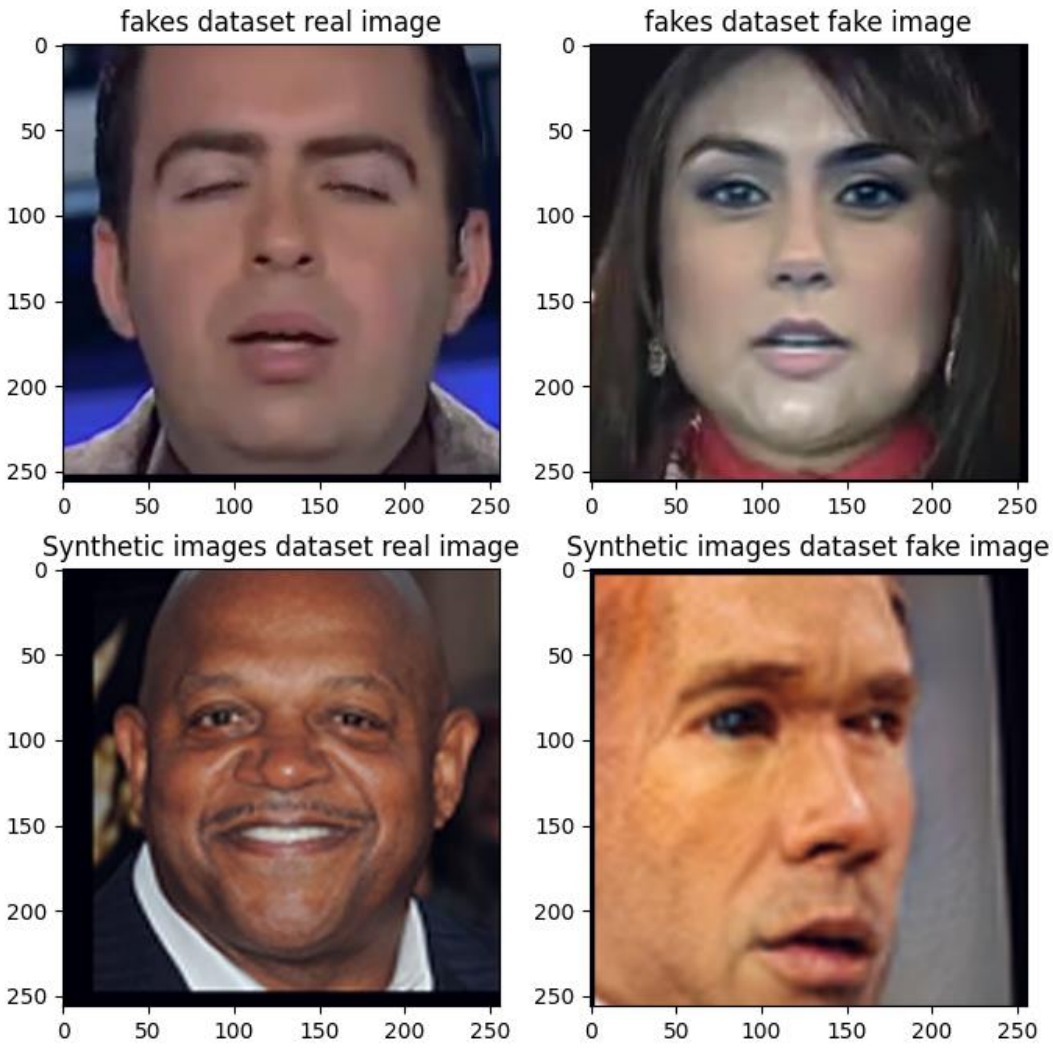
Ori Gilad – 315471375
Daniel Shalev - 207024621

Assignment 4 – Facial Manipulation Detection

Chapter 2 – Build Faces Dataset

1. We implemented the **__getitem__** and **__len__** dunder functions to create our custom pytorch dataset.
2. Running the **plot_samples_of_faces_dataset.py** script yield the following images:

