GAN Generated Deepfake Identification and Transferability of Deepfake Detectors

Zhongqi Ma, Vidya Silai, Zeyu Chang, Parth Raut
University of Michigan
500 S State Street, Ann Arbor, MI
mzhongqi@umich.edu, vsilai@umich.edu, zychang@umich.edu, praut@umich.edu

1. Introduction

1.1. Motivation

With the rapid technological advancement throughout the recent years, there has been a consistent development in the generation of deepfakes. Deepfakes are synthetic images or videos that mimic a real person. These synthetically generated media contents are constantly used maliciously to generate fake news, abusive content, and other fraudulent material, causing financial impacts and misconceptions for the general audience. The malicious use of deepfakes not only have deceptive influences, but are becoming extremely hard to spot. With the continuous rise in the power of media, being able to detect the existence of deepfakes has become an important goal.

1.2. Objective

Our project aims to analyze the images generated by Generative Adversarial Networks—a common AI technique used to create deepfakes. We look to create a Convolutional Neural Network for GAN generated deepfake recognition that can differentiate real images from images generated by a particular GAN. We also look to determine the transferability of our CNN by training it against a GAN generated image dataset, and testing it against an image dataset generated by a different GAN.

1.3. Related Work

Other studies such as [4], [8], and [9] demonstrate the idea that certain preprocessing tricks are leverageable when trying to detect a deepfake. While deepfake detection using various computer vision techniques is not a new endeavor, the idea of preprocessing the images fed into the CNN using some form of frequency transformation like a DCT has been experimented with a few times. These transforms help elucidate what researchers have deemed a GAN's "fingerprint": uniquely identifiable artifacts that a GAN leaves in the images it generates that a real image would not contain. Moreover, we aim to explore this idea of a GAN fingerprint through our project.

2. Approach

2.1. Dataset Generation

The first part of this project involves generating our own dataset. The plan is to generate fake face images from a pre-trained StyleGAN. Due to tensorflow compatibility issues, we tried several online implementations and successfully used [1] to generate 5000 fake face images. We also found another 5000 real face images of the same size from the Flickr-Faces-HQ dataset. These form the dataset we will work on.

2.2. Classification Model

To get a good classification model, we made several attempts, all using 10 epochs of training. In any case, we constructed an 8-layer CNN inspired from [3]. We used the Adam optimizer with CrossEntropyLoss() as the loss function, a weight decay of 1e-5 and a learning rate of 1e-3. A diagram for the architecture of our CNN is shown in Figure 1.

2.3. Model Training and Transferability

Our first attempt directly fed our images into our CNN without any preprocessing. Unsurprisingly, this gives very bad results and the network is not properly trained. We then attempted to reduce the image size to (256, 256, 3), which also didn't work.

Next, we moved on to analyze the frequency difference between real and fake images. Specifically, we replicated the preprocessing ideas from [4] and generated DCT (Discrete Cosine Transform) images (1 channel grayscale images) on our dataset using OpenCV. Moreover, we hoped to test the idea that GANs leave artifacts that are visible throughout their frequency spectrum; when analyzing their

Layers	Image shape
Input	1024, 1024, 3(1)
Conv2d, BatchNorm2d, MaxPool2d	512, 512, 32
Conv2d, BatchNorm2d, MaxPool2d	256, 256, 64
Conv2d, BatchNorm2d, MaxPool2d	128, 128, 128
Conv2d, BatchNorm2d, MaxPool2d	64, 64, 256
Conv2d, BatchNorm2d, MaxPool2d	32, 32, 256
Conv2d, BatchNorm2d, MaxPool2d	16, 16, 512
Conv2d, BatchNorm2d, MaxPool2d	8, 8, 512
Conv2d, BatchNorm2d, MaxPool2d	4, 4, 512
Flatten, FC	512
FC	2

Figure 1. Architecture of CNN

frequency spectrum in comparison to the real images, there could be evidence of stronger high frequency components. Using this idea, we used the same network to train on the DCT images. This gives us satisfying validation and testing accuracy as described in the Experiments section.

Finally, to check if our trained model is transferable to other GAN-generated images, we generate another set of images using StyleGan-XL [5], another GAN network. We run our trained model on this dataset to see the accuracy. If the accuracy is close to the testing accuracy on our original dataset, we will claim that our model is transferable to other GAN-generated images. If the model does no better than random guessing (i.e., 50% accuracy), then it is not transferable.

