DEEPFAKE RECOGNITION ASSIGNMENT

SECOND SEMESTER 2025

Carolin Gerdes
EPS UAM
carolin.gerdes@estudiante.uam.es

Massimo D'Allesandro
EPS UAM
massimo.dalessandro@estudiante.uam.es

Jade Martin-Lise
EPS UAM
jade.martin-lise@estudiante.uam.es

February 9, 2026

Code Availability

All the code used for this project is supplied in the attached zip folder. However, some of the notebooks were run on kaggle. These notebooks require the upload of the data to kaggle. To run the code please make sure to upload the data so that it matches the input paths displayed in the code. Additionally, the code accessing the ChatGPT AI is no longer runnable because the API Key was removed for security reasons.

Task 1 – Intra-database analysis

The goal of this task is to develop and evaluate DeepFake detection systems over the same database (intra-database analysis). In this Task 1, you should use only the UADFV database included in the folder named "Task_1". This database is divided into "development" and "evaluation" datasets.

Within this task we have developed and implemented two main approaches for Intra-Database DeepFake detection. One method (Method 1) is based only on facial features extracted from the images, while the second method (Method 2) implements an image based recognition utilizing several CNNs.

1.a) Provide all details of your proposed DeepFake detection system

Method 1

This method is based the observation, that all deep fake images show the face of Nicolas cage, while the real images show different faces. In addition all real images of the testing dataset represent pictures of Donald trump. Hence, for this specific dataset, it is not necessary to detect whether or not an image is fake, but only whether or not it displays Nicolas Cage. To do so, we extracted several measures describing the proportions of Nicolas Cage's face, with the underlying assumption that his features are relatively unique and easily distinguishable from those of other people. In total 9 such measures were extracted, 7 of which were kept. These include:

- 1: Nose length to face length ratio (kept)
- 2: Inner eye distance in relation to outer eye distance (discarded)
- 3: Nose width to face width ratio (kept)
- 4: "Archedness" of left brow (brow height to brow width ratio) (kept)
- 5: "Archedness" of right brow (brow height to brow width ratio) (discarded)

- 6: Height of left brow (Distance between Browarch and lower eyelid in relation to eye size) (kept)
- 7: Height of right brow (kept)
- 8: Face Width Ratio (Distance between outer eyes in relation to total face width) (kept)
- 9: "Undereye bag measure" (Percentage of pixels identified as eyebags in left lower eye region) (kept)

A detailed explanation of how these features are calculated can be extracted from the appended code. Features 1-7 are based on the specific proportions of Nicolas Cage's face (or at least of the image that was used for the deepfakes) While measures 8 and 9 are based on more general observations. Feature 8 is based on the observation, that in some fake images, the inserted face seems too small for the face it is inserted to (Figure S3). Measure 9 is chosen with the test set specifically in mind. Nicolas Cage has a significantly younger appearance than Donald Trump (used in the validation set) and than many other people in the training set. This effect is further enhanced by the seemingly lower resolution of the inserted face of Nicolas Cage, giving the deepfake faces a more youthful, airbrushed look (Figure S3). This property is aimed to be extracted by measuring the undereye bags of the individuals displayed in the images.

As described above, this method relies on facial landmarks extracted from the images. To maximize the comparability of the landmarks, all images were cropped to the displayed faces and rotated, so that all faces were horizontally aligned. The effect of these transformations can be seen in Supplementary Figure S1 Faces were detected using dlibs face detection model (mmod_human_face_detector) [**?**]. The facial landmarks are extracted using dlibs "shape_predictor_68_face_landmarks" (Figure **??** and are used to both rotate the faces and extract the final facial features.

Each image is thus represented by a feature vector of length 7.

Method 2

This method relies on traditional deep learning approaches. The architecture chosen is a CNN model. The details of this system are as follows :

- Detect the face in the image using MTCNN from the facenet_pytorch library and crop the image to contain the face only.
- Load the dataset. Transformation of the Data, resizing of the data to get appropriate input size for the CNN.
- Define the CNN Model. The model consists of two convolutional layers (with ReLU activation and max pooling), two fully connected layers, and a sigmoid activation in the output layer for binary classification.