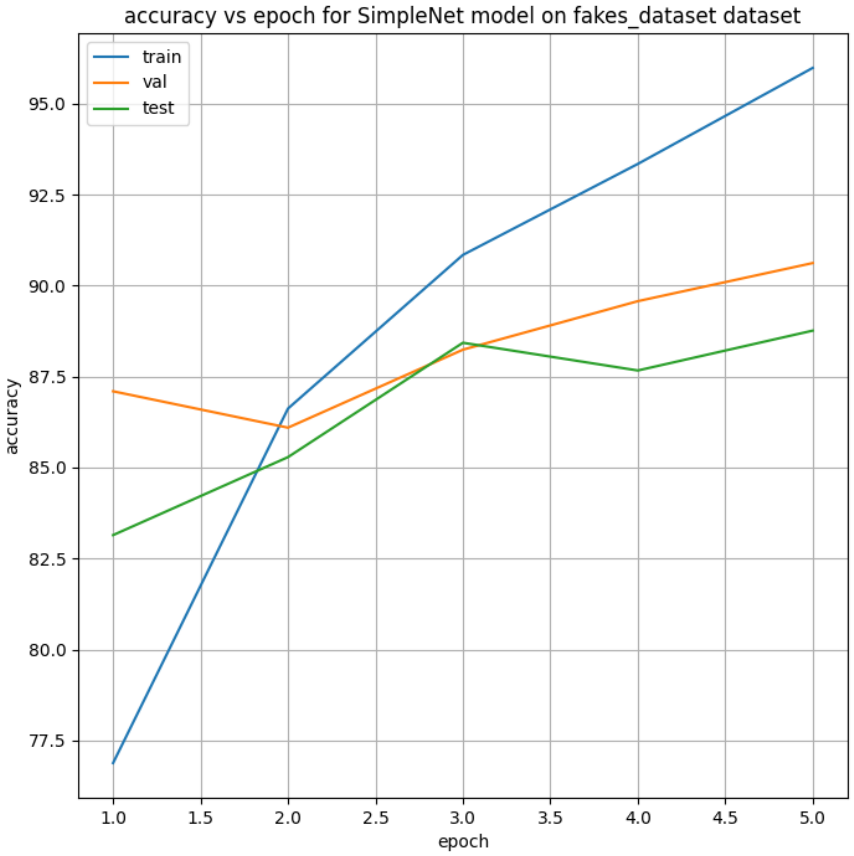


Ori Gilad – 315471375
Daniel Shalev - 207024621

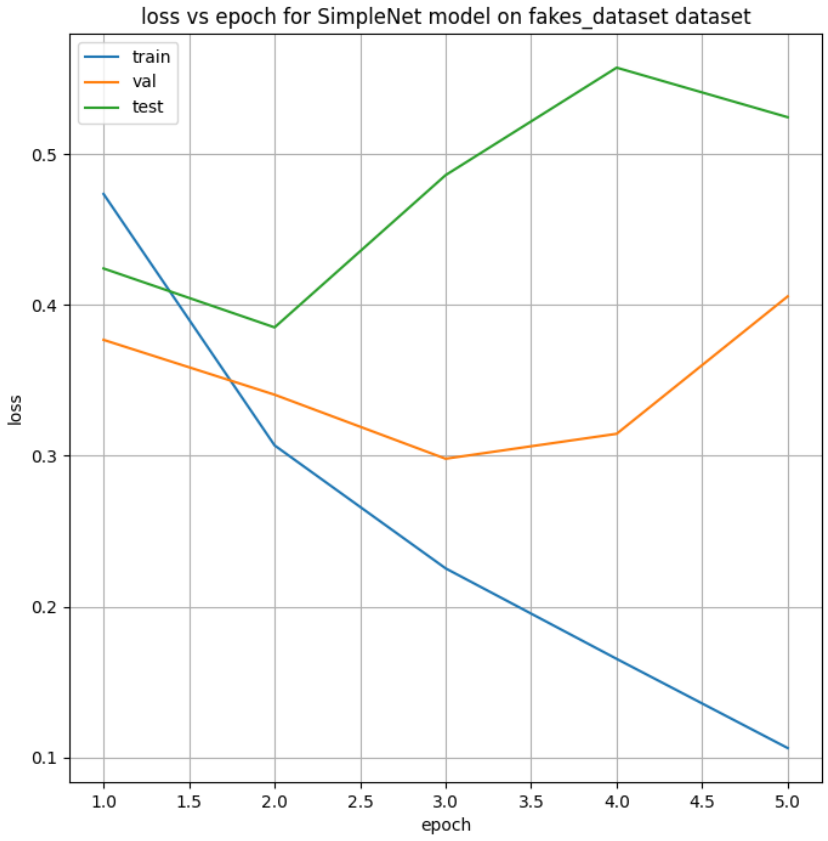
Chapter 3

3. We implemented the **train_one_epoch** function.
4. We implemented the **evaluate_model_on_dataloader** function.
5. We trained SimpleNet with the appropriate hyperparameters using our **train_main.py** script.

6. Looking at the numbers of the training loss and accuracy, we see that we get reasonable results. As the training process advances, the loss decreases while the accuracy improves.
   But, as we will see in the following question, the validation and test loss do not decrease along training time. Moreover, the validation and test accuracy do not increase monotonously over training time. This phenomenon is caused because we use only the training data during the training process.
7. We got the following plots:



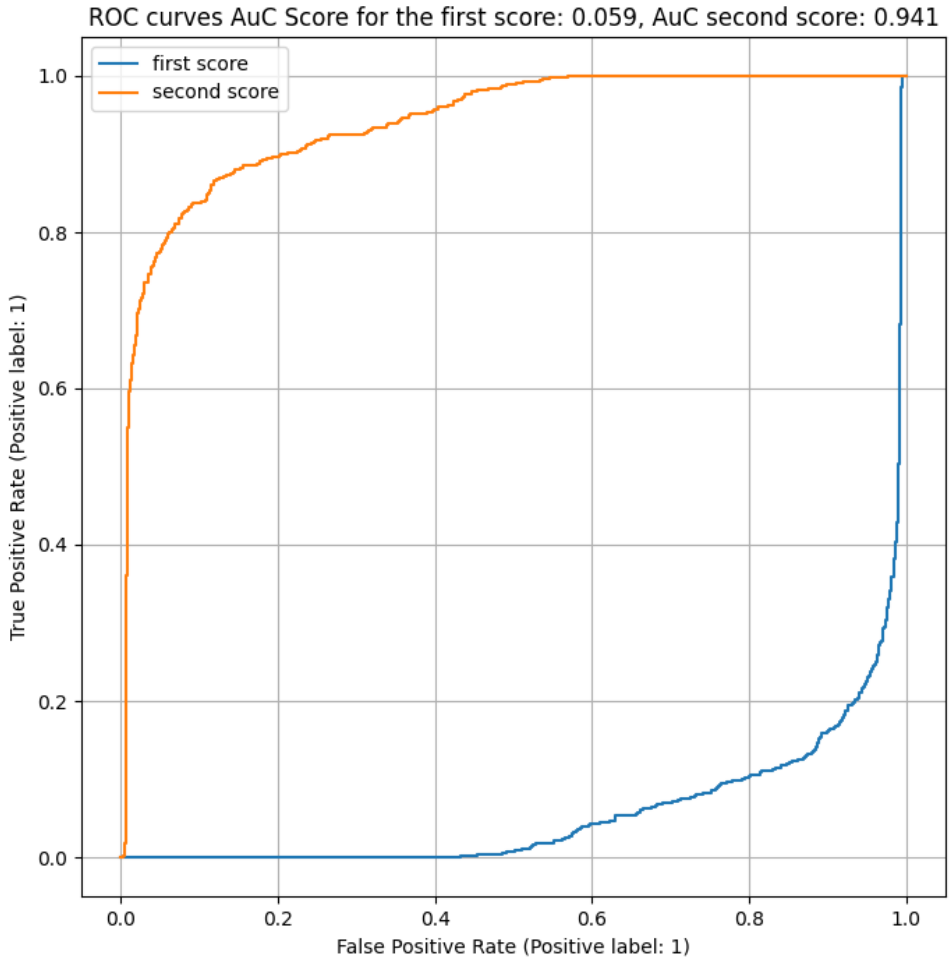
Ori Gilad – 315471375
Daniel Shalev - 207024621



8. Test accuracy: **88.7619**
9. On the fake dataset test set, there are 700 fake images and 1400 real images. Hence the proportion is $\frac{1}{2}$.