3. Experiment

In total, we executed three different experimental setups to explore the idea of a GAN "fingerprint". Throughout all these experiments, we used a dataset comprising 10,000 1024x1024 images total to train the model. In order to maintain class balance in this binary classification model, we had 5,000 images from each class (real and fake) so that the training process would not be affected by suboptimal class representation. The real images were taken from the FFHQ dataset as described above, which we chose since it is considered a benchmark dataset that many industry grade GANs (including the pretrained models we used to generate fake data) are trained on. Also, we generated another 5,000 fake images from the StyleGAN-XL model for testing transferability of models. One important aspect of all these images was that they all conformed to the same "style": they all consisted of a person's face taken up close with the background blurred, which is important in allowing for the classifier to use only the face (and not perhaps the background) in making classification decisions. All images were normalized before being used in the CNN.

Figure 2. Raw Image: real

Figure 3. Raw Image: StyleGAN2-ADA generated

In addition to using this dataset throughout the three experiments, we also utilized accuracy as our quantitative metric for evaluating the success of our approaches. Given that all of our experiments were set up as a binary classification problem with a class-balanced dataset, we felt that this was an appropriate metric. One downside, however, of using accuracy as our sole metric is that it does not capture the nuances in the mistakes that model makes when predicting incorrectly. A measure like AUROC would have been more useful in understanding these nuances, as it captures the false positive rate in its value, which could have been

Figure 4. Raw Image: StyleGAN-XL generated images

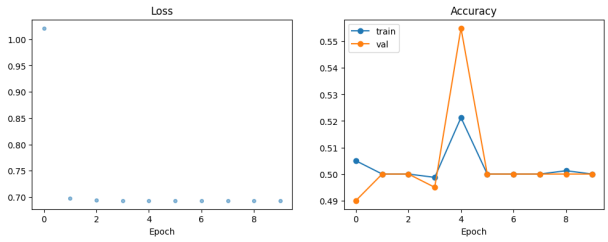

Figure 5. Training and Validation Curves for Classification

useful in helping adjust the model throughout our experimentations.

3.1. Experiment I: Classifying Raw Images

In this first experiment in which we did not preprocess the images at all, we saw the model produce an extremely poor accuracy. As demonstrated by the training and validation curves in Figure 5, we achieved a maximum validation accuracy of around 0.56, and a final test accuracy of 0.5; this meant that model was essentially guessing, and therefore not actually learning anything at all. Given that there were really no distinguishing features between the real and fake images in regards to facial features, we then leaned into the frequency analysis idea to see if a different view of the images would help model performance.

3.2. Experiment II: Classification DCT-Transformed Images

In this next experiment, we transformed all 10,000 raw images using DCT. Before running the CNN, we examined

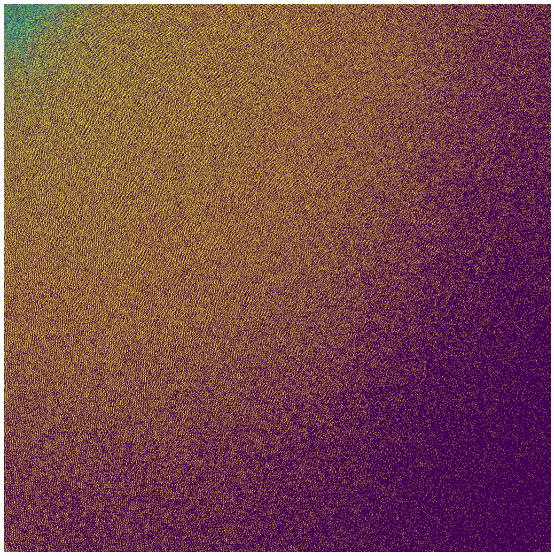

Figure 6. Fake Image DCT Heatmap

the DCT-transformed images and applied a heat map to see if we could discern any noticeable "fingerprint" with the human eye (Figures 6 and 7). The example heat maps shown demonstrate that the fake images have some more high frequency artifacts compared to the real images and less lower frequency artifacts. Then, we moved on to actually running the CNN. As shown in the training and validation curves in Figure 8, we were able to significantly improve our model's performance, achieving validation accuracies in the high 90% range and a final test accuracy of 0.99. Furthermore, this helps validate the idea that doing some sort of frequency-based training can be more effective in allowing the model to learn something meaningful.

