7 times the number of features than the original frame ($299 \times 299 \times 3$), or $256 \times 18 \times 18$ features

= 3 times less features than the original image). Absolute difference between input or output can be up to 25%.
- UNet1 - basic U-Net architecture, inspired from the original paper [30]. Can use Transpose Convolutions, Bilinear upsampling or Bicubic upsampling. Can use different size convolution kernels and a variable number of layers. All UNet architectures below are varied similar to this one.
- Double Unet - double UNet architecture [6, 19], inspired from the original architecture. We use both outputs from the U-net in data augmentation.
- UNet2 - basic U-Net architecture, implemented from [6].
- R2U_Net - Recurrent Residual Convolutional UNet [6, 7] architecture.
- AttU_Net - Attention UNet [6, 29] architecture.
- R2AttU_Net - Residual Recurrent Block with attention Unet, implemented in [5, 6].
- NestedUNet - Unet++ implementation [6, 45].
- DictUNet - Easy to run UNet implementation with Python dictionaries [6].

We trained the autoencoders in epochs of 2000 random frames from different videos from the full DFDC [13] dataset, preprocessed as explained above (face cropping, background elimination, resizing to $3 \times 299 \times 299$). We picked the epoch where the L1 loss was the smallest but over 3% and at least another epoch with bigger loss. That way, we have models that output almost the same image and models that output a slightly changed version. The autoencoders were also trained with 2 different loss functions: L1 loss and MSE (Mean Square Error) loss.

Figure 3 presents the difference between autoencoder/UNet outputs. Some of the outputs are very high quality (top row) and are basically indistinguishable from the real image. Others differ in some small way (middle row), like some color difference in the skin tone or some differences in the edges of the face. The last row contains models that produced some of the biggest differences from the original. Those can be seen as artifacts, blurry images or a total color shift.

In spite of the differences, all the images are somewhat credible and can be used in data augmentation.

4.3 Training, data augmentation and hyperparameters

We trained 3 deepfake detection models: XceptionNet, Capsule Networks and EfficientNetB4. All the models were pretrained on ImageNet [12]. We used the HQ variant of the FaceForensics++ training dataset, which contains a little compression, and 5 types of data augmentation:

(1) No data augmentation whatsoever
(2) Only basic data augmentation, like blurring, noise, color jitter etc
(3) Data augmentation with 8 basic autoencoder models (2x AE1, 2xAE2, 4x UNet1), as presented in Figure 2
(4) Data augmentation with all the models mentioned above, one model per architecture, as presented in Figure 2
(5) Data augmentation with with all the models mentioned above, multiple models at different epochs and multiple loss functions per architecture, as presented in Figure 2.

Table 1: Comparison of generalization for models trained FaceForensics++ and evaluated on CelebDF, with different levels of data augmentation

Method	CelebDF AUC [%]
Xception - basic aug (2)	68.96
Xception - AE aug (3)	73.67
CapsNet - basic aug (2)	64.32
CapsNet - AE aug (3)	70.4
EfficientNetB4 - basic aug (2)	66.93
EfficientNetB4 - AE aug (3)	**75.85**

(6) Data augmentation with with all the models mentioned above, multiple models at different epochs and multiple loss functions per architecture, but without also using basic data augmentation like blurring, noise, color jitter etc.

We use a Dropout value of 0.25 and a small weight decay of 0.0003 to prevent overfitting. We normalize the data to mean=[0.5, 0.5, 0.5] and std=[0.5, 0.5, 0.5].

All the experiments were run on a machine with an Nvidia 3090 24GB graphics card and 32GB RAM.

4.4 Experimental results

All models achieved over 99% AUC on the test dataset of FaceForensics++ in all augmentation conditions. Therefore, the next logical step is to evaluate the generalization, as that is the focus of this work.

Table 1 presents a comparison between our models with 2 different types of data augmentation: (2) Only basic data augmentation and (3) Data augmentation with 8 basic autoencoder models, focusing on generalization to an unseen dataset (CelebDF). All models were trained on FaceForensics++, in all data augmentation settings. Table 1 shows that using autoencoder-based data augmentation can increase generalization to an unseen dataset by almost 10%. Although the CapsNet architecture achieves the best results on FaceForensics++, it overfits the dataset and does not generalize well. Due to the fact that the rest of our experiments proved that CapsNet models tend to overfit the data more, we excluded this model from further experiments.