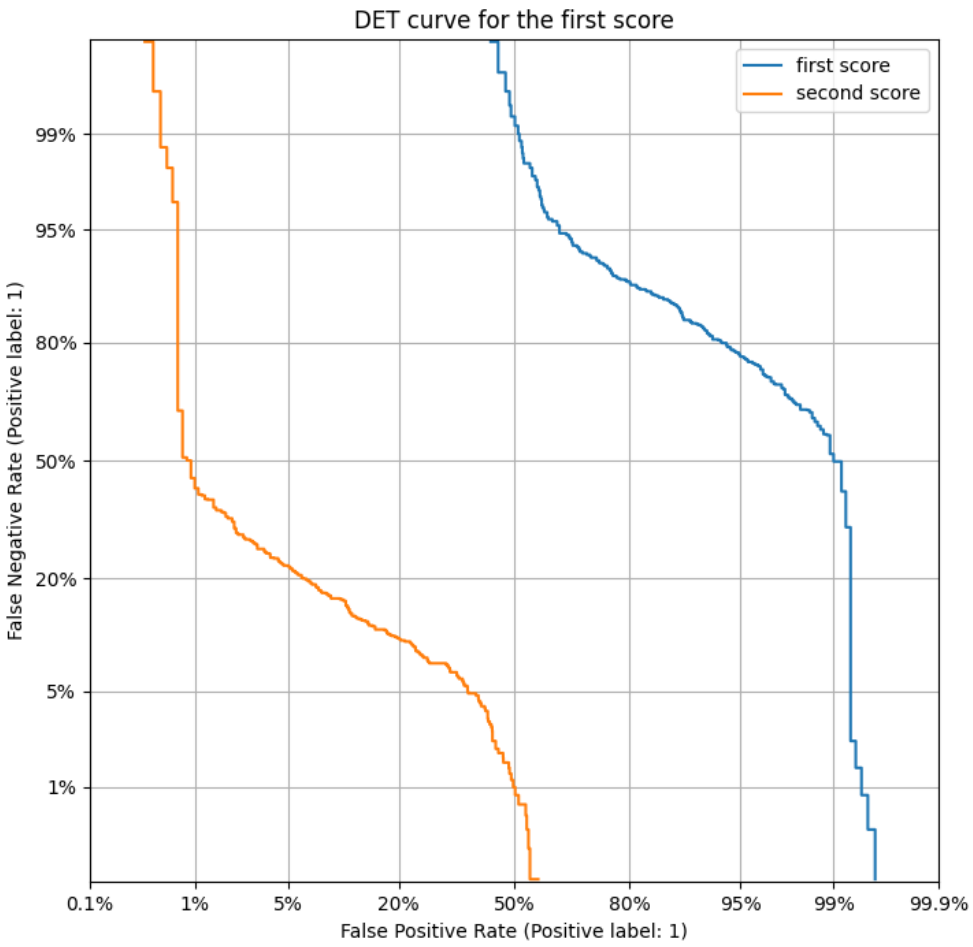
Ori Gilad – 315471375
Daniel Shalev - 207024621

10. After implementing the missing code in the **numerical_analysis** file, we generated the ROC and DET graphs for the SimpleNet trained on the fakes_dataset dataset.
    We got the following graphs:
    ROC:



Ori Gilad – 315471375
Daniel Shalev - 207024621

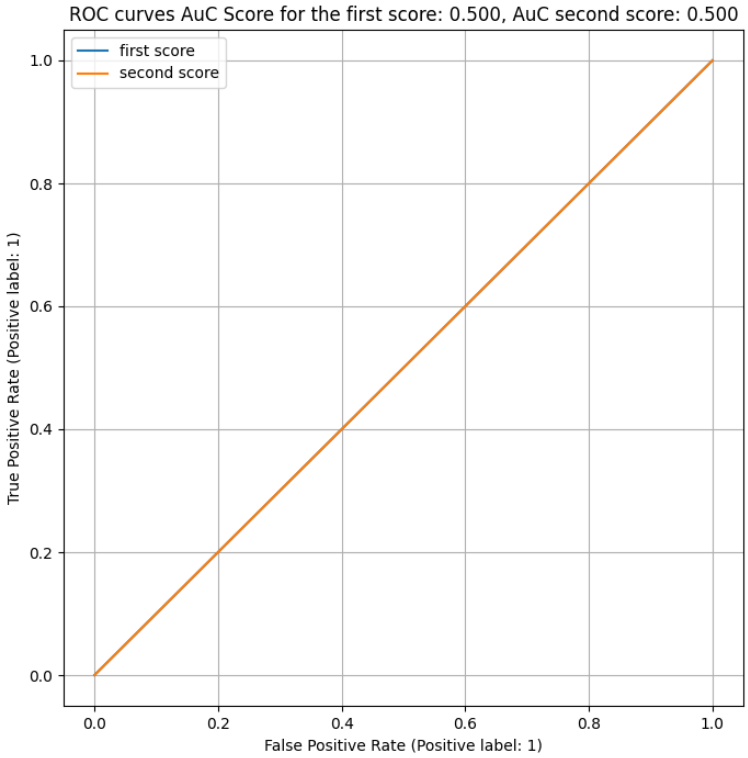
DET:



11. We enter different inputs to the same function - for example, one input is x , and the other is: $1 - x$.
    During the ROC calculation, while comparing each soft score to a certain threshold, we once compare the threshold to x and the other time compare it to $1 - x$, which causes axis inversion and reflection. Hence, we get the above ROC curves.

Ori Gilad – 315471375
Daniel Shalev - 207024621

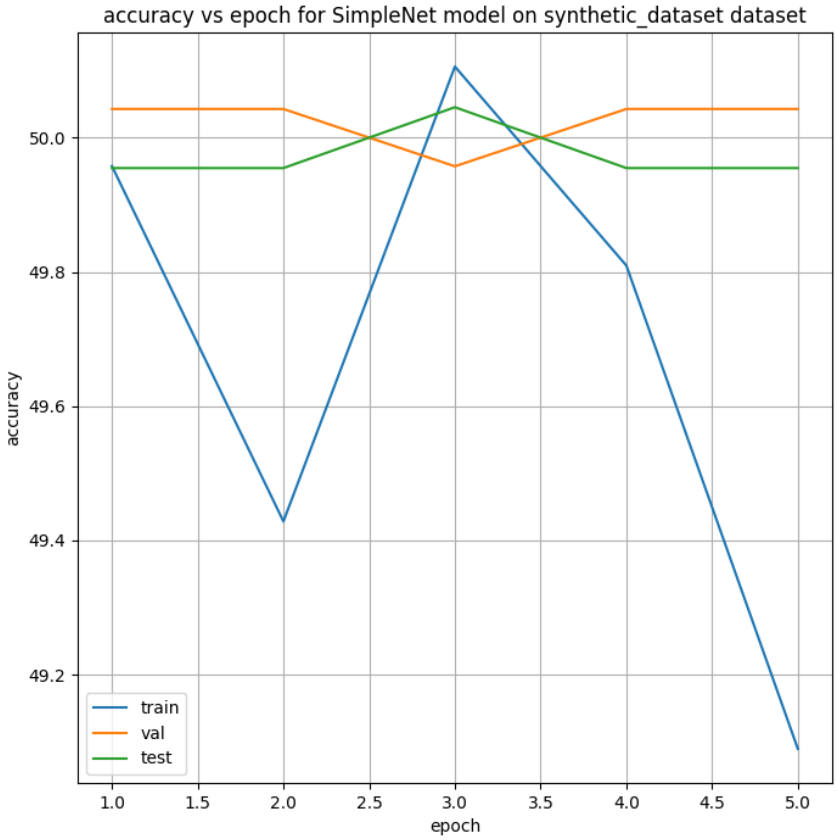
12. We trained to SimpleNet on the Synthetic faces_dataset dataset this time with the appropriate hyperparameters.
    After training, we plotted the ROC graph:



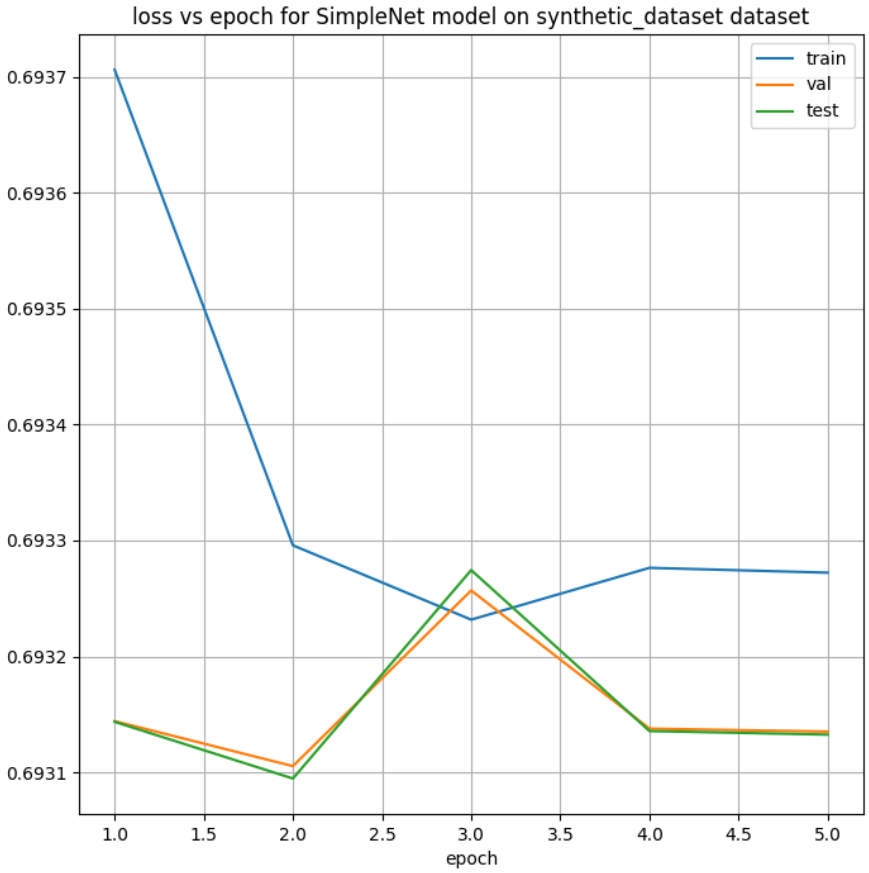
We see two linear ROC curves that coincide.

Ori Gilad – 315471375
Daniel Shalev - 207024621

13. We got the following accuracy and loss graphs for the synthetic_dataset:



Ori Gilad – 315471375
Daniel Shalev - 207024621



We are sad to see that our model failed to differ between real and fake images in this data set. We see an accuracy of 50% even on the train set, which equivalent to randomly guess the class of each image.

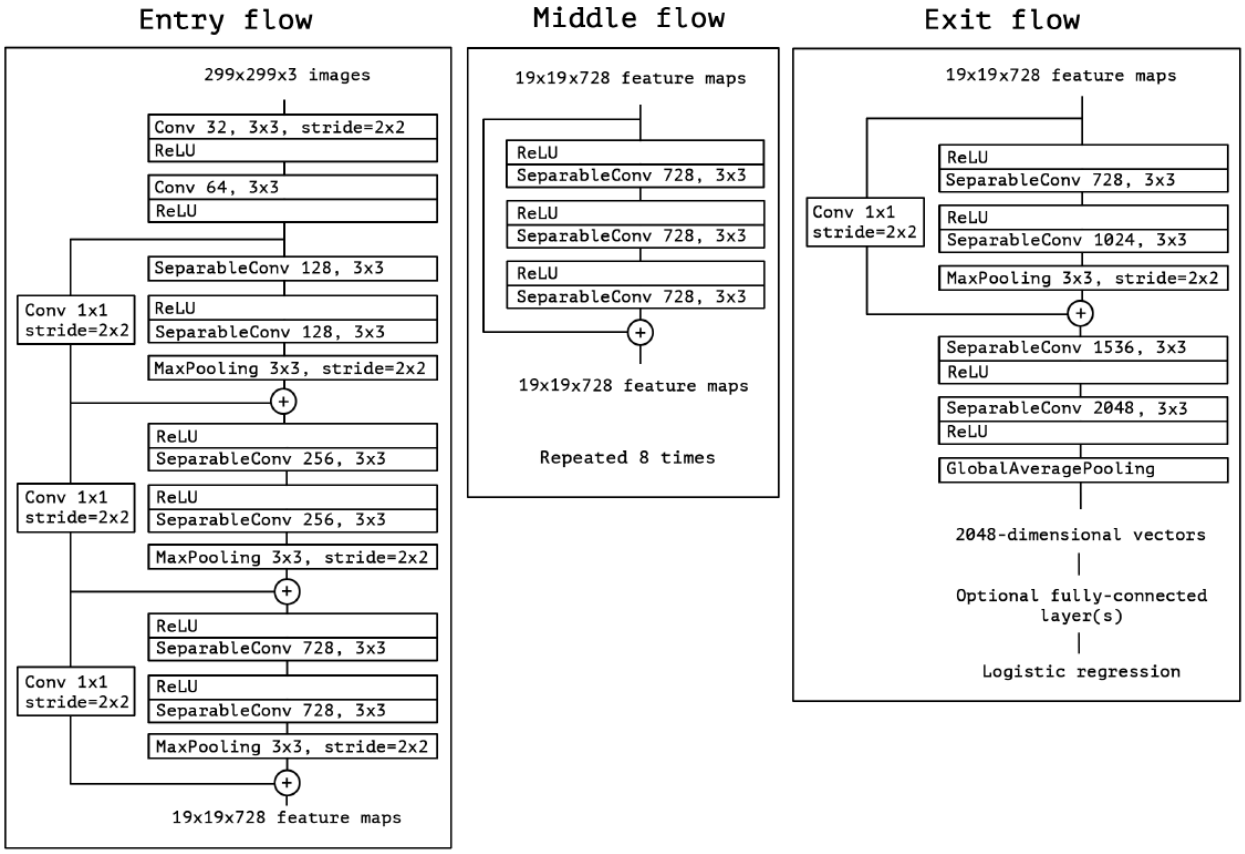
14. Test accuracy :**49.9546**
15. On the synthetic dataset test set, there are 552 fake images and 551 real images. Hence the proportion is **~1**.
16. We get a completely random classifier that performs poorly similar to just tossing a coin and classifying according to the result.
17. These results kind of make sense, the synthetic images are hard to distinguish from the real ones.

