Project Report

Omkar Joglekar ID - 932731094
Samuel Lilek ID - 954437307

22 January 2022

Contents

Chapter 1

Introduction

pass

Chapter 2

Build Faces Dataset

2.1 Intro

Question 1

Implemented in *faces_dataset.py*

Question 2

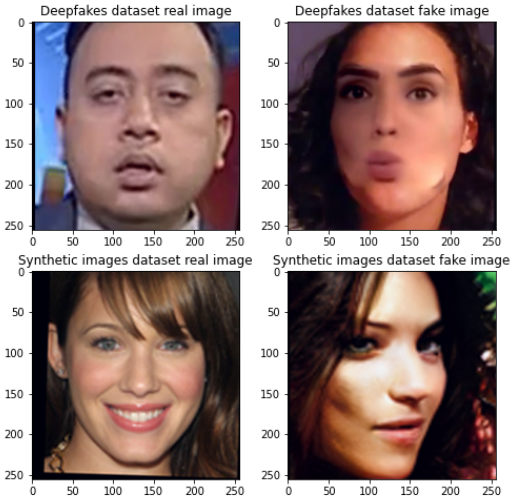

Figure 2.1: Fakes dataset samples

Chapter 3

Write an Abstract Trainer

3.1 Implement an Abstract Trainer

Question 3

Implemented in trainer.py

Question 4

Implemented in trainer.py

3.2 Train a Deepfake Detection Classifier

Question 5

Ran the line:
$python\ train\_main.py\ -d\ fakes\_dataset\ -m\ SimpleNet\ --lr\ 0.001\ -b\ 32\ -e\ 5\ -o\ Adam$

3.3 Analyze the Deepfake Detection Classifier

Question 6

The json makes sense. Over the course of five epochs, training loss continuously decreases as training accuracy continues to go up. However, there is evidence of overfitting: while training accuracy continues to improve, validation and test set accuracies both begin to plateau.

Question 7

(figures 3.1 and 3.2)

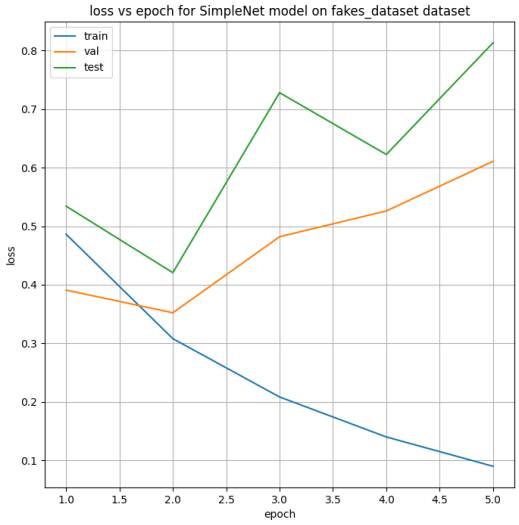

Figure 3.1: Fakes dataset SimpleNet Loss

Question 8

Test accuracy corresponding to highest val accuracy: 84.62% at epoch 3 (val acc was 87.29%)

Question 9

Test set has 1400 real images and 700 fake images. Fake/Real ratio of: 0.5

Question 10

(figures 3.3 and 3.4)

Question 11

The graphs for the scores in (A) and (B) look completely different because we have created a (real/fake) binary classifier; therefore, each curve shows the reflection of the other. For a given real image, a correct model output of "real"

