DETECTING DEEPFAKES WITH SELF-BLENDED IMAGES

ARIJIT SAMAL (19051) ARIHANT MOHAPATRA (19049)
KUNAL SHAW (19353)

GROUP NUMBER- **09**
Date of Submission: NOVEMBER 22, 2022
Under the guidance of: DR. VINOD K KURMI

1 INTRODUCTION

As GANs are becoming more and more powerful over the years i.e they are generating more realistic fake images, deepfake detection is becoming more difficult. Previous detection methods performed well on the training dataset, where they could detect forgeries that they learned in training, but when these same detection methods were applied on new datasets, which contained manipulation not there in the training dataset, the detection performance dropped significantly. This happened because the models used to overfit the training dataset.

Hence keeping all this in mind, the reference paper that we have used for our project has proposed the use of a new type of traing data called self-blended images(SBI) for deepfake detection.

2 HOW TO RUN THE CODE

The sub folder SelfBlendedImages-master inside the folder grp09 contains the complete code.
path- **/home/dl/dl/grp09/SelfBlendedImages-master**
we have generated 2 script files- train.sh and test.sh. Both of them are present in the grp09 folder.
Steps to run the code-

1. change the working directory to grp09 using the change directory terminal command mentioned below
   cd /home/dl/dl/grp09
2. then use **./train.sh** in the terminal to start the training phase.
3. finally, use the **./test.sh** command to start the testing phase.

3 GENERATION OF SELF-BLENDED IMAGES

The key idea is to create images that are hardly recognisable and contain common face forgery techniques. This will allow the model to learn more general and robust representations for face forgery detection.

The process of creating a self-blended image consists of two things: the source target generator(STG) and the masked generator(MG). The STG takes a real image and uses image processing on it to produce a pair of pseudo source and target images. MG uses facial landmarks from the real image and generates different blending masks. Then, in the end, an SBI is created by blending the source and target images with the masks generated by the MG.

The advantages of using SBI while training is that it reduces the computational cost and hence increases training efficiency. This is because, unlike previous methods, SBIs don't need nearest landmark search for source target pair selection(which is computationally expensive).

Then evaluation is done using cross-dataset evaluation and cross-manipulation evaluation. In the cross-dataset evaluation we train on one dataset and test on new datasets. In the cross-manipulation evaluation we test our model on data manipulations not seen in the training dataset.

3.1 Source Target Generator

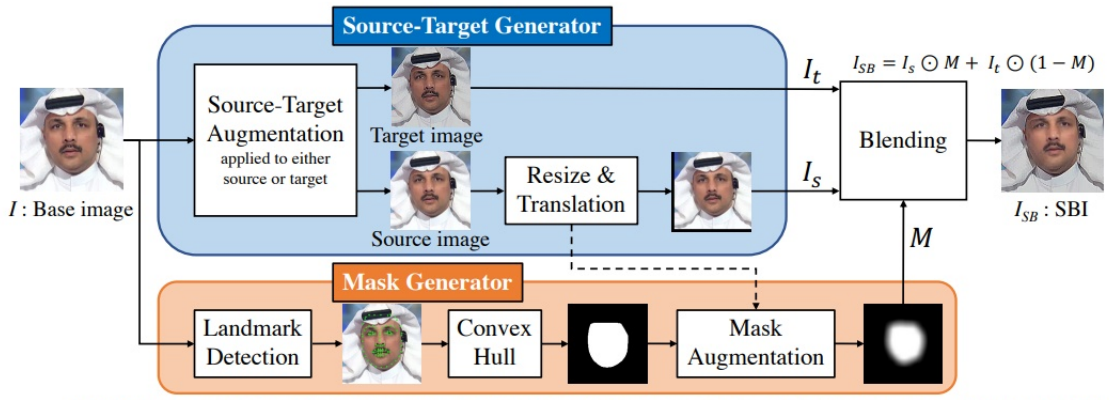
Given an input image, STG copies it to initialize a pseudo-source and target image. Then it randomly applies image transformations to either of the 2 images to create statistical inconsistencies between the 2 images. First colour transformation techniques such as hue, saturation and contrast of input are used. Then the image is down sampled or sharpened as part of frequency transformations. Then it is resized and translated.

3.2 Mask Generator

The mask generator provides a gray scale mask image which is used to blend the source and target images. First the facial features are detected using the landmark detector and then a mask is initialised using the convex hull from predicted facial landmarks. Then mask is deformed using the elastic deformations and then smoothed using 2 gaussian filters.