Chapter 4

18. The Xception network is pre-trained on more than a million images from the ImageNet dataset. The pretrained network can classify images into 1000 object categories (labels).
19. The basic building blocks of the Xception network are the inception modules.
    A convolution layer attempts to learn filters in a 3D space, with 2 spatial dimensions (width and height) and a channel dimension; thus a single convolution kernel is tasked with simultaneously mapping cross-channel correlations and spatial correlations.
    The idea behind the Inception module is to make this process easier and more efficient by explicitly factoring it into a series of operations that would independently look at cross-channel correlations and at spatial correlations. More precisely, the typical Inception module first looks

at cross channel correlations via a set of 1x1 convolutions, mapping the input data into 3 or 4 separate spaces that are smaller than the original input space, and then maps all correlations in these smaller 3D spaces, via regular 3x3 or 5x5 convolutions. This is illustrated in figure 1. In effect, the fundamental hypothesis behind Inception is that cross-channel correlations and spatial correlations are sufficiently decoupled that it is preferable not to map them jointly.

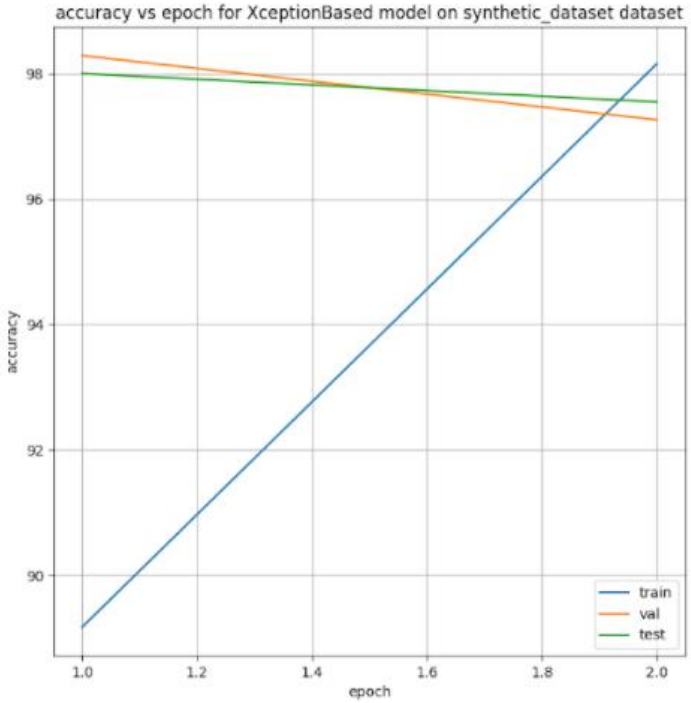
20. It is the same question as Q18.
21. The input feature dimension to the final classification block “fc” is 2048.
    We can see the architecture of the Xception model in the following photo:

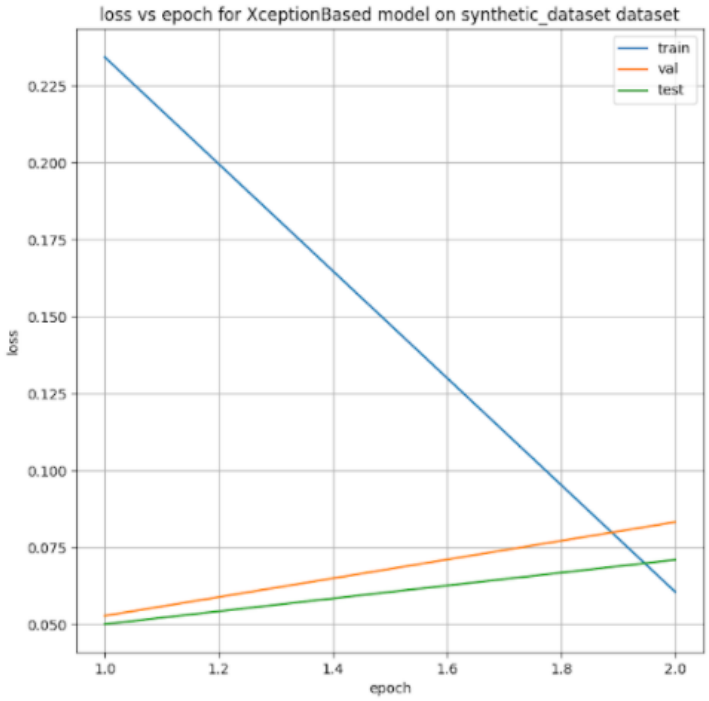


22. According to the given Xception paper, the number of parameters is 22,855,952.
    When running the **get_nof_params** with the Xception model we get **22855952** as well.
23. We implemented the **get_Xception_based_model** that overrides the “fc” block with the MLP architecture.
    By running the **get_nof_params** scripts with this model, we got **23128786** parameters as expected.
24. The new model has 23128786 parameters and the previous one had 22855952, so the number of parameters added is **272834** and that's the number of the MLP model parameters.
25. We trained the Xception-Based network on the Synthetic faces_dataset dataset.

