Deep Fake Detection on Human Faces

Sai Manchikalapati
Georgia Institute of Technology
ssm34@gatech.edu

Pranav Sreedhar
Georgia Institute of Technology
psreedhar6@gatech.edu

Vignesh Sreedhar
Georgia Institute of Technology
visreedhar@gatech.edu

Abstract

Given the dramatic increase in deepfakes and GAN generated images and videos, we propose a fake detection CNN model that performs very well and is efficient and relatively generalizable as well. We hope to leverage our findings to build the basis for a simpler and more general form of deepfake detection.

1. Introduction

Generative AI (GenAI) has been at the forefront of modern artificial intelligence inquiry. As technologies such as GPT, DALL-E, and Sora continue changing content generation and how we interact with online content, distinguishing original content and GenAI content is becoming increasingly difficult. For example, humans are evolutionary trained to recognize, distinguish, and analyze other human faces subconsciously. Even with this evolutionary advantage, it is often nearly impossible for humans to differentiate between GenAI pictures of human faces and real images of human faces.

The objective of this paper is to propose, analyze, and document various discriminator models trained to effectively and efficiently differentiate between images of GenAI faces and real faces. Technical research in this domain promotes understanding of GenAI since, at their core, our distinguishers identify specific features of AI generated images not present in real images and/or vice versa, illuminating shortcomings of image generators. Deeper analysis of the discriminator networks may provide insight on how to improve generators. Additionally, evaluating the processes behind our discriminators can contribute to relevant topics of contest in deep learning, such as the question of whether CNNs operate on the macro level, using shapes and boundaries, or on the micro level, processing textures and localized features.

2. Background

Given the proliferation in generative AI and specifically generative adversarial networks, the creation of realistic deep fakes has skyrocketed. As a result, multiple papers and models have been discussed to detect such deep fakes [1]. In [4], the authors implement a four-phase approach that includes pre-processing, feature extraction methods, and final classification using CNNs. [2] first creates deep fakes via GANs of various resolutions and sizes and then exploits VGG-Net to extract features of deep fakes and classify them accordingly. More complex, hybrid models have been discussed as well: [3] employs pairwise learning to delineate between features of GAN-generated and real images, and [5] trains a two-stream network to specifically detect features resulting from face tampering.

Overall, the majority of current practices are specific to the characteristics of the underlying datasets. They normally train and test with the same data or same types of data and are highly complex, often reducing its ability to perform well for more different types of data. We plan to take inspiration from these various successful approaches and also propose and analyze a simpler, more generalizable model.

3. Motivation

Aside from technical implications of our research, the domain as a whole has deep sociopolitical implications. The rise of GenAI has empowered content creators and bolstered efficient ideation. However, at the same time, it has made it extremely difficult to distinguish between real and fake images, making it easier to mislead, scam, and defame people (especially with GenAI faces). Especially given the speed at which information can spread on the internet, a generator creating faces that are indistinguishable from real faces to the human eye is a dangerous tool to make available publicly without adequate discriminators.

Figure 1: Comparing a real (left) and GenAI (right) human face from our dataset: notice both images seem "real."

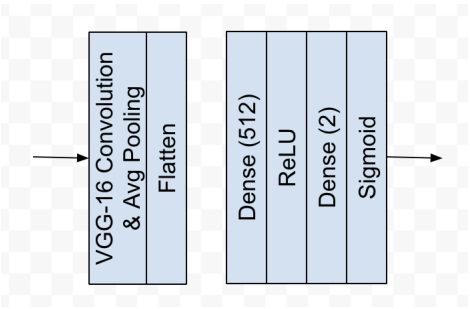

Figure 2: Our modified VGG-16 architecture.

4. Data

Our dataset is composed of over 120,000 real and fake images of human faces. The real images are collected from FFHQ, a popular dataset containing over 50,000 high-quality, diverse, images of human faces at 1024 by 1024 resolution. A comparable amount of fake images are generated via GANs. The dataset owner recommends leveraging the dataset specifically for real vs. GAN generated classification problems, which is our principle aim. Figure 1 shows a sample real and fake image in the dataset.

5. Approach

We treated the problem as a binary image classification problem. The dataset consisted of standard headshot-like images of real and fake facial images. The models output a label classifying the image as real or fake along with an associated confidence. Given the size of the dataset, we decided to use image data generators to optimize image batching, loading, and training. The data was split into training, testing, and validation datasets and loaded into an image generator.

With respect to the architecture of the models, we began by exploring existing research in deep fake detection technologies. Research operating in more technical deep fake detection domains often leverage large, commonly used CNN architectures in their analysis. For example, [2] use VGG-Net as a discriminator during for their deep fake classification task. As such, we began our search with investigating VGG-Net and EfficientNet architectures. For our first model, we attempted to apply transfer learning to an out-of-the-box VGG-16 model with pretrained ImageNet weights. We modified the VGG-16 model to comply with our input size and our output classes as shown in Figure 2.