3.3 Blending

By blening the source and target image we get the self blended image. Once the SBIs are generated, we can train any binary classifier on them. The loss is calculated using the binary cross entropy loss function.



Overview of generating an SBI. A base image I is input into the source-target generator (STG) and mask generator (MG). STG generates pseudo source and target images from the base image using some image transformations, whereas MG generates a blending mask from the facial landmarks and deforms it to increase the diversity of the mask. Finally, the source and target images are blended with the mask.

4 EXPERIMENTAL SETUP

4.1 Training Phase

We use the FaceForensics++(FF++) dataset for training. We extract original sequences of YouTube videos in compressed C23 format for training the model. We generate 8 frames for each video of the c23 format of the dataset. Then, we preprocess the frames, generate SBIs and use them for training with the efficient-net b4 convolutional neural network. Finally, we save the top 5 checkpoints which contains the optimal weights which we reuse for testing the test datasets.

4.2 Testing Phase

We test our model on the Celeb DF datasets. We use the celeb-DF-V1 and Celeb-DF-V2 datasets for testing purpose. The V1 dataset contains 100 videos and the V2 dataset contains 512 videos sampled

from both real and synthesized videos. We use the weights obtained from training phase on the FF++ dataset for evaluation on the celeb DF datasets and hence, perform a type of cross-dataset evaluation.

5 IMPLEMENTATION DETAILS

5.1 Preprosessing the data

Dlib and RetinaFace are used to extract facial landmarks and bounding boxes from each video frame, respectively. We use an 81 facial landmarks shape predictor in Dlib. For the width and height of the face calculated from the bounding box, the face region is cropped with a random margin of 4–20% for training and a fixed value of 12.5% for inference.

5.2 Evaluation types

Mailnly of 2 types-

1. In-dataset evaluation and
2. cross-dataset evaluation.

5.3 Evaluation metrics

We report the area under the receiver operating characteristic (ROC) curve also called the AUC score to compare with prior works. Typically, frame-level predictions are averaged over video frames to get the final auc score and video level predictions directly produce the auc score of that video.

6 TRAINING

We adopt the state-of-the-art convolutional network architecture EfficientNet-b4 (EFNB4) pre-trained on ImageNet as the classifier and train it for 1000 epochs with the SAM optimizer. The batch size and learning rate are set to 16 and 0.001, respectively. We sample 10 frames per video for training. If two or more faces are detected in a frame, the one with the largest bounding box is extracted. Each batch consists of real images and their SBIs, and the same augmentation is applied to each real image and its SBI. We also use some data augmentations, i.e., ImageCompression, RGBShift, HueSaturationValue, and RandomBrightnessContrast.

7 MODEL VALIDATION

We use a validation set that consists of real videos and their SBIs after each epoch and select the weight with the highest number of epochs among the five weights with the highest AUC. Therefore, no manipulated images are used even for the model validation in our approach.

8 INFERENCE STRATEGY

We sample 38 frames per video for inference. If two or more faces are detected in a frame, the classifier is applied to all faces and the highest fakeness confidence is used as the predicted confidence for the frame. Once the predictions for all frames are obtained, we average them to get the prediction for the video. We use Celeb-DF-V2 for testing.

9 FIRST PHASE RESULTS

1. For the training phase, for 20 epochs and batch size 32 we get a training accuracy of 91.76%, validation accuracy of 92.33%, and validation AUC score of 97.65.

2. For the training phase, for 20 epochs and batch size 16 we get a training accuracy of 94.24%, validation accuracy of 94.44%, and validation AUC score of 98.24.

3. For the training phase, for 100 epochs and batch size 16 we get a training accuracy of 98.38%, validation accuracy of 97.57%, and validation AUC score of 99.67.

4. In the testing phase, when we test on 100 sample videos from both real and synthesized images from the Celeb-DF-V1 dataset, We get an AUC score of 51.27 from the ROC curve.

5. In the testing phase, when we test on 512 sample videos from both real and synthesized images from the Celeb-DF-V2 dataset, We get an AUC score of 61.92 from the ROC curve.