Table 2 presents a comparison of performance of the XceptionNet model with different levels of augmentation, presented in Subsection 4.3. As we can see, adding more models increases the performance of the models. This happens up to a point, as adding more models than in Full AE aug (5) proved to be unfruitful. We can see that using no data augmentation overfits the data, making it impossible to generalize. A basic data augmentation only increases generalization by under 5%. By using the first level of autoencoder data augmentation, we get an increase in performance of over 5%. By using over 40 total models with the augmentation (4), we achieve an over 80% AUC on CelebDF. Doubling the number of models to 80 - augmentation (5) does not increase performance by much.

Table 2: Comparison of generalization for models trained FaceForensics++ and evaluated on CelebDF, with different levels of data augmentation

Method	CelebDF AUC [%]
Xception - no aug (1)	64.55
Xception - basic aug (2)	68.96
Xception - AE aug (3)	73.67
Xception - AE aug (4)	80.73
Xception - Full AE aug (5)	**82.62**
EfficientNetB4 - Full AE aug (5)	**82.87**
Xception - AE aug (6)	80.61

Table 3: Comparison of generalization for models trained FaceForensics++ and evaluated on CelebDF and DFDC Preview, compared to other state-of-the-art approaches

Method	CelebDF AUC [%]	DFDC Preview AUC [%]
LipForensics [18]	82.4	73.5
Face X-Ray [22]	79.5	65.5
3D R50-FTCN [44]	86.9	74.0
Xception [31]	73.7	70.9
CNN-aug [40]	75.6	72.1
PCL + I2G [43]	90.03	74.37
EFNB4 + SBIs [32]	**93.18**	**86.15**
Ours - Xception Full AE aug (5)	82.62	71.52
Ours - EfficientNetB4 Full AE aug (5)	82.87	72.6

Using autoencoders to augment the datasets ultimately helps with generalization, achieving results almost 14% bigger compared to using only basic data augmentation techniques.

Table 3 presents the performance of our best models - XceptionNet with augmentation (5) and EfficientNetB4 with augmentation (5), compared to the state of the art. Our models outperform many state-of-the-art approaches like LipForensics [18], Face X-Ray [22] or CNN-aug [40]. However, out method falls short of best performance, with 2 methods outperforming it. What is essential to remember is that out method can easily be combined with other approach. More than that, some of the methods presented also use the temporal dimension, an approach that we have not yet tried and one that can improve performance even further.

The results in Table 4 present our model's performance against 4 types of unseen manipulation: Color Jitter (brightness between 0.5 and 1.5, saturation between 0.5 and 1.5, hue between -0.1 and 0.1), Random Rotation between 10 and 30 degrees, Random Affine Transform (rotation between 10 and 40 degrees, scaling between 50% and 100% and translation between 10% and 30% of the image's size) and Gaussian Blur (3x3 kernel size, sigma between 20 and 25). The results show that out model is very invariant to changes in the image, as our model that is trained only on data augmented with autoencoders is able to maintain a performance within 2% versus the case where data is not perturbed for 3 types of perturbation (Color Jitter and Random Rotate) and drops less than 8% in performance against some very aggressive affine transforms. What is interesting is that we actually managed to increase the performance of our model when it deals with blurred images, compared to normal images.

Table 4: Comparison of generalization for XceptionNet, trained on FaceForensics++ using only the autoencoder augmentation, without basic data augmentation. Evaluation on CelebDF.

Method	Perturbation	CelebDF AUC [%]
Xception Full AE aug (6)	Color Jitter	79.44
Xception Full AE aug (6)	Random Rotate	78.52
Xception Full AE aug (6)	Random Affine	72.58
Xception Full AE aug (6)	Gaussian Blur	82.45

4.5 Discussion

From the results presented above, we can draw the conclusion that augmenting the data with deep learning models increases the model's ability to generalize. More than that, we concluded that using a bigger number of autoencoder/UNet models resulted in increased performance, up to a point.