1.b) Details of the development/training procedure using the development dataset

-provide details, show results (ROC, AUC)

Method 1.1

The extracted feature vectors were normalized, and a SVM was trained on them. An 'rbf' kernel was used. This was chosen, as all but one feature showed an increased performance by using this kernel (More on this later). To fine-tune the model, one SVM was trained for each feature, using only said feature for the deepfake detection. Features with a performance of < 0.6 were discarded.

As the features are specifically selected to identify Nicolas Cage, and he is only present in the deep-fake images, a high performance is expected. However, images, in which no face is detected, or images showing faces from different angles impact the facial landmarks and might thus lead to misclassification. The ROC-curve and matching AUC of 0.97 confirm this hypothesis (Figure 1).

Method 1.2

During the fine-tuning of Model 1, it was discovered, that Feature 9 (Undereye measure) exhibited an AUC when chosen as individual feature and trained using a linear kernel SVM. Thus it was kept for further analyses to demonstrate the power of one individual feature for deepfake detection. While it does not perform very well on the training data, we expect a higher performance on the training data. This is because the feature was specifically selected to distinguish Nicolas Cage from Donald Trump, as the "eyebag" effect was, upon visual inspection, most noticeable in the validation images. The rather peculiar shape of the ROC curve in Figure 2 most likely reflects the score assignment process. Images, in which no "eyebags" could be detected were given a score of 0. As the model associates higher eyebag scores with real images, real images without detectable eyebags (hence assigned with score 0) are thus largely misclassified, leading to an inverted performance for low scores.

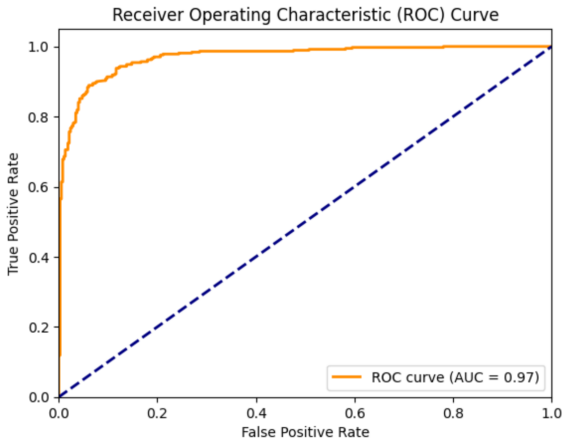

Figure 1: Performance of SVC (Method 1) on training data. The model shows a very high predictive power. 7 features describing facial proportions of Nicolas Cages face were used for training.

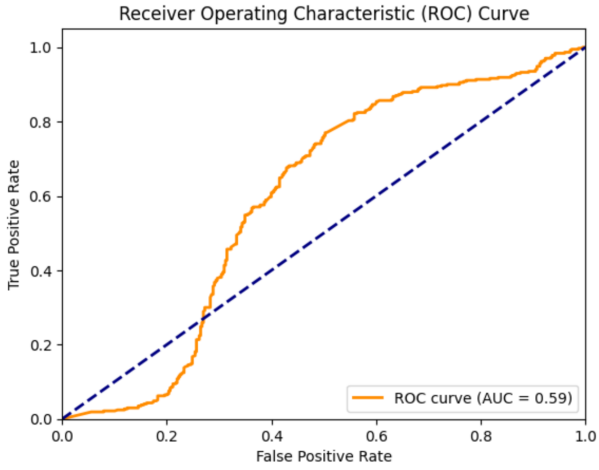

Figure 2: **ROC curve of performance of SVM trained on eyebag feature on training data.** The model shows an AUC of 0.60, indicating very limited predictive power.

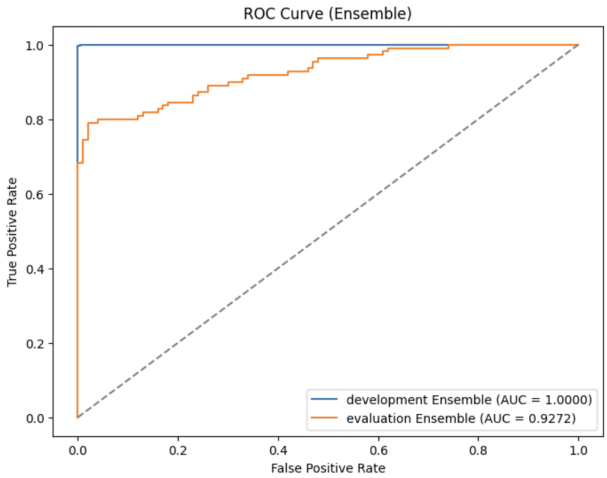

Figure 3: **ROC curve of performance of CNN** In blue, the performance on the Development dataset and in orange on the Evaluation Dataset.