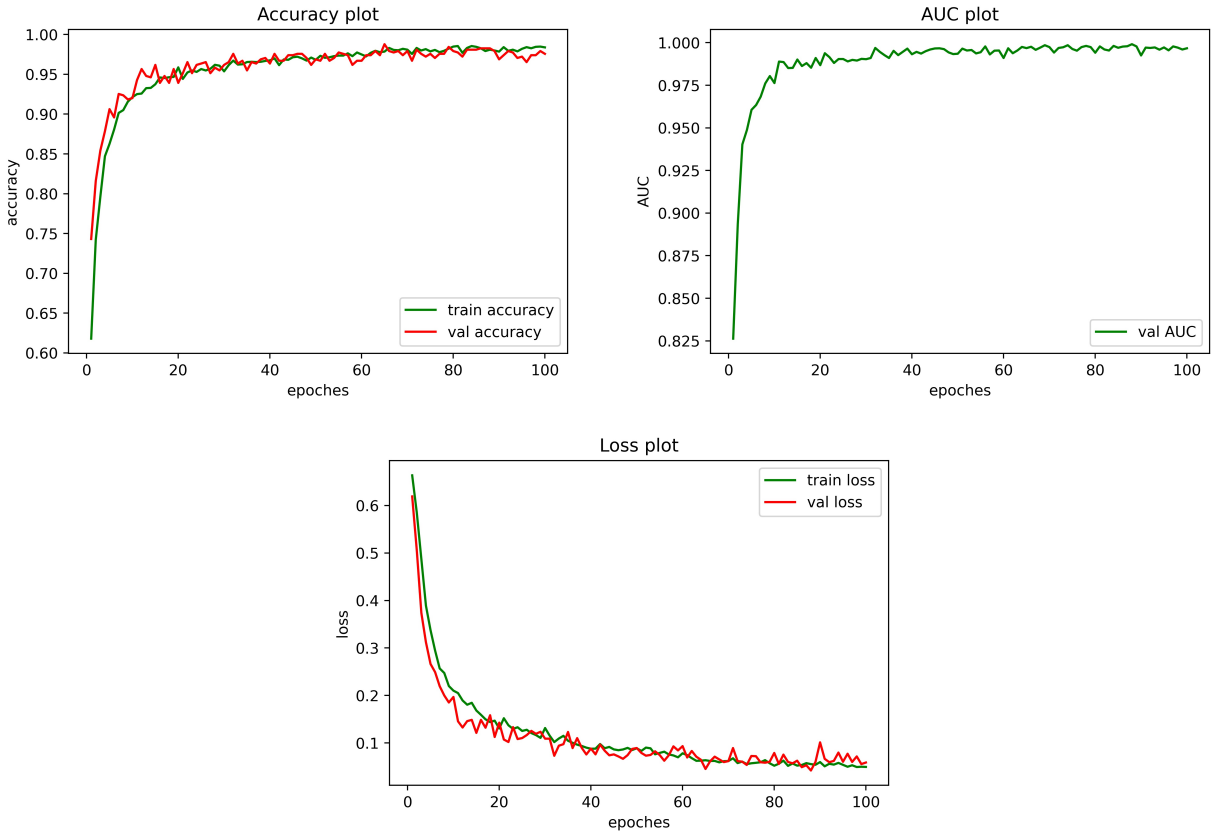


Figure 1: accuracy, AUC and loss of original data for 100 epochs

10 NOVEL IDEA: DATA AUGMENTATION

We used the following data augmentation techniques of our own which were not there in the original paper. The data augmentation techniques we use are- Rotation, Horizontal flip and Vertical flip, Colour jitter, Pixel dropout, Gaussian noise, Solarize, Spatter, Posterized, tosepia, Gaussian blur, Zoom blur.

11 EFFECT OF DATA AUGMENTATION TECHNIQUES

1. For the first method, Augmentations such as Rotation, Horizontal flip, Vertical flip, Colour jitter, Pixel dropout, Gaussian noise when combined with one of the methods gaussian blur or zoom blur. We observe the performance on the training data.

2. For the second method, the above data augmentation techniques are combined with Solarize, Spatter, Posterized, tosepia and it is observed that this does not perform well as compared to the first method. Then we observe the performance on the training data.

3. we use 20 epochs and batch size 16 to run the training phase and observe the train, test and validation losses, accuracy and auc scores.

4. The obtained test accuracy of the first method of augmentation is 95.66% whereas, the obtained test accuracy of the second augmentation is 91.49%. From this result we infer that the first data augmenation method is performing better than the second method.

   Therefore, the first data augmentation technique is the best augmentation technique. We use the first method to extrapolate it to 100 epochs using batch size 16 and obtain the final weights from the same, which is later used for testing the celeb-DF-V2 dataset.

