Detecting Fake Images using the Discriminator of GANs
MIT 6.867 Final Project, Fall 2019

Sujay Thakur **Charlotte Loh** **Bilha-Catherine Githinji**

Abstract

The rise of AI has accelerated the ability to create falsified images that look incredibly real, especially with the advent of Generative Adversarial Networks (GANs). In this work, we propose a simple model to detect fake images by taking advantage of the exact process that can be use to generate them, that is, by extracting the discriminator of a trained GAN. We show that the discriminator of a trained GAN performs decently well in detecting manipulated facial images, achieving an accuracy of 86.1% (AUC of 0.882), even though it has no knowledge of any of the manipulated image instances. We also benchmark its performance against two state-of-the-art fake image detection approaches, and show that these approaches work better only when the model has complete knowledge of all possible facial image manipulation techniques.

1. Introduction

The rapid improvements of image manipulation techniques has increased both the ease with which images can be manipulated and the difficulty in distinguishing between altered and real images. With the advent of Generative Adversarial Networks (GANs) in 2014, this process was greatly accelerated. Although image editing techniques can provide significant aesthetic or entertainment value, they are also increasingly being used with malicious intent. This raises significant societal concerns in areas such as the credibility of digital content and the spread of falsified information.

In this paper, we propose a GAN-based approach for image forgery detection, particularly, in the area of facial manipulation. An end-to-end GAN system is first trained to generate fake images; we then extract the discriminator of the GAN which will serve as our fake image classifier. This simple approach is based upon the idea that, since the GAN-based classifier replicates the generative process of the falsified images, it would be better able to identify the subtle nuances of the fake images and hence would serve as a good fake image detector. In order to benchmark the performance of our GAN-based classifier, we will also present other state-of-the-art forgery detection techniques, namely using a Convolutional Neural Network (CNN)-based classifier integrated with statistical feature extraction (N. Rahmouni, 2017) and a Support Vector Machine (SVM) that uses hand-crafted features (D. Cozzolino, 2014).

2. Methodology

2.1. Dataset

Facial manipulation methods can be broadly classified into two categories: facial expression manipulation and facial identity manipulation. In this work, we use the FaceForensics++ Dataset (Rössler et al., 2019) to train and test our GAN-based forgery detector and benchmarks. In particular, this dataset consists of four manipulation techniques: manipulations based on classical computer graphics-based methods *Face2Face* and *Faceswap*, and learning-based approaches *DeepFakes* and *NeuralTextures*. This large-scale dataset includes 1000 original video sequences from YouTube as well as versions of them being manipulated using the above four techniques. In the interest of computational efficiency, our main analysis uses a single video, where we extracted 390 frames and pre-processed them to obtain 450 x 450 pixel images of a single subject, Barack Obama. Examples of the pre-processed original images as well as the frames manipulated from the four techniques are shown in Figure 1. Even though our main analysis considers only a single subject, we will later also briefly discuss the generalization of our GAN-based classifier to other subjects of different ethnicity and gender.

2.2. Benchmark 1: CNN-based approach

Before discussing our GAN-based classifier, we first discuss the approach used in the first of our benchmarks, the CNN-based classifier adopted from (N. Rahmouni, 2017). In the original paper, the CNN-based classifier was used to distinguish between photographic (PG) and computer graphics (CG) images. As shown in Figure 2, the image was first decomposed into N smaller patches of resolution 100 x 100 pixels (the authors argue that this decomposition helps in the detection of local splicing and allows for computational

Figure 1. Test and training samples. Going from top-left to bottom right, belonging to the Original, DeepFakes, Face2Face, FaceSwap and NeuralTexture frames.

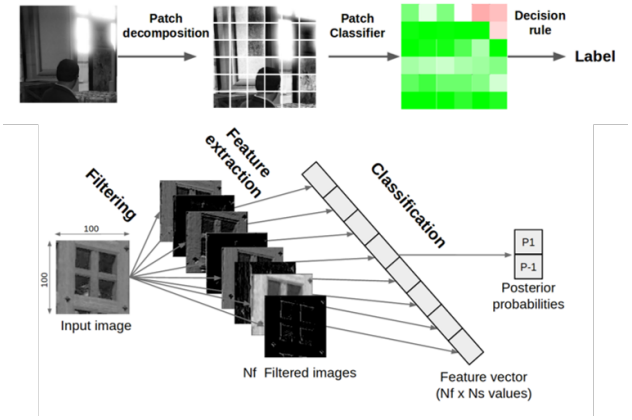

Figure 2. Approach and architecture of the CNN-based detector adapted from (N. Rahmouni, 2017) showing the patch decomposition as well as patch classification