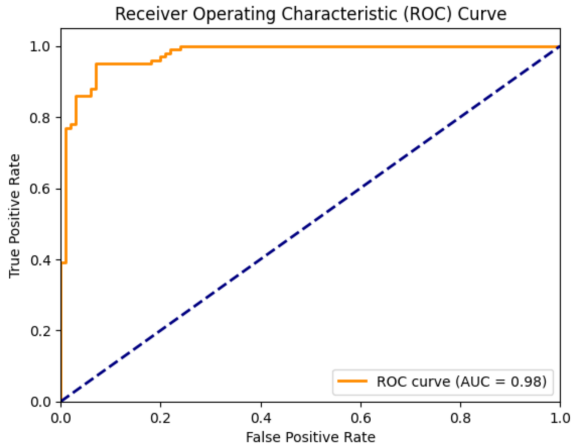

Figure 4: **Performance of SVM on Validation data (Method 1.1)**. An AUC of 0.98 was achieved. 7 Features were considered for model training

Method 2

For this method, three separate CNN models are trained independently. After training, we combine them in an ensemble for improved performance. Each CNN model is trained using Adam optimizer (lr = 0.001) and Binary Cross-Entropy Loss (BCE Loss). The ROC of the model on the development dataset is given in figure 3. On the training data, the model obtains an AUC of 1 in this specific instance. Overall, in other instances, the model obtained an AUC of around 0.98 on the development dataset.

1.c) Final evaluation of your proposed DeepFake detection system

Method 1.1

Using the SVM on the validation dataset returned an AUC of 0.98. The ROC and the AUC of this model are visualized in Figure 4. As the features were specifically selected and extracted in a way to identify the deepfakes based on facial features of Nicolas Cage, and both the training and the validation set show image of Nicolas Cage, a similar performance on the validation set and the testing set was expected. This is confirmed by the AUC.

Method 1.2

Using an SVM with linear kernel and only the extracted eyebag feature, an AUc of 0.90 can be achieved. The corresponding ROC curve is plotted in Figure 5.

The results are quite surprising, since the model seems to perform quite well, despite performing poorly on the training data. Despite the eyebag feature not classifying the training data correctly, it seemed to have correctly identified the trend, that the deepfake images show more of an "airbrushed" look. As this difference is much more noticeable, a better result on the validation data was expected. The eyebag detection relies on enhancing edges and then binary classifying each pixel as eyebag or not. To do so a threshold has to be selected. This threshold was selected quite arbitrarily, failing to recognize some visual eyebags in some images (As can be seen in Figure 5). While in theory this feature should separate the images in this dataset almost perfectly, a significantly worse performance was expected due to the rudimentary extraction of the feature and the ambiguities in the training data.

Method 2

Figure 3 shows the ROC curve for the evaluation dataset. The model performs well with an AUC of 0.92. However compared to the Machine Learning approach of Method 1, the Deep Learning approach is less performant. For this dataset, studying the data and extracting the features manually proves to be more efficient than letting the model extract the features on its own.

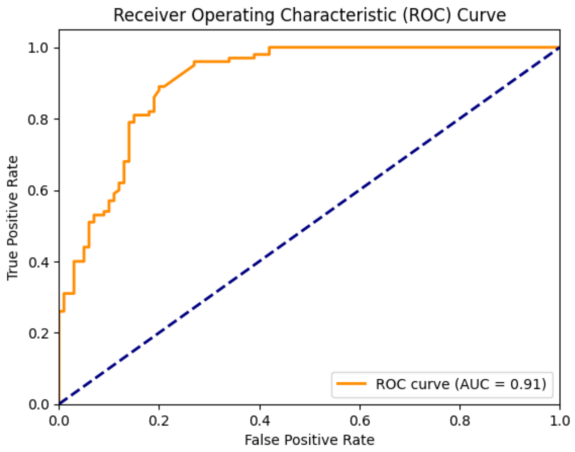

Figure 5: Performance of SVM on Validation data (Method 1.2). An AUC of 0.91 was achieved. Only the eyebag measure was considered for training.

