Fake-Buster: A Lightweight Solution for Deepfake Detection

Nathan Hubens[a,b], Matei Mancas[a], Bernard Gosselin[a], Marius Preda[b], and Titus Zaharia[b]

[a]ISIA Lab (UMONS), Belgium
[b]Artemis (IP Paris), France

ABSTRACT

Recent advances in video manipulation techniques have made synthetic media creation more accessible than ever before. Nowadays, video edition is so realistic that we cannot rely exclusively on our senses to assess the veracity of media content. With the amount of manipulated videos doubling every six months, we need sophisticated tools to process the huge amount of media shared all over the internet, to remove the related videos as fast as possible, thus reducing potential harm such as fueling disinformation or reducing trust in mainstream media. In this paper, we tackle the problem of face manipulation detection in video sequences targeting modern facial manipulation techniques. Our method involves two networks: (1) a face identification network, extracting the faces contained in a video, and (2) a manipulation recognition network, considering the face as well as its neighbouring context to find potential artifacts, indicating that the face was manipulated. More particularly, we propose to make use of neural network compression techniques such as pruning and knowledge distillation to create a lightweight solution, able to rapidly process streams of videos. Our approach is validated on the DeepFake Detection Dataset, consisting of videos coming from 5 different manipulation techniques, reflecting the organic content found on the internet, and compared to state-of-the-art deepfake detection approaches.

Keywords: DeepFake Detection, Image Manipulation, Neural Network Compression

1. INTRODUCTION

In 2017, the term *Deepfakes* was coined to refer to a deep learning based technique able to swap the face of a person with the face of another. It was later expanded to any deep learning based method able to manipulate image, video and/or audio content and whose objective is to deceive people for a harmful usage. The definition "Believable media generated by a deep neural network" has been recently proposed,[1] to include methods such as reenactment, replacement, editing and synthesis of media content. While such manipulation techniques can be prolific for certain usages such as special effects or movie dubbing, they can also be harmful when used in other contexts, to create hoax and propaganda.

For a long time, video editing and manipulation was in the hands of a few people, requiring lots of expertise and high-end programs and resources. But over the last few years, generation neural methods such as Autoencoders (AE) or Generative Adversarial Networks (GAN) have made so much progress, both in terms of quality and in terms of required resources, that nowadays many DeepFakes generation techniques have become widely accessible on many phone applications and social media, or even on marketplace services, allowing anyone to create fake videos without any preliminary experience in the field. For that reason, the amount of detected deepfakes on the internet has been increasing exponentially, at a rate which is roughly doubling every 6 months. As a result, Deepfakes could flood social media very quickly if one does not pay attention. Moreover, access to even better computation resources allow deepfakes to perform real-time impersonation, which can be used for social engineering and fraud. Eventually, without any method able to discern the true from the fake, this would irrevocably damage our trust in any video or image we see on conventional media.

Nowadays, a wide field of research is now dedicating growing efforts for detecting facial manipulation in image and video. However, the lack of data makes the task of deepfake detection in the real-world setting pretty challenging. For that reason, the research community has released many deepfakes datasets[2,3] to promote

Further author information: (Send correspondence to Nathan Hubens)
[a]: E-mail: {name.surname}@umons.ac.be
[b]: E-mail: {name.surname}@telecom-sudparis.eu

and help researchers in the field. More importantly, the Deepfake Detection Challenge was recently organized, gathering the experts in the field, competing to release the best deepfake detection models. This lead in more than 2,000 teams submitting their solution. However, participants were judged on the final performance of their model, so few of them considered the size and speed of their model being a limitation and generally proposed solution involving ensembles of models of millions of parameters, which is unsuitable for such systems to be able to cope with the potential amount of videos to analyse on a daily basis on social networks or for real-time applications.

In this paper, we tackle the problem of deepfake detection by proposing a lightweight model, able to quickly analyze a video. Our contributions can be summarized as follows:

- We propose a single lightweight solution for DeepFake Detection, able to detect manipulation on still images, as well as videos.
- Empirically show that the proposed method is able to reach competitive performance when compared to state-of-the-art classifiers, while showing less parameters and operations.
- We propose a small web-based application for deepfake detection. This application, as well as the code to reproduce our results, can be found at the following link: https://github.com/nathanhubens/deepfake-buster.