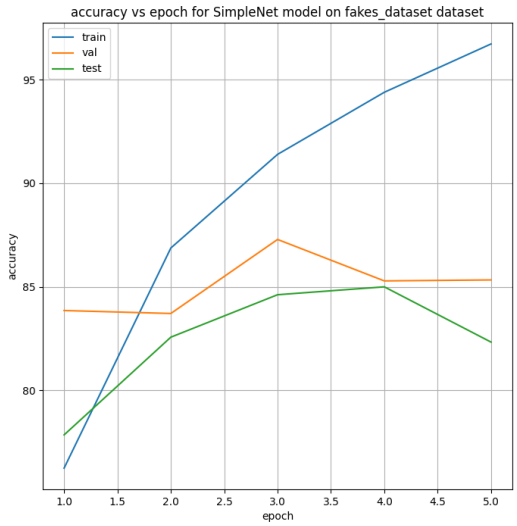

Figure 3.2: Fakes dataset SimpleNet Accuracies

results in a "true positive" for the real image score curve and a "true negative" for the fake image score curve - correct for both. For that same real image, an incorrect model output of "fake" results in a "false negative" for the real score curve and a "false positive" for the fake score curve - incorrect for both. The "fake" ROC curve reflects the opposite of the "real" ROC curve and vice-versa; the same is also true for the DET curves.

3.4 Train a Synthetic Image Detection Classifier

Question 12

Ran the line:
python train_main.py $-d$ *synthetic_dataset* $-m$ *SimpleNet* $--lr$ 0.001 $-b$ 32 $-e$ 5 $-o$ *Adam*

3.5 Analyze the Synthetic Image Detection Classifier

Question 13

(figures 3.5 and 3.6)

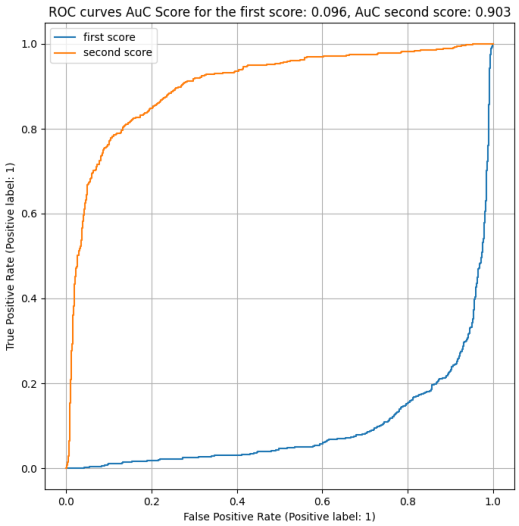

Figure 3.3: Fakes dataset SimpleNet ROC curve

Question 14

Test accuracy corresponding to highest val accuracy: 50.04% at epoch 1 (val acc was also 50.04% - reflects a single output classifier)

Question 15

Test set has 551 real images and 552 synthetic images. Fake/real ratio of: 1.0018 Note: with a total of 1103 images, guessing "fake" every time on this dataset would result in an accuracy of (552/1103)*100 = 50.04% - exactly what we observed for Question 14.

Question 16

We developed a single-output classifier. The network is not strong enough to make a good prediction; as a result, it simply guesses "real" or "fake" for everything that it sees. The accuracy rate is approximately 50%, meaning the classifier is not even better than chance.

Question 17

From looking at the sample images, this result makes sense. The fake images in the "Fakes" dataset were of fairly low quality, making it easy for a relatively small network to make accurate predictions. However, simply by looking at the

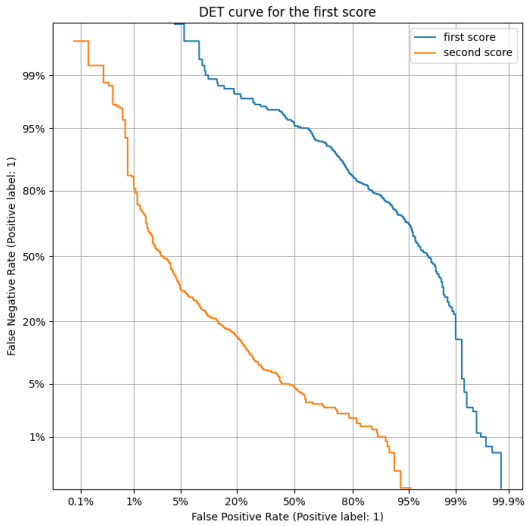

Figure 3.4: Fakes dataset SimpleNet DET curve

"Synthetic" dataset it is clear that the fake images are of a much higher quality. As a human reviewer, many of the images are extremely difficult to assess, so it makes sense that the SimpleNet would be unable to accurately predict real/fake in this instance.