savings). An image was then classified as PG or CG according to a decision rule; in our case, we use a weighted voting scheme where each patch contribution is the log likelihood of the label:

$$y_{pred} = \text{sign}\left(\sum_{i=1}^{N} \log \frac{\mathbb{P}(Y=1|X_i=x_i)}{\mathbb{P}(Y=-1|X_i=x_i)}\right) \quad (1)$$

where Y and X_i are the labels and patches, and x_i the real observations. For each small patch, the usual 3-step procedure of filtering, statistical feature extraction and classification was used. In this work, four statistical features were used: the estimate mean, estimate variance, maximum and minimum of the pixel values in that patch. By adopting a transfer learning approach, we will discuss in section 3 how this CNN-based classifier can be adapted to perform forgery detection on our single-subject FaceForensics++ dataset. We will then compare this against our GAN-based approach.

2.3. Benchmark 2: SVM-based approach

The next approach that our benchmark suite included was a method based on an image forgery detection algorithm that place first in the 2013 Image Forensics Challenge (D. Cozzolino, 2014). The aim of this approach was to distinguish between the real and forged images based on their noise characteristics. The technique leveraged features extracted from residual images that summarized statistical properties within neighborhoods of pixels. These features were inspired by those developed by Fridrich et al. (J. Fridrich, 2012) to design residual noise models for image steganalysis, a form of digital forensics. Once extracted from a real or fake image, the steganalysis features were labeled and provided as input to a standard SVM classifier for training. We found that the method reasonably discerned between real and fake within categories of forged images. Section 3 details our findings.

2.4. GAN-based approach

The GAN-based classifier aims to replicate the generative process of the falsified images. Figure 3 shows the layout of a typical GAN system, originally proposed by Goodfellow (Goodfellow et al., 2014), which consists of two networks: the generator and the discriminator. The generator transforms noise into a sample, and the distribution of these samples should ideally be the same as the training distribution. The discriminator is trained to differentiate the real training samples from these generated fake samples. Both of these networks are then trained against each other to improve their generative and discriminative performance respectively. In our case, an end-to-end GAN system is first trained to generate fake images, and the discriminator is then extracted to classify real and fake images, with the hope that since it has been trained alongside the generative process, it would be better able to identify the subtle nuances of the fake images.

Figure 4 shows the generator and discriminator architectures used in the overall GAN system. Note that the generator upsamples a lower dimensional noise vector into the same dimensions as the training images, while the discriminator follows the architecture of a standard CNN.

3. Results and Discussion

In this section, we will first present and discuss how the two benchmark models, specifically the CNN-based approach and the SVM-based approach, were modified and applied

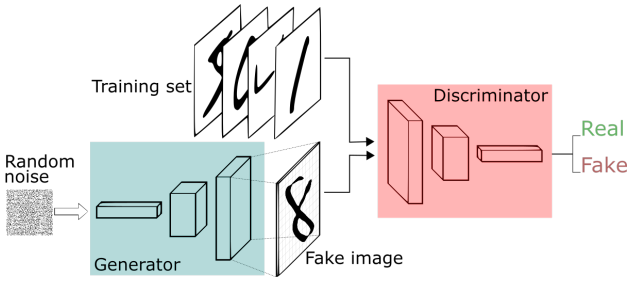

Figure 3. Layout of a typical GAN (Nicholson, 2019).

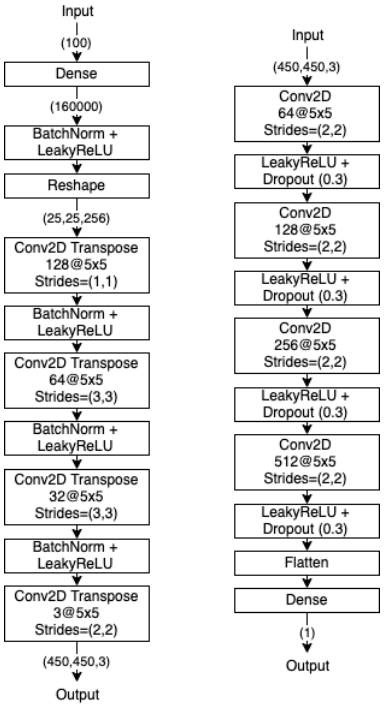

Figure 4. Generator (left) and discriminator (right) architecture of the GAN.

to our single-subject dataset. Only a single subject was used due to limited computation resources. From these we obtain performance metrics that we will use to compare our GAN-based approach against. We will then discuss how the discriminator was extracted and discuss how it performs in contrast to our benchmarks. Finally, we will also briefly study how well our GAN-based classifier performs on other subjects of different ethnicity and gender from our single-subject.

