Final Project Report for CS 175, Spring 2020 - Deepfake Detection

Ariel Jiang

Student ID: 86225946
E-mail: shuxinj@uci.edu

1. Introduction

The state-of-the-art of deep learning evolves rapidly in recent years. One of the most widely-dicussed media type of deep learning is “Deepfake”, in which a person in an existing image or video is replaced with someone else's likeness[1]. With the rise of generative adversarial network (GAN) proposed by Goodfellow et al. [3], the techiniques that synthesize Deepfake have advanced so far that they are causing ethical concerns across all social aspects with their potentially malicious intents. As a result, the importance and relvevancy of developing a “countermeasure” that is capble of detecting Deepfake raise. In this project, I will be focusing on utilizing the given data set of the “Deepfake Detection Challenge”[2] on Kaggle and exploring the multiple approaches to develop efficient machine learning algorithms that can distinguish Deepfake videos.

Although this topic is rather new in the machine learning area, there are already plentiful studies showcasing the possible ways of detecting Deepfake, each of them proving novel insights into solving the problem itself. I will be studying their research methods and employing some of them in this project. That being said, this project will not be able to thoroughly use and test every file provided in the 50 GB data set. Rather, it focuses on validating the available techniques that are already developed by other people in this area as well as comparing and analyzing their performance. Consequently, this project will only train and test on a small portion of the data set (10 GB) to produce fast feedback. Most of the techniques used will be an adaption of other people’s research studies and the focus will be the examinatin of their robustness and effectiveness.

2. Related Work

With recent research efforts from many machine learning scholars, the frontline of Deepfake detection is constantly moving forward[9]. In a 2019 paper written by Irene Amerini et al., they propose using *optical flow fields* to distinguish Deepfake from orignal videos[4]. “Optical flow” is a vector field which is computed on two consecutive frame f(t) and f(t + 1) to extract apparent motion between the observer and the scene itself. They believe that optical flow is able to exploit discrepancies in motion across frames synthetically created with respect to those that are real. In *FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals*, an effort made by Umur Aybars Ciftci et al., they believe the extraction of biological signals (remote photoplethysmography, or PPG) contain valuable information in the context of fake videos[5]. They have discovered that inter-consistency of PPG signals from various locations on a face is higher in real videos than synthetic ones. Meanwhile, Li et al. proposed a method based on the observations that current Deepfake algorithm can only generate images of limited resolutions, which need to be further warped to match the original faces in the source video. Such transforms leave distinctive artifacts in the resulting Deepfake videos, and these artifacts can be effectively captured by convolutional neural networks (CNNs)[6].

In this project, I will selectively test and evaluate thses methods and some other methods based on the scope of this class and my own limitations on time and resource.

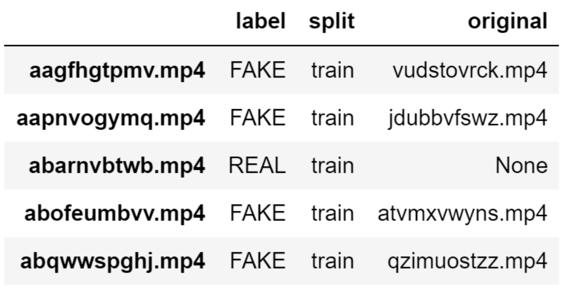
3. Data Set

The data set used in this project is provided by Kaggle's "Deepfake Detection Challenge"[2]. The challenge is to correctly identify the classification labels, eiter "fake" or "real", for a given video with using method available.

Originally, this data set contains 50 GB of data. For the sake of time and computational recourses, I reduce the data set to 4GB (sample set) and 10 GB (train set) to only focus on algorithm exploring and studying rather than accuracy.

3.1 Data Exploration

3.1.1 Structure Exploration. The sample data set is split into two parts, training and testing[8]. After a quick scan of the files containing in the data set, there are 401 files in the training folder and 400 files in the testing folder. All files are in mp4 format except one metadata json file located in the training folder. Each training data point is labelled either "fake" or "real" in the json file. In the case that the file is fake, the json file also provides the original video that is used to produce the fake one.



	label	split	original
aagfhgtpmv.mp4	FAKE	train	vudstovrck.mp4
aapnvogymq.mp4	FAKE	train	jdubbvfswz.mp4
abarnvbtwb.mp4	REAL	train	None
abofeumbvv.mp4	FAKE	train	atvmxvwyns.mp4
abqwwspghj.mp4	FAKE	train	qzimuostzz.mp4

Figure 1. *Visualization of the metadata file*

As of the distribution of the two labels, the real videos are only 19.25% in train videos, the fake videos acounting for 80.75% of the samples.