From our testing, the quality of the autoencoder models did not affect the performance of the deepfake detectors very much, as it remained more or less the same even when running the same experiments without the worst models.

One drawback of this augmentation technique is that we can only apply it on stationary frames. We tried using temporal networks like LSTM or Transformer together with features from the XceptionNet model and the results did not exceed just averaging the outputs for each individual frame. For this to be possible, the data augmentation must be done by 3D CNN models, so that it would also incorporate the changes in time. Without that, there were sometimes inconsistent artefacts in time for the frames.

Although our method did not achieve state-of-the-art performance, it is important to note that this technique can be used together with other ideas very easily.

5 CONCLUSION

This paper introduces a new method of data augmentation that aims to increase generalization. This method uses neural networks to try to generate slightly different images from the training dataset. The autoencoder neural networks add some kind of variation or noise and may hide the frequency signature of deepfake generators as it passes the image through another neural network. We tested a number of architectures and models for autoencoders, like convolutional neural networks and UNet architectures. When augmenting the training data using our method, we saw a significant increase in the generalization ability of our deepfake detection models. More

than that, the models became very invariable to different kinds of perturbations. Although our method does not achieve state-of-the-art performance, it has the advantage that it can be used together with any models to augment their performance.

Lastly, we wish to improve this method in the near future, aiming at augmenting temporal data and finding the right level of error for which this autoencoder augmentation technique produces the best results.

ACKNOWLEDGMENTS

This work was supported under projects AI4Media, A European Excellence Centre for Media, Society and Democracy, H2020 ICT-48-2020, grant #951911 and PIMS-IAT, Platform Specialized in Identifying and Evaluating Early Warning Indices for Crisis Management, grant 27SOL/2021, UEFISCDI.