The remainder of the article is organized as follows. Section 2 provides a general description of DeepFake detection methods, as well as model compression techniques. We then describe in Section 3 the proposed method, including the architecture, the data used for training and the methods applied to obtain a lightweight architecture. We then provide our results in Section 4, as well as an ablation study of the techniques that we used, and also an interpretation of the decision of our solution by presenting the attention maps given by our model. Lastly, we provide in Section 5 our concluding remarks and open issues.

2. RELATED WORK

2.1 DeepFakes Detection

DeepFakes creation often generates artifacts that are usually exploited by recent methods. Those artifacts can generally be separated into two groups: *spatial* and *temporal*.

Spatial Artifacts Detection. Artifacts generally appear when the generated content is encrusted into the original frame. Treating the problem as image classification allows CNN models to detect the artifacts present in the frames,[4] thus classifying each frame individually. Those artifacts can come in many forms and dimensions, requiring different techniques to be detected. For example, mesoscopic information can be analyzed to detect the forgery.[5] Because the generated and original contents usually do not come from the same source, some warping artifacts may appear during the blending phase, which can also be detected.[6]

Temporal Artifacts Detection. Because a majority of DeepFake generation techniques are used frame-by-frame on videos, this can leave temporal inconsistencies or artifacts such as flickering or jitter that can be detected.[7,8] For example, irregular eye blinking patterns can be exploited to detected manipulated faces.[9] Recent method even study the visual heartbeat rhythm to assess the veracity of a video.[10]

Hybrid Techniques. Because both spatial and temporal artifacts usually appear in DeepFakes videos, they can both be exploited in order to detect forged videos. Combining those techniques usually require more data intensive models such as 3D CNNs.[11]

2.2 Network Compression

There exist many techniques that can be applied in order to obtain a lightweight architecture. The next subsection will outline the main ones.

Efficient Architectures: To obtain a lightweight architecture, it is most important to design it using parameters efficient building blocks. For example, convolutions have been shown to be more efficient than fully-connected layers when the task involves natural signals data. Also, early CNN architectures used to possess a MLP in the top layers,[12,13] which have been replaced by a pooling layer, followed by a single linear layer, drastically reducing the number of parameters without impacting the performance. More recently, several conducted researches have been interested in creating even more efficient building blocks. Those blocks involve Depthwise Separable Convolutions,[14,15] but also bottlenecks to further improve the computation efficiency. Some architectures have even been discovered by performing Neural Architecture Search, designed to be as efficient as possible, notably in terms of computations.[16,17]

Pruning. Neural networks have been known to be widely overparametrized, even for already parameter efficient ones. An efficient way to reduce the size of trained neural network is to remove redundant weights, *i.e.* prune them. Pruning can be applied at different granularities, *i.e.* structured when the goal is to remove blocks of weights, e.g. kernels or filters, or unstructured when the weights are removed individually. There exist many criteria of selection used to decide which parameters to remove, but the most widely used is Magnitude Pruning, i.e. remove parameters that have the smallest l_1-norm, which as been shown to better generalize across architectures and datasets.[18]

Quantization. Representing the weights of a neural network using a smaller arithmetic precision makes it lighter to store but also faster during inference. A limitation of using smaller precision arithmetic is that it hinders the SGD updates when computing the backward pass, usually involving small changes in the weights, as the small precision may cause number rounding errors. To solve that problem, it is possible to store the model using a smaller precision but to make the updates using high precision.[19] By doing so, the model still benefits from the low precision for each forward passes, making both the training and inference faster and the updates of SGD are done using high-precision, avoiding any rounding problems.

Knowledge Distillation. It is possible to distill the knowledge of a big model into a smaller one by training the later to copy the output of the big one. This technique is called Knowledge Distillation[20] and is known to expose the *Dark Knowledge* present in the big model, helping the smaller one to understand more about the data distribution and to reach a better performance.

Inspired by the state of the art, we propose a CNN-based technique, able to detect manipulation on both still images or videos, thus exploiting spatial artifacts, and which is lightweight, thanks to the use of compression techniques, to be able to process a large flow of information rapidly.

3. METHODOLOGY

3.1 The Dataset

The dataset used for our research is the DFDC dataset.[21] This dataset belongs to the so-called third generation of Deepfakes datasets,[2] and is the largest currently available dataset consisting of 124k videos, distributed in three subsets.

Training Set. The training set provided in DFDC is comprised of 119,154 video clips, of 486 unique subjects. A total of 100,000 videos contain Deepfakes, created with DFAE, MM/NN face swap,[22] NTH[23] and FS-GAN.[24] Each video is 10 seconds long, representing 300 frames.