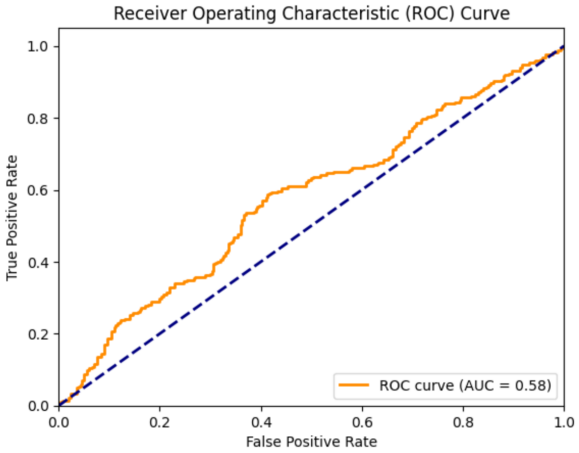

Figure 6: Performance of Model 1 on the New Database.

Task 2 - Inter-database analysis

The goal of this task is to evaluate the DeepFake detection system developed in Task 1 with a new database (not seen during the development/training of the detector). In this Task 2, you should use only the Celeb-DF database included in the folder named "Task_2_3". You only need to evaluate your fake detector developed in Task 1 over the evaluation dataset of Celeb-DF, not training again with them.

Results achieved on evaluation dataset

0.0.1 Method 1.1

The model that performed the best in Task 1 is the one classifying the images based on features specific to Nicolas Cage. Thus a very bad generalization of the model is expected. This is reflected in the AUC of the predictions on the new dataset (0.xx, Fig 6). The slight predictive power might be due to the inclusion of the eyebag feature, which shows some predictive performance on the validation (Figure 7)

Method 1.2

Given that the eyebag feature is not specifically tuned to detect Nicolas Cages face, but rather describes that the inserted images are of lower quality and thus appear airbrushed, and with softer facial features. Thus we expect a slightly higher

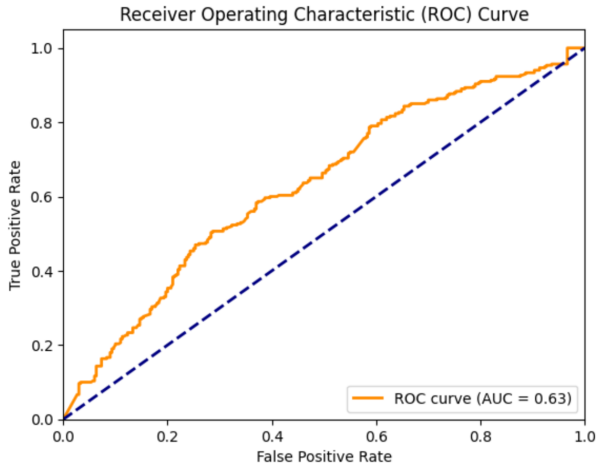

Figure 7: Performance of Model 1.2 on the New Database

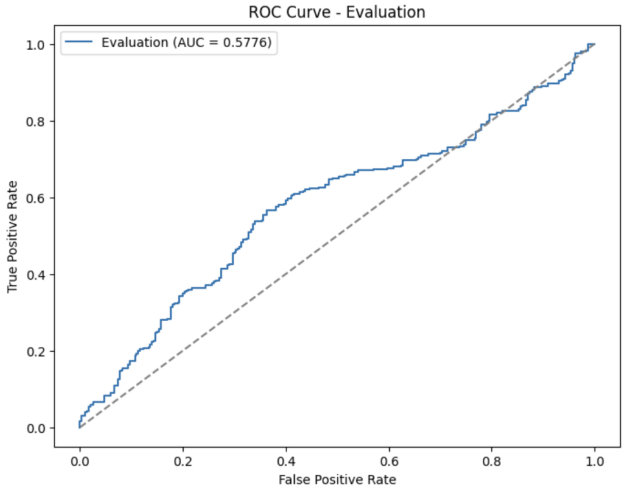

Figure 8: Performance of Method 2 on the New Database

predictive power for this model. However, as seen during the training the model also does not perform well with images of people with less pronounced eye bags. The model performs slightly better, displaying an AUC of 0.63 (Figure 7. The decrease in the performance is most likely due to the increased heterogenicity of the dataset. In contrast to the test set of task 1, not all images are of Donald Trump, but show people of various people and of both genders. However, the general trend of deepfake images appearing slightly airbrushed remains.

