Facial Manipulation Detection

Computer Vision / 0510-6251

By

Nimrod Curtis 311230924

Tel Aviv University

Winter 2024

Contents:

1. Chapter 1 – Introduction

Given two distinct datasets for analysis:

Deepfake Detection Dataset ("fakes dataset"): This dataset comprises a collection of real images and fake images, where the latter are generated by embedding faces of existing identities into disparate contexts.
Synthetic Image Detection Dataset ("synthetic dataset"): This dataset contains real images as well as synthetic images. The synthetic images are crafted by generative models, which are trained on a pristine set of photographs to produce highly convincing "real" faces.

The real images in both datasets serve as genuine representations of identities, while the fake and synthetic images represent manipulated versions aimed at testing the robustness of detection models.

Objective:
The primary goal of this project is to design and implement detection models that can accurately classify an image as either real or fake.

2. Chapter 2 – Build Faces Dataset

2.1 Intro

- Q1 – Code
- Q2

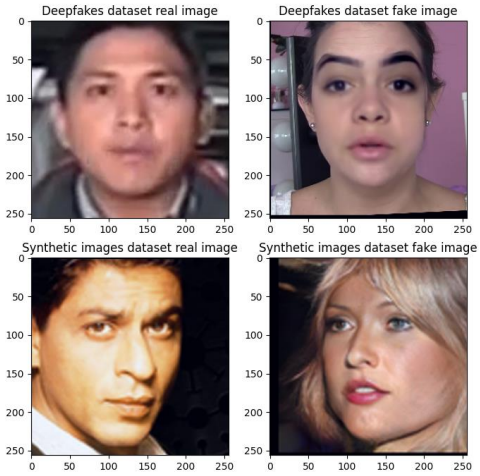

Figure 1 - Datasets samples

The figure illustrates a sampling of images, showcasing two sets of images: one from a Deepfakes dataset and the other from a Synthetic images dataset. Each set consists of a real image and its corresponding manipulated version. The transformations applied to the images can be seen, such as cropping and flipping randomly.

3. Chapter 3 - Write an Abstract Trainer

3.1 Implement an Abstract Trainer

- Q3 – Code
- Q4 - Code

3.2 Train Deepfake Detection Calssifier

- Q5

Training was executed using the train_main.py script with arguments of SimpleNet architecture, 0.001 learning rate, batch size of 32 samples, 5 epochs and Adam optimizer.

3.3 Analyze The Deepfake Detection Classifier

- Q6

It seems that indeed there is a learning process and the results values are make sense at overall.

- Q7

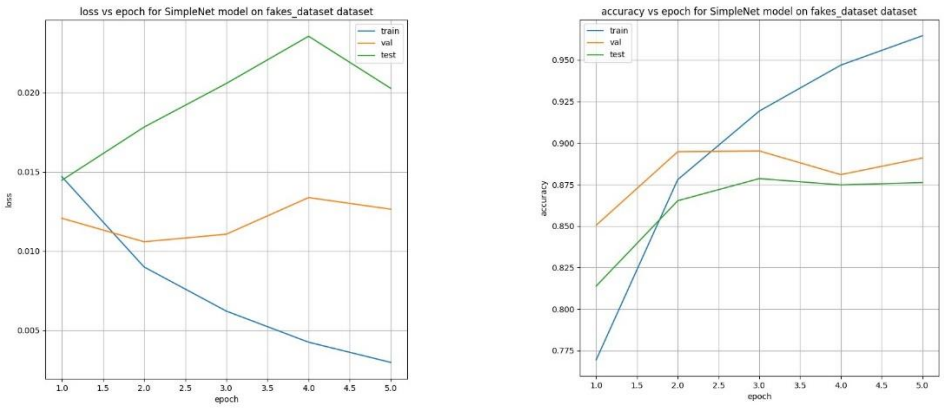

Figure 2 - Train deepfake detection classifier using SimpleNet - losses curves (left) & accuracy curves (right)

The training phase exhibited a steady increase in model accuracy, signifying effective learning from the training dataset. Contrastingly, validation and test accuracies showed lower improvement rates and exhibited fluctuations, which may point towards overfitting tendencies and suboptimal generalization to new data. The loss analysis paralleled these observations with a consistent decrease in training loss, indicating positive learning strides. However, a pronounced spike in validation loss during the third epoch suggested difficulties in generalizing to complex validation set patterns.