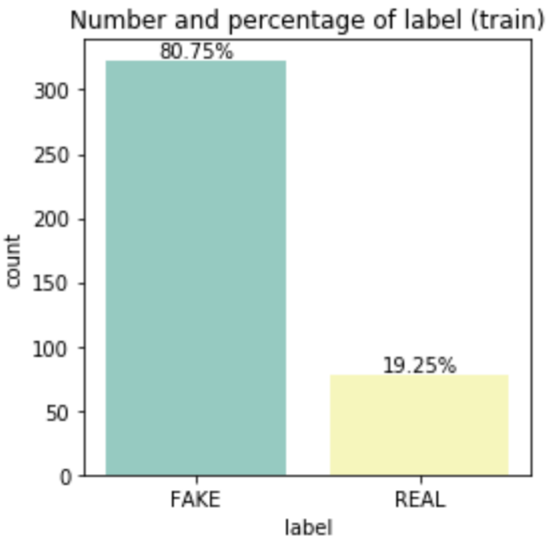


Figure 2. *Distribution of FAKE and REAL train videos*

3.1.2 Video Exploration. We now examine the content of the video files using libraries such as OpenCV[7]. Specifically, I try to use the open-source detection algorithm from "haarcascade" to identify the body parts of the subject in the videos. In this case, the eyes are labelled with red circles, face with green rectangles, and profile with blue rectangles. This helps narrow down the area the problem is dealing with.

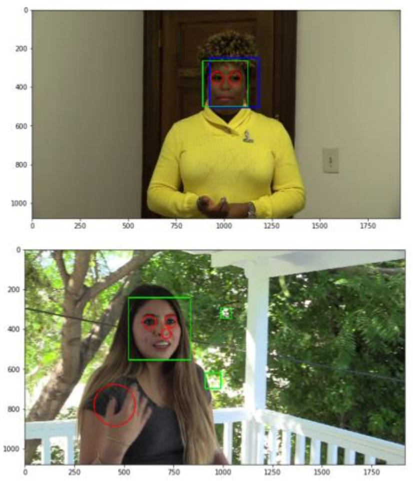


Figure 3. *The accuraccy of face detection fluctuates when the subject is not looking frontaly or when the luminosity is low.*

4. Description of Technical Approach

4.1 CNN

Since this challenge ultimately deals with computer vision and image processing, my first intuition is to focus on building different CNN models based on other people's research work and comparing their performance later.

4.1.1 Region of Interest (ROI). To simplify the process, I need to find the region of interest (ROI) of each video file, which is the face of the subject. This is done by the cascade tools provided by OpenCV library combined with the "haarcascade" face detection classifier provided on Github, However, some of the videos in the training set do not have a clear region of where the face is due to bad lighting. In that I case I used a 256 x 256 blank space to replace it. Some of the videos have faces that have dimensions way less than 256 x 256, in that case I resize the images.

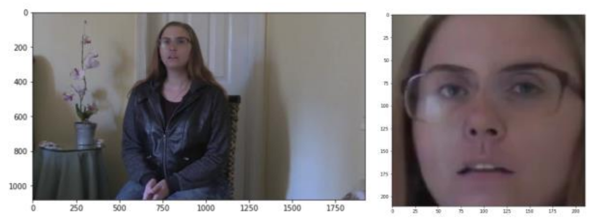


Figure 4. *Finding the region of interest of a frame using face detection.*

4.1.2 Model Construction – Model I. For the first model, I decide to choose Deep Feature Extraction in VGGFace [10] for deep face extraction through the VGG-Net described in Fig 5[11]. I adopted this model (originally fits 512 x 512) to fit my input size, which is 256 x 256. I used 1000 data points as my training data set and 200 data points to be my validation data set during the training. For optimizer, I choose to use the Adam optimization, which is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments[13]. Because of the data set is heavily imbalanced, I also adjusted the initial bias value to *log([pos/neg])* to accommodate it. Additionally, the accuracy of the prediction does not really speak of the performance of the model, since more than 80% of the data is labelled "FAKE", it is relatively easy to get a high accuracy by just predicting "FAKE". As a result, I added class weight which treats REAL as multiple instances of FAKE videos when calculating loss for gradient descent.