Method 2

As expected, the performance of the model on the new database is bad. The ROC curve is shown in figure 8. The AUC is of 0.58 which is similar to the AUC obtained with models from Method 1. The model has only been trained on the UADFV Database and as result seems to present a Dataset Bias. The CNN architecture of the model is simple so its ability to extract robust, high-level features (which are needed in order to generalise across datasets) is limited. In other words, the features that the model has learned when training on the UADFV Dataset are not effective on the Celeb-DF database.

Figure 9: deepfake-and-real-images dataset

Task 3 – Inter-database proposal

The goal of this task is to improve the DeepFake detection system originally developed in Task 1 in order to achieve better inter-database results.

3.a) Improvements in comparison with Task 1

Since there were no restrictions for this task, we decided to try to utilize existing frameworks for Deepfake Detection. This was chosen because the best performing approaches in the 2020 Facebook deepfake detection model were based on deep learning. Among the most performing models emerging from the competition are EfficientNet, Xception, and ResNet, which demonstrated remarkable generalization capabilities on complex and diverse datasets.

Selected Architecture

The ResNet50 model was chosen for feature extraction. ResNet50 was preferred for several reasons: firstly, it is a well-established model, pre-trained on a large dataset (ImageNet), making it particularly suitable for transfer learning. Secondly, its relatively lightweight architecture compared to other models like EfficientNet makes it suitable for contexts where storage space and computational power are limited.

Method 1

As a first attempt, the deepfake-and-real-images dataset [Figure 9], available on Kaggle, was used. ResNet-50 was fine-tuned to this dataset. Details on the training process can be found in section 3b

Method 2

The images from the provided evaluation dataset show snapshots from videos. Thus the facial expressions and poses are mostly different to those from the Kaggle deepfake dataset. To increase the similarity and thus the generalization between training and testing dataset, another dataset was chosen, It was generated from taking snapshots of deepfake videos and their respective original videos. Details of this extraction are listed in 3c).

Method 3

In this approach we tried to incorporate video snapshots from the DeeperForensics 1.0 dataset to increase the training data. The procedure was analogous to Method 2 [Figure 10].

Method 4

Based on the approaches chosen for task 1, another attempted approach was to train four separate models, each specialized in a specific facial region: nose, mouth, and two for the eyes (left and right). The idea was that, by focusing on individual parts of the face, the models could better capture the artifacts or imperfections typical of deepfakes, which often manifest differently depending on the facial area. All models were based on ResNet50, pre-trained on ImageNet, and were connected via a fully connected layer to combine the extracted features and produce a final binary classification.

Figure 10: Dataset of video snapshots

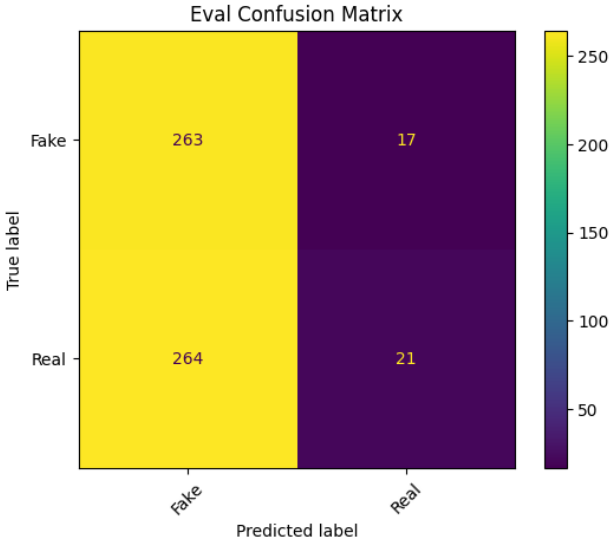

Figure 11: Confusion matrix method 1

0.0.2 Method 5

Lastly, we also tried to use ChatGPT for DeepFake detection, and used ChatGPTs API to do so.