Validation Set. The validation set consists of 4,000 video clips, which are also 10 seconds long, in which 2,000 clips contain Deepfakes. This set was created using 214 unique subjects, none of which were present in the training set. The same Deepfake generation techniques were used, with the addition of a new one, StyleGAN.[25] Authors also include augmentations and distractions to 80% of the videos. Augmentations perform modifications such as noise, frame rate change, JPEG artifacts, blur, while distractions are composed of logo or face overlay.

Test Set. The test set is comprised of 5,000 video clips of 260 unique subjects that have not been seen before. Again, augmentations and distractions were added to 80% of the set, including new distractions such as dog masks or flower crown filters.

3.2 The Architecture

In this paper, we propose a lightweight solution, for DeepFakes detection, whose general framework is presented in Figure 1. This solution is composed of two networks: (1) a face identification network, extracting the faces contained in an image or video, and (2) a manipulation recognition network, classifying the extracted faces as real or fake.

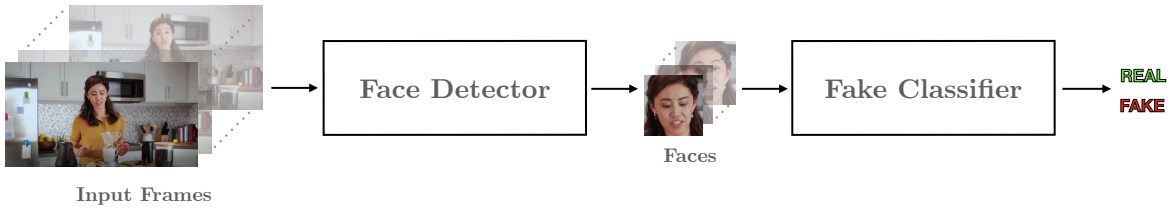

Figure 1. The general framework of our solution. It takes a single frame (an image) or several frames (a video) as an input, extracts the face(s) and performs classification.

The Face Detector that we use is RetinaNet,[26] as we empirically find that it provides a good trade-off between detection performance and speed of processing. Concerning the Fake Classifier, the architecture that we use is inspired by the design of ResNets,[27] but replacing the common convolution blocks by an inverted residual module, as used in MobileNetV2.[28] This inverted residual module is composed of a first pointwise convolution, allowing to expand the amount of channels before performing the spatial feature extraction by a depthwise convolution. The amount of channels is squeezed by another pointwise convolution. We also append a Squeeze-Excite module[29] before adding back the residual, as it allows to increase the classification performance at the cost of a slight increase in the total parameter amount.

Our building block is represented in Figure 2. Each Convolution is followed by a batch normalization layer and a Rectified Linear Unit.

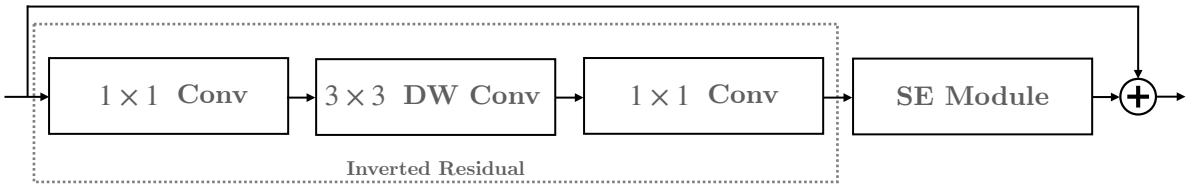

Figure 2. The simplified building block of our model (non-linearities and normalization layers are omitted).

Our architecture, represented in Figure 3, is composed of a stem of 3 convolutional layers, then 4 double blocks, each followed by a MaxPooling layer. After the first block, we insert a Self-Attention module, that not only helps improving performance, but also helps to understand and interpret the decision of the network.