12 SECOND PHASE RESULTS

1. When we trained for 20 epochs and batch size 32 using the first data augmentation technique, we obtain a training accuracy of 88.88%, validation accuracy of 89.24%, validation AUC score of 96.75, test accuracy of 91.28%., and test AUC score of 97.05.

2. When we trained for 20 epochs and batch size 16 using the first data augmentation technique, we got a training accuracy of 95.76%, validation accuracy of 97.75%, validation AUC score of 99.46, test accuracy of 95.66%, and test AUC score of 99.21.

3. When we trained for 20 epochs and batch size 16 using the second data augmentation technique, we got a training accuracy of 88.10%, validation accuracy of 92.53%, validation AUC score of 97.52, test accuracy of 91.49%, and test AUC score of 96.39.

4. Finally, Extrapolating it to 100 epochs and batch size 16 using the first data augmentation technique, we got a training accuracy of 99.19%, validation accuracy of 99.13%, validation AUC score of 99.85, test accuracy of 99.31%, and test AUC score of 99.95.

5. Testing the model with Celeb-DF-v2 with c23 videos using the weights obtained from the above training phase with 100 epochs and batch size 16, we get an AUC score of 92.87.

13 PLOTS

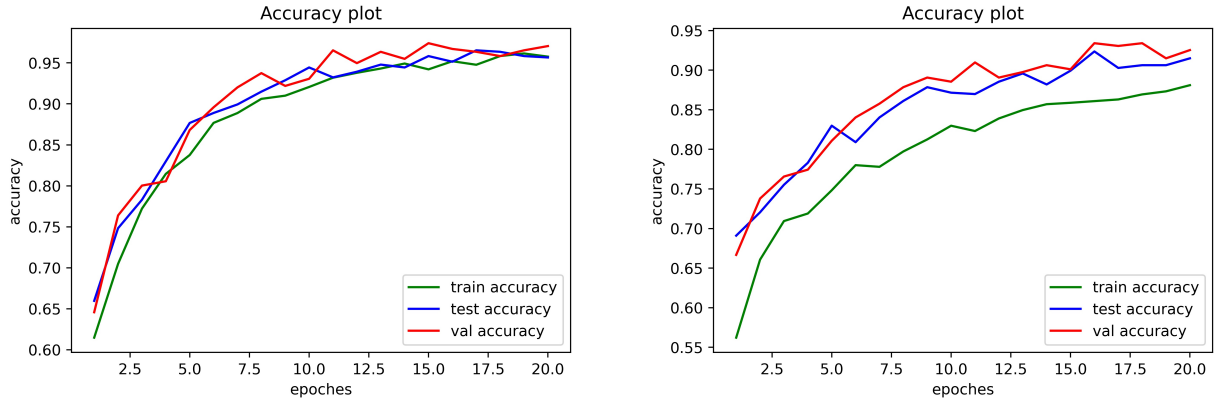


Figure 2: comparing accuracies of best augmentation (first data augmentation technique) with extra augmented (second data augmentation technique)

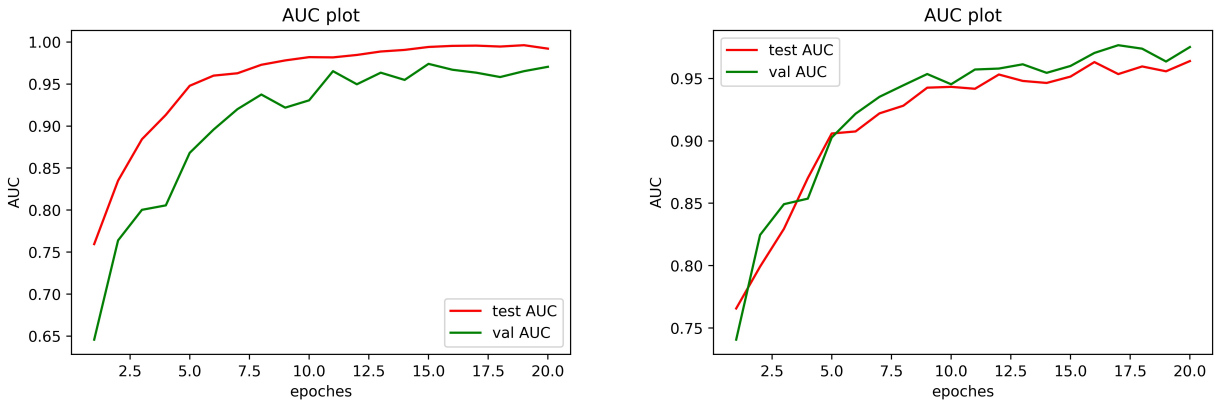


Figure 3: comparing AUC of best augmentation (first data augmentation technique) with extra augmented (second data augmentation technique)