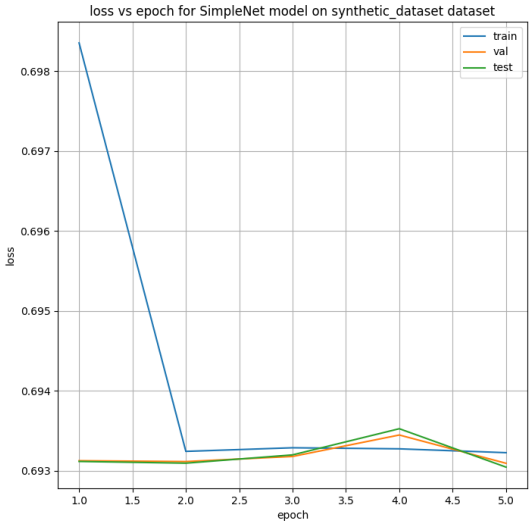

Figure 3.5: Synthetic dataset SimpleNet Loss

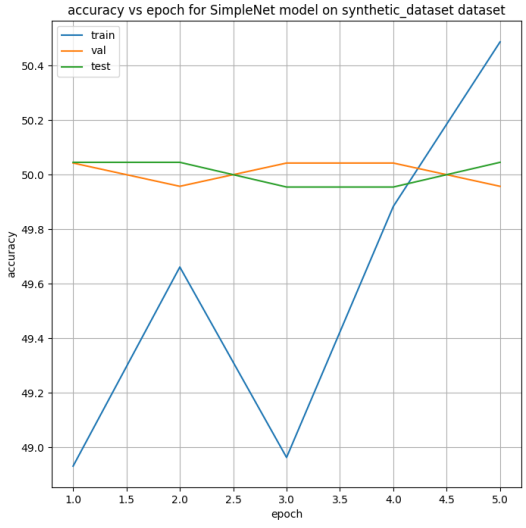

Figure 3.6: Synthetic dataset SimpleNet Accuracies

Chapter 4

Fine Tuning a Pre-Trained Model

4.1 Intro

Question 18

Xception is pre-trained on the ImageNet dataset (1000 classes).

Question 19

The basic building blocks of the Xception model are called Separable convolutional layers. In these types of convolutions, we separate the spatial convolutions from the pointwise convolutions. This saves memory space as well as increases computation speed. The idea is to perform the image transformation once and then apply multiple pointwise convolutions to obtain the desired output channels.

Question 20

(same as Q18).

Question 21

The final "fc" classification block is a (2048, 1000) linear layer. The size of the input dimension is 2048.

Question 22

The original Xception network has 22,855,952 parameters.

4.2 Attaching a new head to the Xception backbone

Question 23

Implemented in xception.py

Question 24

We added 272,834 parameters on top of Xception, for a total of 23,128,786 parameters

4.3 Train and evaluate a new architecture

Question 25

Ran the line:
$python\ train\_main.py\ -d\ synthetic\_dataset\ -m\ XceptionBased\ --lr\ 0.001\ -b\ 32\ -e\ 5\ -o\ Adam$

Question 26

(figures 4.1 and 4.2)

Question 27

Test accuracy for the highest validation accuracy we received: 97.10 at epoch 2 (val acc was 97.27)

Question 28

(figures 4.3 and 4.4)