Furthermore, test loss displayed a rise in later epochs, reinforcing concerns regarding overfitting and generalization capabilities. Given these observations, extending the training duration and refining the batch size could be beneficial for improved model convergence.

- Q8

Table 1 – Deepfake with SimpleNet, best accuravy of Valid/Test sets, found @epoch 5

Validation	Test
0.89	0.88

- Q9

The proportion of the test set is $\frac{\text{fake}}{\text{real}} = \frac{1}{2}\left(\frac{700}{1400}Samples\right)$

- Q10 + 11

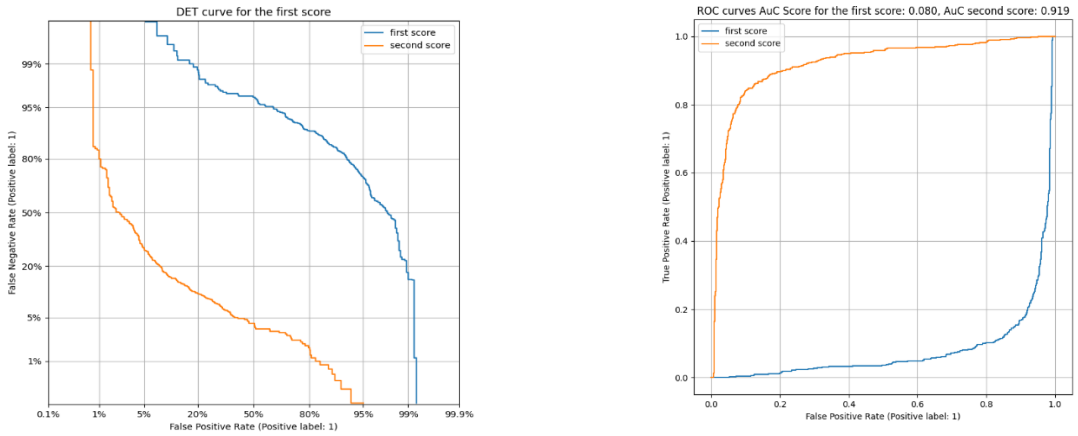

Figure 3 - Train deepfake detection classifier using SimpleNet - DET curve (left) & ROC curve (right)

ROC & AUC:

- Considering that the positive label represents fake images (label 1), which indicates that the classifier's task is to detect fake images, the direction of the curves indeed reflects contrasting performance measures. The curves appear to be inversed due to the differing objectives of each score:

First score (real images): The ROC curve for the real images (first score) stays very close to the x-axis as the false positive rate increases, which indicates a very low true positive rate. Given that the positive class is "fake images," this means the classifier is rarely wrong when it claims an image is real (low false positive rate), but it also rarely correctly identifies fake images (low true positive rate). The AUC is very low (0.080), which suggests that the classifier, when using the first score, is almost ineffective at detecting fakes, as it would perform only slightly better than random guessing.

Second score (fake images): In contrast, the ROC curve for the fake images (second score) rises sharply and then levels off close to the y-axis, showing that the classifier has a high true positive rate for most thresholds. This means that the classifier is quite effective at detecting fake images (high true positive rate) while keeping the false positive rate relatively low. The AUC of 0.919 indicates that the classifier performs very well when using the second score to detect fakes.

DET :

The desirable trait for a classifier tasked with detecting fake images is for the curve to be as close to the bottom left corner as possible, indicating both low false positives (real images incorrectly labeled as fake) and low false negatives (fake images missed by the classifier). The orange curve's initial position near the lower left corner indicates that the classifier starts with a strong performance. However, its upward trend as the false positive rate increases indicates a performance drop-off: the classifier starts to struggle to differentiate between fake and real images as the criterion for classifying an image as fake becomes less strict.

3.4 Train a Synthetic Image Detection Classifier

- Q12

Training was executed using the train_main.py script with arguments of SimpleNet architecture, 0.001 learning rate, batch size of 32 samples, 5 epochs and Adam optimizer.

