DeepFake Detector

Final Project

Ryan Joseph
CIS 542: Digital Forensics
Dr. Gokhan Kul
12-09-23

In my original project proposal I stated I wanted to create a DeepFake Detection system using AI. I stated how I thought the field of digital forensics is coming to an age where it's going to be very hard to distinguish what is real and fake. The reason I said this is because of the advancements of AI generated content. I referenced a photo (figure 1) that is hyper-realistic that the lay person would never imagine is fake.

Figure 1:AI generated image [2]

It's going to be very important to tell what's real and fake in these polarizing times. For that reason, I attempted to create a DeepFake Detector built using AI. As you can imagine these AI generated photos are only going to get better. That is why I chose to make my project about DeepFake Detection because in the very close future we need to know if what we are seeing is reality or fake. The goal of this project is to create an AI DeepFake Detector that can recognize AI generated content (images). The scope of my project is not going to go in the realms of me creating my own deep fake generator. This is often the case where some people will create a Deep Fake AI Generator to train another AI Deep Fake Detector what Deep Fakes look like. The scope will purely lie in the field of detecting not generating ai made content. In respect to deliverables, I plan on creating multiple python files that will be used to code up and train the model. However, I will upload the trained model to a Jupyter Notebook which will allow me to better show the class what the AI Detector is doing. I plan on using the Midjourney App during my presentation by prompting it to create an AI generated image. I then plan on putting that image into my AI Detector (in Jupyter Notebook) and showing the class what the AI prediction of the image is, whether it's fake or not. I will also test it with real photos I actually took in my life.

The first thing I did for this project was dive into the literature on previous research attempts to detect AI generated content. I realized that there is a great difference between different types of AI generated content detection (images vs. videos vs. sound). So I decided to reduce the scope of my project even more and focus on AI-generated Images. After diving into the literature I then moved to find a suitable dataset. I ended up finding a wonderful dataset called GenImage. This dataset had 1,000,000 Fake/Real image pairs. This dataset would have been perfect to train my model however the entire dataset was written in Chinese so I wasn’t able to use it. I ended up using a Kaggle dataset called CIFAKE which contained 60,000 Fake/Real image pairs. It is considerably smaller but it would be easier to train my model on. The question now was to determine which AI model to use.

The literature was split between training from scratch completely new models and using existing pre-trained Convolutional Neural Networks (CNN) models. I vied for the latter as it was least compute intensive (my only limiting factor). There are a variety of pre-trained CNN models to use ResNet, DenseNet, and etc. I chose DenseNet towards the end because it's a relatively small model that can be more easily trained. Training the model was difficult because I had to

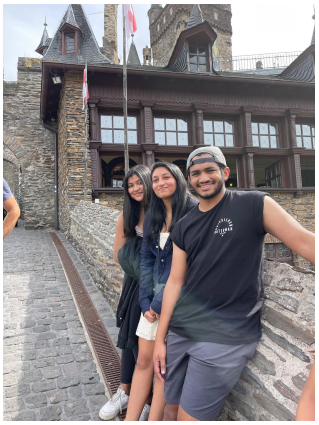
Layer (type)	Output Shape	Param #
densenet121 (Functional)	(None, 7, 7, 1024)	7037504
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 1024)	0
dense_49 (Dense)	(None, 256)	262400
dropout_27 (Dropout)	(None, 256)	0
dense_50 (Dense)	(None, 1)	257

Total params: 7300161 (27.85 MB)
Trainable params: 262657 (1.00 MB)
Non-trainable params: 7037504 (26.85 MB)

Figure 2: Final Model Architecture

make crucial decisions on the added structure on top of the DenseNet model. The way you train a pre-trained CNN is you add more layers to the end of the CNN that you further train on your dataset. The problem is you don't know the right amount of layers and nodes to add. In figure 2, you can see the model configuration I ended up using. The addition of only one extra large layer with 256 nodes seemed to be the best. The way I trained the model required a lot of trial and error, mainly with the learning rate.

Testing my trained model was by far the most fun. Usually, the way people test models is they just get a section of their dataset they don't train on and test on that. I decided however to test my AI generated Image Detector on images I in fact AI generated and images I took myself (so I know they're real). I found and used a series of photos of myself in Germany as you can see in figure 3. I also subscribed to Midjourney, a premium AI-generated Image generator to date. The subscription was $10 dollars but worth it.



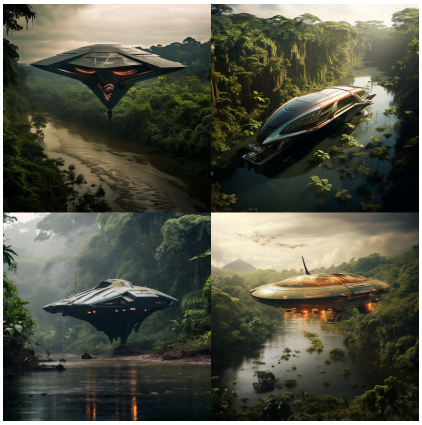


Figure 3: Photo taken of me
Figure 4: Image I created on Midjourney using prompt "a starship landing in the Amazon Rainforest"

The way it works is you feed it a prompt and it generates an image of your text prompt. One of the prompts I used was "a starship landing in the Amazon Rainforest" which you can see in figure 4. The reason I used such a weird prompt is to ensure that there was no real image of that object that could have been in my dataset. The tests went well however it had a slight tendency to choose fake images when it was confident.

I created a very well maintained readme on my Github page [3]. It was very interesting documenting all the material that I used for this process. I listed where all the files were, how to

run the model and even how to train your own. It took a lot longer to document how to run and use everything then I previously thought. I also linked a video of my recorded presentation in references and on my github [4].

References

[1] https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images
[2] MilesZim Images came from:
https://twitter.com/mileszim/status/1613965684937224192/photo/4
[3] My Github: https://github.com/rjoseph6/DeepFake-AI-Detector/tree/main
[4] https://youtu.be/eBz0ecB39Zk