Ori Gilad – 315471375
Daniel Shalev - 207024621

26. <u>We got the following plots:</u>



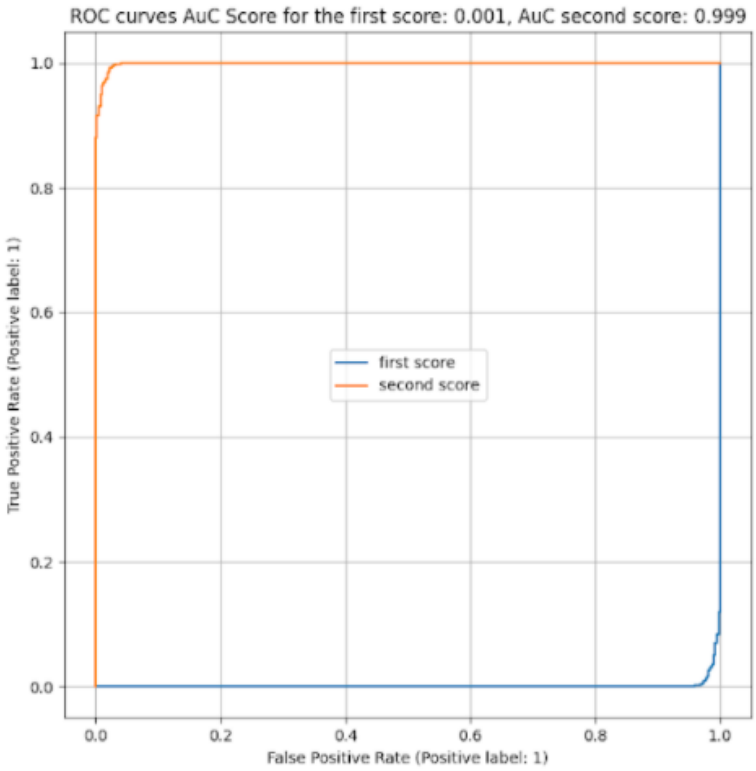


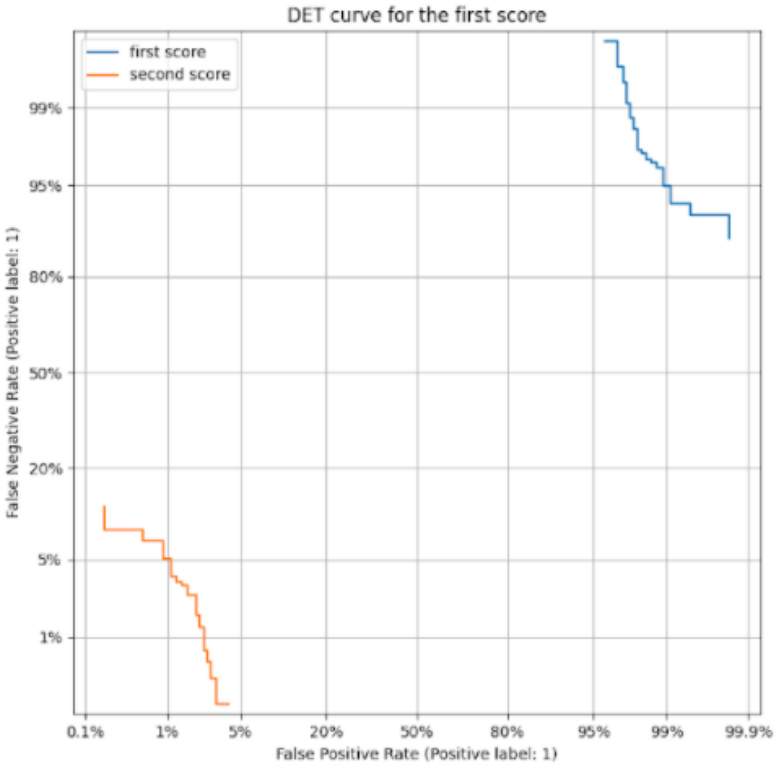
Notice that the performance on the train set seems poor since we average the loss and accuracy along the epoch, and not evaluate them at the end of each epoch.
We can see that we get much better results on the validation and test set.
<u>Interesting point we encountered:</u> We also tried to "freeze" the original Xception weights and only calculate gradients for the new MLP head of the model. This approach didn't give us better result from training the whole network weights. We added a "FREEZE" flag in the script that makes this option available.

27. Test accuracy = **98.005%**

Ori Gilad – 315471375
Daniel Shalev - 207024621

28. We got the following plots:



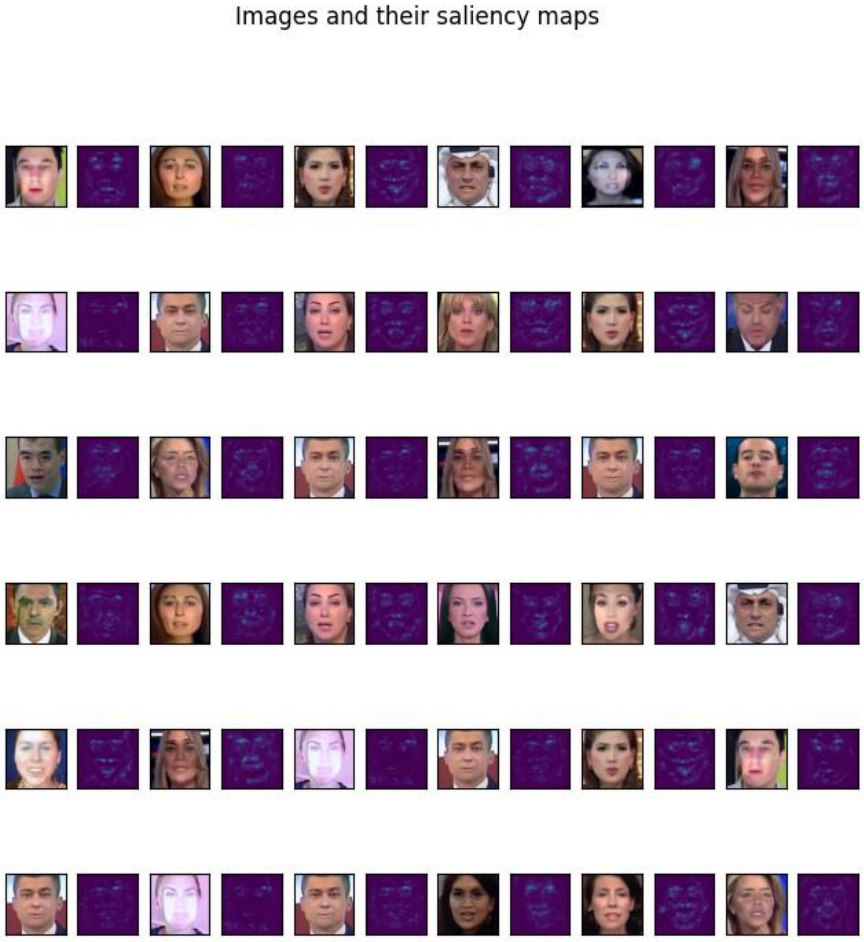


29. Image-Specific Class Saliency Visualization tries to visualize which areas of the image affects the network's classification result. They do so by visualizing the derivative of the output class score of the image with respect to the image itself, while keeping the network's parameters fixed. In that way, we can find how a small change in certain pixels affect the network's output score.