3.5 Analyze the Synthetic Image Detection Classifier

- Q13

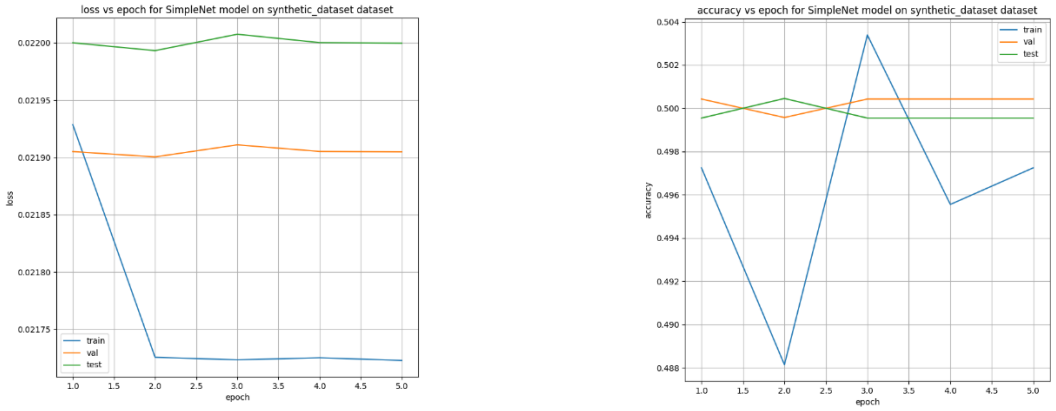

Figure 4 - Train synthetic detection classifier using SimpleNet - losses curves (left) & accuracy curves (right)

The fluctuations in the training accuracy around the value of 0.5 suggests that the learning process is not stable due to significant variance with a mean of 0.5 which is not good enough. Also the flat validation and test accuracies suggest that the model is not improving its performance on unseen data as the training progresses. Ideally, we would expect these to increase over time as the model learns.

The loss curve of the training set shows a sharply drop initially, indicating that the model is rapidly learning from the training data on the first epoch but then it immediately stabilized which indicating that the model stopped learning.
The validation and test losses are flat, suggesting that the model's ability to generalize to unseen data is not improving as well.

- Q14

Table 2 - Synthetic with SimpleNet, best accuravy of Valid/Test sets, found @epoch 4

Validation	Test
0.50	0.49

- Q15

The proportion of the test set is $\frac{\text{synthetic}}{\text{real}} \approx 1\left(\frac{552}{551}Samples\right)$

- Q16

For both validation and test accuracies, the flat lines near the 50% mark suggest that the model is not effectively learning from the data to make better-than-random predictions on unseen data. This result is indeed the expected outcome of a **random classifier** in a binary classification task.

- Q17

Upon examining the examples from the Synthetic Images dataset, we can assert with certainty that the results obtained are not make sense, given that most images in the fake dataset are evidently artificial, even to the human eye.

4. Chapter 4 – Fine Tuning a Pre-Trained Model

4.1 Intro

- Q18

Pre-trained Xception model would typically be trained on the ImageNet dataset.

- Q19

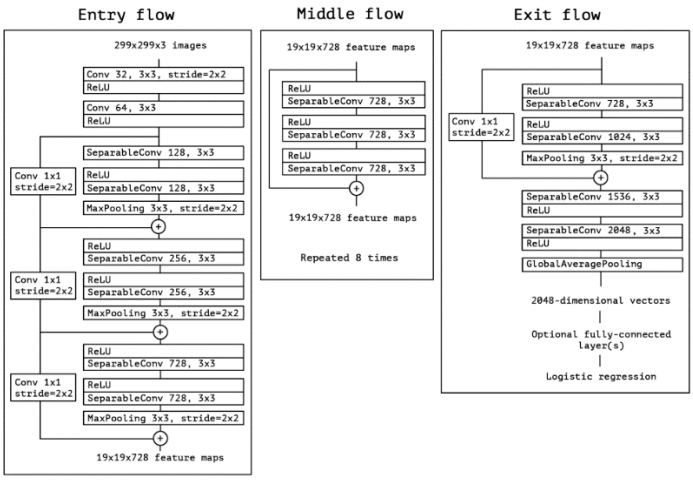

Figure 5 - Xception architecture

The core of Xception architecture is: **Depthwise Separable Convolutional blocks**, consists of two parts:

1. A depthwise convolution, where a single filter is applied to each input channel separately.
2. A pointwise convolution, which is a 1x1 convolution used to combine the outputs of the depthwise convolution.