3.b) Description of results

Method 1

During the retraining of the model's head, the training stabilized at the seventh epoch, reaching a test accuracy of 71%, with no significant improvements up to the tenth epoch. After fine-tuning, the accuracy on the test set increased to 83%. However, the model showed disappointing performance on the evaluation set, with an accuracy of 50% and a strong tendency to classify almost all images as fake.

Method 2

During training, the model showed good performance, with high accuracy on both the training set and the test set. However, the results on the evaluation set were disappointing: the model classified almost all elements as "real". This behavior is likely due to the lower quality of the deepfakes in the training dataset compared to those in the evaluation set. The deepfakes used for training might have been less realistic, leading the model to recognize only coarse artifacts and classify as "real" everything that did not exhibit such characteristics. Additionally, many original videos were filmed by the same actors, limiting the model's ability to generalize across different faces and contexts.

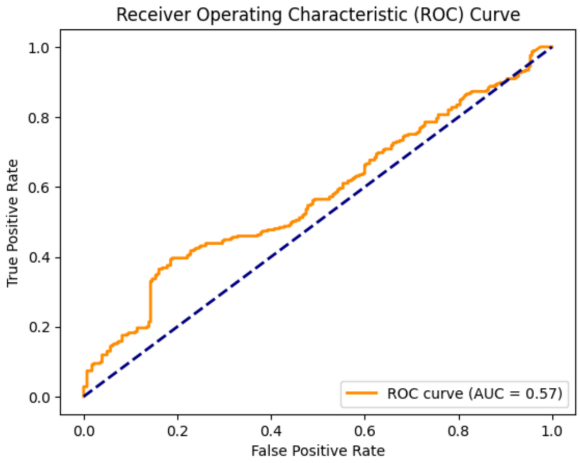

Figure 12: Results of DL Method 3 on the evaluation dataset.

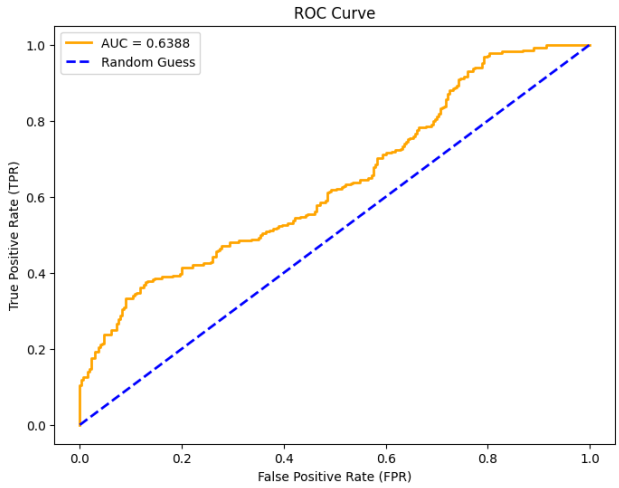

Figure 13: Results of DL Method 4 on the evaluation dataset.

Method 3

The results were the most promising so far: the model achieved an AUC (Area Under the Curve) of 0.57, indicating an improvement over previous attempts. Additionally, no imbalance towards either class ("real" or "fake") was observed, unlike previous cases where the model tended to favor one class over the other. The introduction of the DeeperForensics 1.0 dataset contributed to improving the model's performance thanks to the greater variety and quality of the data. The ROC and AUC are visualized in Figure 12

Method 4

The outcomes obtained from this approach are the most encouraging to date, with an AUC of 0.64. We believe that with a larger dataset, which was not utilized due to computational constraints, the results could significantly improve. This approach demonstrates that by manually directing the model's attention to critical features, its performance can be notably enhanced. This suggests a potential pathway for refining model accuracy through focused feature analysis, paving the way for more sophisticated and effective facial recognition systems in the future.

Method 5

The results of ChatGPT were very bad with an AUC of 0.42 (Graph omitted due to low performance). ChatGPT almost always either returned a score of 0.2 or 0.9, and the results seemed really random. While engineering the prompt further, ChatGPT even refused to provide information about the fakeness of the images, claimed it was not trained to do so. Because of this (and also because of the cost of the API) the approach was not further explored.

