Deep Fakes (Final Report)

Sam Forbush, Kalel Lopez, Robert Smithers

A. Introduction

Deep fakes have only been around publicly since about 2015, and while academic institutions have safeguarded the discovery of this technology beginning in the 1990s, it has since been adopted by the industry. Deep fakes are a type of artificial media in which an object or person is masked as something or someone else. Requiring powerful generative neural network architectures, the existence of deep fakes was relatively unknown – that was until machine learning became even more widespread.

By 2020, many competent machine learning enthusiasts have had the ability to produce deep fakes. With a channel to generate false imagery or video, not to mention the difficulty for humans to assert its validity, deep fakes have been a real problem. Most notably with celebrities, government officials, and other noteworthy persons, deep fakes have posed a national security threat, challenging the public trust and self-image of individuals and even government entities. Numerous governmental agencies have released statements on the danger associated with deep fakes, and I would not be surprised to see the term appear in future National Security Strategy publications.

The goal of this project was to create a model to effectively detect deep fakes. This has significant value in the world of entertainment, such as in Hollywood and pop culture. There are also political stakes in the technology, as it can potentially lead to massive misinformation campaigns and public distrust. Deep fake technology is known to be dangerous; therefore, a program that detects the usage of it is necessary to counter misinformation.

The process of creating this network involved designing the model with the help of previous projects, downloading a curated data set, training the model and adjusting it. The data set we ended up choosing was from the Kaggle Deepfake Detection Challenge, which included over 400 GB of video.

Originally, we were going to use a Generative Adversarial Network in order to detect the DeepFakes. However, we decided to instead take frames of videos and analyze those images through our model. This allows us to easily manipulate and compare the sets of data as well as make the program more lightweight.

B. Related Work

Many different methods of recognizing DeepFakes have been proposed within the last few years. This is due to the increased potential for social risks that arise from developing facial manipulation technology. We searched for many different approaches to the problem. Some models focused on audio and visual data from input videos to detect discrepancies with lip movement and audio feedback [7]. Other approaches focus on frame-by-frame analysis of video. For example, in [8], the authors use a MesoNet model with the objective of detecting fake faces. These works provided base ideas from which we could work off of. The following are sources that directly helped us:

DFDC 3D and 2D inc cutmix with 3D model fix:
This notebook provided us with the ResNet34 model we needed in order to identify and classify the images we took from the frames of video. What this code sought to do was take the faces from a video, run it through the 3D CNN, and transform the frames given to predict whether the video is a DeepFake or not. While we used the ResNet34 model presented in this notebook, we created our own model for predicting DeepFakes from frames of a video. However, it did provide a guideline for transforming the images while training.

Kaggle DeepFake Detection Introduction:
This notebook provides a quick introduction on how to start with the DeepFake Detection Challenge (DFDC). What it does is take frames from a video and extract the faces in the images. We followed a similar approach for face detection. We first took the frames of a video and converted them to images we can manipulate. From there, we used DeepFace, a lightweight face recognition program. This detects and isolates the faces in an image. We end up being able to use the output just like in the notebook. Another difference in our model is that we use an SSD based DeepFace module versus the OpenCV package used in this notebook.

Video Face Manipulation Detection Through Ensemble of CNNs
This research paper formed our base idea of using a CNN to analyze frame data. This paper's method relies on ensembles of CNNs. It puts focus on investigating whether and how it is possible to train different CNN-based classifiers to capture different high-level semantic information that complement one another.

C. Method

As we expected, dealing with the massive amounts of video data from our data set came with a few problems. We wanted to train our model on images rather than videos, and the the data set itself was also heavily lopsided towards fake data, with over 80 percent of the data being fakes. In order to resolve this, we created a function to extract a certain number of frames from each video to train with, making

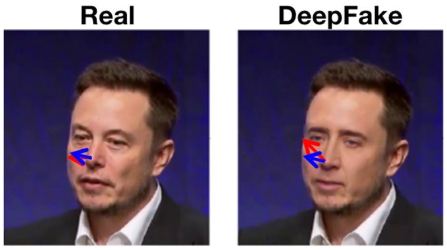

Figure 1. Example of a DeepFake

sure the frames were as spread out as possible to reduce overfitting. Then, during training, we extracted 4 times as many frames from the real data to help balance the number of reals and fakes that our model trained on.

One of the most important aspects of our final model was the inclusion of the DeepFace facial recognition algorithm. Because the face is the only important aspect of the image, the most efficient way for the model to learn the correct features would be to only train on the data within the face itself. The way this was implemented was, during training, each frame would be passed through the DeepFace algorithm in order to isolate the face. The only downside of this was that training would take significantly longer due to the computational complexity of such an algorithm.

For the model itself, we used a pre-trained ResNet34 due to its proficiency for image classification. For each frame extracted from the data set, it would be run through the ResNet34 architecture to generate a prediction from 0 to 1, $[0, .5)$ being real and $(.5, 1]$ being fake. We calculated our loss with L2 regression, using the label on the data gathered earlier in a .json file as the ground truth.

D. Experiments

Description	Results
Dummy Baseline	80.75 %
MVP (3 Epochs)	86.5 %
Improvements (15 Epochs)	94 %

The data that we used for these experiments are the sample video files from the Kaggle DeepFake Detection Challenge. The dummy baseline represents how lopsided our data was to begin with. If the model predicted fake for every frame, it would have an 80% success rate due to there being approximately 5 times as many fakes as reals.

The first MVP consisted solely of the ResNet34 model, with no face detection or data balancing. Clearly, it is better than the dummy baseline, however the imbalanced data definitely inflates the resulting percentage. The relevant hyperparameters were a batch-size of 1 and a learning rate of 10^{-6}.