3.3 Data Augmentation and Progressive Learning

As we discussed in Section 1, organic content found on the internet is subject to come from many environment and to undergo quality degradation. For those reasons, the data augmentation that we apply during training should be thoughtfully chosen, in order to make our model as robust as possible. The augmentations that we apply on our training data are represented in Figure 4. We also use Cutout[30] to randomly discard some part

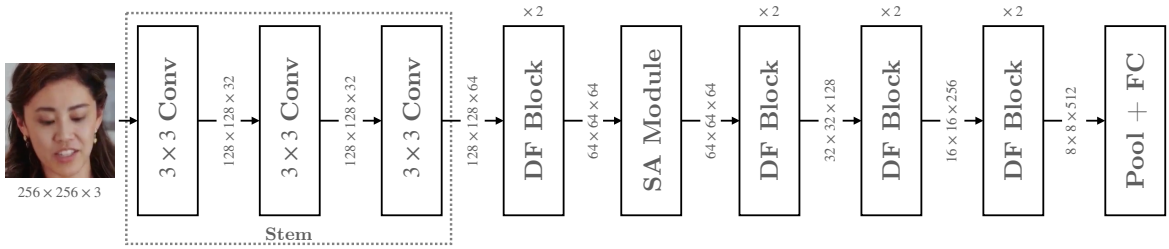

Figure 3. The Architecture of our Proposed model. It consists of a stem of 3 convolutional layers, then 4 double DF Block, each followed by a pooling layer. A single Self-Attention module was included in the model.

of the image, forcing the network to focus on different attributes and also Mixup[31] (not represented), blending images together to improve generalization capacity of the network.

Figure 4. Illustration of the augmentations applied to training data. The transforms have been chosen to create a model robust to organic videos that it might encounter in real-world settings.

To create a model even more robust and less prone to overfitting, we use a technique called Progressive Learning,[17] much inspired by Curriculum Learning. The idea is to gradually increase the effects of augmentations along the training, making the training example more difficult as the model is learning to discriminate more simpler examples.

3.4 Compression of the solution

After designing an efficient architecture, we still can reduce the number of parameter and speed of processing. In this purpose, we consequently use pruning, quantization, knowledge distillation and batch normalization folding. The results of the addition of each method is reported in Table 1.

Pruning. In order to reduce the amount of parameters, we apply pruning on our architecture. To remove as much parameters as possible, we apply unstructured pruning, *i.e.* we remove individual weights. Doing so enables to reach higher sparsity level without suffering from a performance degradation. Unlike most commonly used pruning methods, which start pruning after the model has been trained, we perform pruning right from the start of training, using the One-Cycle Pruning schedule.[32] This not only allows to reach higher accuracy than most pruning schedule, but allows to do it in a more constrained training budget. The weights that show the smallest L_1-norm are the ones that are removed. We decided to remove the same percentage of weight at each layer, *i.e.* local pruning, and empirically found a sparsity amount of 50% being the upper bound above which the performance of the model starts to decrease.

Quantization. The quantization that we apply in our work is using the Mixed-Precision Training[19] technique. It allows us to obtain a model using FP16 precision, but performing all the training updates using FP32. As the forwards pass is performed using FP16, Mixed-Precision thus allows us not only to have a model that has a faster inference, but also a faster training.

Knowledge Distillation. Pruning and Quantization are usually not lossless, *i.e.* a trade-off must be found between the reduction of the size of the network and the performance drop. In order to mitigate that drop, we perform Knowledge Distillation[20] where the student model is the pruned and quantized one and the teacher model is the model before compression. By doing so, we encourage the compressed model to better recover its lost performance.

Batch Normalization Folding. After training, the parameters of the Batch Normalization layers in a model are fixed. This means that we can blend the effect of normalization into the weights of the convolution operation preceding it. In practice, we can remove the batch norm layers and replace the values of the weights W and biases b of the convolutional layers preceding it by W_{fold} and b_{fold}, expressed by the following equations:

$$w_{\text{fold}} = \gamma \cdot \frac{W}{\sqrt{\sigma^2 + \epsilon}}$$
$$b_{\text{fold}} = \gamma \cdot \frac{b - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

With γ and β being the learned parameters of the batch normalization, and σ, μ, being respectively the standard deviation and mean values, stored by the batch normalization layer.

4. RESULTS

4.1 Ablation Study

We perform an ablation study to further highlight the impact of each technique used to create our final lightweight model.

	Accuracy	Loss	Model Size (Mb)	ROC-AUC
Baseline	74.34	0.5563	46.22	0.893
+ Self-Attention	75.36	0.5511	46.35	0.903
+ Squeeze-Excite	75.54	0.5445	49.17	0.909
+ Mixup	78.24	0.5312	49.17	0.917
+ Progressive Learning	81.64	0.5098	49.17	0.935
+ Pruning (50%)	81.52	0.5109	24.74	0.930
+ Mixed-Precision	81.16	0.5122	12.45	0.927
+ KD	81.90	0.5109	12.45	0.939
+ BN Folding	81.90	0.5109	12.43	0.939

Table 1. Ablation Study of the proposed model.