This separation of filtering and combining allows for reduced computational complexity while maintaining the network's effectiveness.

Based on that and according to the figure above, here are the basic building flow, which utilizing the depthwise seperable conv:

1. **Entry flow**: It begins with standard convolutions for initial feature extraction.
   The entry flow then uses several blocks of depthwise separable convolutions with increasing numbers of filters and strides, typically with max-pooling layers to reduce spatial dimensions.
2. **Middle flow:** This consists of a series of identical blocks that repeat depthwise separable convolutions, refining the network representation learning.
3. **Exit flow:** Similar to the entry flow, the exit flow uses blocks of depthwise separable convolutions. However, it also starts to increase the depth of the network while reducing spatial dimensions.
   It often includes additional layers such as separable convolutions with upsampled filters for high-level feature learning, and it prepares the feature maps for final classification.

- Q20 = Q18
- Q21

Based on the network architecture depicted in the figure and the accompanying code, it is evident that the input dimension for the optional classification block consists of **2048 features.**

- Q22

The number of parameters the Xception network holds is **22,855,952**. Parameter count is reported on ImageNet (1000 classes, no fully-connected layers).

4.2 Attaching new head to the Xception backbone

- Q23 – Code
- Q24

Adding the MLP on top of the original Xception model introduces an additional **272,834 parameters.**

4.3 Train and evaluate a new architecture

- Q25

Training was executed using the train_main.py script with arguments of Xception-Based model architecture, 0.001 learning rate, batch size of 32 samples, 2 epochs and Adam optimizer.

- Q26

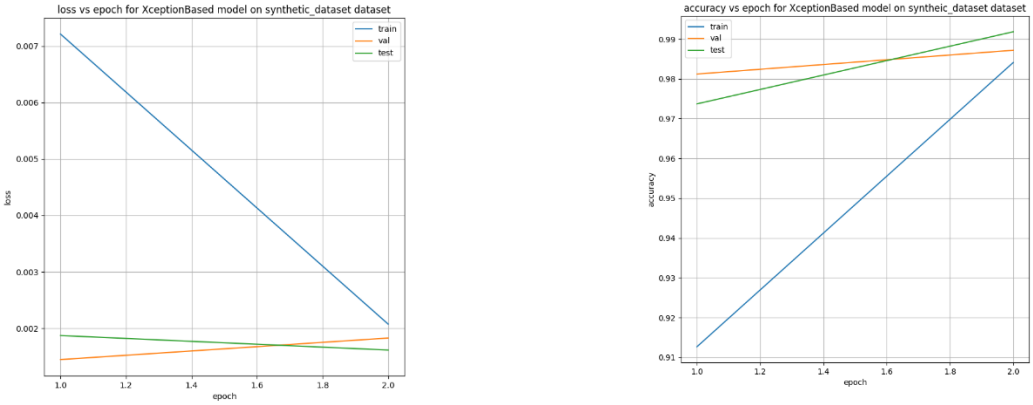

Figure 6 - Train synthetic detection classifier using Xception - losses curves (left) & accuracy curves (right)

In this case, we observe a typical convergence behavior. The training loss decreases sharply, suggesting that the model is learning and improving its predictions over the training dataset. However, the validation and test losses remain relatively flat and close to each other after an initial decrease, indicating that the model is not improving its performance on unseen data after the first epoch. This could suggest that the model has quickly learned to generalize to new data or, conversely, that it has reached a performance plateau where additional training does not yield significant improvements on the validation and test sets.

In the second figure, depicting accuracy versus the number of epochs, the training accuracy increases dramatically to near-perfect levels, while the validation and test accuracies are relatively high but stable. Since the validation and test accuracies do not decrease, it is possible that the model has reached its capability for the given task within the first epoch, and further training simply reinforces the patterns it has already learned without overfitting.