Finally, after the addition of face recognition using DeepFace and data balancing, our model became 94% accurate. This increase in proficiency can be attributed to the model learning better features due to isolating the face and also training overnight (15 epochs) rather than just briefly. The relevant hyper-parameters were a batch-size of 1 and a learning rate of 10^{-7}.

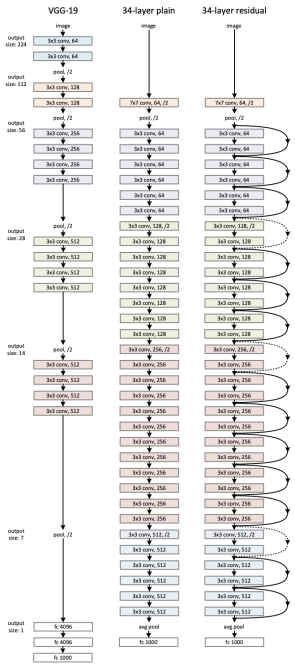

Figure 2. ResNet34 model

After further analyzing our trained model to better understand its decision-making process, we believed the model could have been learning heavily on the facial features of the 10 sample subjects. Things such as typical skin colors in lit and shaded environments likely had a heavy weight into the model's determination when compared to the potential deep fake video. For this reason, along with the endless pros of a larger dataset, we decided to look at increasing our data further. Since we used 5 frames from each 10 second video centered on the middle of the video, we instead decided to change it to clip 10 frames, 1 per second, of the 10 second video clips. This would not only give us more data for our model to look at, but also could test our new hypothesis of whether it could have been overfitting to the sample subjects and their skin colors.

We found that with 10 frames per video sample, the model strangely had much greater difficulty accurately identifying real video clips and had slightly better accuracy with identifying faked video clips. This might indicate that there was indeed some sort of overfitting, or perhaps that the reduced number of centered video clips were more/less well lit, making it easier to identify real versus fake samples.

E. Conclusion

One of the biggest takeaways from completing this project is that even with low-performing models on limited data, sometimes the best next step is to generating your own similar data. In our case, it was incredibly valuable to have such a function that could easily influence the amount of data we collect and train on by changing a couple values. Not only does it give the model more data to train on, but it allows us as the coders to better parse the type of data we want to input and have the model focus on. By doing this, we were able to determine at which points we were overfitting the model.

Additionally, this project is a great example of how complex neural network architectures are. More specifically, it shows that due to this complexity, models can often be focused on smaller details that humans may not see, possibly learning on areas that are not effective. For example, our model had difficulty making predictions from the entire 400x400 pixel images. This is likely because it could not discern which parts of the background were normal and which ones could be deepfaked. The purpose was not for the model to identify deepfakes in objects, but rather deepfakes of people. This means that by using a pretrained model to identify the face and crop to its specific area, our model could focus solely on the details in the face itself, which is presumably where any deepfake would be making changes. As you can see from the output, it was considerably more effective (and is even more of a change than the numbers show, since the first MVP had highly unbalanced data).

In terms of future work, mentioned in our reflections as well, one of the biggest differences between our model's method of discerning real versus fake video and our human method of doing so is that for humans, we have a full understanding of the entire video. Our model, on the other hand, looks at images of the video and makes predictions on each individual image. Unfortunately, this is not entirely holistic of the video itself, especially if an almost perfect deepfake has a major, obvious indication of it being a deepfake in just a few frames, in which case the effective other portions of the video would outweigh the ineffective portions and would cause the model to predict it as being a real video.

So, best advice to overcome this, and what I would like to see happen next, is to look at taking differences between frames. What could be done is to use some sort of motion differentiation analysis to find not only where the face is moving to, but also how the colors move. From visibly analyzing the training data, I could see some splotches of gray or other abornmal facial color in some frames. With a proper color difference analysis, a model could tell that the skin tone is changing throughout the video in highly abnormal manners (more than just changing lighting). If this could be implemented, I would imagine that we could close or eliminate entirely the gap between our model and some of the leading ones out there. If given more time, I would have tried to implement this, because I have never seen it be done, but I think it would be highly effective.

F. Contributions

- Robert
    1. Introduction
    2. Experiments
    3. Conclusion
- Sam
    1. Introduction
    2. Method
    3. Experiments
- Kalel
    1. Introduction
    2. Related Works

References

[1] Kaggle DeepFake Detection Challenge, 2020 https://www.kaggle.com/c/deepfake-detection-challenge/data

[2] Enoch Kan, GAN reference database, 2020 https://github.com/enochkan/awesome-gans-and-deepfakes

[3] N. Bonettini, E. D. Cannas, S. Mandelli, L. Bondi, P. Bestagini and S. Tubaro, "Video Face Manipulation Detection Through Ensemble of CNNs," 2020 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 5012-5019, doi: 10.1109/ICPR48806.2021.9412711. https://github.com/polimi-ispl/icpr2020dfdc

[4] Sefik Ilkin Serengil, DeepFace: A Lightweight Face Recognition and Facial Attribute Analysis Framework (Age, Gender, Emotion, Race) for Python, August 2021 https://pypi.org/project/deepface/

[5] Ian Pan, DFDC 3D and 2D inc cutmix with 3D model fix, 2020 https://www.kaggle.com/vaillant/dfdc-3d-2d-inc-cutmix-with-3d-model-fix/notebook

[6] Rob Mulla, Kaggle DeepFake Detection Introduction, 2019 https://www.kaggle.com/robikscube/kaggle-deepfake-detection-introduction/notebook

[7] P. Korshunov and S. Marcel, "Deepfakes: a new threat to face recognition? assessment and detection," CoRR, vol. abs/1812.08685, 2018.

[8] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "MesoNet: a compact facial video forgery detection network," in IEEE International Workshop on Information Forensics and Security (WIFS), 2018.