3.1. Benchmark 1: CNN-based approach

3.1.1. Transfer Learning applied to our Dataset

In order to circumvent the small sample size and limited variability of our single-subject FaceForensics++ dataset, a transfer learning approach was adopted. The CNN-based classifier discussed in section 2.2 was first pre-trained on the same datasets discussed in the original paper, leveraging on the availability of the large corpora of 1000 Photographic (T.-T Ng, 2004) and 1000 Computer graphics (Piaskiewicz, 2017) images in the respective referenced datasets. We used the optimal hyperparameters presented in the original paper for training on 15k epochs. These pre-trained weights were then transferred, and the model was further trained on our single-subject FaceForensics++ dataset. As seen in Figure 5, transfer learning (indicated in the yellow and orange lines) improved the validation accuracy significantly over training the CNN classifier from randomly initialized weights (indicated in the blue and grey lines). We also explored varying the patch size, specifically 50 and 100 pixels in width. It was found that 100 pixel patches gave significantly better validation accuracy. A possible reason for this could be because most manipulation techniques are mainly focused around the facial area of the subject (see Figure 1) and hence having pixels too small would tend to over-emphasize the areas of the images that were not manipulated.

3.1.2. Generalization to other Manipulation Techniques

We first considered training the classifier only on the *Original* and a single manipulated class (*DeepFakes*) in order to simulate the realistic situation in which one might not have knowledge of all manipulation techniques available and hence study how well this classifier will be able to generalize to unknown or newly developed manipulation techniques. The patch decision labels (visualization of Equation 1) from the pre-trained CNN classifier using 100-pixel patches and trained for further 20k epochs on our single-subject dataset are visualized in Figure 6. The validation accuracies tested on individual manipulated classes are shown in brackets (in black font). As evident, the model was able to predict the *Original* and *DeepFakes* test images with perfect accuracy, but fails to generalize well to any other manipulation techniques. This suggests over-fitting to the *DeepFakes* dataset and hence we also explored stopping the training much earlier, to 2k epochs (indicated in the pink line in Figure 5). This improved the prediction accuracy for the other manipulation techniques (accuracy indicated in pink in Figure 6). Interestingly, the model was able to detect images manipulated by the *NeuralTextures* technique extremely well, even though this manipulation technique is hard for the human eye to detect (see Figure 1). This

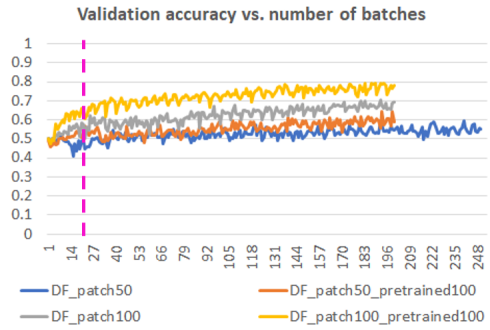

Figure 5. Validation accuracy against number of batches (axis in hundreds) with the use of pre-trained vs randomly initialized weights. Blue: 50-pixel patch (random); Grey: 100-pixel patch (random); Orange: 50-pixel patch (pre-trained); Yellow: 100-pixel patch (pre-trained)

Figure 6. Decision labels from the CNN-based classifier. The intensity of the green (red) patches shows the posterior probability for the patch belonging to the real (manipulated) class.

could be because the CNN model was heavily pre-trained on the CG dataset, allowing it to detect computer generated images very well. Compared to other manipulation techniques (which mainly involve face swapping between different subjects, i.e. less graphics-based generation), the *NeuralTextures* manipulation technique relies more heavily on computer generated graphics (i.e. learned feature maps to synthesize photo-realistic images).