- Q27

Table 3 - Synthetic with Xception, best accuravy of Valid/Test sets, found @epoch 2

Validation	Test
0.99	0.99

- Q28

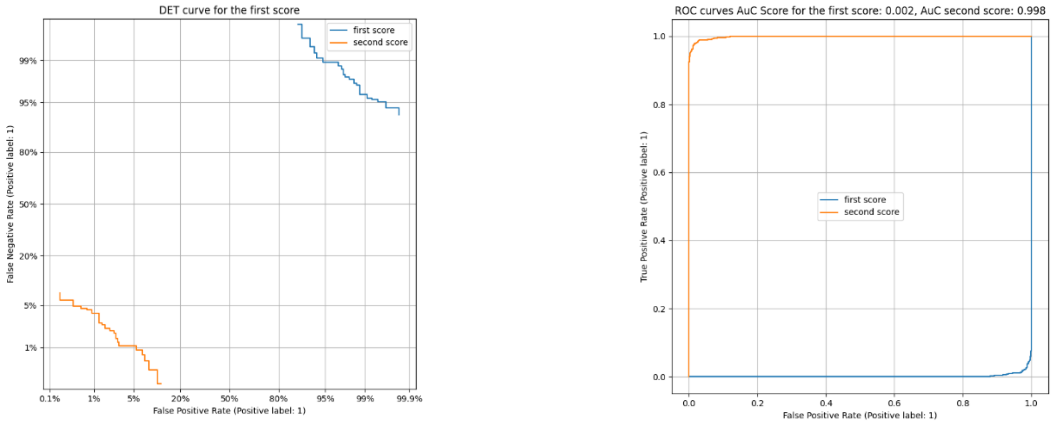

Figure 7 - Train synthetic detection classifier using Xception - DET curve (left) & ROC curve (right)

ROC & AUC:

The ROC curve for the fake images score (second score) stays extremely high along the y-axis, which indicates an excellent true positive rate across almost all thresholds. This high true positive rate is maintained even as the false positive rate increases from 0 to 1. The area under the curve (AuC) is 0.998, which is near perfect, signifying that the classifier has an outstanding ability to distinguish fake images from real ones.

DET:

Analyzing the orange curve since the fake is the positive label, the DET curve starts with a very low false negative rate when the false positive rate is also low, indicating that the classifier is highly accurate in identifying fake images as fakes when it is very strict. As the false positive rate increases, the false negative rate remains low for quite a range, suggesting that even if we lower the strictness a bit, the classifier still performs well at identifying fake images. However, as we continue to increase the threshold (becoming less strict), the false negative rate gradually starts to increase, indicating that more fake images are being misclassified as real.

5. Saliency Maps and Grad-CAM analysis

5.1 Intro

- Q29

Image-Specific Class Saliency Visualization in simple words refers to highlighting the areas of the image that are most important for the classification decision of a CNN network, effectively visualizing the "reasoning" behind the CNN decision for a specific image and class.

The process involves generating a saliency map for a given image and class, where the map ranks pixels based on their influence on the class score. This is done by computing the derivative of the class score

with respect to the input image, which indicates how changes in pixel values would affect the class score.\\

The formula for computing the saliency map:

$$w = \frac{\partial S_c}{\partial I}|_{I_0}$$

where S_c is the class score for class c, I is the input image, and I_0 represents the specific image for which the saliency is being calculated.

- Q30

Grad-CAM is a visualization technique that explains CNN model decisions by highlighting image regions important for predictions, offering a clearer, region-based view compared to the pixel-level focus of saliency maps. While saliency maps can be noisier with finer details, Grad-CAM uses gradient information from the last convolutional layer to identify significant features for specific classes, making model decisions more interpretable.

The formula used in Grad-CAM to obtain the localization map is as follows:

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right)$$

Here, $L^c_{\text{Grad-CAM}}$ is the localization map for class c, A^k represents the feature maps of the last convolutional layer, and α_k^c are the weights capturing the importance of the feature map k for the target class c. The ReLU function is applied to the linear combination of feature maps to only consider the features that have a positive influence on the class of interest, essentially highlighting the areas crucial for identifying the class.