REFERENCES

[1] 2016. *FaceSwap.* Retrieved March 13 2023 from https://github.com/MarekKowalski/FaceSwap
[2] 2018. *Faceswap-GAN.* Retrieved March 13 2023 from https://github.com/shaoanlu/faceswap-GAN
[3] 2019. *DeepFake FaceSwap.* Retrieved March 13 2023 from https://github.com/deepfakes/faceswap
[4] 2019. *Faceapp.* Retrieved March 13 2023 from https://www.faceapp.com
[5] 2019. *Residual Recuurent Block with attention Unet.* Retrieved March 13 2023 from https://github.com/LeeJunHyun/Image_Segmentation
[6] 2019. *Unet-Segmentation-Pytorch-Nest-of-Unets Github.* Retrieved March 13 2023 from https://github.com/bigmb/Unet-Segmentation-Pytorch-Nest-of-Unets
[7] Md Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, Tarek M Taha, and Vijayan K Asari. 2018. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *arXiv preprint arXiv:1802.06955* (2018).
[8] Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. 2018. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*. IEEE, 59–66.
[9] François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1251–1258.
[10] Davide Alessandro Coccomini, Nicola Messina, Claudio Gennaro, and Fabrizio Falchi. 2022. Combining efficientnet and vision transformers for video deepfake detection. In *Image Analysis and Processing–ICIAP 2022: 21st International Conference, Lecce, Italy, May 23–27, 2022, Proceedings, Part III*. Springer, 219–229.
[11] Oscar de Lima, Sean Franklin, Shreshtha Basu, Blake Karwoski, and Annet George. 2020. Deepfake detection using spatiotemporal convolutional networks. *arXiv preprint arXiv:2006.14749* (2020).
[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
[13] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. 2020. The deepfake detection challenge (dfdc) dataset. *arXiv preprint arXiv:2006.07397* (2020).
[14] Brian Dolhansky, Russ Howes, Ben Pflaum, Nicole Baram, and Cristian Canton Ferrer. 2019. The deepfake detection challenge (dfdc) preview dataset. *arXiv preprint arXiv:1910.08854* (2019).
[15] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. 2020. Leveraging frequency analysis for deep fake image recognition. In *International conference on machine learning*. PMLR, 3247–3258.
[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).
[17] Alexandros Haliassos, Konstantinos Vougioukas, Stavros Petridis, and Maja Pantic. 2021. Lips don't lie: A generalisable and robust approach to face forgery detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5039–5049.
[18] Alexandros Haliassos, Konstantinos Vougioukas, Stavros Petridis, and Maja Pantic. 2021. Lips don't lie: A generalisable and robust approach to face forgery detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5039–5049.
[19] Debesh Jha, Michael A Riegler, Dag Johansen, Pål Halvorsen, and Håvard D Johansen. 2020. Doubleu-net: A deep convolutional neural network for medical image segmentation. In *2020 IEEE 33rd International symposium on computer-based medical systems (CBMS)*. IEEE, 558–564.
[20] Aminollah Khormali and Jiann-Shiun Yuan. 2022. DFDT: an end-to-end deepfake detection framework using vision transformer. *Applied Sciences* 12, 6 (2022), 2953.
[21] Akash Kumar, Arnav Bhavsar, and Rajesh Verma. 2020. Detecting deepfakes with metric learning. In *2020 8th international workshop on biometrics and forensics (IWBF)*. IEEE, 1–6.
[22] Lingzhi Li, Jianmin Bao, Ting Zhang, Hao Yang, Dong Chen, Fang Wen, and Baining Guo. 2020. Face x-ray for more general face forgery detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5001–5010.
[23] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. 2020. Celeb-df: A large-scale challenging dataset for deepfake forensics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3207–3216.
[24] Honggu Liu, Xiaodan Li, Wenbo Zhou, Yuefeng Chen, Yuan He, Hui Xue, Weiming Zhang, and Nenghai Yu. 2021. Spatial-phase shallow learning: rethinking face forgery detection in frequency domain. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 772–781.
[25] Yuchen Luo, Yong Zhang, Junchi Yan, and Wei Liu. 2021. Generalizing face forgery detection with high-frequency features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 16317–16326.
[26] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. 2019. Do gans leave artificial fingerprints?. In *2019 IEEE conference on multimedia information processing and retrieval (MIPR)*. IEEE, 506–511.
[27] Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. 2019. Use of a capsule network to detect fake images and videos. *arXiv preprint arXiv:1910.12467* (2019).
[28] Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. 2022. Capsule-Forensics Networks for Deepfake Detection. In *Handbook of Digital Face Manipulation and Detection*. Springer, Cham, 275–301.
[29] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. 2018. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999* (2018).
[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 234–241.
[31] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. 2019. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1–11.
[32] Kaede Shiohara and Toshihiko Yamasaki. 2022. Detecting deepfakes with self-blended images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18720–18729.
[33] Dan-Cristian Stanciu and Bogdan Ionescu. 2022. Uncovering the Strength of Capsule Networks in Deepfake Detection. In *Proceedings of the 1st International Workshop on Multimedia AI against Disinformation*. 69–77.
[34] Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. 2017. Synthesizing obama: learning lip sync from audio. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1–13.
[35] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.
[36] Shahroz Tariq, Sangyup Lee, and Simon S Woo. 2020. A convolutional LSTM based residual network for deepfake video detection. *arXiv preprint arXiv:2009.07480* (2020).
[37] Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred neural rendering: Image synthesis using neural textures. *Acm Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.
[38] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2016. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2387–2395.
[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[40] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. 2020. CNN-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8695–8704.
[41] Wei Wang, Jing Dong, and Tieniu Tan. 2010. Tampered region localization of digital color images based on JPEG compression noise. In *International Workshop on Digital Watermarking*. Springer, 120–133.
[42] Ning Yu, Larry S Davis, and Mario Fritz. 2019. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *Proceedings of the IEEE/CVF international conference on computer vision*. 7556–7566.
[43] Tianchen Zhao, Xiang Xu, Mingze Xu, Hui Ding, Yuanjun Xiong, and Wei Xia. 2021. Learning self-consistency for deepfake detection. In *Proceedings of the*

IEEE/CVF international conference on computer vision. 15023–15033.

[44] Yinglin Zheng, Jianmin Bao, Dong Chen, Ming Zeng, and Fang Wen. 2021. Exploring temporal coherence for more general video face forgery detection. In *Proceedings of the IEEE/CVF international conference on computer vision*. 15044–15054.

[45] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. 2018. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*. Springer, 3–11.