CSCI 1430 Final Project Report:
Spot the Difference: Real vs. StyleGAN-generated Faces

Turtle Flippers: Colden Bobowick, Spencer Dellenbaugh, Alexander Halpin
TA name: Kelly Patel
Brown University

Abstract

Generative Adversarial Networks (GANs) have gained significant attention in recent years for their ability to generate photorealistic human faces. However, this capability also raises concerns about the spread of misinformation, fake profiles, and deepfake videos. This paper addresses the problem of detecting GAN-generated faces using Convolutional Neural Networks (CNNs). Two approaches are presented: the canonical CNN approach and a Siamese Neural Network (SNN) approach that analyzes each eye. The CNN approach achieved a validation accuracy of approximately 93%. The SNN approach, although not fully implemented due to time constraints, showed promise in eye extraction and preprocessing. The paper discusses the related work, dataset used, model architectures, and the results obtained. Furthermore, it acknowledges the technical challenges faced during the project. Overall, the research provides a foundation for detecting GAN-generated faces and emphasizes the importance of responsible computing in developing such technologies.

1. Introduction

Generative Adversarial Networks (GANs), first developed in 2014, have become increasingly impressive in recent years. Many networks, like StyleGAN, can generate photorealistic human faces that can be surprisingly difficult to detect visually without a thorough, up-close examination. This has made GAN-generated faces popular for spreading misinformation, creating fake online profiles, and creating deepfake videos of well-known people.

Recent work has shown that Convolutional Neural Networks (CNNs) can be utilized to automate the detection of GAN-generated faces to a relatively high degree of accuracy. One difficult aspect of this solution, however, is creating a robust solution that generalizes to unseen GAN architectures.

This paper outlines two distinct approaches to this problem. Both approaches utilize the feature extracting capabilities of CNNs to find specific visual artifacts of GAN image generation. These artifacts are then passed through a classifier to determine if the image is of a real face or has been synthetically generated by a GAN. The first of these approaches is the canonical CNN approach, which resulted in around a 93% validation accuracy. The second approach utilizes a Siamese Neural Network (SSN) to analyze each eye. Although the eye-extraction and preprocessing was a success, the network was not completed due to the inherent time constraints of the project.

2. Related Work

Recent literature has seen significant success in using convolutional neural networks for GAN-generated face detection. Chapter 9 of the Handbook of Digital Manipulation and Face Detection [1] provides a detailed overview of current methods and relevant techniques for detecting artificially generated faces.

The SNN was based on a paper by Wang et. al. [3], which saw near-perfect accuracy by comparing the left and right eye. Wang et. al. uses Dlib to extract facial landmarks (see Figure 1). By creating a bounding box around facial points 37-42 for the left eye and 43-48 for the right eye, we were able to capture nearly every eye from the testing and training datasets. These eyes were cropped and saved for use in the SNN.

For training computation and code development, we used Google Colab with GPU accelerators. The dataset used for training and validation was 140k Real and Fake Faces from Kaggle [5], which consisted of 100,000 training images, 20,000 testing images, and 20,000 validation images. Each section of the dataset had a 50% split between real and fake images.

An initial attempt to crop the eyes utilized a full frontal face classifier and eye classifier from the CV2 cascade classifier library, although its performance was deemed unsatisfactory due to the quantity of eyes misidentified, so the Dlib eye extractor was used instead.

Finally, the VGGFace model from keras_vggface on GitHub [2] was utilized for pre-trained weights of the CNN.

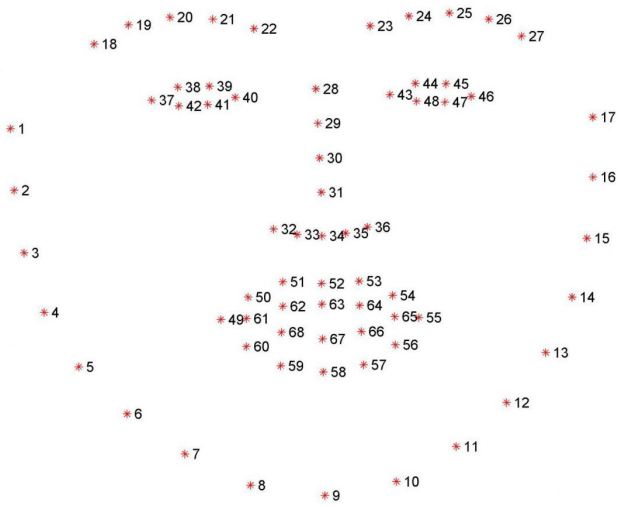

Figure 1. Facial feature layout extracted via Dlib.

The head was omitted and a custom feature classifier was fine-tuned. Although the SNN did not produce results, the Xception feature extractor weights were downloaded from tensorflow.