Initial training and testing of our VGG-16 architecture was evaluated (see in section 6) and deemed inadequate in the eyes of our criteria. Not only was the VGG architecture extremely large and therefore inefficient, but it only reached an accuracy of approximately 58%. Though we anticipated our modified VGG-16 architecture to produce a sufficient model, analyzing the architecture and parameters of the model with respect to the problem domain illuminated various possible explanations for our observations. In particular, we postulate that the observed sub-par performance was likely a result of two main reasons:

1. Discrepancies between the Architecture and the Problem: A common point of controversy in literature currently is the question of whether CNNs rely on global features or local textures in classification domains. Performing extensive convolutions in a bottleneck-like structure (as it is done in VGG-16), thus, risks masking and averaging out extremely localized features. Intuitively, the purpose of GenAI is to fool human observers by mimicking global features of faces, which means discriminator models likely look to far more localized features of images to differentiate between real images of human faces and GenAI generated images. This intuition is reflected in our subsequent custom discriminator model by our elimination of back-to-back convolutional layers.

2. Over-Complexity in the Architecture: Often, too much complexity (represented by the number of trainable parameters in the model) cause the significance of each weight with respect to the output to diminish, leading to vanishing gradients. We believe the VGG-16 architecture was structurally more complex than necessary for the problem space. To compensate for this, we modeled our second iteration model more closely to Efficient Net, limiting our model's depth to 7 convolutional layers, increasing the kernel size of the last layers, and reducing the size of the fully connected classification layers at the top of the model.

We incorporated our analysis of the VGG-16 model into our second iteration, custom model. To promote information compression, prevent sparseness of features, and increase efficiency, we downsized the images from $(512, 512)$ to $(256, 256)$. The final architecture we arrived at is shown

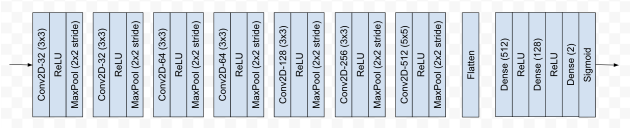

Figure 3: Our final proposed model architecture (after refinements from VGG-16 model).

in Figure 3. Our custom discriminator is more than $20x$ smaller than our original VGG-16 model. After finetuning, we enumerate our final model's specifications and hyperparameters.

Model/Training Specifications	Value
Total Parameters	4,826,914
Optimization Alg.	Adam
Learning Rate	0.001
Loss Function	Binary Cross Entropy
Training Epochs	15
Training Dataset Size	72,574
Validation Dataset Size	24,190
Test Dataset Size	24,190

Table 1: Our refined model's specifications.

6. Experiments and Results

Before delving into our results, we will first enumerate our evaluation criteria:

1. Strength of Model: The first and most important criterion for our model is testing the strength of the model, for which we fortunately have multiple quantitative metrics. Not only do we want to construct a model with high accuracy (which consists of the percentage of accurate predictions), but we also calculate precision, recall, and F-1 score in order to get a better understanding of the model's performance given imbalances in the data.
2. Computational Cost: As mentioned in our approach (Section 5), we experimented with and drew inspiration from a variety of models ranging from VGG-16 to EfficientNet. In order to scale better with more data and features, we want to minimize computational cost without taking much away from the strength of the model. We will calculate the number of parameters and even note runtimes (given platform information and number of epochs).
3. Generalizability: Another important goal for our project based on our survey of related work was to propose a simpler, more generalizable model that can be used as a foundation for different types of data. We also want to make sure the simpler model retains (or ideally exceeds) the strength that comes with more complex models. We will make sure that our model does not overfit to our training data and test our model manually against face images of different sizes and resolutions, including images not in the dataset as well.

6.1. Strength of Model

As mentioned in our approach, we first began with a pre-trained VGG-16 model, which is a 16-layer deep CNN composed of convolutional kernels of size 3x3 and maxpool kernels are of size 2x2 with a stride of 2. Despite consisting of more than 130 million parameters, the model performed fairly poorly with our training loss barely decreasing past the first epoch and training and validation accuracy plateauing at about 0.579 as soon as the second epoch. We believe that the root cause for the poor results was the discrepancy between the model's complexity and the data. The model learning seemed to saturate fairly quickly (as soon as the first epoch), suggesting that the model was unable to learn much more given the training data. The model was probably too complex, leading to difficulties in optimization and most importantly generalization. The number of parameters and layers as well probably led to vanishing gradients, where gradients become very small during backpropagation, not allowing the hundreds of millions of parameters to be updated effectively.