3.1.3. Complete Data Training

Finally, we also explored training the CNN classifier using the complete dataset, i.e. including all four manipulation classes, *DeepFakes*, *Face2Face*, *FaceSwap* and *NeuralTextures*, as well as the *Originals* for both training and validation. Here we assume we have complete knowledge of all possible manipulation techniques and train a classifier to discriminate between two classes: fake (with samples from all four manipulation techniques) and real (samples from the original images). The test accuracy with this complete data set is shown in Table 1, where we contrast against the accuracy metrics we obtained from section 3.1.2 when we assumed we had an incomplete knowledge of all manipulation techniques. We again used the pre-trained weights and trained for 4k epochs (where the validation accuracy was seen to reach a plateau). As expected, the test accuracy in the complete data case were significantly higher for all the classes. This is unsurprising since all instances of manipulation techniques are now included in the training set and hence the CNN classifier need not generalize to discern manipulated images that it has never seen before.

Table 1. Test accuracy for each manipulation class when the CNN model is trained only on the DeepFakes (plus Originals) dataset vs when trained with the complete dataset.

Fake image class	Accuracy (DeepFakes only)	Accuracy (Complete Data)
Originals	0.795	0.914
DeepFakes	1.000	0.971
Face2Face	0.297	0.671
FaceSwap	0.243	0.986
NeuralTextures	0.831	1.000
Total	0.686 (AUC=0.746)	0.914 (AUC= 0.972)

3.2. Benchmark 2: SVM-based approach

3.2.1. Computing Steganalysis Features From the Dataset

As introduced in section 2.3, the SVM-based approach consisted of two key components: steganalysis feature extraction and SVM classifier. First, we used a variety of highpass filters to generate residual images. We explored both first and second order filters, independently, but chose to use a Sobel filter along the vertical direction because it was simple to compute and captured better structure within images. Figure 7 shows examples of the first frame from the original image dataset and each manipulated image dataset. The computed residuals underwent a post processing phase in preparation of feature extraction. We applied k-means quantisation and truncation to the residual values, as suggested in (J. Fridrich, 2012) (see Table 3). Then each image was decomposed into M-by-M patches from which we computed Grey-level Co-occurrence Matrices (GLCM). We tested various values for M and found that 32 was a good choice for this dataset. Smaller values resulted in drastically higher training times. Much like histograms, these matrices account for the occurrence of grey-level values

Table 2. Statistics computed from GLCM. The indices i and j are the grey-levels of pairs of pixel neighbors. The weight $P_{i,j}$ is an element in the co-occurrence matrix.

STATISTIC	FORMULATION
CONTRAST	$\sum_{i,j=0}^{levels-1} P_{i,j}(i-j)^2$
DISSIMILARITY	$\sum_{i,j=0}^{levels-1} P_{i,j}\|i-j\|$
HOMOGENEITY	$\sum_{i,j=0}^{levels-1} \frac{P_{i,j}}{1+(i-j)^2}$
CORRELATION	$\sum_{i,j=0}^{levels-1} P_{i,j}\left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}\right]$

observed between a pair of neighboring pixels. This computation required careful selection of patch size and relative angular configuration of a reference pixel's neighbor. We chose values shown in Table 3. Figure 8 shows a summary of this feature extraction pipeline. From the per-patch co-occurrence matrix we gathered the following GLCM statistics: dissimilarity, homogeneity, correlation, and contrast, shown in Table 2. Each image was transformed into a set of GLCM feature vectors, one for each statistic, through this process. A soft margin SVM classifier was trained on this set of feature vectors marked by real or fake label.

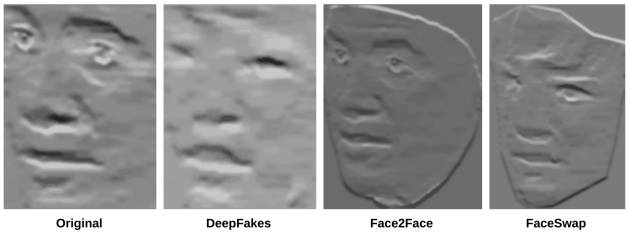

Figure 7. Example of residual images computed using Sobel filter along vertical direction. This example shows the first frame's results from each sequence. For the sake of efficiency, a manipulation mask was applied to generate residual images for the real frame and fake frame for each category.