3. Method

GAN-generated images can be difficult to detect via visual inspection, which can be exploited in harmful ways such as misinformation campaigns and fake online profiles. Our proposed solution uses a vgg16 convolutional neural network followed by a fully-connected feature classifier to determine if an input image was generated by a GAN. See Figure 2 for a visual representation of the model architecture.

The solution takes in a 256x256 input image of a face. This image is first passed through an augment block, which randomly applies horizontal flips, brightness and contrast adjustments, zoom, and rotation. The image is then passed through the VGGFace pre-trained convolutional block. Both the augment and convolution blocks are untrainable so that learning is concentrated in the feature classifier.

The pretrained VGGFace feature classifier provided unsatisfactory accuracy results, so the head was removed and a custom fully-connected classifier was constructed. The classifier was the subject of many modifications, and the final result consisted of a 512-neuron dense layer with the LeakyReLU activation function ($\alpha = 0.5$), a batch normalization layer, a 256-neuron dense layer with the same activation function, another batch normalization layer, and a 2-neuron softmax dense layer for classification.

Ample time was spent adjusting a multitude of variables in the augment and classification blocks to maximize accuracy. The arbitrary starting point for the classifier consisted of a 24-neuron ReLU layer and the 2-neuron softmax layer. Through rigorous testing of dropout layers, classifier depth,

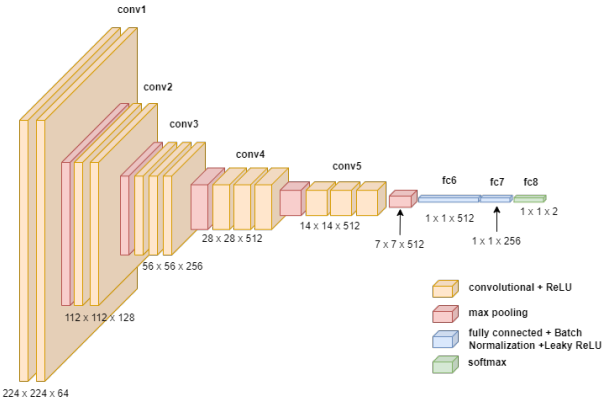

Figure 2. Convolutional neural network architecture featuring a vgg16 feature extractor and custom classifier.

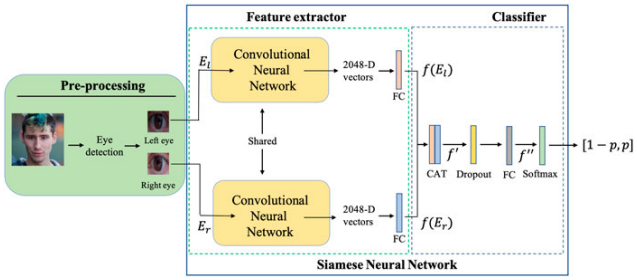

Figure 3. Convolutional neural network architecture featuring a vgg16 feature extractor and custom classifier.

number of neurons in each layer, batch normalization, and augmentation, the final classifier saw a 6% increase in testing accuracy from the initial tests. Eventually, the final model was trained over 10 epochs for a 93% resulting accuracy.

Deeming this accuracy acceptable, we moved on to attempt the SNN solution by comparing the left and right eye. A significant portion of the project was dedicated to preprocessing the dataset to crop the left and right eye from each face. The first attempt utilized the HAAR feature cascade from the CV2 library. Due to the unexpectedly high data loss caused by this process, the Dlib face extractor was used.

The SNN was constructed such that two 71x71 pixel images were passed in as inputs. First, each image passed through the augmentation block described above. Then, each of the eyes was passed through the Xception feature extractor individually. The resulting 2048-dimensional vectors were concatenated into one 4096-dimensional vector, which was then passed through the classification block described above. See Figure

Due to multiple obstacles throughout the preprocessing and training of the SNN, we were unable to achieve functionality within the time constraints of the project. The first issue was caused by a Google Drive upload throttle. Although the Dlib eye extractor was reporting that nearly 20,000 eyes were found in the test portion of the dataset, the target Google

Measure	Result
Validation Accuracy	93.00%
Validation Loss	0.1736
Training Accuracy	90.61%
Training Loss	0.2238

Table 1. Best performing results of our CNN model.

Drive directory contained only 13,000 files. Once this issue was identified, we saved the eye files locally and then zipped them for storage in Google Drive. Once this was sorted out, attempts to fit the model to the dataset resulted in a graph error, which we were unable to accurately diagnose. We believe it may be caused by either malformed data or incorrect usage of the Xception model.

4. Results

The results obtained from the most successful training epoch of our CNN model are summarized in Table 1. As far as the process of tuning our dense layer architecture, see Table 2 for a summary of the changes we made in the and the effects they had on test accuracy. We also produced two explainability metrics for our model, LIME visualization (see Figure 4) and saliency maps (see Figure 5).