As a result, we propose a simpler model with fewer trainable parameters that attempts to combine the feature extraction ability of VGG-16 and efficiency of EfficientNet to create a more effective model for our binary classification. After finetuning hyperparameters (Table 1), we managed to achieve a training accuracy and loss of 0.983 and 0.047 respectively and validation accuracy and loss of 0.955 and 0.162 respectively after 15 epochs. The final accuracy on the test dataset is 0.938.

As mentioned in our evaluation criteria, we go further with our analysis of our final model, calculating the precision, recall, and F-1 score of our model with respect to our test dataset. This is especially important since a simple proportion of true predictions does not necessary tell the full story. For example, in a dataset that consists of disproportionately more positive or negative instances, a model that falsely predicts positive or negative will usually not be penalized as much. Therefore, we calculate precision, recall, and F-1 score. The final precision on the test dataset, which measures the ratio of true positives and total number of positive predictions, is 0.938. The final recall, which measures the ratio of true positives to all the actual positive samples, is also 0.938. Thus, the F-1 score, which is the harmonic mean of precision and recall, is 0.938 as well.

6.2. Computational Cost

In terms of computational cost, as expected, our final model performs extremely well. Compared to our intial VGG-16 model of 138 million trainable parameters and 16 layers, our custom model only consists of 4.8 million trainable parameters. This is a 96.52% decrease on the number of trainable parameters from our VGG-16 model, and it seems large enough (and small enough) to extract important, general features and perform significantly better than VGG-16 as evidenced by the resulted detailed in 6.1.

Using Google Colaboratory's T4 GPU, we saw an average runtime of 1206 seconds or 20 minutes per epoch for our VGG-16 model. However, for our final model we see an average runtime of 224 seconds or 3.74 minutes per epoch, which is an 81.46% decrease in runtime per epoch. This is yet another measure of the low computational cost of the model.

6.3. Generalizability

In a more micro scale, the first test for good generalizability is to check for overfitting. Figure 4 shows the training and validation accuracy curves, and Figure 5 shows the training and validation loss curves. The training accuracy and loss consistently increased and decreased respectively after every epoch, and although our validation accuracy and loss saw more oscillations at later epochs, the general trend was similar to our training accuracy and loss. This is especially apparent seeing that our validation accuracy and loss began at 0.826 and 0.401 respectively and eventually ended at 0.955 (which is the global optima in our 15 epochs) and 0.162 respectively. This suggests that the model did not overfit the training data as the general trend in our validation curves are promising. Figure 6 shows the training and validation precisions, and Figure 7 shows the training and validation recalls, both of which follow a very similar trend as accuracy. Furthermore, as an added test of overfitting, the accuracy, precision, and recall on the final test data is 0.938, suggesting very good predictive power and generalizability.

Furthermore, we did a more macro, qualitative check by testing the accuracy of our model on 2-3 random samples each of different sizes, different angles, different resolutions, as well as real and fake generated images not from the dataset, which yielded the right prediction almost every time. Figure 8 shows an example of correctly predicted real and fake images not from the dataset.