5.2 Saliency Maps

- Q31 – Code
- Q32

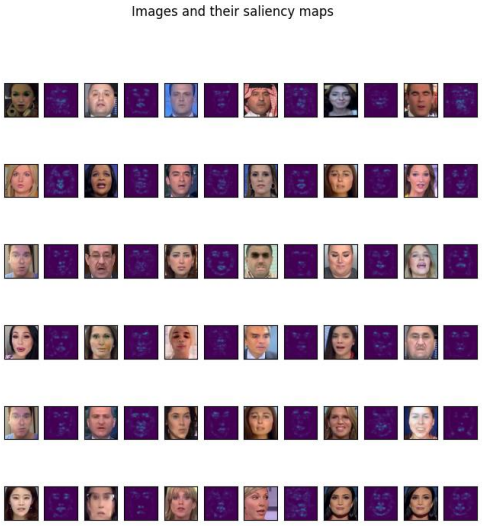

Figure 8 – Fakes dataset, SimpleNet - Saliency maps

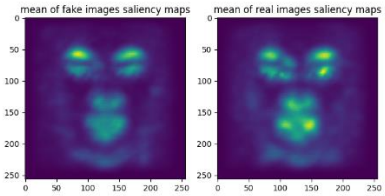

Figure 9 - Fakes dataset, SimpleNet - Mean of saliency of real/fake samples

The mean saliency maps represent the average areas of focus across multiple images for each class (real and fake). The mean map for both real and fake images show similar general structure of bright spots on the eyes, nose and mouth, suggesting that these areas are commonly scrutinized by the network when identifying the images.

Upon closer examination of the finer details, we observe variances in the regions that exhibit the highest saliency averages (indicated by the bright yellow areas). In the fake images, the most prominent region appears to be the right eyebrow, whereas in the real images, the mouth and left eye seem to be the areas with the most influence.

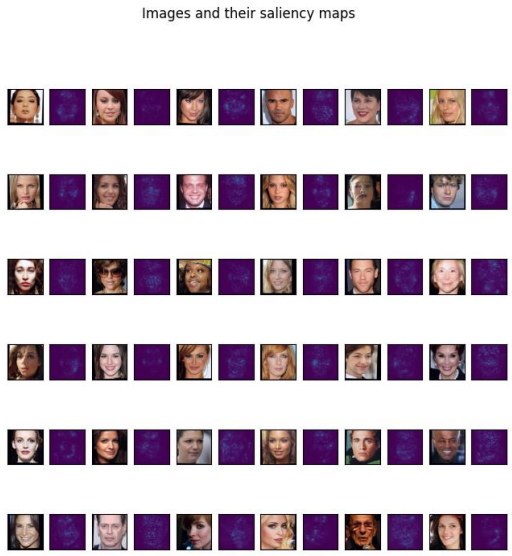

Figure 10 - Synthetic dataset, Xception - Saliency maps

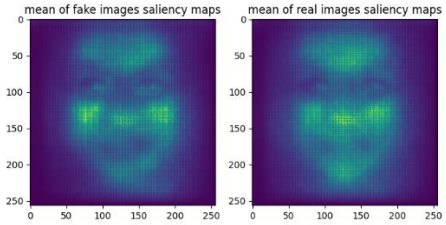

Figure 11 - Synthetic dataset, Xception - Mean of saliency of real/fake samples

In this instance, as with the previous example, the overall structure of the mean saliency maps is quite similar between the synthetic and real classes. However, the real images likely exhibit more diverse and authentic facial traits. For the synthetic images, the saliency map seems to focus on the area of the right cheek, which might reflect characteristics that the network identifies as indicative of synthetically generated faces, potentially resulting from the artificial creation process.

5.3 Grad-CAM

- Q33 – Code
- Q34 – Code
- Q35

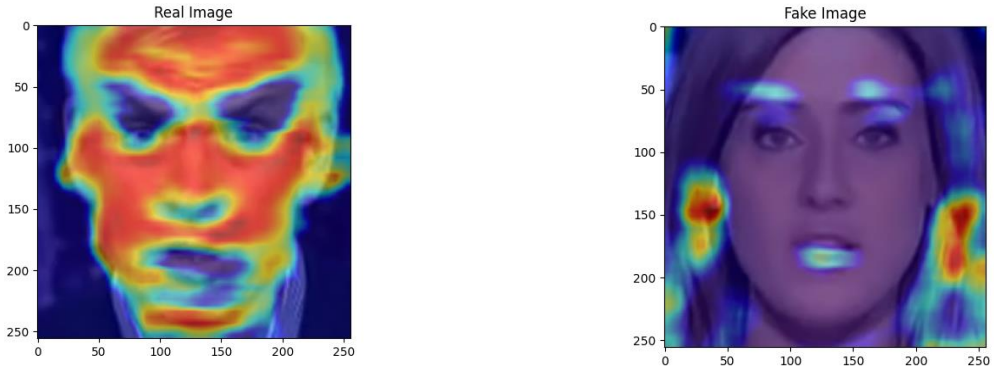

Figure 12 - Fake dataset, SimpleNet - Grad-CAM - real image (left) & fake image (right)