These show the respective metrics carried out on one batch of test data. Looking at these results, we were relieved but unsurprised to see that the model is in fact "looking" at the faces in the images most of the time rather than the backgrounds. This makes sense, especially since it makes use of the VGG-face feature extractor. We also noticed that the nose and center area of the face are often more influential to the model's classification. While this does make sense on some intuitive level, it was surprising to us since some of the papers we referenced placed great emphasis on other features such as eyes [4]. This goes to show that there is not necessarily any one facial feature that is most central to distinguishing between real and GAN-generated faces.

4.1. Technical Discussion

Since our validation accuracy is higher than our training accuracy, it is unlikely that our model is overfitted to the training data. There is some tradeoff between accuracy and computational complexity; a more complex dense layer was able to classify face images as real or GAN-generated with greater accuracy but, by virtue of its greater complexity, took longer to train and evaluate. However, these differences in compute and runtime were relatively marginal in practice, and in general did not prove to be a limiting factor.

One interesting question that our model raises is whether or not it would be equally effective on GAN-generated faces produced by a GAN other than StyleGAN (which generated the fake faces in the dataset we used). We were not able to find a comprehensive dataset generated by another type of GAN, but this would definitely be a worthwhile endeavor if we were to continue our research.

4.2. Socially-responsible Computing Discussion via Proposal Swap

The first critique is not related to socially-responsible computing, and highlighted a lack of technical detail in our proposal, stating that "the proposal only mentions that they will be using a CNN to detect GAN-generated faces." This is true, and it serves to succinctly summarize our goal for the project. At the time of proposal we did not know what our eventual solution would look like in full, but we did end up using a CNN as we initially planned to do. In hindsight we could have stated our plan to use a CNN more clearly in our proposal, but it is reasonable to omit greater technical detail at that point in the project since we had yet to explore various avenues.

The second critique has to do with fairness and bias, and notes our proposal's lack of "a clear plan to address fairness and bias in the project." This is a fair critique; while our proposal does address the potential impact on privacy and how this could lead to discriminatory scenarios, it does not address the fact that a system to detect real and GAN-generated faces could be biased against certain groups. For example, if such a model were used in a verification system for online services, groups for whom it is less accurate might suffer increased inconvenience and difficulty in registering. This in turn would contribute to a greater failure of recognition, a possibility that raises significant ethical concern. We admittedly did not do much to address this issue in our project because we were more focused on creating a baseline proof of concept that we are able to distinguish real faces from GAN-generated faces using a CNN. Now that we have done so successfully, and certainly well before such a model is implemented in an actual application, establishing fairness and lack of bias among different groups is a great next step. This could take the form of testing the model on data separated by different characteristics (such as race) ensuring that it performs comparably well across all groups; any inequalities could potentially be addressed by increasing the representation in the training data of the groups on which the model underperforms.

The third and final critique once again was not related to socially-responsible computing, and suggested "more specific accuracy and performance metrics." Team HORD points out that our proposal "does not provide a plan for measuring [...] accuracy," but we believe that it is clearly implied that accuracy would simply be the percentage of data points in the testing/validation data that are correctly classified. There is some merit to this critique though, as we did end up looking at some metrics (such as false positives and false negatives

Variable	Initial Attempt	Initial Accuracy	Final Attempt	Final Accuracy
Batch Normalization	0 Layers	84.95%	2 Layers	87.20%
Dropout Layers	1 Layer of 0.5	90.45%	0 Layers	91.60%
Number of Dense Layers	3 Layers	82.75%	2 Layers	86.40%
Leaky ReLU Alpha Value	0.1	90.45%	0.5	91.00%
Number of Neurons	26 (24+2)	87.45%	770 (512+256+2)	91.90%
Augmentation (Bright., Cont.)	(0.1, 0.1)	91.40%	(0.3, 0.2)	93.40%
Trainable vgg16	Trainable	55.34%	Untrainable	92.92%

Table 2. Hyperparameter tuning results.

Figure 4. Results of LIME Visualization.

in the confusion matrix) that we were not thinking about at the time of writing our proposal. Additionally, it might be interesting to look at metrics such as f-measure which factors in both precision and recall to gain greater insights into the effectiveness of our model. Going forward this is definitely something we would consider, and flesh out more explicitly in advance.

5. Conclusion

In conclusion, we consider our project to be a success. We did what we set out to do, constructing and training a CNN to achieve greater than 90% accuracy distinguishing between real and GAN-generated faces. We explored other more complex avenues of doing this as well, and even though we were not able to get the eye-based SNN working, the research we did toward that end leaves us with valuable understanding of how and why such a system might work. As GANs get better and better, it will become increasingly important to be able to distinguish between real and generated images, whether it is to ensure that the person you're talking to on Zoom is actually real or to verify the authenticity of photographs in a news article. While GANs do have discriminators that aim to do this very thing, the generative portion of the model is trained to fool these very discriminators, rendering them unhelpful for the aforementioned and similar applications. Our project shows that an external approach is definitely possible, and provides a foundation for future research into this topic going forward.