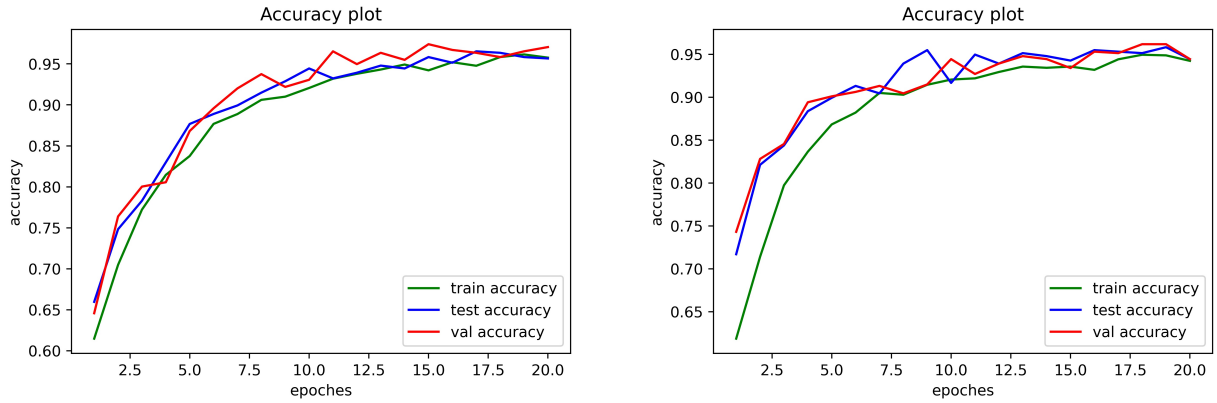


Figure 4: comparing accuracies of best augmentation (first data augmentation technique) with original augmentation

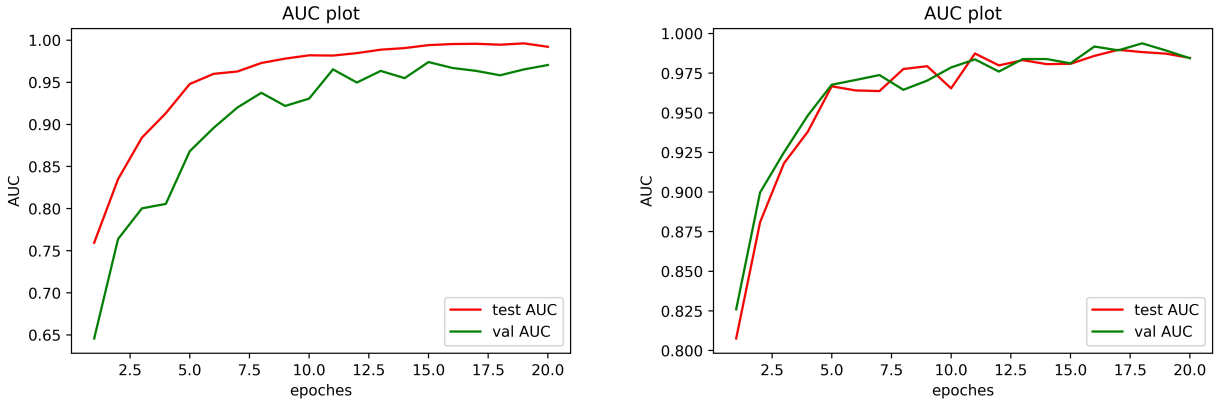


Figure 5: comparing AUC of best augmentation (first data augmentation technique) with original augmentation