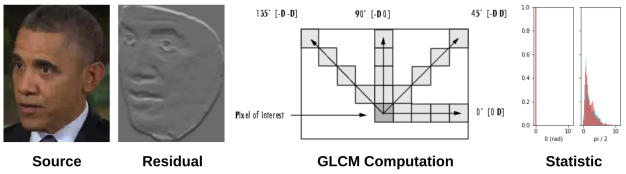

Figure 8. A summary of the feature extraction pipeline. Highpass filters were used to compute residual from input images. GLCMs were generated for multiple pixel neighborhoods in the residual images. Finally, co-occurrence statistics were computed per neighborhood. The right most plot shows a histogram of dissimilarity statistic across all image patches in the *Face2Face* dataset.

Table 3. Feature extraction parameters used in SVM approach.

PARAMETERS	VALUES
QUANTISATION STEP	5
TRUNCATION THRESHOLD	0.2
PATCH SIZE	32, 64
ANGULAR ORIENTATION	0, 1/2 π, 2/3 π, 4/3 π
HIGH-PASS FILTERS	SOBEL-X, SOBEL-Y, GABOR, LAPLACIAN

3.2.2. TUNING SVM FEATURES AND PARAMETERS

The training and testing performance for SVM was measured with respect to different choices of GLCM features, SVM hyper-parameters, and selection of training data.

Based on Sobel residuals we studied the effect of different steganalysis features on classification accuracy. For these tests we set the SVM classifier hyper-parameters to $C = 1$ for the slack variable and with no kernalisation. Table 4 shows that the contrast and dissimilarity metrics yielded the highest performing classifiers at 80% across the entire Obama dataset while others achieved less than 60% accuracy. We noticed that the dissimilarity variant out-performed the contrast variant by more than 5% on the *DeepFakes* and *FaceSwap* categories, but under-performed by less than 3% on the other categories. The lowest performance overall occurred on the troublesome *NeuralTextures* data with 72% accuracy for contrast-based SVM, the best classifier for *NeuralTextures*. When generalizing across techniques, however, we saw a significant drop in performance.

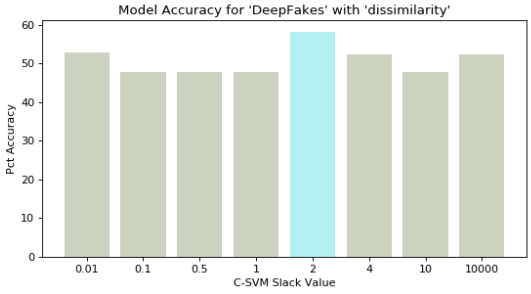

Figure 9. Parameter sweep of C-SVM slack variable.

3.2.3. GENERALIZATION TO OTHER MANIPULATION TECHNIQUES

To test the generality of this model we trained it on one category and evaluated its performance on the types of forged images. Again, we kept the SVM classifier hyper-parameters fixed and used the dissimilarity metric for feature extraction. Figure 11 shows the performance of this model for different training sets. We also ran more experiments to

Table 4. Accuracy of SVM models trained with different GLCM statistics computed from each dataset.

DATASET	DISSIMILARITY	HOMOGENEITY	CORRELATION	CONTRAST
DeepFakes	**0.999**	0.952	0.611	0.592
Face2Face	**0.980**	0.576	0.568	0.600
FaceSwap	**0.995**	0.982	0.816	0.721
NeuralTextures	**0.806**	0.478	0.467	0.612
Full Dataset	**0.930**	0.748	0.608	0.634

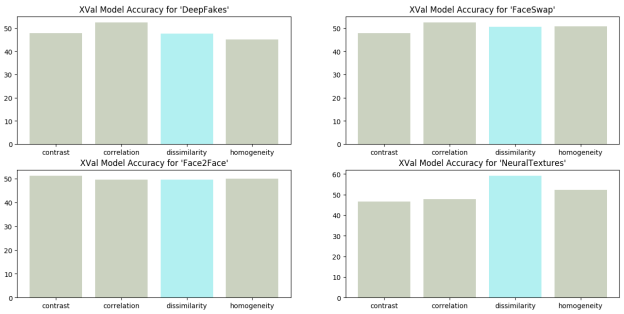

Figure 10. Performance for models trained with different GLCM statistics on single manipulation dataset and tested on other manipulation datasets. For example, the top-left plot shows performance of SVM model trained on *DeepFakes* dataset and tested on *FaceSwap*, *Face2Face*, and *NeuralTextures*.