3.3. Experiment III: Transferability of a Frequency-Based Model

For our final experiment, we aimed to use the model trained in Experiment II to discern whether the fingerprints learned from one GAN architecture were transferable in recognizing the fake images produced by another GAN architecture. After first inspecting the heat map images (Figure 9) in a similar manner to Experiment II, we did indeed notice some subtle differences in the fingerprint patterns generated by the StyleGAN-XL, which we also observed producing more realistic images than the first GAN. When actually evaluating the model's performance on classifying the StyleGAN-XL generated images as fake, we attained a performance of 74% accuracy. Therefore, this highlights the idea that different GANs potentially produce different fingerprints and that GAN architectures that fall under the same general umbrella (StyleGAN-XL and StyleGAN2-

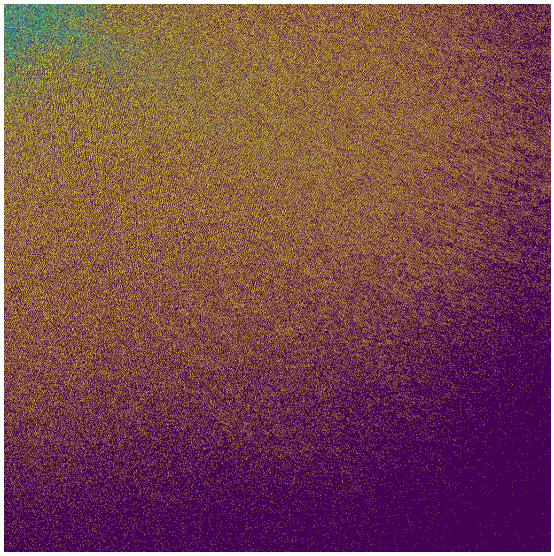

Figure 7. Real Image DCT Heat Map

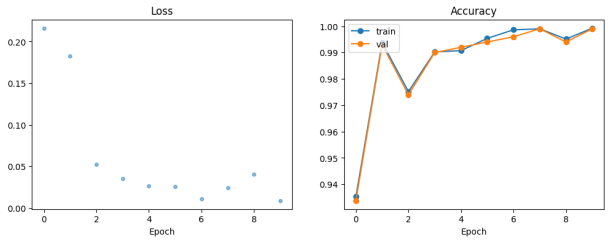

Figure 8. Training and Validation Curves for Frequency-Based Classification

ADA are StyleGANs with some differences in underlying architecture) demonstrate some transferability in recognizing fingerprints.

4. Implementation

All of our code is implemented in Python and runs on the Great Lakes computing cluster. To generate the images for our dataset, we utilized the official PyTorch implementation of StyleGAN2-ADA [6] from [1]. For labeling, we wrote our own code. We used PyTorch as our main library, and built our 8-layer CNN and the entire training/evaluating process with it. To realize the research done in [4], we utilized the OpenCV library to first turn our images into grayscale and then generate DCT-transformed images. We finally also utilized another GAN, StyleGAN-XL [7] using the implementation from [5] to check the transferability of the original model. Regarding graphing out our training and testing results, we used the Matplotlib library.

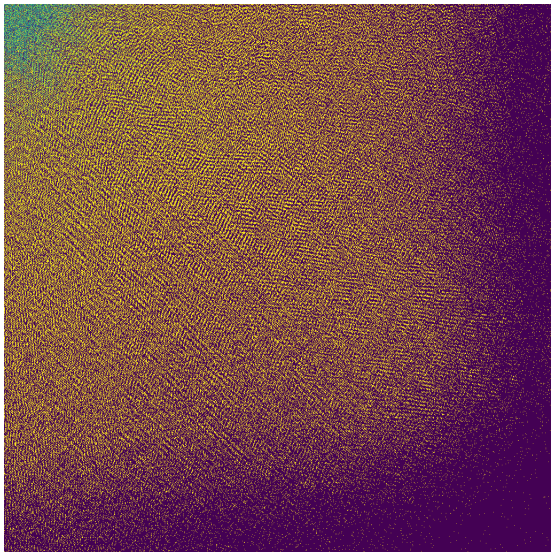

Figure 9. StyleGAN-XL DCT Heatmap

References

1. https://github.com/rkuo2000/stylegan2-ada-pytorch.git
2. https://github.com/NVlabs/stylegan2.git
3. https://philarchive.org/archive/SALCOR-3
4. https://arxiv.org/pdf/2003.08685.pdf
5. https://github.com/autonomousvision/stylegan-xl.git
6. https://arxiv.org/abs/2006.06676
7. https://arxiv.org/abs/2202.00273
8. https://arxiv.org/pdf/2205.12543.pdf
9. https://arxiv.org/pdf/1812.11842.pdf