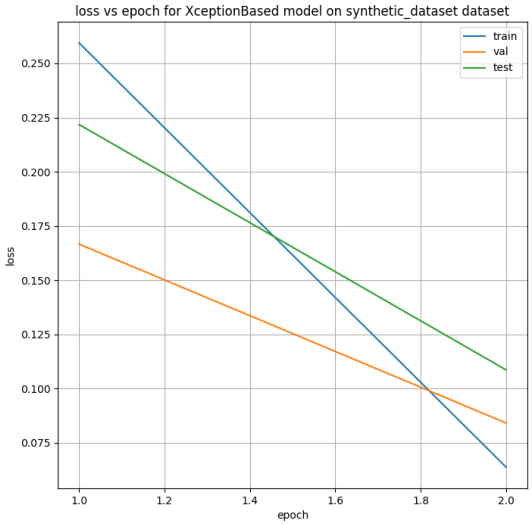

Figure 4.1: Synthetic dataset XceptionBased Loss

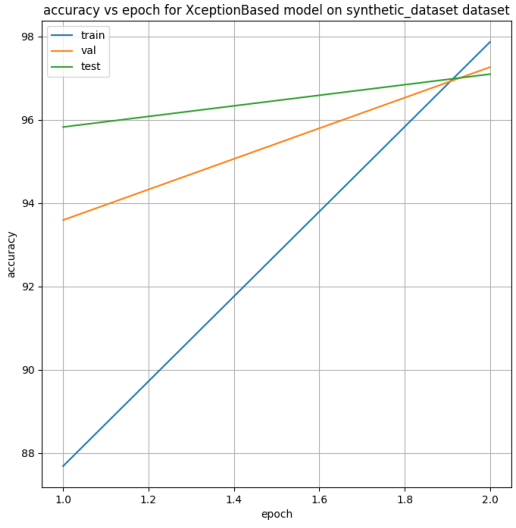

Figure 4.2: Synthetic dataset XceptionBased Accuracies

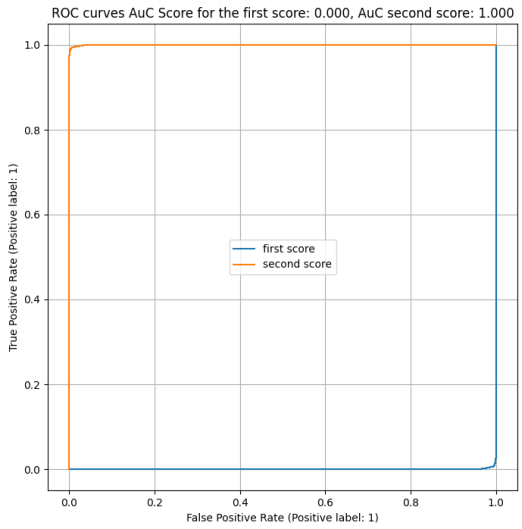

Figure 4.3: Synthetic dataset XceptionBased ROC curve

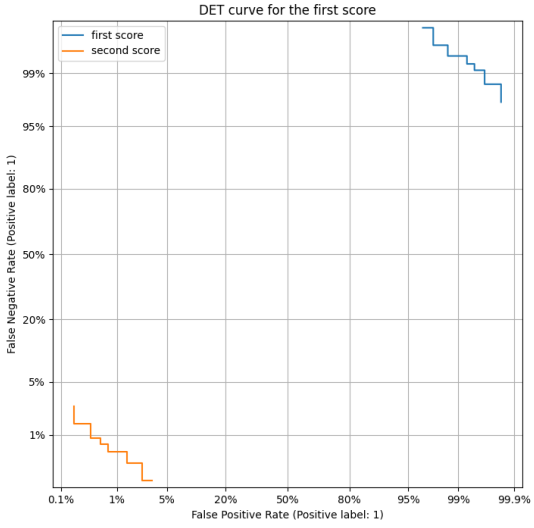

Figure 4.4: Synthetic dataset XceptionBased DET curve

Chapter 5

Saliency Maps and Grad-CAM analysis

5.1 Intro

Question 29

Saliency maps are used to visualize which regions in the image are considered important by the network when it is making its decision. It is calculated by computing the gradient of the class score of interest with respect to each pixel in the image. After gradient computation, you take the max of the absolute value of the gradient tensor in order to get a final saliency map.

Question 30

CAM or Class Activation Maps use Global Average Pooling on the outputs of the final convolutional layer in an image classification network to highlight the regions of interest in an image which led to the network deciding a particular class. Grad-CAM improves on this method and provides a more generalized scheme for interest region visualization.
Grad-CAM computes the gradients of the scores (before softmax) with respect to the feature map activations of the convolutional layer. These gradients flowing back go through the global average pooling operation to generate neuron importance weights for each class 'c' and channel 'k'.