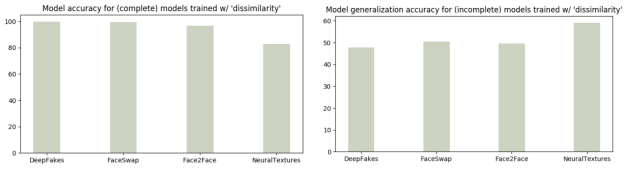

Figure 11. Performance of (left) SVM models trained on the full Obama dataset and tested on individual manipulation dataset and (right) models trained on single manipulation dataset and tested on other manipulation datasets.

verify our selection of GLCM features. The results in Figure 10 indicate that the correlation metric generally performs better than dissimilarity by almost 3%, depending on the training category.

In an attempt to improve generalization, we tuned the classifier using cross-validation. A parameter sweep of C-SVM slack variable showed that C=2 was the best choice over the entire Obama dataset, as seen in Figure 9. Brief experimentation with RBF kernel, e.g., setting γ to either $1/\#samples$ or 0.5, did not ameliorate the situation. Ultimately, we expect that supplementing the training corpus with significantly more video frames capturing both a broader a) range of movement by the subject and b) range of subjects would increase generalization capabilities of SVM models.

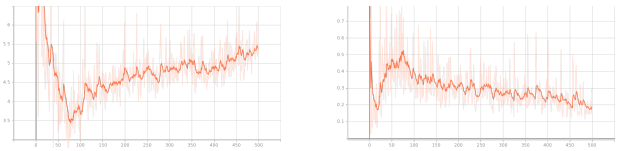

Figure 12. Generator (left) and discriminator (right) loss against training epochs.

3.3. GAN-based approach

3.3.1. Optimal stopping condition

Figure 12 shows the generator and discriminator losses as training progresses. Note that in typical GAN implementations, we would be interested in the generator, and hence would stop training when the generator is optimal. However, we are now interested in the discriminator, so it is not immediately obvious what the optimal stopping condition should be.

From Figure 12, we note that the discriminator loss has a minimum at epoch 20, hence it is reasonable to assume that this could be a potential stopping point. However, we found that the convolution kernels in the discriminator had barely been updated from their initialization at this point, and hence had not learned anything useful. This shows that finding the minimum discriminator loss was not a good method to determine the optimal stopping point for our case.

We thus chose to tune this by tracking various other metrics as training progresses, shown in Figure 13. The first metric, validation accuracy, tracks the accuracy of the discriminator at every 10 epochs on the real and fake (all four classes) validation sets. This would be a good indication of how well the model was generalizing. Next, note that we can create a histogram of the discriminator output for real and fake images, as shown in Figure 15, and we would ideally like a large separation between the two classes because that signifies that the discriminator is distinguishing well between real and fake images. The remaining three metrics, correlation, intersection and Bhattacharyya distance were used to compute how well-separated both histograms were. For histogram H_1 and H_2, these metrics are:

$$(H_1, H_2)_{corr} = \frac{\sum_i (H_1(i) - \bar{H_1})(H_2(i) - \bar{H_2})}{\sqrt{\sum_i (H_1(i) - \bar{H_1})^2 \sum_i (H_2(i) - \bar{H_2})^2}}$$

$$(H_1, H_2)_{int} = \sum_i \min(H_1(i), H_2(i))$$

$$(H_1, H_2)_{BD} = \sqrt{1 - \frac{1}{\sqrt{\bar{H_1}\bar{H_2}N^2}} \sum_i \sqrt{H_1(i)H_2(i)}}$$

where $\bar{H_k} = \frac{1}{N}\sum_j H_k(j)$ and N is the total number of histogram bins. We want high validation accuracy, low correlation, low intersection and high Bhattacharyya distance. From Figure 13, using a combination of these metrics suggests 90 epochs to be the terminating point for the ideal discriminator.

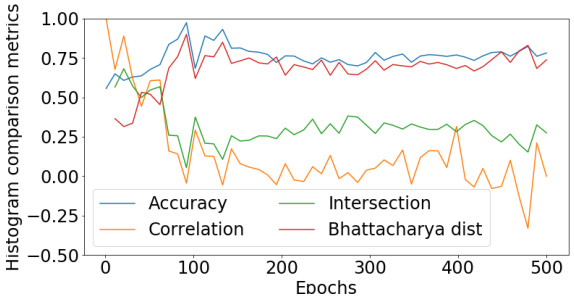

Figure 13. Validation accuracy, correlation, intersection and Bhattacharyya distance during training.