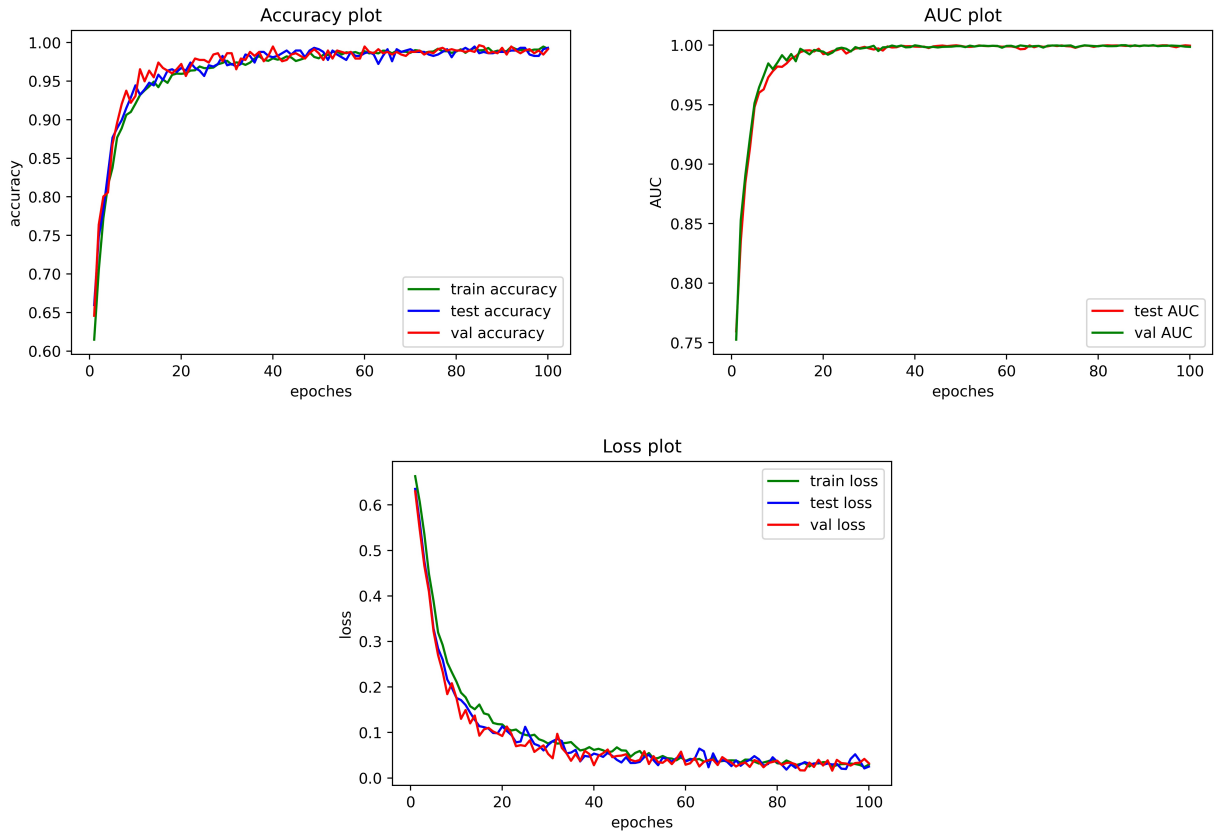


Figure 6: accuracy, AUC and loss using best data augmentation (first data augmentation technique) for 100 epochs and batch size 16

14 CONCLUSION

The novelty we proposed is the use of data augmentation techniques to improve the training and test accuracy for detecting deepfakes. From the above comparison plots and the results obtained in both the phases we conclude that the first data augmentation technique with 100 epochs and batch size 16 that we proposed performed better than the method proposed in the paper which uses 100 epochs and a batch size of 32. Hence, we achieve better results in both training and testing phase when we use the first data augmentation technique proposed by us as compared to the original paper.

self-blended images (SBIs), based on the idea that more general and hardly recognizable fake samples encourage classifiers to learn more generic and robust representations. SBIs are generated by blending pseudo source and target images that are slightly transformed from single real images to reproduce forgery artifacts. Using SBIs, we could train detectors without forged face images. Our proposed novelty performs better than the original paper.

15 REFERENCES

1. Kaede Shiohara, Toshihiko Yamasaki. Detecting Deepfakes with Self-Blended Images. https://arxiv.org/abs/2204.08376
2. Detecting Deepfakes with Self-Blended Images https://github.com/mapooon/SelfBlendedImages
3. Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks.
4. 81 facial landmarks shape predictor https://github.com/codeniko/shape_predictor_81_face_landmarks
5. Contributing data to deepfake detection research https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html
6. A beginner's guide to tmux
7. FaceForensics++: Learning to Detect Manipulated Facial Images
8. Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics
9. D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen. Mesonet: a compact facial video forgery detection network.
10. Liming Jiang, Ren Li, Wayne Wu, Chen Qian, and Chen Change Loy. DeeperForensics-1.0: A large-scale dataset for real-world face forgery detection.
11. Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. Celeb-df: A large-scale challenging dataset for deepfake forensics.