The baseline model is built using only inverted block modules and trained using the data augmentations showed in Figure 4. We then gradually add components to the architecture, i.e. Self-Attention and Squeeze-Excite, showing that they help the model to reach higher performance at the price of a small increase in the parameter amount. The data augmentation is then modified, by adding Mixup and Progressive Learning. Again, those method help to obtain a better model and do not change the parameters count. Pruning and Mixed-Precision training reduce slightly the performance of the model but allow to decrease the model size by a factor of $\sim 4\times$. Knowledge Distillation allows to recover from the lost performance after Pruning and Quantization, and Batch Normalization Folding finally reduces the parameter count a bit more while keeping performance intact.

4.2 Comparison to other methods

We compare our DeepFake classifier to other state-of-the-art classifiers such as Mesonet,[5] ResNet-18,[27] EfficientNet-B0,[16] Xception,[15] that were used for deepfake classification, but also to the two best solutions proposed at the DeepFake Detection Challenge.

	ROC-AUC	Model Size (Mb)	FLOPS (1e6)	Inference Time (ms)
MesoNet	0.786	0.91	0.06	3.84 ± 0.22
Resnet18	0.886	357.53	1.81	4.35 ± 0.24
EfficientNet-B0	0.913	127.65	0.39	16.17 ± 0.13
Xception	0.929	665.08	38.92	13.52 ± 0.32
Selim Seferbekov[33]	0.984	6109.12	111.00	197.02 ± 21.76
WM[34]	0.985	1722.85	95.84	56.21 ± 8.63
Our Method	0.939	12.43	0.77 (fp16)	6.31 ± 0.23

Table 2. Comparison state-of-the-art DeepFake classification methods. Inference Time is computed on a Nvidia GTX 1080 and averaged over 10 iterations.

As can be seen in 2, our proposed model is able to outperform most of state-of-the-art classification models, at the exception of the solution of Selim Seferbekov and WM, being the two top solutions of the DFDC. However, those two solutions involve an ensemble of models, making both the storage and inference pass very costly. The lightest and fastest solution is MesoNet, however, even though it was able to accurately classify on the Face2Face dataset,[35] the DFDC dataset is more challenging and requires more power than MesoNet is able to provide. We thus find that our solution is able to provide a good trade-off between performance and computation costs.

4.3 Interpretation

As DeepFakes detection is becoming more and more difficult, especially for the human eye, it is important to be able to understand the decisions of our model. For that reason, we can get a few indications by studying the effect of the attention block that we introduced in the network. Not only does it helps our model getting better results, but it also allow us to highlight the part in the images that are the most important for the decision of our model. As shown in Figure 5, the highlighted parts generally concern high frequency information, i.e. eyes, nose, mouth. It was also reported that the artifacts were usually localized around those facial features.[4] This shows that DeepFakes generation methods still create artifacts that can be detected by classification models.

Figure 5. Extracted faces and their corresponding attention map.

5. DISCUSSION AND CONCLUSIONS

In this paper, we proposed a novel, lightweight solution for DeepFake detection. This solution has been created by the use of efficient computation layers, and application of several compression techniques, namely pruning, knowledge distillation, quantization, batch normalization folding. We have shown that our solution was able to compete with state-of-the-art classifiers, while providing comparable performances to state-of-the-art deepfake detection models. Further work could however include the use of temporal artifacts in our classification.

REFERENCES

[1] Mirsky, Y. and Lee, W., "The creation and detection of deepfakes: A survey," *ACM Comput. Surv.* **54**(1) (2021).

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

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

[4] Tariq, S., Lee, S., Kim, H., Shin, Y., and Woo, S. S., "Detecting both machine and human created fake face images in the wild," in [*Proceedings of the 2nd International Workshop on Multimedia Privacy and Security (MPS)*], (2018).

[5] Afchar, D., Nozick, V., Yamagishi, J., and Echizen, I., "Mesonet: a compact facial video forgery detection network," in [*International Workshop on Information Forensics and Security (WIFS)*], (2018).

[6] Li, Y. and Lyu, S., "Exposing deepfake videos by detecting face warping artifacts," in [*IEEE Conference on Computer Vision and Patten Recognition (CVPR) Workshops*], (2019).

[7] Güera, D. and Delp, E. J., "Deepfake video detection using recurrent neural networks," in [*IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*], (2018).