3.3.2. Performance

Figure 14 shows the generator outputs as training progresses. We are able to converge to good images.

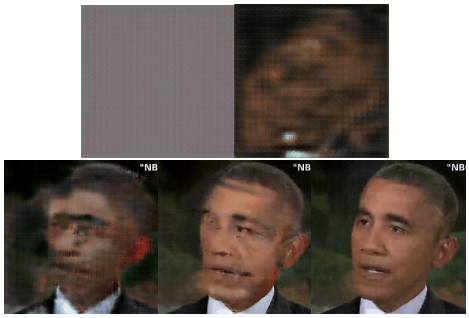

Figure 14. Samples from the generator as training progresses (top-left to bottom-right.

We then extract this discriminator to classify real and fake images. Figure 15 (top) shows the discriminator outputs for real and fake images (all four classes aggregated together) using the ideal stopping condition, and we see a good separation of the two histograms. It was found that using the average of the means of the two clusters as a threshold for classification gave the best test accuracy, which was 86.1%. Figure 15 (bottom) shows the output of the discriminator separated by each fake image class, including fake images generated by the GAN's generator itself. We can see that the discriminator best distinguishes real images from the generator's fake images. This is to be expected since the discriminator was specifically trained alongside this generator, and is hence best attuned to the nuances of these fake images. We also see that the discriminator does the worst when distinguishing between real images and *NeuralTextures* images. This could be because comparing Figures 1 and 14, we see that the generator's images are very different from the *NeuralTextures* images, and hence the discriminator is not well trained for this. It should also be noted that the *NeuralTextures* images are arguably the best fake images, since they look very realistic to even the human eye.

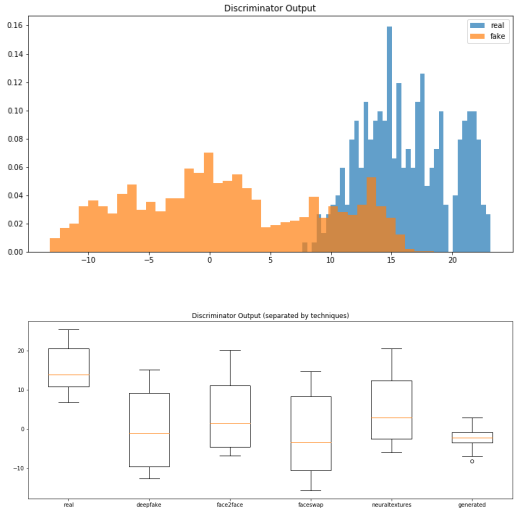

Figure 15. Top: Output of the discriminator separated by real (blue) and fake (orange) images. Bottom: Output of the discriminator for real and all fake images, showing real, DeepFakes, Face2Face, FaceSwap, NeuralTextures and images generated by the GAN's generator (left to right).

3.4. Comparison with Benchmarks

Table 5 compares the relevant accuracies and Areas Under ROC Curve (AUC) for all methods. We see that in the presence of perfect adversarial information, where we know all manipulation techniques we are up against and have sufficient samples from all these techniques, traditional methods like CNNs and SVMs outperform GANs. However, we ar-

gue that this assumption does not hold well in real situations. It is reasonable to assume that the adversary is able to generate manipulated images using new methods that we are not aware of, and in this situation, a GAN provides much better performance. It is able to achieve good performance without seeing any manipulated images at all. Additionally, we observed in Section 3.3.2 that it performed best with fake images similar to its own generated images. With manipulation techniques becoming increasingly GAN-based in recent years (Karras et al., 2018), this only works as an advantage for the GAN-based method.

Table 5. Test accuracy and AUC for all methods. "Incomplete" model refers to training on only the Deepfakes (NeuralTextures) dataset for CNN (SVM) approaches

METHOD	ACCURACY	AUC
CNN (COMPLETE)	0.914	0.972
CNN (INCOMPLETE - ONLY DF)	0.686	0.746
SVM (COMPLETE)	0.947	0.948
SVM (INCOMPLETE - ONLY NT)	0.591	0.603
GAN	0.861	0.882

3.5. Generalization to other subjects

Since our main analysis focussed only on a single subject, Barack Obama, we also studied how well the GAN-based approach would work on other types of faces. The GAN was re-trained individually for three other subjects of different ethinicity and gender. Figure 16 shows the discriminator output separations while Table 6 reports the accuracies and AUC achieved for them. As evident, the GAN-based approach works decently well across all types of faces, showing accuracies and AUCs in a similar range. Hence we conclude that the GAN-based approach works well on most types of face structures. We also note that the high accuracy observed in Subject B could possibly be due to the lower number of frames in the training set - while the other subjects had 390 frames, subject B only had 290 frames available in the dataset.