3.c) Additional information

The Training procedure used is as follows:

Method 1

Retraining the Model's Head The "head" of the model, i.e., the final classification layers, was replaced with a new structure suitable for binary classification. Instead of softmax, a sigmoid activation function was used, which returns a value between 0 and 1, indicating the probability that an image is a deepfake. As the loss function, Binary Cross-Entropy (BCE) was employed, ideal for binary classification problems. During this phase, a learning rate (LR) of 0.001 was used, chosen to balance convergence speed and training stability. Only the weights of the new head were updated, while the pre-trained convolutional layers of ResNet50 remained frozen.

Fine-Tuning the Entire Model After retraining the head, a fine-tuning phase was performed to further refine the model. In this phase, some convolutional layers of ResNet50 were unfrozen and retrained with a reduced learning rate of 0.0001. This lower value was chosen to avoid excessively perturbing the pre-trained weights, allowing for gradual adaptation to the new task.

Database: This dataset contains approximately 190,000 facial images, divided into train set, test set, and validation set, with a good variety of angles, occlusions, and lighting conditions. The model was trained on a subset of 20,000 training images and 4,000 test images.

Method 2

In the second attempt, a video dataset was used, consisting of 3,434 videos (365 original and 3,069 deepfakes). To balance the classes, the first 365 deepfakes and all 365 original videos were selected. From each video, a frame was extracted every 10 seconds, with the goal of capturing facial expressions during the video and making the training dataset more similar to the evaluation set. Faces were extracted using the MTCNN detector, keeping only snapshots with clearly visible faces and a minimum bounding box size. The ResNet50 model was retrained on this new dataset, following the same transfer learning approach: first, retraining the head with a learning rate of 0.001, followed by fine-tuning with a learning rate of 0.0001.

Method 3

For this approach the DeeperForensics 1.0 dataset was introduced, a very large dataset (over 280 GB of videos) that includes both deepfake videos and source videos (recordings of the people used to create the deepfakes, seen from different angles). For practical reasons, only 769 deepfake images were extracted from this dataset, using the same frame and face extraction technique via MTCNN as previously applied. This new dataset was combined with the one used in the previous attempt, creating a more diverse and robust dataset. The ResNet50 model was retrained on this combined dataset, following the same transfer learning approach: retraining the head with a learning rate of 0.001 and fine-tuning with a learning rate of 0.0001.

3.d) Conclusions and Possible Future Improvements

In conclusion we were not able to achieve improved results with the DL approaches in comparison to the original "eyebag" approach taken in task 1. Thus the best final approach we could get was 0.63 AUC. If one further wants to improve the performance of the deepface detections, one could fine-tune the eyebag/wrinkle detection approach. The edge enhancement and threshold for the binary selection (is eyebag / is not eyebag) very implemented only roughly and could definitely be improved. One could even try to train a DCNN only to wrinkly areas in deepfakes, though we expect a strong gender bias here (Makeup, smoother skin in women). In en even more advanced approach one might include gender estimation and train separate deepfake networks for the genders. Concluding the DL methods, we could show that increasing the variance and amount of our training data can improve the training performance.

Figure S1: Exemplary Processing of the Images as used for Method 1. Images are rotated and cropped to horizontally align the faces.

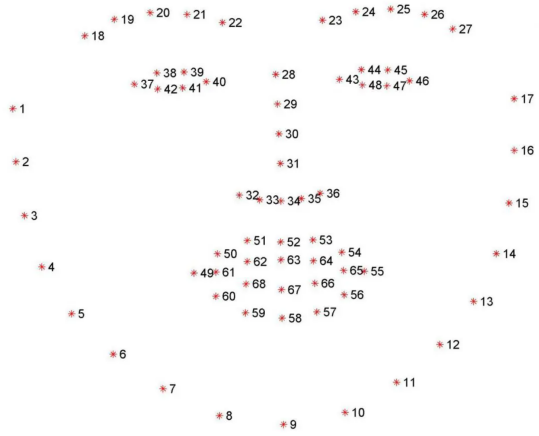

Figure S2: Landmarks extracted by dlib. Image credits: `https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/`

Supplementary Figures

Figure S3: Comparison of an image of Elon Musk with its deepfake equivalent. The inserted face of Nicolas Cage seems too small for the face. Moreover, the lower quality of the deepfake image gives the face a more blurred, airbrushed look, making the visible eyebags of Elon Musk less pronounced.