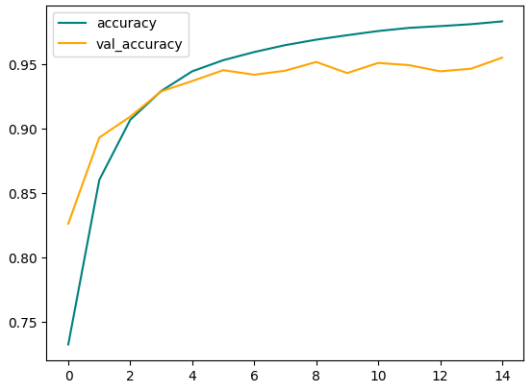

Figure 4: Training and Validation Accuracy of Final Model

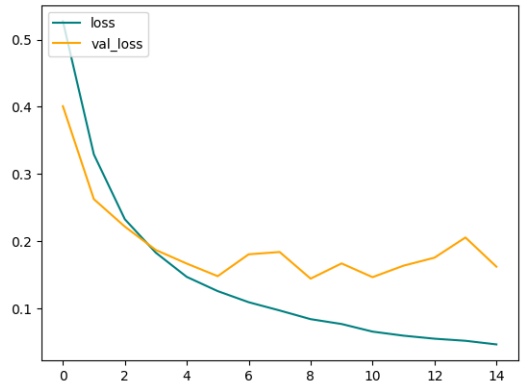

Figure 5: Training and Validation Loss of Final Model

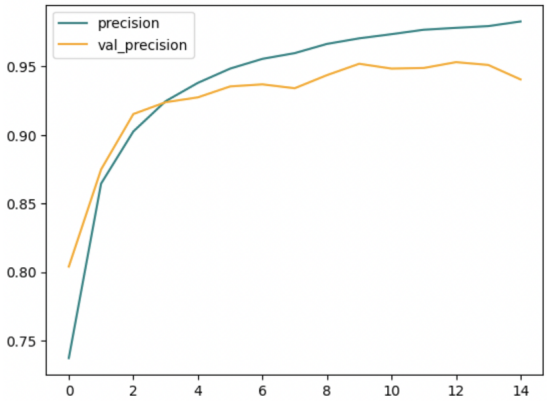

Figure 6: Training and Validation Precision of Final Model

7. Conclusion and Discussion

As a whole, our proposed model does a fantastic job in terms of all of our evaluation criteria proving to be accurate, efficient, and generalizable. Our approach and results indicate that a very complex model architecture with multiple convolutional layers and trainable parameters is unnecessary and even ineffective with the classification of real and fake images. Fewer parameters are required to be trained in order to achieve higher higher accuracy, precision, and recall. Evidently, the computational cost and runtime is also significantly reduced. The lower number of trained parameters seem to make it more robust to unseen data as it per-

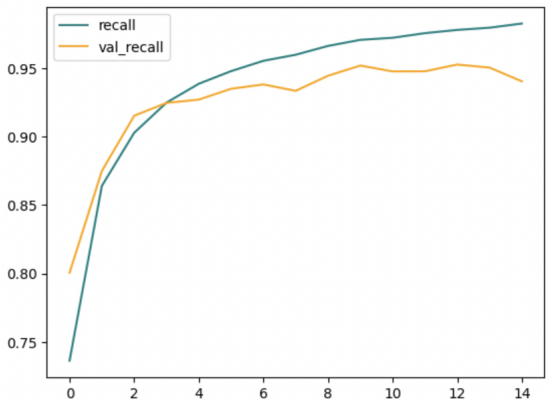

Figure 7: Training and Validation Recall of Final Model

Figure 8: Example of correctly predicted real (right) image of Tom Hanks and a famous deepfake (left) of his.

forms very well with different types of images.

The model is not without its limitations. While it performs very well on predicting the test data, the confidence with which it predicts that the image is fake is significantly lower on average than when it predicts that the image is real. This suggests that the model would benefit from more GAN generated images in the training data and perhaps more complexity in the model architecture to confidently declare an image fake. The lack of a clear generalizability measure of the model with respect to different data is also noteworthy since only small-scale tests were done.

Future work could be done on visualizing specific features that contribute the most to the real or fake decision via interpretation methods such as Grad-CAM and Saliency Maps. More work could also be done to derive a quantitative measure of the model's generalizability on a range of different data (in terms of a set number of metrics such as angles, resolutions, image content differences...etc). Perhaps the model could also be used as a basis for a more generalized model on a much larger range of data.

Student Name	Contributions
Vignesh Sreedhar	*Experiments and Results*: Evaluated the model via various quantitative and qualitative methods. Analyzed the strength, computational cost, and generalizability of the model.
Sai Manchikalapati	*Model Implementation*: Constructed the initial VGG-16 model and performed initial testing. Incorporated research, empirical findings, and intuition to reduce model complexity and construct the final custom model.
Pranav Sreedhar	*Data Preprocessing and Figures*: Loaded the data into training, validation, and test dataframes and preprocessed the image data for training. Created figures/plots and tested the model with various images.

Table 2: Contributions of team members.

8. Work Division

Table 2 concisely depicts the division of work among the team members. Each of us contributed to the entire project, helping each other when needed and offering ideas and alternative approaches. Pranav preprocessed the data and made the figures, Sai implemented our approach, and Vignesh evaluated our approach.

References

[1] Abdulqader M. Almars. Deepfakes detection techniques using deep learning: A survey. *Journal of Computer and Communications*, 9(5), 2021. 1

[2] N.-T. Do, I.-S Na, and S.-H. Kim. Forensics face detection from gans using convolutional neural network. *Journal of Computer and Communications*, 9(5), 2021. 1, 2

[3] C.-C. Hsu, Y.-X. Zhuang, and C.-Y Lee. Deep fake image detection based on pairwise learning. *Applied Sciences*, 10(1), 2020. 1

[4] R. Sabitha, A. Aruna, S. Karthik, and J. Shanthini. Enhanced model for fake image detection (emfid) using convolutional neural networks with histogram and wavelet based feature extractions. *Pattern Recognition Letters*, 152:195–201, 2021. 1

[5] P. Zhou, X. Han, V.I. Morariu, and L.S. Davis. Two-stream neural networks for tampered face detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017. 1