Table 6. Test accuracy & AUC for GAN approach applied on various types of faces. Faces of subjects are shown in Figure 16

FACE	ACCURACY	AUC
SUBJECT A	0.813	0.838
SUBJECT B	0.924	0.850
SUBJECT C	0.871	0.855

4. Conclusion

In this paper, we proposed a method to detect manipulated images using the discriminator of a GAN, with the hopes

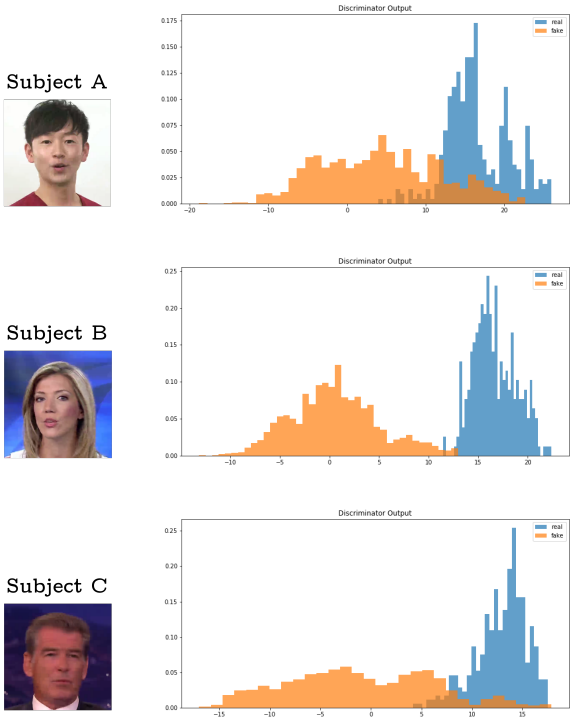

Figure 16. Output of the discriminator separated by real (blue) and fake (orange) images for other subjects of different ethnicity and gender, whose facial image is shown alongside.

that since it was trained alongside the generative process, it would be better attuned to the nuances of fake images. We then compared this to benchmarks of two state-of-the-art fake image detection approaches, a CNN-based approach and a SVM-based approach. We found that these two methods perform better than the GAN only when we assume perfect adversarial information. In the more realistic scenario where we would not know the manipulation techniques and would not have access to these fake images beforehand, the GAN-based method performed much better since it did not require access to any fake image. In future, with access to more computation resources, we will consider training the methods on larger data sets with more variance in the images to check how well they generalize.

5. Contributions

S.T. came up with the idea of extracting the GAN's discriminator as a fake image classifier and proposed the FaceForensics++ dataset. He implemented the GAN model and perfected the discriminator outputs. C.L. extracted the datasets and worked on the CNN-based approach. The results and performance metrics of the GANs were produced in collaboration between S.T. and C.L. B.G worked on the SVM-based approach. All three members contributed to the final report.

References

D. Cozzolino, D. Gragnaniello, L. V. Image forgery detection through residual-based local descriptors and block-matching. *In IEEE International Conference on Image Processing*, pp. 5297–5301, 2014.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks, 2014.

J. Fridrich, J. K. Rich Models for Steganalysis of Digital Images. *IEEE*, pp. 5297–5301, 2012.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018. URL http://arxiv.org/abs/1812.04948.

N. Rahmouni, V. Nozick, J. Y. I. E. Distinguishing computer graphics from nautral images using cnn. *2017 IEEE WIFS*, 2017. doi: 10.1109/WIFS.2017.8267647.

Nicholson, C. *A Beginner's Guide to Generative Adversarial Networks (GANs)*. 2019. URL https://skymind.ai/wiki/generative-adversarial-network-gan.

Piaskiewicz, M. Level-design reference dataset, 2017. URL http://level-design.org/referencedb/.

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.

T.-T Ng, S.-F. Chang, J. H. M. P. Columbia photographic images and photorealistic computer graphics dataset. *ADVENT Technical Report, Columbia University*, (205-2004-5), 2004.