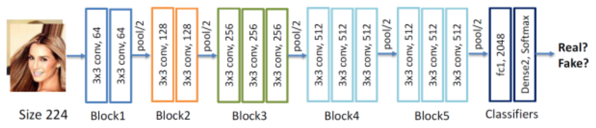


Figure 5. *VGG-Net architecture.*

4.1.2 Model Construction –Model II. For the second model, I adapt the approach proposed by Alfcher et al.[14], who used a low number of layers to focus on the mesoscopic properties of images. In Fig 6, the architecture of this model is illustrated in details. In total, there are 27,977 trainable parameters for this network.

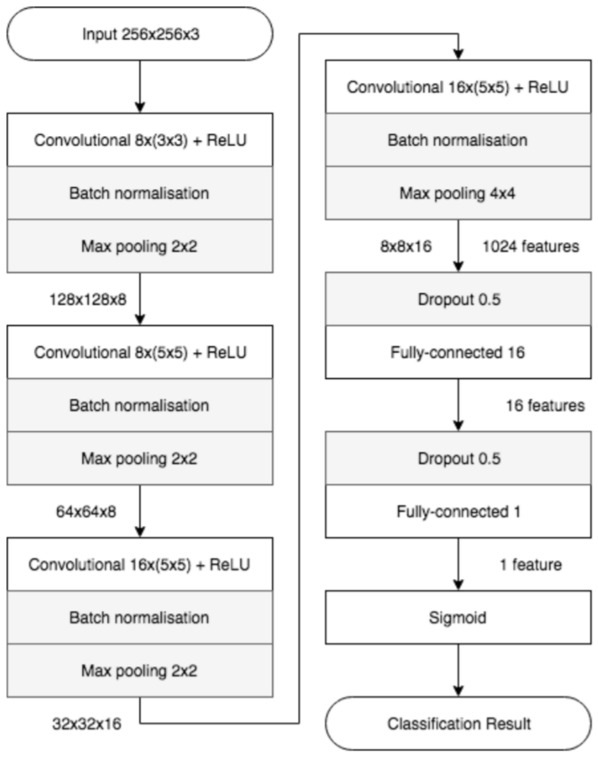


Figure 6. *The network architecture of Meso-4[14].*

I used the same optimizer, and measure the same loss as of the previous model. Additionally, I tried another version of the MesoNet, called MesoInception. The idea of the module is to stack the output of several convolutional layers with different kernel shapes and thus increase the function space in which the model is optimized[14]. Instead of the 5 × 5 convolutions of the original module, I use 3 × 3 dilated convolutions [15] in order to avoid high semantic.

5. Software

5.1 My Script

Data Exploration	Data preprocessing – writing the logic for *picking outliers, eliminating data noise, metafile information extraction*
Model Buidling and Training	Writing code for *constructing CNN model*
Evaluation	Writing code to *plot and compare test accuracy*

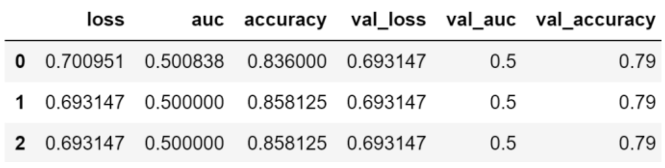
5.2 Code from Others

Data Exploration	Used library: For data processing – *python standard libraries, numpy* For visualization – *Matplotlib, pandas* Used Code: *DeepFake Starter Kit,*
Model Buidling and Training	Backend: *Tensorflow* Face Detection: *OpenCV, haarcascade* Used Code: *Face Detection with OpenCV*
	Finding ROI: *Basic EDA Face Detection, split video and ROI*
Evaluation	Used library: numpy & matliplot

6. Experiments and Evaluation

6.1 Model I - VGGNets

This model has a really high volume of parameters (3,745,890), thus takes a significant amount of time to train. I was only able to run 3 epochs before the computer runs out of memory. It is not hard to tell that the model produces an overfitting result and has no improvement over training. The AUC value of .5 shows that there is no distinction between the two classes. It is indeed a highly disappointed model in this case.



	loss	auc	accuracy	val_loss	val_auc	val_accuracy
0	0.700951	0.500838	0.836000	0.693147	0.5	0.79
1	0.693147	0.500000	0.858125	0.693147	0.5	0.79
2	0.693147	0.500000	0.858125	0.693147	0.5	0.79