5.2 Saliency Maps

Question 31

Implemented in *saliency_map.py*

Question 32

The saliency maps for fakes dataset show that the features such as eyes and noses and the pixels around these regions are of more importance when determining the class of the image (real/fake). The saliency maps of the XceptionBased model on the Synthetic dataset show that more important pixels in the image are around the mouth and cheeks and overall face texture. (figures 5.1, 5.2, 5.3 and 5.4)

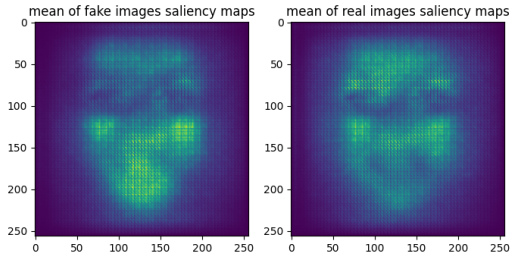

Figure 5.1: Synthetic dataset XceptionBased Mean Saliency map

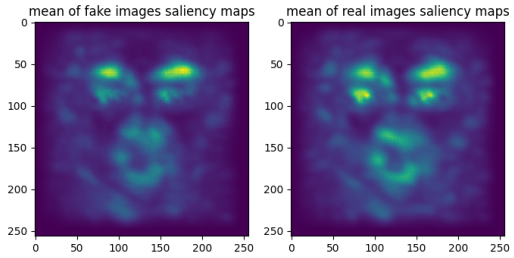

Figure 5.2: Fakes dataset SimpleNet Mean Saliency map

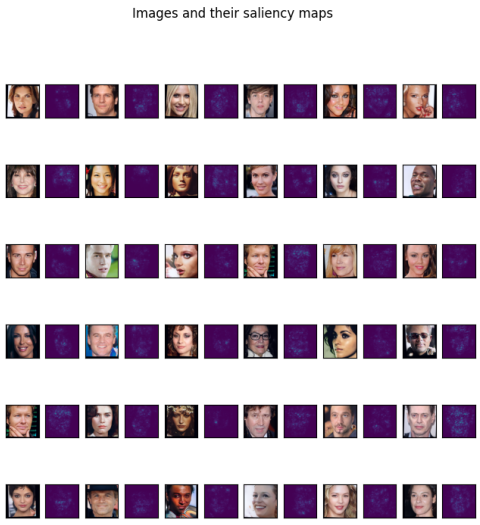

Figure 5.3: Synthetic dataset XceptionBased saliency maps and image pairs

5.3 Grad-CAM

Question 33

Installed version (1.3.6) and uploaded: *updated_environment.yml*

Question 34

Implemented in *grad_cam_analysis.py*. Because of the recent update in the API, the show_cam_on_image function is broken and showed errors while running. We have implemented the function manually and run that instead. All other code has been used according to the README.md provided in the repository.

Question 35

(figures 5.5, 5.6, 5.7, 5.8, 5.9 and 5.10)