[8] Sabir, E., Cheng, J., Jaiswal, A., AbdAlmageed, W., Masi, I., and Natarajan, P., "Recurrent convolutional strategies for face manipulation detection in videos," in [*IEEE Conference on Computer Vision and Patten Recognition (CVPR) Workshops*], (2019).

[9] Yuezun, L., Ming-Ching, C., and Siwei, L., "In ictu oculi: Exposing ai created fake videos by detecting eye blinking," in [*2018 IEEE International Workshop on Information Forensics and Security (WIFS)*], (2018).

[10] Qi, H., Guo, Q., Juefei-Xu, F., Xie, X., Ma, L., Feng, W., Liu, Y., and Zhao, J., "Deeprhythm: Exposing deepfakes with attentional visual heartbeat rhythms," *ACM International Conference on Multimedia* (2020).

[11] Wang, Y. and Dantcheva, A., "A video is worth more than 1000 lies. Comparing 3DCNN approaches for detecting deepfakes," in [*International Conference on Automatic Face and Gesture Recognition*], (2020).

[12] Simonyan, K. and Zisserman, A., "Very deep convolutional networks for large-scale image recognition," in [*3rd International Conference on Learning Representations, ICLR*], (2015).

[13] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," in [*Advances in Neural Information Processing Systems (NeurIPS)*], (2012).

[14] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR* **abs/1704.04861** (2017).

[15] Chollet, F., "Xception: Deep learning with depthwise separable convolutions," *CoRR* **abs/1610.02357** (2016).

[16] Tan, M. and Le, Q., "EfficientNet: Rethinking model scaling for convolutional neural networks," in [*International Conference on Machine Learning (ICML)*], (2019).

[17] Tan, M. and Le, Q. V., "Efficientnetv2: Smaller models and faster training," *ArXiv* **abs/2104.00298** (2021).

[18] Gale, T., Elsen, E., and Hooker, S., "The state of sparsity in deep neural networks," *ArXiv* **abs/1902.09574** (2019).

[19] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H., "Mixed precision training," in [*International Conference on Learning Representations (ICLR)*], (2018).

[20] Hinton, G., Vinyals, O., and Dean, J., "Distilling the knowledge in a neural network," in [*NeurIPS Deep Learning and Representation Learning Workshop*], (2015).

[21] Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., and Ferrer, C. C., "The deepfake detection challenge dataset," (2020).

[22] Huang, D. and De La Torre, F., "Facial action transfer with personalized bilinear regression," in [*European Conference for Computer Vision (ECCV)*], (2012).

[23] Zakharov, E., Shysheya, A., Burkov, E., and Lempitsky, V., "Few-shot adversarial learning of realistic neural talking head models," *International Conference on Computer Vision (ICCV)* (2019).

[24] Nirkin, Y., Keller, Y., and Hassner, T., "Fsgan: Subject agnostic face swapping and reenactment," *International Conference on Computer Vision (ICCV)* (2019).

[25] Karras, T., Laine, S., and Aila, T., "A style-based generator architecture for generative adversarial networks," *Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[26] Lin, T.-Y., Goyal, P., Girshick, R. B., He, K., and Dollár, P., "Focal loss for dense object detection," *International Conference on Computer Vision (ICCV)* (2017).

[27] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," in [*Conference on Computer Vision and Pattern Recognition, (CVPR)*], (2016).

[28] Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L.-C., "Mobilenetv2: Inverted residuals and linear bottlenecks," *Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[29] Hu, J., Shen, L., and Sun, G., "Squeeze-and-excitation networks," *Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[30] Devries, T. and Taylor, G. W., "Improved regularization of convolutional neural networks with cutout," *ArXiv* **abs/1708.04552** (2017).

[31] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D., "Mixup: Beyond empirical risk minimization," in [*International Conference on Learning Representations (ICLR)*], (2018).

[32] Hubens, N., Mancas, M., Gosselin, B., Preda, M., and Zaharia, T., "One-cycle pruning: Pruning convnets under a tight training budget," *ArXiv* **abs/2107.02086** (2021).

[33] Seferbekov, S. in [*https://github.com/selimsef/dfdc_deepfake_challenge*],

[34] Zhao, H., Cui, H., , and Zhou, W. in [*https://github.com/cuihaoleo/kaggle-dfdc.*],

[35] Thies, J., Zollhöfer, M., Stamminger, M., Theobalt, C., and Nießner, M., "Face2face: real-time face capture and reenactment of RGB videos," *Commun. ACM* **62**(1), 96–104 (2019).