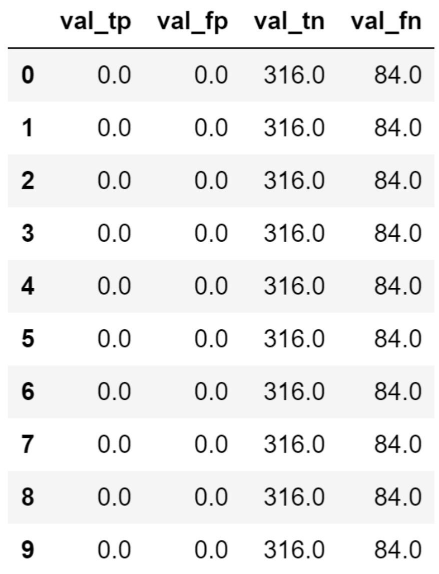
Figure 7. *VGGNets loss, AUC, and accuracy for 3 epochs*

6.2 Model II - MesoNet

This model has a reasonable amount of parameters (around 20,000) so I was able to experiment with multiple class weights while training to see the difference between their performance. Since the data set is very imbalanced and more than 80% are “FAKE” videos, I did not choose “accuracy” as the metric I’m measuring, instead, I used AUC, which I think is a better representation.

The parameter I’m experimenting with is mainly class weight, which tells the loss function to penalize a wrongly labelled “REAL” video over

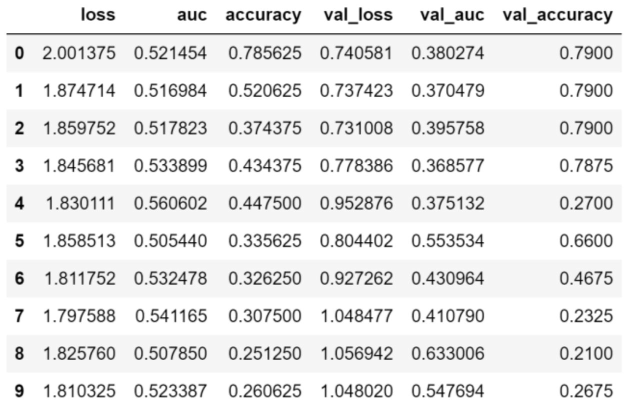
"FAKE" video. The class weight ratios I chose were 1:1, 1:5, 1:10, 1:15, and 1:25.



	val_tp	val_fp	val_tn	val_fn
0	0.0	0.0	316.0	84.0
1	0.0	0.0	316.0	84.0
2	0.0	0.0	316.0	84.0
3	0.0	0.0	316.0	84.0
4	0.0	0.0	316.0	84.0
5	0.0	0.0	316.0	84.0
6	0.0	0.0	316.0	84.0
7	0.0	0.0	316.0	84.0
8	0.0	0.0	316.0	84.0
9	0.0	0.0	316.0	84.0

Figure 8. *When class weight ratio is 1:1 (no discrimination), the model always predicts all negative. (tp, fp, tn, fn, stand for true positive, false positive, true negative, false negative)*

I notice that when the ratio is small, such as 1:1, no matter how the model trains, it will always end up predicting all negative (Fig 8), as by nature there are more FAKE videos. Thus, the AUC of these trials are around .5. However, I notice that when I increase the class weight to 1:15, there is a significant improvement to the AUC value (Fig 9).



	loss	auc	accuracy	val_loss	val_auc	val_accuracy
0	2.001375	0.521454	0.785625	0.740581	0.380274	0.7900
1	1.874714	0.516984	0.520625	0.737423	0.370479	0.7900
2	1.859752	0.517823	0.374375	0.731008	0.395758	0.7900
3	1.845681	0.533899	0.434375	0.778386	0.368577	0.7875
4	1.830111	0.560602	0.447500	0.952876	0.375132	0.2700
5	1.858513	0.505440	0.335625	0.804402	0.553534	0.6600
6	1.811752	0.532478	0.326250	0.927262	0.430964	0.4675
7	1.797588	0.541165	0.307500	1.048477	0.410790	0.2325
8	1.825760	0.507850	0.251250	1.056942	0.633006	0.2100
9	1.810325	0.523387	0.260625	1.048020	0.547694	0.2675

Figure 9. *When the class weight ratio is 1:15 (15 instances of "FAKE" count as 1 instance of "REAL")*

While the ROC curve shows improvement as the ratio increases, the accuracy starts to drop, which is an understandable behaviro in this case, since the model is not predicting all negative anymore. Fig 10 shows the AUC curve comparison of the different class weight.