The Grad-CAM for the deepfake image shows concentrated areas of high activation (red color corresponds to high values), particularly below the ears on the hair area. In contrast, the Grad-CAM for the real image displays a more evenly distributed activation across the face, with a particularly strong focus on the central features, such as the forehead, nose and cheeks.

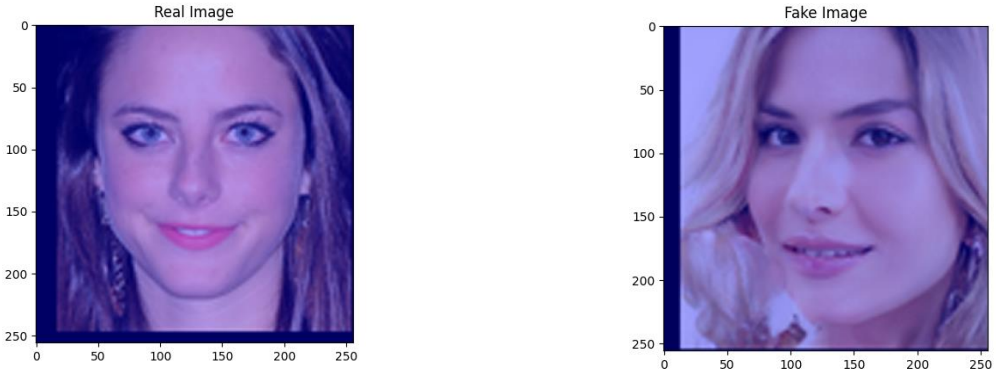

Figure 13 - Synthetic dataset, SimpleNet - Grad-CAM - real image (left) & fake image (right)

Aligned with the section describing SimpleNet's training on synthetic images, where we observed that the trained model's image labeling probabilities resemble random guesses, it appears that there are no specific regions in the images that significantly influence the image classification.

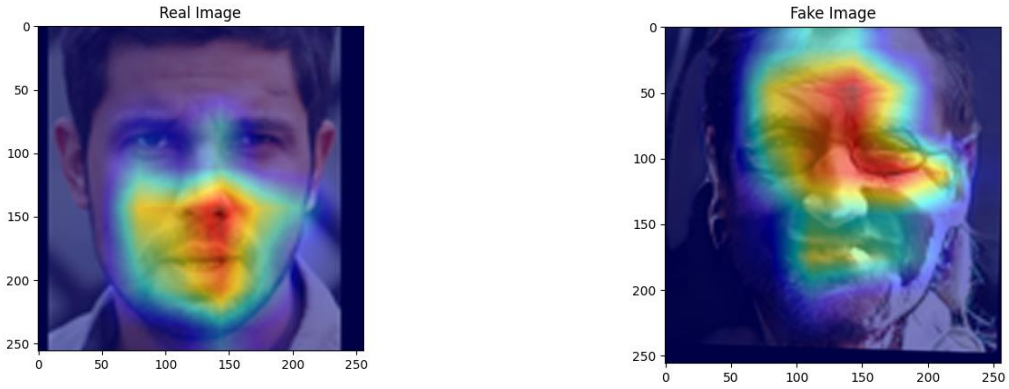

Figure 14 - Synthetic dataset, Xception - Grad-CAM - real image (left) & fake image (right)

In this case, training the Xception model on the dataset has yielded a clear identification of influential regions within each image, as evidenced by the Grad-Cam values, which is a notable improvement over the previous example.

The differences with relate to the networks architecture could mean that in the SimpleNet case the network's architecture is not complex enough to capture the defining features of the different classes. This may suggest that the network lacks the depth or the complexity needed to learn discriminative features that are vital for accurate classification.

On the other hand, the Xception based model case indicates that the network's architecture has sufficient depth and complexity to learn and focus on the most important features for each class.