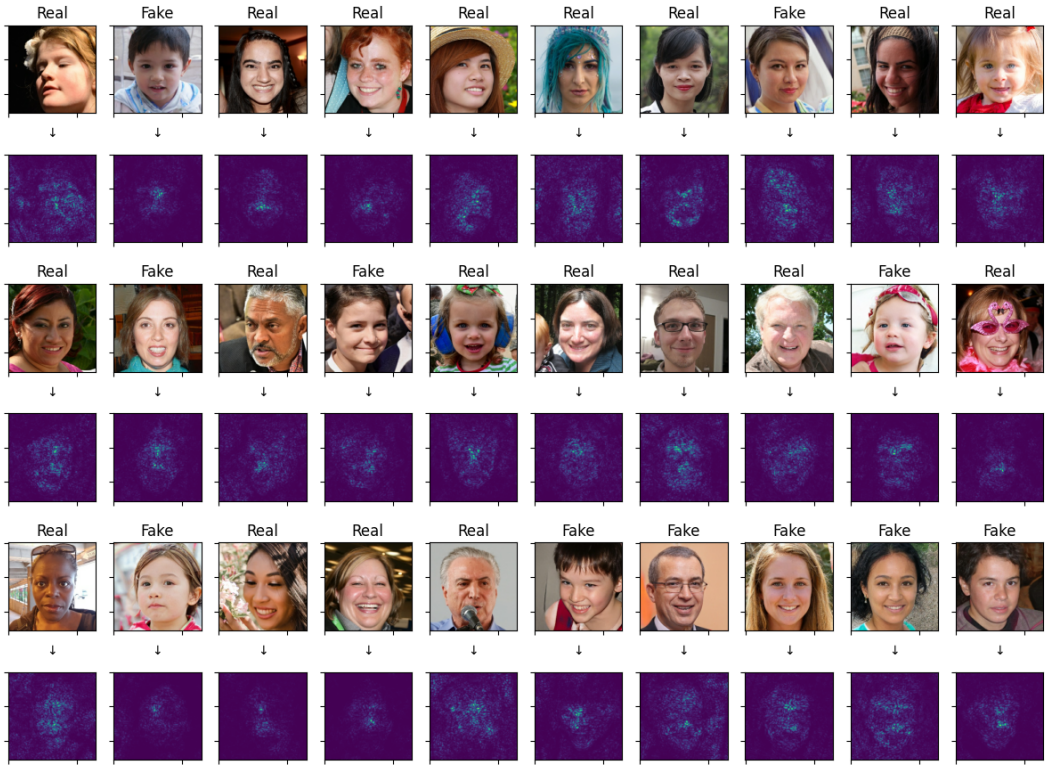

Figure 5. Results of saliency maps for sample dataset images.

References

[1] Ruben Vera-Rodriguez Christoph Busch Christian Rathgeb, Ruben Tolosana. *Handbook of Digital Face Manipulation and Detection*. Springer, 2014. 1

[2] Refik Can Malli. keras-vggface. https://github.com/rcmalli/keras-vggface, 2016. [Online]. 1

[3] Jun Wang, Benedetta Tondi, and Mauro Barni. An eyes-based siamese neural network for the detection of gan-generated face images. *Frontiers in Signal Processing*, 2, 2022. 1

[4] Xin Wang, Hui Guo, Shu Hu, Ming-Ching Chang, and Siwei Lyu. Gan-generated faces detection: A survey and new perspectives, 2023. 3

[5] xhulu. 140k real and fake faces. https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces, 2020. [Online]. 1

Appendix

Team contributions

Colden Bobowick

Colden spent most of his time split between two tasks. The first task was the optimization of the CNN. Colden altered many variables and tested for an estimated 100 total epochs. The result of this testing was a 6% increase in validation accuracy. The rest of Colden's time was spent constructing the SNN and attempting to preprocess the eye images. Finally, Colden contributed to the proposal, progress report, proposal swap, poster, and this document.

Spencer Dellenbaugh

Spencer worked on producing the LIME visualization and confusion matrix for the model, as well as updating the code for the saliency maps as the model architecture changed. These provided valuable insights toward explaining the model and identifying trends in its classifications. Additionally, Spencer contributed to the socially-responsible computing portions of the project as well as the progress report, this report, our poster, and the presentation thereof.

Alexander Halpin

Alex designed and implemented the original model architecture. This included researching previous implementations of face-detection CNNs and incorporating the appropriate model into our classification architecture. Alex then implemented the saliency map functionality, enabling the interpretability of the model and the dataset.