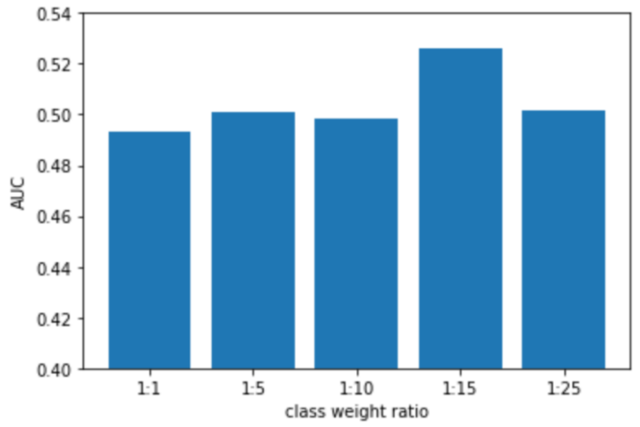


Figure 10. *AUC curve comparison of the different class weight.*

6.2 Comparison and Takeaways

It is clear that a too complicated CNN net is not suitable for this challenge. Meanwhile, because of the imbalance of the data set, I believe that the key to construct an accurate CNN model is to play with the class weights. Even though my limited experiments and trails on these models do not produce very accurate result, I think it hints at the direction where a better performance can be achieved.

7. Discussion and Conclusion

Personally speaking, this project has been an eye-opening learning experience for me. First of all, I was given the opportunity to explore a very popular and relevant topic of the current world, allowing me to keep track of the rapid development of modern Machine Learning. In order to find a place to start, I researched and read through many papers about this topic. They broadened my knowlege on the scope of the current AI development and showed me the infinite things that machine learning can achieve, which then gave me motivation to further expand the scope of my own project.

After reading and carefully understanding their methods, I was able to construct complicated CNN modelds and experiment with them. This process has deepened my understanding of neutral networks and why they are at the frontline of current machine learing study. I was able to see how the configuration of hyperparameters play a role in deciding overfitting/underfitting, and how different metrics are able to tell different stories about the same model. At the same time, however, from experimenting with CNNs, I can not help but notice the limitation of this algorithm and how more advanced Deepfake generation techniques can surpass its detection. Most of the CNNs approaches rely on the "reverse-enginerring" of GANs, which is the most popular technique that generates Deepfake. It picks up on the several flaws GAN has, such as motion blurriness and edges detachment. However, as more advanced techniques emerge in the future, I am uncertain this "reverse-engineering" approach will always work.

While I was doing extensive research for this project, however, I came across some interesting papers. One of them talked about picking up the biological signal inside of an image, while the other one talks about mesosphysics. Both of these features are basically exlusive to computer and are almost impossible to be detected by human, which I think is actually a more reliable approach. If I were to further study this topic and improve the performance of current detection techniques, I would focus more on micro-level of the Deepfake images and go from there.

References

[1] Brandon, John (16 February 2018). "Terrifying high-tech porn: Creepy 'deepfake' videos are on the rise". *Fox News.* Retrieved 30 May 2020.

[2] "Deepfake Detection Challenge". *Kaggle.* Retrieved 30 May 2020.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y.Bengio. Generative adversarial nets. In Z.Ghahramani, M. Welling, C. Cortes, N. D.Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27,* pages 2672–2680. Curran Associates, Inc., 2014.

[4] Amerini, Irene, Leonardo Galteri, Roberto Caldelli, and Alberto Del Bimbo. "Deepfake Video Detection through Optical Flow based CNN." In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0-0. 2019.

[5] Ciftci, Umur & Demir, Ilke. (2019). FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals.

[6] Li, Yuezun, and Siwei Lyu. *"Exposing deepfake videos by detecting face warping artifacts."* arXiv preprint arXiv:1811.00656 (2018).

[7] *Face Detection with OpenCV*, Kaggle Notebook.

[8] *DeepFake Starter Kit,* Kaggle Notebook.

[9] *State of the Art. Deepfake Detection references,* Kaggle Discussion.

[10] Do Nhu, Tai & Na, In & Kim, S.H.. (2018). Forensics Face Detection From GANs Using Convolutional Neural Network.

[11] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," in Procedings of the British Machine Vision Conference 2015, 2015.

[12] *Basic EDA Face Detection, split video and ROI*, Kaggle Notebook.

[13] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[14] Afchar, Darius, et al. "Mesonet: a compact facial video forgery detection network." *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2018.

[15] J.-W. Lee, M.-J. Lee, T.-W. Oh, S.-J. Ryu, and H.-K. Lee. Screenshot identification using combing artifact from interlaced video. In

Proceedings of the 12th ACM workshop on Multimedia and security, pages 49–54. ACM, 2010.