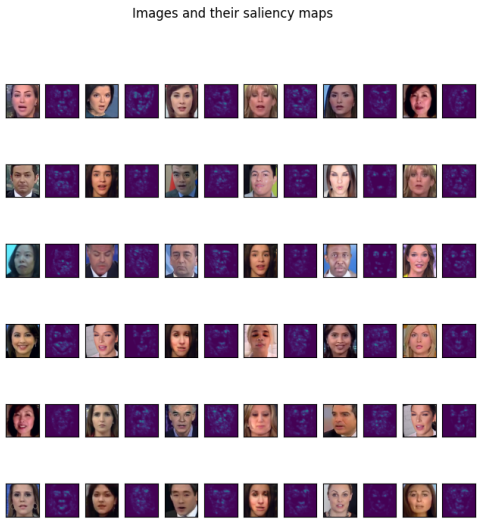

Figure 5.4: Fakes dataset SimpleNet saliency maps and image pairs

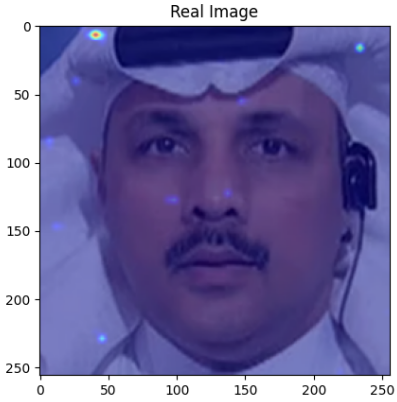

Figure 5.5: Real Image(Fakes Dataset), SimpleNet Grad-CAM

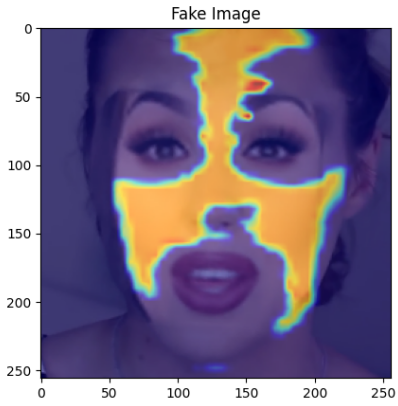

Figure 5.6: Fake Image(Fakes Dataset), SimpleNet Grad-CAM

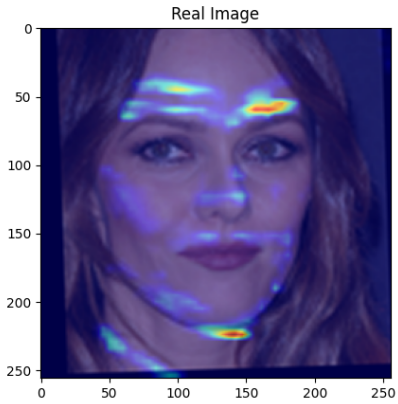

Figure 5.7: Real Image(Synthetic Dataset), SimpleNet Grad-CAM

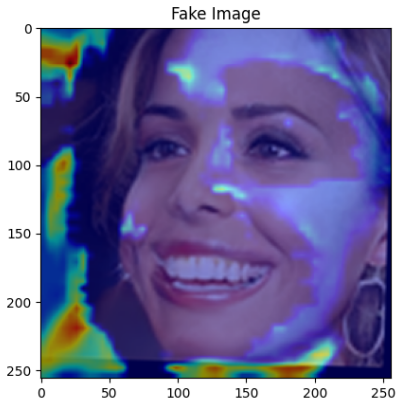

Figure 5.8: Fake Image(Synthetic Dataset), SimpleNet Grad-CAM

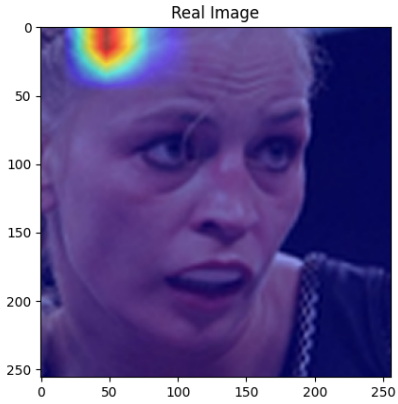

Figure 5.9: Real Image, XceptionBased Grad-CAM

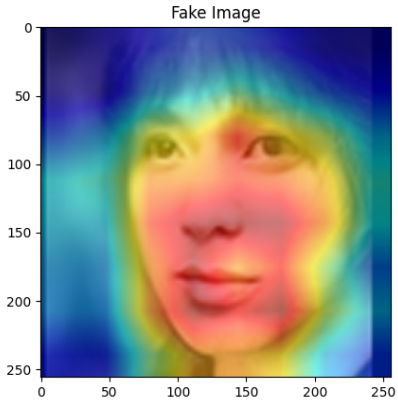

Figure 5.10: Fake Image, XceptionBased Grad-CAM