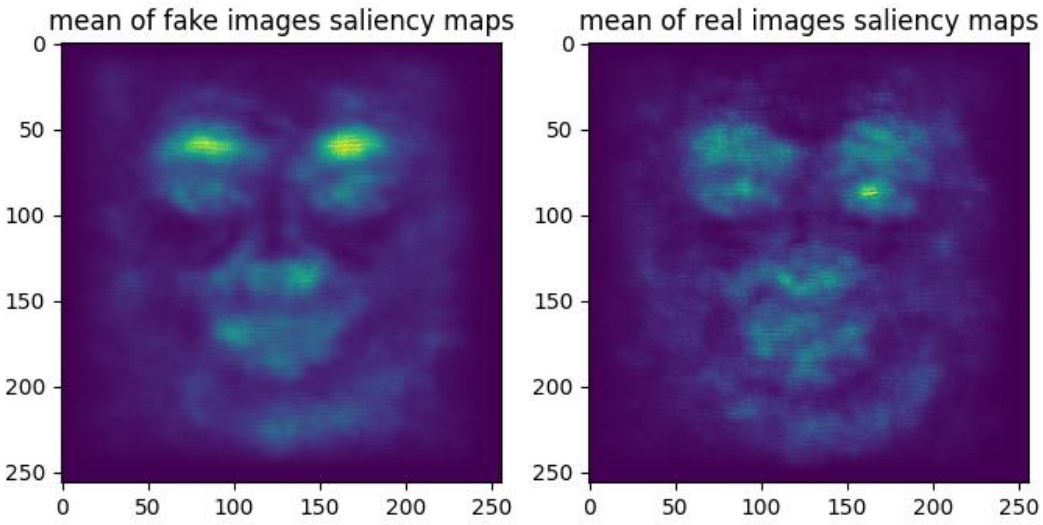
30. Grad-CAM is also a technique to visualize the areas that correspond to a specific class that the network tries to label. It does this by calculating a weighted average of the derivative of the class score with respect to each of the feature maps at the last convolutional layer.
31. We implemented the **compute_gradient_saliency_maps** function.

Ori Gilad – 315471375
Daniel Shalev - 207024621

32. First, we ran the saliency map computation on the SimpleNet network and the fakes_dataset data set. We got the following results:



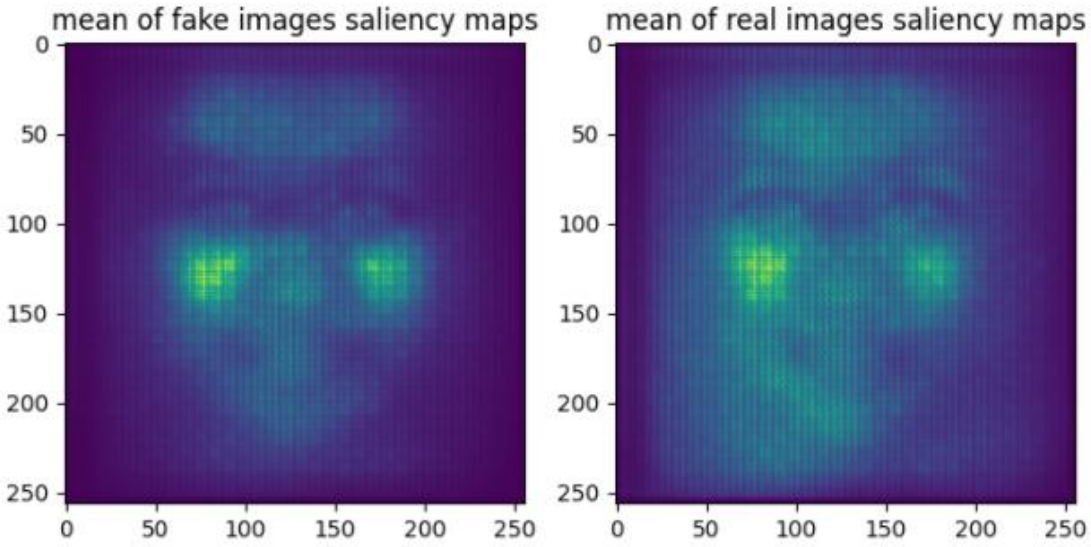
Ori Gilad – 315471375
Daniel Shalev - 207024621



Ori Gilad – 315471375
Daniel Shalev - 207024621

Focusing on the mean saliency maps of each category, we can see that it succeeds to visualizing the importance of each pixel to the classification. We can see that the saliency map emphasizes textures and edges in the faces.
More specifically we can see that the network labels the images as fake mostly just by looking at the person's eyebrows, while the real image classification concentrates on the whole face.
Next, we ran the saliency map computation with the Xception network and the synthetic_dataset dataset and got the following results:

Ori Gilad – 315471375
Daniel Shalev - 207024621

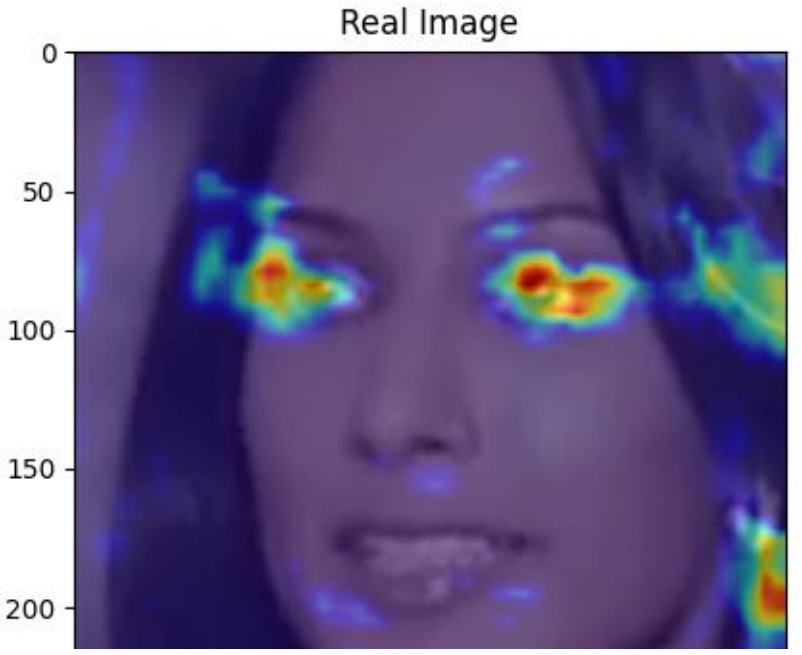


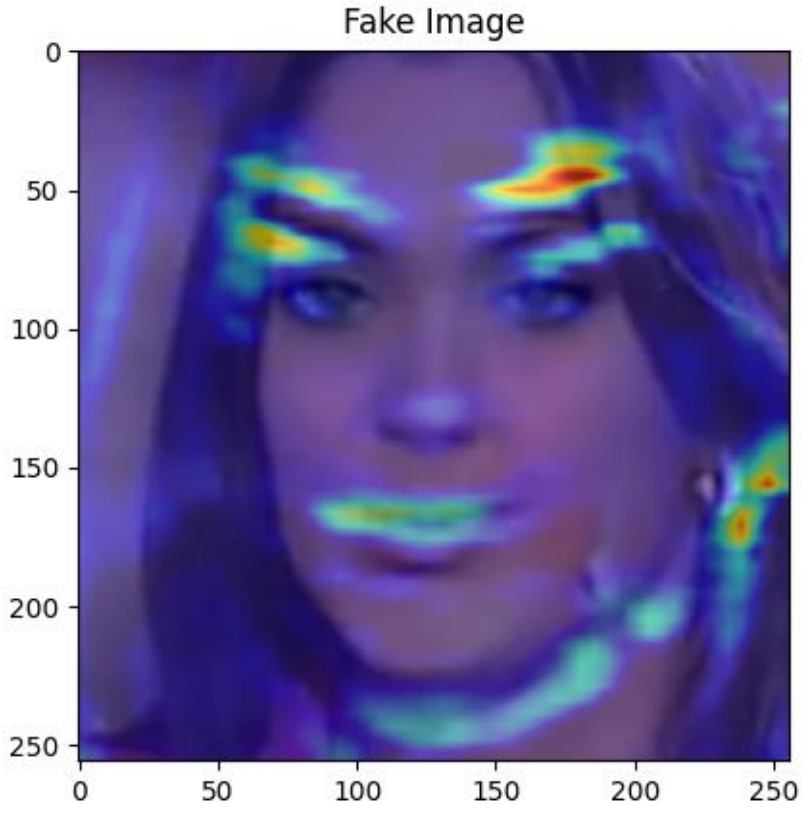
Here we can see that the network concentrates on the same area of the face (the cheeks) for both classes. Moreover, we can see that for real images the network gives importance to wider areas, unlike for fake images where it concentrates solely on the face.

33. We installed the grad-cam package (version 1.3.6) and added it to the new_environment.yml file.
34. We implemented the **get_grad_cam_visualization** function.

Ori Gilad – 315471375
Daniel Shalev - 207024621

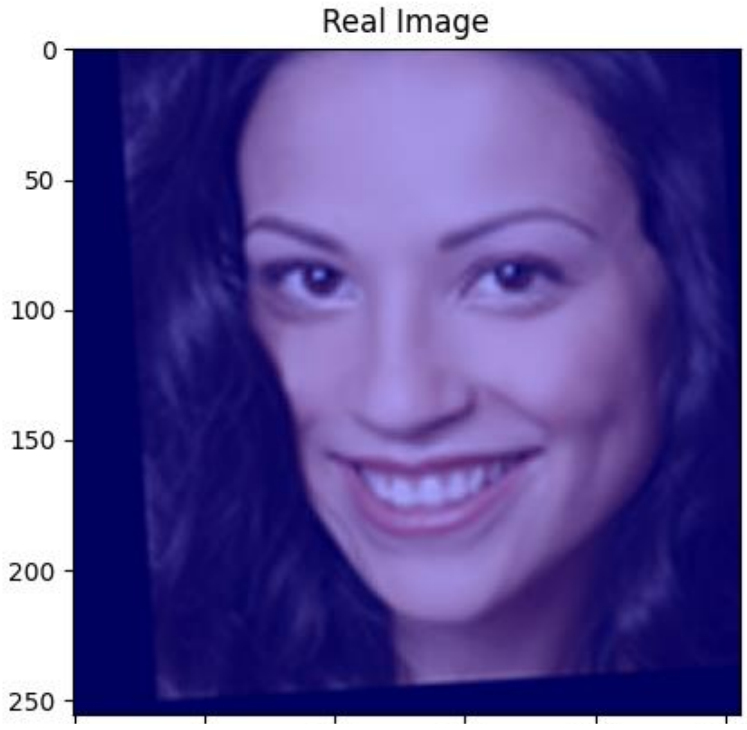
35. We run the **grad_cam_analysis.py** script for the following scenarios:
    a. <u>SimpleNet trained on Deepfakes:</u>

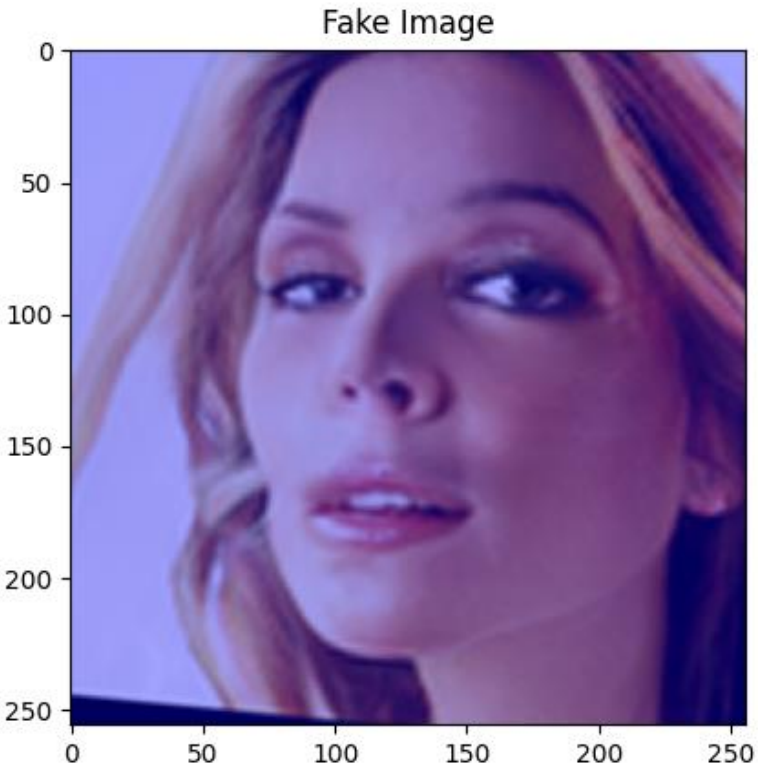




Ori Gilad – 315471375
Daniel Shalev - 207024621

b. SimpleNet trained on Synthetic faces:





Ori Gilad – 315471375
Daniel Shalev - 207024621

c. Xception-Based trained on Synthetic faces: