Università degli studi di Torino

SCUOLA DI SCIENZE DELLA NATURA

Corso di Laurea Triennale in Informatica

Tesi di Laurea Triennale

On the use of DeepFakes for Data Augmentation

Relatori:
Prof. Idilio Drago
Prof. Elvio Amparore

Candidato:
Enrico Cassano

Anno Accademico 2021/2022

Life's but a walking shadow, a poor player
That struts and frets his hour upon the stage
And then is heard no more. It is a tale
Told by an idiot, full of sound and fury,
Signifying nothing.

William Shakespeare, *Macbeth, Act V, Scene 5*

DICHIARAZIONE DI ORIGINALITÀ

Dichiaro di essere responsabile del contenuto dell'elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d'autore. Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata.

A Lara, e alla mia Famiglia,
grazie di cuore.

Abstract

Data augmentation is a technique consisting of extracting or generating more training data from a starting dataset. It can help in the training of a model, by reducing one of the most persistent issues in this task: the overfitting. This problem is often caused by insufficient data in the training set, and manifests itself in a poor performance of the model when tested on previously unseen data. Generating bigger datasets with real data is anything but a cheap and trivial task, requiring new approaches to reduce the resources consumption to inflate a dataset.

The main goal of this thesis is to develop a tool capable of augmenting data, to help in the process of handling human behavioural and facial classification.

In addition to that, we wanted to study how data augmentation (more precisely, data augmented by AI) can affect the performance of a classifier model.

To do so, a data augmentation tool was developed, called FaceSwapAugmentation: it takes an existing video from a dataset, and replaces its face with second one. To this thesis' scope, the model considered is trained upon the MaskDAR[1] dataset. The model's purpose is to understand and classify the behaviour of a driving person.

One of the key aspects of FaceSwapAugmentation, is that the latter face can be retrieved from another pre-existent video, or can be AI generated, making possible to generate an unlimited number of faceswaps. These face-swapped videos will then be used to inflate the original dataset, that will then be employed to train the classifier. The objective is to understand whether this technique can be useful to improve the model's performance, without the need of acquiring a bigger dataset composed by real data. In addition to that, we also wanted to do a comparison between the generating methods put at disposal by faceSwapAugmentation.

To answer the first question, the best and most feasible test was to build an augmented dataset with FaceSwapAugmentation, and then use it to train the classifier to see whether the performance improved. The first results showed a slightly worse performance of the model trained with fake data and tested on fake data, in comparison to the performance of the model trained and tested with real data. This, however, needs further consideration and study, taking into account also the resources that can be saved using AI generated data, and also the diminished privacy issues when making a dataset public.

The second research question was answered by giving an objective score to each face swap made in each workflow of FaceSwapAugmentation. For each flow, 30 deepfakes were made: this allowed us to generate a distribution of values for each flow of usage, and to use these distributions to do a comparison. This gave as result that the AI-generated face swaps (ThisPersonDoesNotExists) produce the best deepfakes.

Italian abstract

L'obiettivo della tesi è di studiare come l'uso di DeepFakes con il fine di augmentare un dataset (MaskDAR[1] in questo caso), influenzi le prestazioni di un classificatore. Il modello considerato è il classificatore descritto in [1]. Ha come classi 15 possibili azioni eseguite da un utente al volante, ed ha lo scopo di capire quando esso è distratto. Per fare ciò, è stato sviluppato un tool chiamato FaceSwapAugmentation, il quale prende come input un video e ne produce un deepfake, sostituendo il suo volto. Il volto nuovo può essere ottenuto a partire da un altro video esistente, oppure può essere generato tramite AI. Questa seconda possibilità, rende illimitato il numero di possibili face swaps ottenibili a partire da un singolo video. I deepfakes così generati verranno poi utilizzati per addestrare il classificatore, verificando se sia possibile ottenere un aumento delle prestazioni. In aggiunta a ciò, si vuole anche capire quale dei metodi di generazione messi a disposizione da FaceSwapAugmentation sia il migliore.

Per rispondere al primo quesito, la metodologia è basata sul generare un dataset composto da deepfakes, usarlo per allenare il modello, e poi verificare se ci sia stato o meno un aumento delle prestazioni. I primi risultati mostrano come la performance del modello allenato e testato con deepfakes sia leggermente peggiore rispetto alla sua performance addestrandolo e testandolo su dati reali. È necessario, in ogni caso, analizzare in modo più accurato questi risultati, anche in ottica del risparmio in termini di risorse e della maggiore semplicità dal punto di vista della privacy per la diffusione del dataset.

Per rispondere alla seconda domanda, è stato assegnato un punteggio oggettivo ad ogni deepfake sulla base della somiglianza fra il volto di base e quello sostitutivo. Per ogni workflow di FaceSwapAugmentation sono stati prodotti 30 deepfakes, generando così una distribuzione di valori per ogni metodologia di utilizzo. Lo studio della distribuzione suggerisce come migliore il metodo di generazione del deepfake basato su ThisPersonDoesNotExist.

Contents

Chapter 1

Introduction

Since the beginning of its journey - located in the second half of the 20th Century - Artificial Intelligence has always been strictly related to the data upon which it gets trained. Among all the several techniques available to improve the performance of a Deep Neural Network (DNN from now on), the one with the biggest payoff is to train the DNN with *more* data.

The collection of data, however, is anything but a trivial task. In addition to being intrinsically complicated, it may also demand a big amount of time to manage all the people involved - to collect the data itself, and, for projects with an objective similar to the one of this thesis, to also generate the data. Finally, to produce and collect big amounts of data, money investments are generally required.

This thesis and the resulting project are an attempt to mitigate this kinds of issues that are needed to be tackled when generating a dataset. More specifically, the proposed project aims to *augment* a dataset: this means that it starts from an existing dataset and tries to extract more data from it, or to build additional data upon the existing one by tweaking some features. This thesis objective is to go beyond that, and augment an existing dataset by adding more subjects to it by generating the subjects from AI.

The tool built in order to achieve the former described results is called *faceSwapAugmentation*. It's job is to take as input two main elements: **a destination video** containing the footage of a person performing some action - this will presumably be a recording from an existing dataset that we want to augment, and a **source face**, that will be used as data from which to extract the new face that will be applied upon the **destination video**. As said, the tool allows the user to give as input the destination video only, and to request the source's face to be AI generated.

The scope of this project is to operate an inflation of a dataset that contains videos of people whom's face is the main focus of the recording. The concrete objective of this project was to inflate the **MaskDAR** [1] dataset, containing recordings of 30 different subjects, both with and without a facial mask (during this thesis' production, the World's Pandemic of Covid-19 was still taking place). The augmented dataset will be composed of footages of new subjects consisting of a new face applied to an existing video. This will be useful to avoid biases and overfitting, as more deeply explained in the *Background* Chapter.

Regardless of the actual scope of faceSwapAugmentation, it can be use on any dataset that contains videos of people, in which changing the face of the subject can be beneficial for the training of a DNN.

In the first phase of this thesis, it was necessary to search and understand whether previous attempts were made in this specific field, and how advanced were they. It was intresting to discover that were no existing paper or project that aimed to augment or inflate a dataset using deepfakes. To build and test a tool that has what's required, we had to start from scratch by studying and understanding how DeepFakes are generated, which tools were available and what was the current state of the art. The most important element of faceSwapAugmentation is DeepFaceLab [9], a very complex and complete tool that allows the user to mesh two faces, taking as input two mp4 footages. It can also be used for post-production purposes, containing several advanced sub-tools that can be used to perform adjustments of color, face shape and size, its blurness, and so on. DeepFaceLab performs the face substitution by training an internal DNN (an Autoencoder) that *learns* the most important features of the face, and extracts landmarks from both source's and destination's images in order to apply one upon the other. The most impressive element of this product is the facial expressions and realisticness of movements in the resulting video. In addition to that, this tool was selected because it's the current state of the art for deepfakes generation.

DeepFaceLab alone was not sufficient to build the requested tool, we also wanted the option to retrieve the source face without giving it as input. To do so, two main methods of face generation are provided: *ThisPersonDoesNotExists* and *StyleGAN 2* [7]. They both are tools that simply consist of a script that uses their APIs, and both output a single 2D image, representing an AI generated subject.

It's not possible, however, to use the so-produced images as direct input to DeepFaceLab, for the following reasons:

- They have zero angle variance (it's trivial, since they're 2D images)
- They have zero emotional variance (which is very important to generate a realistic deepfake)

The first issue is resolved in the following way:

- We build a 3D model of the 2D image using Deep3DFaceReconstructor [5]
- We turn the 3D model of the face into a batch of images that represent the reconstructed face from multiple angles.

The batch of images is then fed into DeepFaceLab.

As an alternative to the previously described input (source footage and image generation), it is also made possible to retrieve the source's faces from an existing dataset, skipping the steps of facial generation and transformation into batches. In this thesis, the tests for this method were done using the KDEF dataset [4], but many other similar datasets can be used. KDEF was chosen due to its structure and the fact that each contained individual contains several images, each one having a good angle and emotional variance.

Two main questions are the core of this thesis' research:

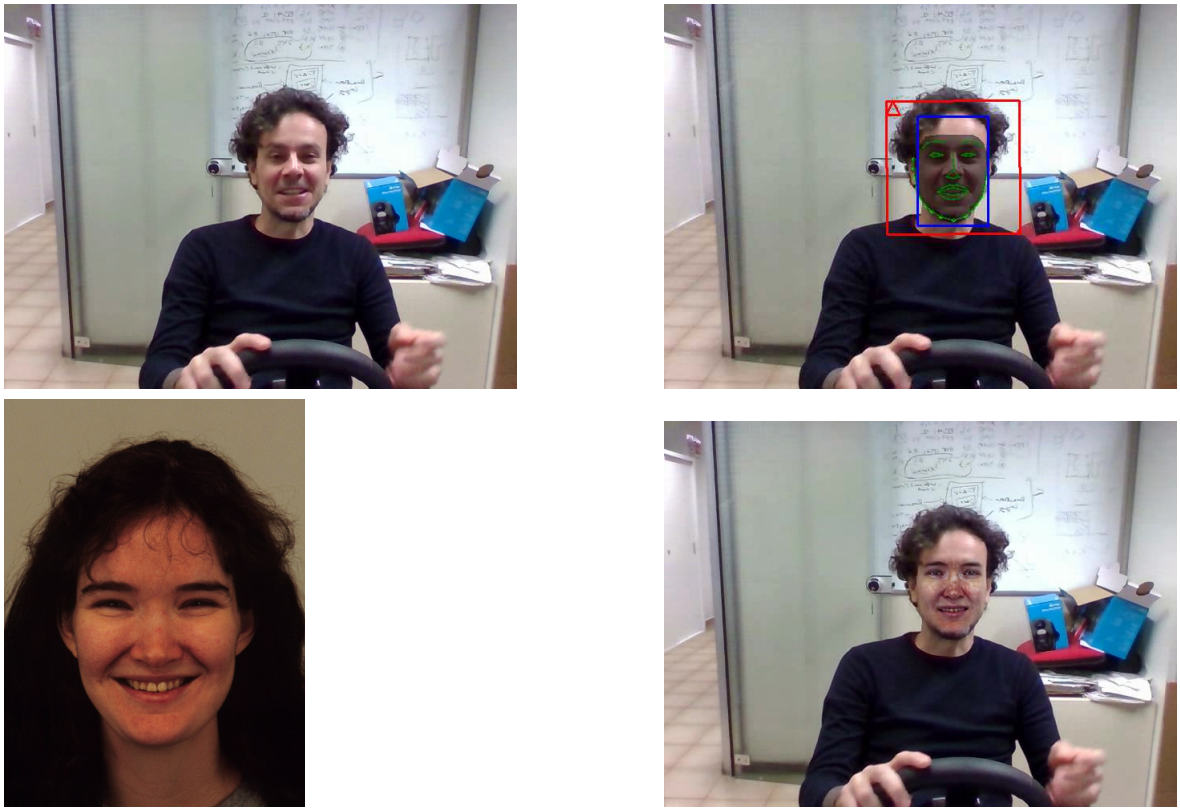

Figure 1.1: A deepfake built with FaceSwapAugmentation by using an AI generated source face. In this case, the face was generated with ThisPersonDoesNotExist.

1. What is the best strategy to generate a dataset? (Among the ones put at disposal by faceSwapAugmentation)

2. Does deepfake data augmentation approach improve the classification performance of a Drive Action Recongition? Does the quality of the replacement impact the performance of the trained model?

Images 1.1 show an example of deepfake generated with FaceSwapAugmentation and an AI generated source face. In this case, the face was generated with ThisPersonDoesNotExist.

The first research question, *What is the best strategy to generate a dataset?*, requires some sort of comparison method between the available strategies to be coherently answered. To do so, its necessary to define a quantitative score that enables us to choose the objective best among the possible methods.

First of all, we defined a score to evaluate a singe deepfake, based on the similarity between the landmarks extracted from the source image and the destination image. We then developed a second scoring method to be applied to a strategy itself, rather than the single swaps. This second scoring strongly relies on the scores given to the face swaps done with the to-be-evaluated flow of usage. Thus, we use the scores given to the deepfakes done with one method to evaluate the method itself. This is done by studying the distribution of the values obtained by the same workflow. A cumulative density

function is plotted for each flow, in order to qualitatively see which one performed the best. As seen in figure 4.13, the best results are given by ThisPersonDoesNotExist.

The second question, *Does deepfake data augmentation approach improve the classification performance of a Drive Action Recongition? Does the quality of the replacement impact the performance of the trained model?*, can be answered in a less theoretical way. To do so, we created a new dataset based on the given one (to this matter, the starting dataset is MaskDAR[1]), containing 30 deepfakes. 15 were made using external source videos, and 15 were made using source faces generated by ThisPersonDoesNotExist.

Since the state of the art of *data augmentation using deepfakes* was not very well defined when this thesis began being developed, *faceSwapAugmentation* may represent a starting point itself. Being so, there are several elements and aspects that can certainly be improved. The most trivial situation that causes some trouble to the tool, are obstruction objects that are placed between the camera and the to-be-extracted face. To this matter, are included both temporary obstructive objects, but also semi-permanent elements. In the first category, we consider items such as a cup, or a water bottle, that partially cover the face for a limited amount of time, generally for a matter of seconds. The second category of obstructive elements, are the facial covers and protection devices required during the Covid-19 pandemic.

Both this kinds of elements cause issues in two steps of the deepfake generation:

- During the face detection and extraction phase, since they interfere with the landmark placement. This is caused by the fact that bottles, cups, facial masks are not similar to face elements (such as mouth, cheekbones, eyes, ...) and thus lead the face extractor into mistakenly see faces where there aren't, and to fail to recognize partially covered faces.

- During the face substitution phase, especially when one of the two subjects is covered, and the other is not. This will result in a messy and unrealistic swap, since the obstructive element will partially be represented in the result.

A solution to this issue can be represented by a face detector that is also trained to spot common face covering elements: if a facial mask/water bottle/drinking cup is recognized in the destination footage, the face replacement will be performed only for the section of the source face that is not covered by the obstructive element.

A similar issue is caused by elements such as glasses, a beard and hair. To this matter, the solution seems to be less trivial compared to the masks issue, since it may require some more specific and practical knowledge to achieve realistic results. The most interesting (and arguably the hardest) of the issues of this pipeline, is the lack of emotional variance that is present when the source subject is AI generated. An intresting approach would be the one descripted in the [8] paper, in which is attempted to add emotional variance to the 3D Morphable Face Model by modifying it accordingly to common pattern in faces.

Chapter 2

Background and State Of The Art

2.1 Background

The main goal of this thesis is to be able to augment an existing dataset, consisting of footages of driving subjects. In order to inflate the dataset dimension, this thesis makes possible to create a deepfake of the original video, replacing the original face with a second one. This approach allows the user to inflate the dimension of a training set, putting at disposal more material that can be used to train a Deep Neural Network model, but can also be used for cross-testing and evaluation. In addition to creating simple deepfakes, faceSwapAugmentation allows the user not only to mesh two videos given as input, but also to start from a single video and make an unlimited number of face swaps. This is made possible by using AI generated faces, using the techniques and tools described in this chapter. By doing so, the advantages are not limited to the generation of a bigger dataset, but also to save time and money, avoiding the need of involving other people in the process.

2.1.1 Deep Neural Networks

Thanks to the increasing computing capabilities and to the very large quantity of data at disposal, both gained approximately in the last ten years, this kind of Neural Networks started to be vastly used and researched. They're called *Deep* because of the many layers that can be inserted in between the input and the output layer. Deep Neural Networks (DNNs) have been indeed useful for multiple problems, such as classifying images and voice, as well as time series prediction. Different DNN architectures can be built in order to choose one that suits best the problem that needs to be tackled [13].

- **Recurrent Neural Networks**: each node in the hidden layers forwards its result not only to the next layer, but also to itself. This structure allows the netowork to remember patterns of previous data instances.
- **Convolutional Neural Networks**: nodes use convolutional matrices in order to compute outputs. This scheme is useful for filtering and extracting complex features from the input data. Convolutional Neural Networks have been used

successfully for image processing, e.g. due to their capacity to identify image borders.

2.1.2 Encoder-Decoder

The Encoder-Decoder Structure (also known as *Autoencoder*) is composed by an encoder and a decoder, and it's a type of Neural Network that extracts from the input data a series of features, that will that will be compressed into a latent space. The compressed data is put into a latent space, from which will be reconstructed the output image. The quality of the reconstruction will then be measured by means of the reconstruction error [13]. The image 2.1, taken from [13], shows the basic structure of an autoencoder, being X the input data, h the data in the latent space, and X' being the output data reconstructed from the h values in the latent space.

Encoder

It's job is to encode data: *encoding* means to compress the input data into a latent space. The encoder is built by stacking Recurrent Neural Network (RNNs). This type of structure allows the model to understand context and temporal dependencies of the sequences. The output of the encoder, the hidden state, is the state of the last RNN timestep.

Decoder

Its job is to reconstruct the input based on the latent variables. In the machine learning model, the core job of the decoder will be to convert the encoded values in the vector into the output sequence. It is built with RNN layers aswell.

2.1.3 GANs

Generative Adversarial Networks[2, 12, 10], are a class of machine learning frameworks in which, two neural networks (a generative network and a discriminative network) contest each other in a game (a sum zero game). Given a training set, this tecnique learns to generate new data with the same statistics as the training set. The generative network generates candidates while discriminative network evaluates them. The generative network's training objective is to increase the error rate of the discriminative network (pratically, it tries to fool the discriminative network).

The generator G takes noise as input, and its goal is to generate instances that resemble the real data, whilst the discriminator D does a binary classification between *real* and *generated*. The used loss function on the training set takes into account the similarity between generated data and real data. [13].

Figure 2.2, taken from [13] shows the basic structure of a GAN, in which the Generator G gives as input to the Discriminator D some generated data, named $G(z)$. D, that is trained upon the real data x, tries to classify the given input as real or fake.

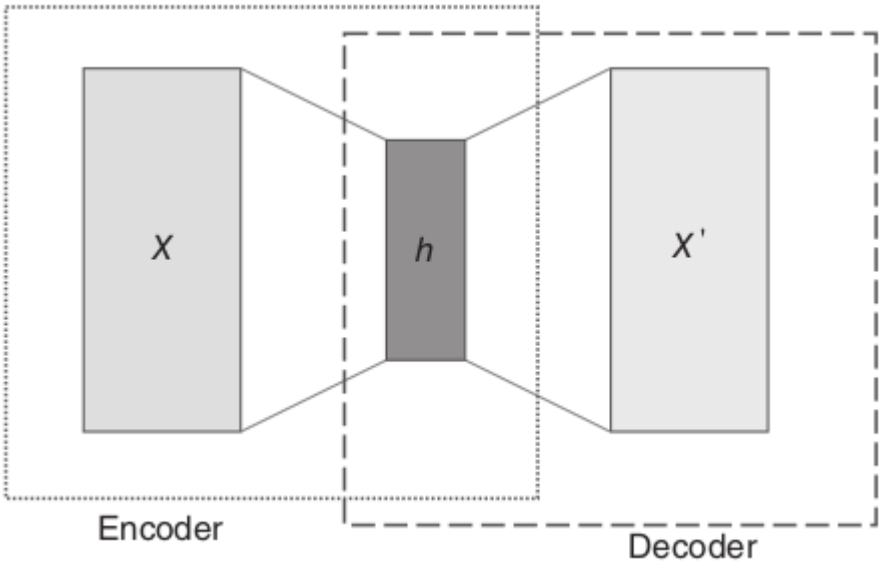

Figure 2.1: Basic Autoencoder Structure taken from [13]. X represents the input data, h is the function that compresses the input data into a latent space, and X' is the reconstructed data from the latent variables.

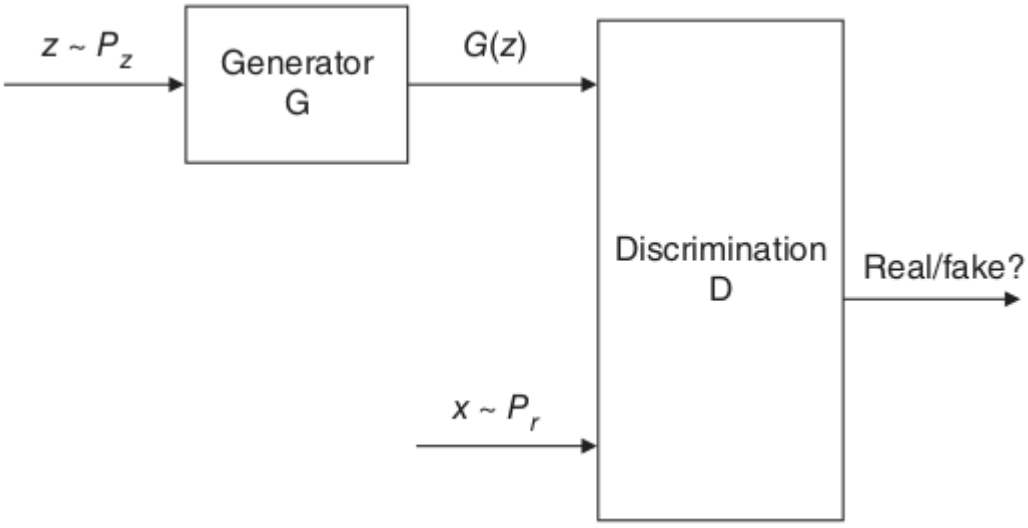

Figure 2.2: GANs basic structure taken from [13]

2.2 Possible problems caused by the training data

When the DNNs are trained to classify some very specific and precise actions, a lot of different examples are required in the training dataset in order to precisely understand which actions or elements of the subject are relevant to the classification. If the dataset is small, the DNN may excessively rely on the presented examples, thus resulting in an insufficient performance with previously unseen data. This issue is named *overfitting*.

Overfitting

Overfitting is an issue that can reveal itself in the testing phase of a DNN model. It consists of a lack of generalization of the trained model, meaning that is has difficulties on classifying previously unseen data. Models with poor generalizability have overfitted the training data. One way to discover overfitting is to plot the training and validation accuracy at each epoch during training.

The trivial solution to this issue is to enlarge the dataset dimension, injecting in it more classified examples. This, however, is not always possible due to the costs of time and money required in order to acquire more data.

The state-of-the-art solution to this issue, today, is represented by *Data Augmentation*.

2.2.1 Data Augmentation

Data Augmentation is a very powerful technique that allows the reduction of the validation error and the training error. This is necessary in order to build useful Deep Learning Models.

The augmented dataset will consist of a more complete set of data elements, thus minimizing the distance between the training and validation set, as well as any future testing sets.

Data Augmentation is a technique that is very useful in order to reduce overfitting, an issue by which the trained model relies too much on the training dataset, making wrong assumptions and correlations on events.

Data Augmentation approaches overfitting from the root of the problem: the training dataset. It tries to extract more information from the original dataset using augmentation tecniques, such as data warping or oversampling [11] .

- **Data warping**: consists of transforming existing images such that their label is preserved. This encompasses augmentations such as geometric and color transformations, random erasing, adversarial training, and neural style transfer.

- **Oversampling**: consists of creating synthetic instances and adding them to the training set. This includes mixing images, feature space augmentations, and generative adversarial networks (GANs).

Oversampling and Data Warping augmentations are not mutually exclusive [11].

2.3 Face Generation Software

Many different products capable of generating faces of non-existent people have been developed and made public in the last years. The current state of the art is called Syle-GAN, developed by Nvidia and based upon Generative Adversarial Networks. In this thesis are used two different version of StyleGAN: the first published version of it, used by ThisPersonDoesNotExist. The second version, called StyleGAN 2, is also implied in this thesis, and the used version is trained upon MetFaces, a dataset composed by the portraits present in the Met Museum.

Among other changes, the second version of StyleGAN improves the former version in two fields:

- It avoids creating visual artifacts, that were present with StyleGAN
- The face generation is done with more angle variance, both of the face and the eyes of the subject.

2.3.1 StyleGAN 2

The resolution and quality of images produced by generative methods, especially generative adversarial networks (GANs), are improving rapidly. The current state-of-the-art method for high-resolution image synthesis is StyleGAN 2 (produced by Nvidia) [7], which has been shown to work reliably on a variety of datasets.

The Style Generative Adversarial Network (StyleGAN), is an extension to the GAN architecture, that proposes the following changes to the generator model (GAN):

- the use of a mapping network to map points in latent space to an intermediate latent space
- the use of the intermediate latent space to control style at each point in the generator model
- and the introduction to noise as a source of variation at each point in the generator model.

The resulting model is capable not only of generating high-quality photorealistic images of faces, but also offers control over the style of the generated image at different levels of detail through varying the style vectors and noise.

StyleGAN 2 [7] yields state-of-the-art results in data-driven unconditional generative image modeling.

2.3.2 ThisPersonDoesNotExist

The AI face generator (ThisPersonDoesNotExist) is powered by StyleGAN, a neural network from Nvidia developed in 2018.

For the user, everything works very simply. As soon as the webpage

(www.thispersondoesnotexist.com) is requested, random face is generated. The generated image can be saved if required. A new random person can be generated by

refreshing the webpage. The website shows the results of the generator's work (which are updated every 2-3 seconds) not the generator itself.

2.4 DeepFake Generation

In the last five to eight years, among many social networks and news platforms, started to become popular the *face swaps*, which consist of a (more or less) realistic face substitution of a starting model, presented in a video or an image. This draw very much attention to the cases in which movie stars or celebrities were involved, because most of the times, the face substitution was made for pornography or defamation purposes.

In order to protect people from a bad usage of this technology, many AI projects consisted in training a model, with the goal of identify deepfakes, and distinguish between real material and face swaps.

None of the previous usages of deepfakes seems to have considered this technology for the field of data augmentation, in particular to generate or inflate a dataset of real people in a very low time-consuming and cheap approach. Among the different methods available to generate deepfakes, we choose to use DeepFaceLab [9] because of two main reasons:

- It's the current state of the art. The pipeline proposed is mature and capable of achieving photorealistic results
- It's open source, meaning that any required change in the code can be made painlessly.

In 2018, DeepFakes introduced a complete production pipeline in replacing a source person's face with the target person's along with the same facial expression such as eye movement, facial muscle movement. However, the results produced by DeepFakes are poor somehow, so are the results with contemporary Nirkin et al.'s automatic face swapping. In order to further awaken people's awareness of facial-manipulation videos and provide convenience for forgery detection researchers, we established an open-source deepfake project, DeepfaceLab (DFL for short), which is used to build enormous high-quality faceswapping videos for entertainment and greatly help the development of forgery detection by providing high-quality forgery data [9].

Compared to the previous face swapping tecniques, DeepFaceLab [9] promised the best possible results, other than being the current state of the art. In addition to that, the choice fell upon this tool because it is open-sourced and thus, modifiable. It also promised to keep up with the progress in computer vision areas and making positive contributions in the field.

2.4.1 DeepFaceLab

[9] DeepFaceLab is the current dominant deep-fake framework for face-swapping. It provides the necessary tools as well as an easy-to-use pipeline to generate high-quality face-swapping.

Face swapping is a challenging task in generating fake content by transferring a source face to the destination while maintaining the destination's facial movements and expression deformations.

It is noteworthy that DFL falls in a typical one-to-one face-swapping paradigm, which means there are only two kinds of data: src and dst, the abbreviation for source and destination.

In order to achieve high-quality face swaps, GANs (Generative Adversarial Networks) are used. DeepFaceLab provides a set of workflow which form a flexible pipeline. In DeepFaceLab, we can abstract the pipeline into three phases: **extraction**, **training**, and **conversion**.

Extraction: The extraction phase is the first phase in DFL, aiming to extract a face from src and dst data. This phase consists of many algorithms and processing parts, i.e., **face detection**, **face alignment**, and **face segmentation**.

- **Face Detection**: the first step in extraction phase is to spot the target face in the input images. S3FD as is the default face detector for DeepFaceLab.
- **Face Alignment**: the second step is face alignment. Facial landmarks are the key to maintaining stability in a long sequence of frames. All the so aligned faces are then put in a data folder with face of standard front/side-view (aligned src or aligned dst).
- **Face Segmentation**: After face alignment, a fine-grained Face Segmentation network is employed on top of aligned source and aligned destination, through which a face covered by either hair, fingers, or glasses could be segmented exactly. In addition to this, since some state-of-the-art face segmentation models fail to generate fine-grained masks in some particular shots, the XSeg was introduced in DFL, which allows users to manually set segmentation points on the aligned images.

Figure 2.3, taken from [9], shows an overview of the steps required in order to generate an aligned face, and an aligned mask, starting from an input image. As shown, the first phase is to detect one or more faces in the input image. After that, landmarks are used in order to align faces, and finally, faces are segmented in order to create an aligned face mask.

Training: The training phase plays the most vital role in achieving photorealistic face-swapping results of DFL. No need for facial expressions of aligned src and aligned dst being strictly matched, DFL aims to provide an efficient algorithm paradigm to solve this unpaired problem along with maintaining high fidelity and perceptual quality of the generated face.

DeepFaceLab's structure consists of an Encoder as well as Inter with shared weights between src and dst, two Decoders which belong to src and dst separately.

Conversion: The first step of the proposed face-swapping scheme in the conversion phase is to transform the generated face alongside with its mask from dst Decoder to the original position of the target image in src.

For the sake of keeping the complexion consistent, DFL provides five color transfer algorithms (Reinhard color transfer: RCT, iterative distribution transfer: IDT, ...)

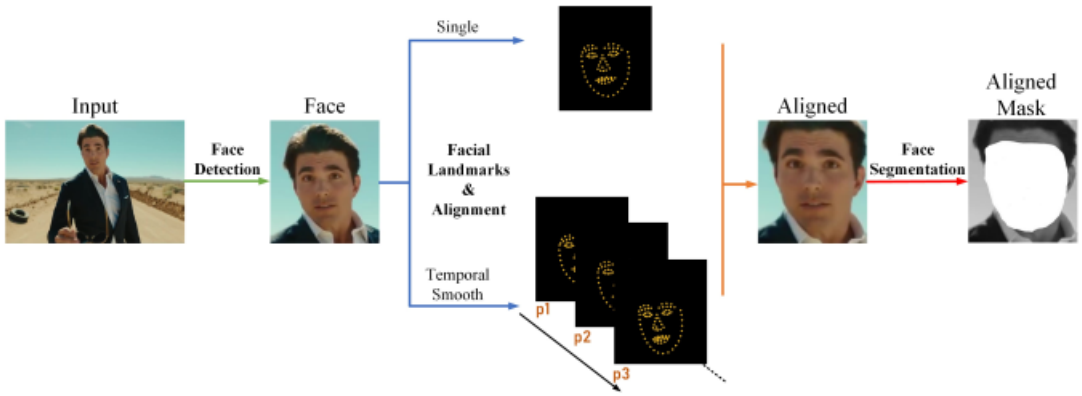

Figure 2.3: Extraction phase dfl: an overview, taken from [9]

to approximate the color of the reenacted face to the target. Any blending must account for different skin tones, face shapes, and illumination conditions, especially at the junctions between reenacted face with the delimited region and target face. DFL implemented this by Poisson blending. For sharpening purpouse, a pre-trained face super-resolution neural network was added to sharpen the blended face since it is noted that the generated faces in almost current state-of-the-art face-swapping works, more or less, are smoothed and lack minor details such as moles or wrinkles.

Chapter 3

FaceSwapAugmentation

Because of their nature and the way in which are designed, Deep Neural Networks, in particular Classifiers, need huge amounts of data to achieve a viable performance. Obtaining the required amount of data is often an issue that needs to be considered and discussed before even starting to design and build the DNN itself.

FaceSwapAugmentation takes as first input a video, which will be the base video upon which perform the face swap. This starting video is taken from the **MaskDAR** [1] dataset, consisiting of 30 individuals performing each 15 labeled actions. FaceSwapAugmentation is required to start from one of the **MaskDAR** videos, apply a new face to it, and then produce a resulting video.

Acquiring a dataset from scratch would almost certainly be a a time consuming activity, but would likely also require a generous money investment to find and coordinate all the needed people for the task. Another worrying aspect of building a dataset from scratch may be the fact that it can possibly be unstructured, not have a good enough quality, or just be wrong for the purpose.

To start from a public dataset could be a better idea, but in this scenario it is also needed to consider the fact that the publicly available dataset may not suite our needs.

The approach presented in this thesis allows the user to start from an existing dataset of a modest size, that can equally be just built or public, and augment it in order to inflate it's dimension. This makes possible to start from a set of data which suits the needs of the purpouse - in this case, it is to train a behaviour classifier - and to add more data to it, structured in the same way of the beginning dataset. FaceSwapAugmentation is built to work around a dataset of videos representing people, the main focus being people's faces.

FaceSwapAugmentation is a system that allows to build an unlimited number of face swaps, starting from a single video.

From the input video will be generated a second video, by swapping it's face with another one. The second face can be retrieved from another picture, another video, or generated by AI. *The fact that the faces can be generated from AI, with tools such as* ***ThisPersonDoesNotExist****, or* ***StyleGAN 2****, allows a truly unlimited number of face swaps for each video.*

In order to build an automatic and reliable tool, I used several tools that are the current SOTA, but also some selfmade scripts in order to connect seamlessly all the components.

3.1 Requirements

The requirements for faceSwapAugmentation (FSA) were to be a tool capable of building an augmented dataset by taking advantage of the deepfake tecniques. In addition to that, its requirements are:

- **To be automatic**: it's necessary for faceSwapAugmentation to be used without the need of any human interaction but giving the input video(s).
- **To be able to augment an existing dataset**: the objective of this thesis is to test whether deepfakes can be a feasible way for training a DNN
- **To be capable of generating an unlimited amount of deepfakes from one destination video**: this is done through the use of tools that generate non-existent people from scratch, using AI
- **To give the user multiple use-modes**: generating the source face from scratch, using an existing video or an existing image from which to extract the source face.

The augmented dataset will then be used to train a classifier of driving actions, in order to better identify and recognize when a car driver is distracted, by doing something else than driving [1].

One of the key aspects is the fact that in the resulting video, the actions performed will be the same of the input video, thus respecting the labels associated with each frame of it. By applying a new face to it, it's possible to reduce or prevent spuriousnesses in the trained model, as an attempt to improve its performance.

3.2 Architecture

The tool, called faceSwapAugmentation, is built upon several tools, such as **DeepFaceLab** [9], **Deep3DFaceReconstructor** [5], **ThisPersonDoesNotExist** [7], **StyleGAN 2** [7]. Some other tools where built ad hoc in order to perform challenging tasks needed to connect all the sections of the tool together.

FaceSwapAugmentation takes as input video `data_dst.mp4`, which will be used as destination upon which apply a source face. This face can be generated by AI, in two different methods: `stylegan2` , or `ThisPersonDoesNotExist`.

The to-be-applied faces can also be gathered from a dataset. To this matter, KDEF[1] face dataset is the suggested dataset, but many others can be used.

[1]KDEF is a dataset of pictures contains 70 individuals, displaying 7 different emotional expressions. Each expression is viewed from 5 different angle. It was chosen because it presented the two most important features required to make a realistic face swap: emotional and pose variance.

Figure 3.1 shows an high level diagram of the structure of FaceSwapAugmentation. The image generation block is by any mean the most complex, and will be futher discussed in the analysis section of this chapter.

It's important to notice that the obtained results will strongly rely on two factors:

- Emotional variance of the source images
- Angle variance of the source images

The former is essential to achieve realistic face muscle motion and facial expressions, the latter is fundamental in order to obtain a realistic face swap, even when the pose reaches extreme angles.

In addition to that, source images can also be retrieved from an `.mp4` video, given as input directly to **DeepFaceLab**.

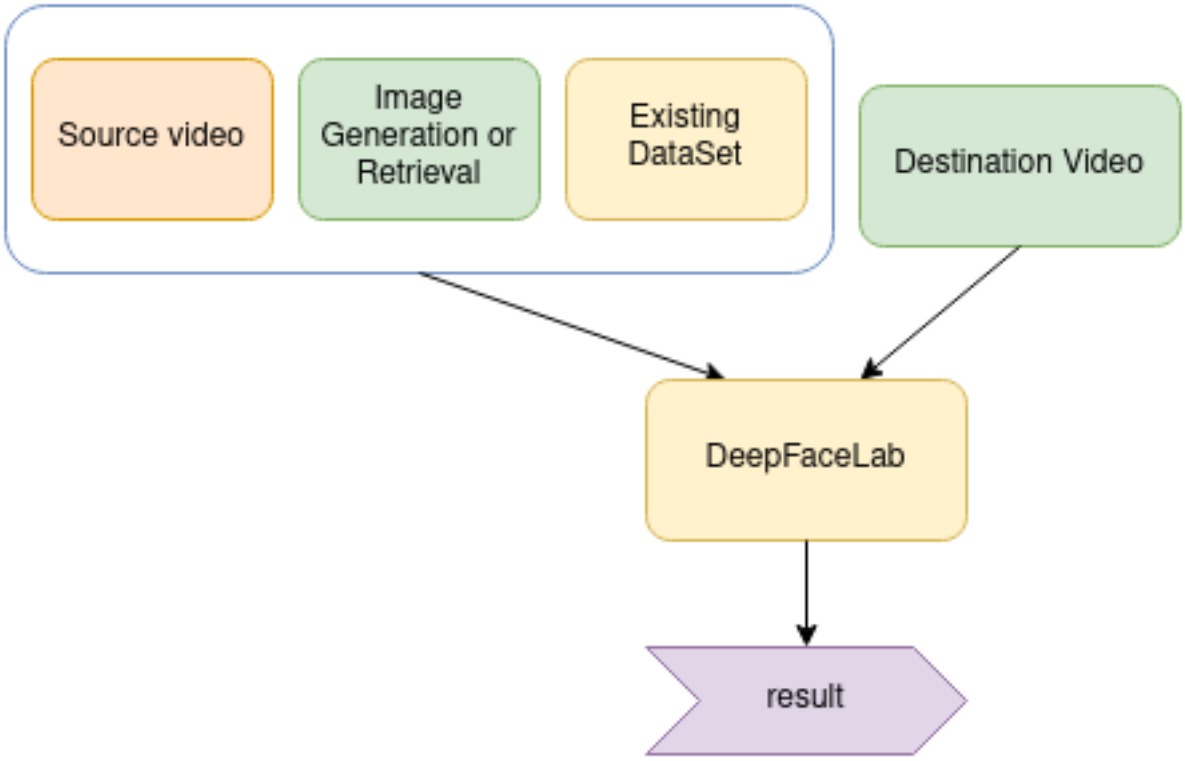

Figure 3.1: FaceSwapAugmentation base structure

3.3 FaceSwapAugmentation Analysis

3.3.1 Base structure

As shown in figure 3.1, the high level structure of FaceSwapAugmentation is quite simple: the main tool used is DeepFaceLab [9], which takes as input two batches of frames: the source's frames, which are the faces that will be applied upon the destination's frames, which are extracted from a base video, given as input. It then generates the `result.mp4` video, containing all the poses and facial expression of the destination video, but with the source image's face.

To do so, we run some specific scripts of the `DeepFaceLab` tool. Figure 3.2 shows all the available scripts in DeepFaceLab [9]. Most of them won't be used in FSA, since they require manual interaction, or are far too specific for the scope of this project.

Scripts 1 and 2 are going to be skipped, since we do not want to delete the material we have in `workspace`, and we don't need to extract images from the source's video, since we're already starting from a pool of images.

The scripts that will be used are:

- `3_extract_image_from_data_dst.sh`, which extracts the images from the base video.
- `4_data_src_extract_faces_S3FD.sh`, which will run the face detector on the source images.

```
(deep3d) npunito@npunito:~/DeepFaceLab_Linux/scripts$ ls
1_clear_workspace.sh
2_extract_image_from_data_src.sh
3.1_denoise_data_dst_images.sh
3_extract_image_from_data_dst.sh
4.1_download_CelebA.sh
4.1_download_FFHQ.sh
4.1_download_Quick96.sh
4.2_data_src_sort.sh
4.2_data_src_util_add_landmarks_debug_images.sh
4.2_data_src_util_faceset_enhance.sh
4.2_data_src_util_faceset_metadata_restore.sh
4.2_data_src_util_faceset_metadata_save.sh
4.2_data_src_util_faceset_pack.sh
4.2_data_src_util_faceset_unpack.sh
4.2_data_src_util_recover_original_filename.sh
4_data_src_extract_faces_MANUAL.sh
4_data_src_extract_faces_S3FD.sh
5.2_data_dst_sort.sh
5.2_data_dst_util_faceset_pack.sh
5.2_data_dst_util_faceset_unpack.sh
5.2_data_dst_util_recover_original_filename.sh
5_data_dst_extract_faces_MANUAL_RE-EXTRACT_DELETED_ALIGNED_DEBUG.sh
5_data_dst_extract_faces_MANUAL.sh
5_data_dst_extract_faces_S3FD_+_manual_fix.sh
5_data_dst_extract_faces_S3FD.sh
5_XSeg_data_dst_mask_apply.sh
5_XSeg_data_dst_mask_edit.sh
5_XSeg_data_dst_mask_fetch.sh
5_XSeg_data_dst_mask_remove.sh
5_XSeg_data_dst_trained_mask_remove.sh
5_XSeg_data_src_mask_apply.sh
5_XSeg_data_src_mask_edit.sh
5_XSeg_data_src_mask_fetch.sh
5_XSeg_data_src_mask_remove.sh
5_XSeg_data_src_trained_mask_remove.sh
5_XSeg_train.sh
6_train_Quick96_no_preview.sh
6_train_Quick96.sh
6_train_SAEHD_no_preview.sh
6_train_SAEHD.sh
7_merge_Quick96.sh
7_merge_SAEHD.sh
8_merged_to_avi.sh
8_merged_to_mov_lossless.sh
8_merged_to_mp4_lossless.sh
8_merged_to_mp4.sh
env.sh
```

Figure 3.2: DeepFaceLab scripts

- `5_data_dst_extract_faces_S3FD.sh`, which will run the face detector on the destination images.
- `6_train_SAEHD_no_preview.sh`, is required to train the model that will perform the face swap between images.
- `7_merge_SAEHD.sh`, this script applies the face masks generated from the former script, asking the user for parameters, in order to achieve the most realistic possible swap.
- `8_merged_to_mp4.sh`, generates the resulting video, concatenating together the merged frames.

In this tool, the following elements can be modified painlessly (since they're provided by DeepFaceLab as default alternatives or were modified by me in order to automatize the process):

- In steps 4 and 5, `S3FD` is used, as default face detector, but [9] allows the user to substitute it easily.
- In steps 6 and 7, the model `SAEHD` is used, but in DeepFaceLab is also provided another default model, `Quick96`.
- The train script (`6_train_SAEHD.sh`) is provided with a popup window, that allows the user to see in real time a preview of the outcoming face swap, but also a plot which shows the models progress and performance in training. This popup window is disabled in order to achieve the automation for FSA, since it required a human interaction in order to quit.
- The merge script (`7_merge_SAEHD.sh`) is also provided with a popup interactive window, which allows the user to choose and modify several parameters that will result in a more realistic face swap, such as mask blur, color intensity, color transfer mode, mask dimension, mask corrosion, and many others. This popup window is also disabled in order to achieve the automation for FSA, since it required a human interaction in order to quit.

3.3.2 Changes on DeepFaceLab source code

Since the goal of FaceSwapAugmentation is to be an automatic tool, all the tools and scripts that required human interaction must have been modified, in order to introduce a `Non-Interactive-Mode`.

All the scripts that were built for this project, were designed not to require human interaction, but only the necessary input data.

Pre-existing tools such as `stylegan2-ada-pytorch`, `ThisPersonDoesNotExist`, `Deep3DFaceRecon_pytorch`, were already as required: a script to be run providing the needed input.

The most challenging element that was not *Human Independent* was `DeepFaceLab`. Every single step of `DeepFaceLab` used in FaceSwapAugmentation requires some sort of interaction: many times it is just parameters (such as the *target iteration* for training purpouses, but also the image format (png/jpg) of the video-extracted frames, or the type of face swap), whilst other times it is required to interact with a GUI window, in order to either quit the training phase, or to do some post-production on the face-merging phase.

This kind of interaction was required to be made optional, allowing the user to create a `json` dictionary containing the default answer to the prompted questions. In addition to that, it's possible to insert into the dictionary the pair of values `"NonInteractiveMode": "y"`, that will keep the tool from showing GUI windows, using pre-defined settings for both the steps.

This changes turned out to be a pull request to the official DeepFaceLab repository on GitHub.

3.3.3 Image Generation

Going more deeply into the FaceSwapAugmentation structure, we start to analyze how the *Image Generation* block of the image 3.1 is structured. As shown in figure 3.3, in order to build a batch of source images, 4 main steps are required:

1. *AI Face Generation Phase or Face Retrieving*
2. *Landmarks Extraction*
3. *3D Model Construction*
4. *Images generation from 3D Model*

Since the *AI Face Generation Phase or Face Retrieving* step has a complex structure itself, it will be more accurately discussed in another section. What's important to understand for now, is that the block's output will be a single 2D image, picturing a human face.

The first step of FaceSwapAugmentation consists of generating or retrieving a source image. It must be a generic 2D image representing a human face.

As shown in figure 3.3, the following step is the *Landmarks Extraction* process. This step is fundamental in order to generate a 3D model of the source image, since a `.txt` file containing facial landmarks is required by the *3D Model Construction* block.

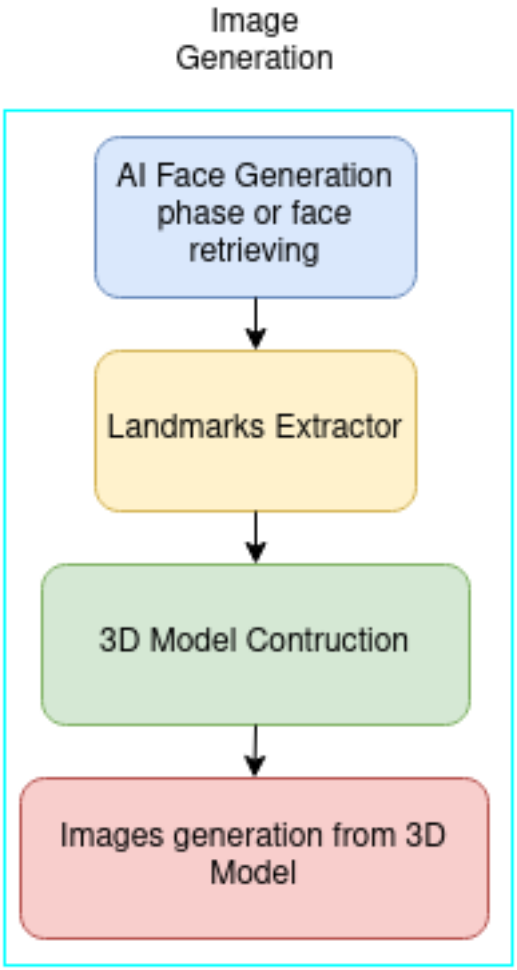

Figure 3.3: Elements that compose the image generation block. As shown, the key components are: AI FaceGeneration or face retrieving, Landmarks Extraction, 3D Model Construction, Images Generation from 3D Model

The Facial Landmarks detection is achieved in the `lmDeep3DFR.py` script, which was written based on the [3]'s work and library.

It takes as input an image, and outputs a `landmarks.txt` file containing all the desired facial landmarks, which are specified inside the script itself. For this thesis's purpose, the required landmarks are detected upon **nose**, **eyes** and **lips**.

One of the most important aspects required to achieve a realistic face swap, is the presence of a good angle variance in the source images. This is needed by the Face Detector, in order to build a swap mask for every possible face angle present in the destination video.

A lack of images for some pose angles, will correspond to a failure swap for the concerned frames.

The *3D Model Construction* and *Images generation from 3D Model* blocks of image 3.3 are critical in order to generate some angle variance, starting from a single 2D image, which has none of it.

The key idea is to use the **starting source** image obtained from the *AI Face Generation phase or Face Retrieving* block and the `landmarks.txt` file produced in the *Landmarks Extraction* phase, in order to generate a 3D model of the source face.

To achieve this, `Deep3DFaceRecon_pytorch` [5] is used. This tool, based on the [5] paper, uses the facial landmarks detected in the *Landmarks Extraction* phase, in order to modify and manipulate a starting face model (**The Basel Face Model** [6]).

This step outputs an `.obj` file, which is intelligible with softwares like *Blender* or *Meshlab*.

After the *3D model construction* , the *Images generation from 3D model* block turns the `.obj` file, outputted by the former step, into a pool of images, which then can be used as input to `DeepFaceLab`.

This batch of images, is produced using *Blender*'s Python APIs, in the

`GIFandSpriteFromModel.py` script of FSA. The script achieves so, doing the following:

- Loads the `.obj` file
- Places multiple light sources, in order to obtain a realistic texture and albedo
- Moves the camera from side to side, from the top to the bottom of the loaded object, whilst taking snapshots of it, and saving them into `.pngs`.

This will result in a set of the `.obj`'s images from all the required angles.

This images are then moved in the `workspace/data_src/` folder in `DeepFaceLab`, where will be used to train the model that will perform the faceswap.

3.4 AI Face Generation or Face Retrieving

As previously said, the *AI Face Generation or Face Retrieving* block is composed itself by other blocks that are alternative to each other. Figure 3.4 shows a visual representation of the alternatives present in the block.

Figure 3.4: Possible alternatives in the AI Face Generation or Face Retrieving block. As show, as base 2D image can either be used StyleGAN 2, ThisPersonDoesNotExist, or any given 2D image.

For the automation purpose, FaceSwapAugmentation provides two equivalent methods to generate a 2D face image from scratch. They both use AI in order to generate the image. As shown in the diagram 3.4, these two AI powered methods are:

- **stylegan2-ada-pytorch** uses *Nvidia*'s stylegan2 neural network to generate a face picture. I choose to make stylegan2-ada-pytorch generate images based upon the `MetFaces DataSet`, which is a dataset containing several Met Museum human portraits.
- **ThisPersonDoesNotExist** is the second method used in FaceSwapAugmentation to generate face-images from scratch. It uses the first version of *Nvidia*'s StyleGAN.

Using **stylegan2-ada-pytorch** as it is, it's possible to generate images only by starting from protraits (as previously said). This ends up in some particular generated faces, a bit less realistic then desired. Both tools work quite well in generating faces, but ThisPersonDoesNotExist is preferred, since it generates real-life plausible images (closer to actual photographs), rather than painted portraits.

As an alternative to the former methods, any 2D image representing a human face can be used in this step.

> It's important to notice that this step is the key element that allows an unlimited number of face swaps, since every time it's used, a brand new face is generated.

It's also important to understand that this steps is not necessary if the user provides a source video to use, from which to extract the faces to apply to the base video.

3.5 Starting from an existing DataSet

As previously said and shown in pictures 3.1 and 3.3, what's required in order to use FaceSwapAugmentation, are two batches of images. One of them, is produced starting from a base video input by extracting its frames, and the other batch can be generated from a 2D image (given as input, or generated by AI).

FaceSwapAugmentation gives the possibility to bypass the *Images Generation* block, and to directly receive as input a folder containing the source's images.

For this thesis' purposes, the KDEF [4] was used and it is currently suggested as benchmark for already-existing datasets, since it the two main features required in order to achieve good face swaps:

- Face angle variace
- Subject's emotional variance

The dataset of pictures contains 70 individuals, displaying 7 different emotional expressions. Each expression is viewed from 5 different angles. Thus, on folder of the dataset contains 35 images of one single individual, so one folder contains enough data to create a face swap.

In the figure 3.5 it's shown how FaceSwapAugmentation is used in comparison with the usage done when starting from a base 2D image.

3.6 Starting from a source's base video

Another way to skip the *Image Generation* block, would be to start from an existing video containing the source's face frames. The source's video can also be directly fed into FaceSwapAugmentation as input along with the base video containing the destination's face frames. As show in figure 3.6, in this scenario, we'll also need to run one additional script, `2_extract_image_from_data_src.sh`, in order to generate source's images staring from the input video.

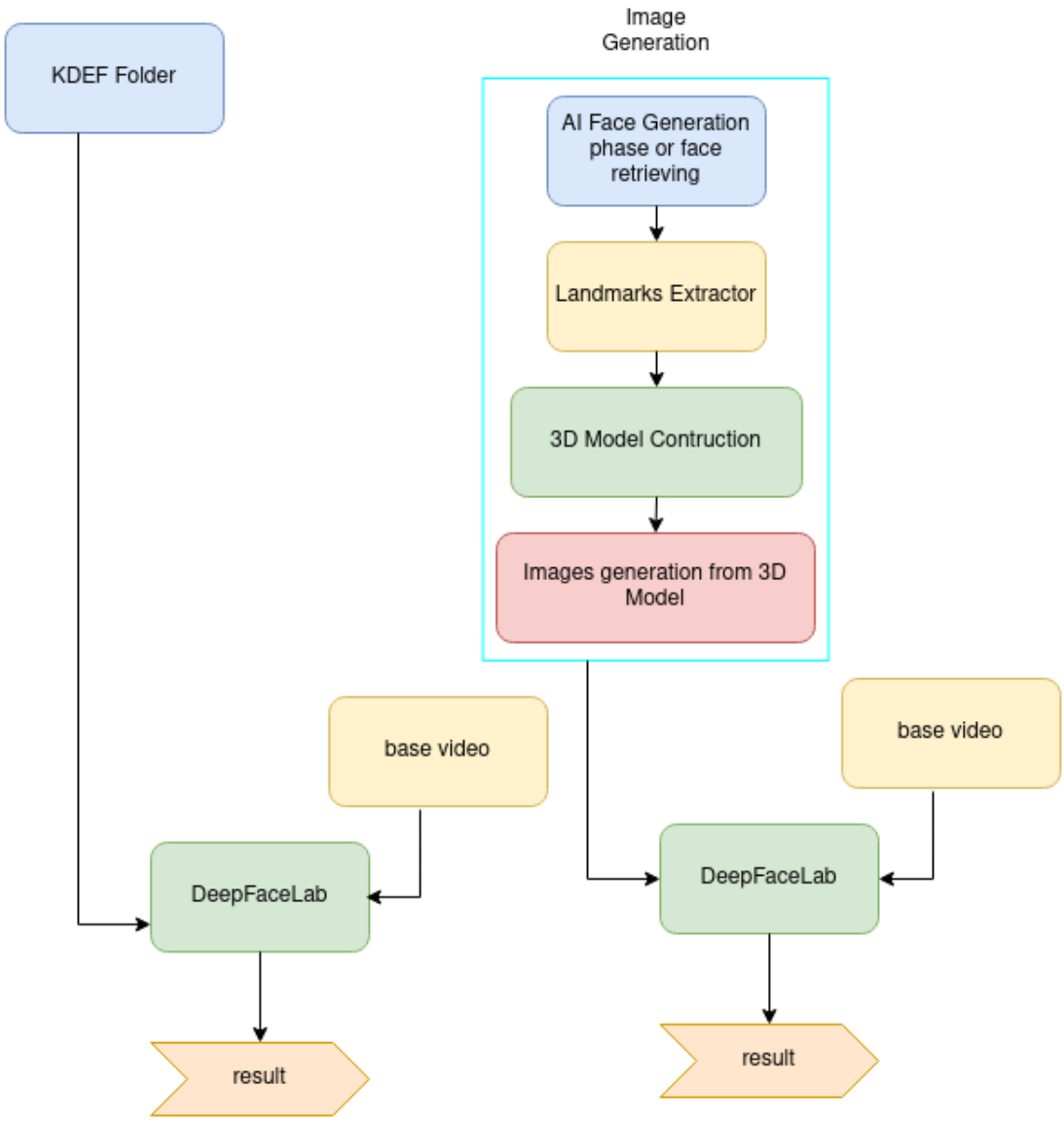

Figure 3.5: Comparison between the usage that involves the image generation, and the usage starting with a KDEF folder, representing an individual, given as input.

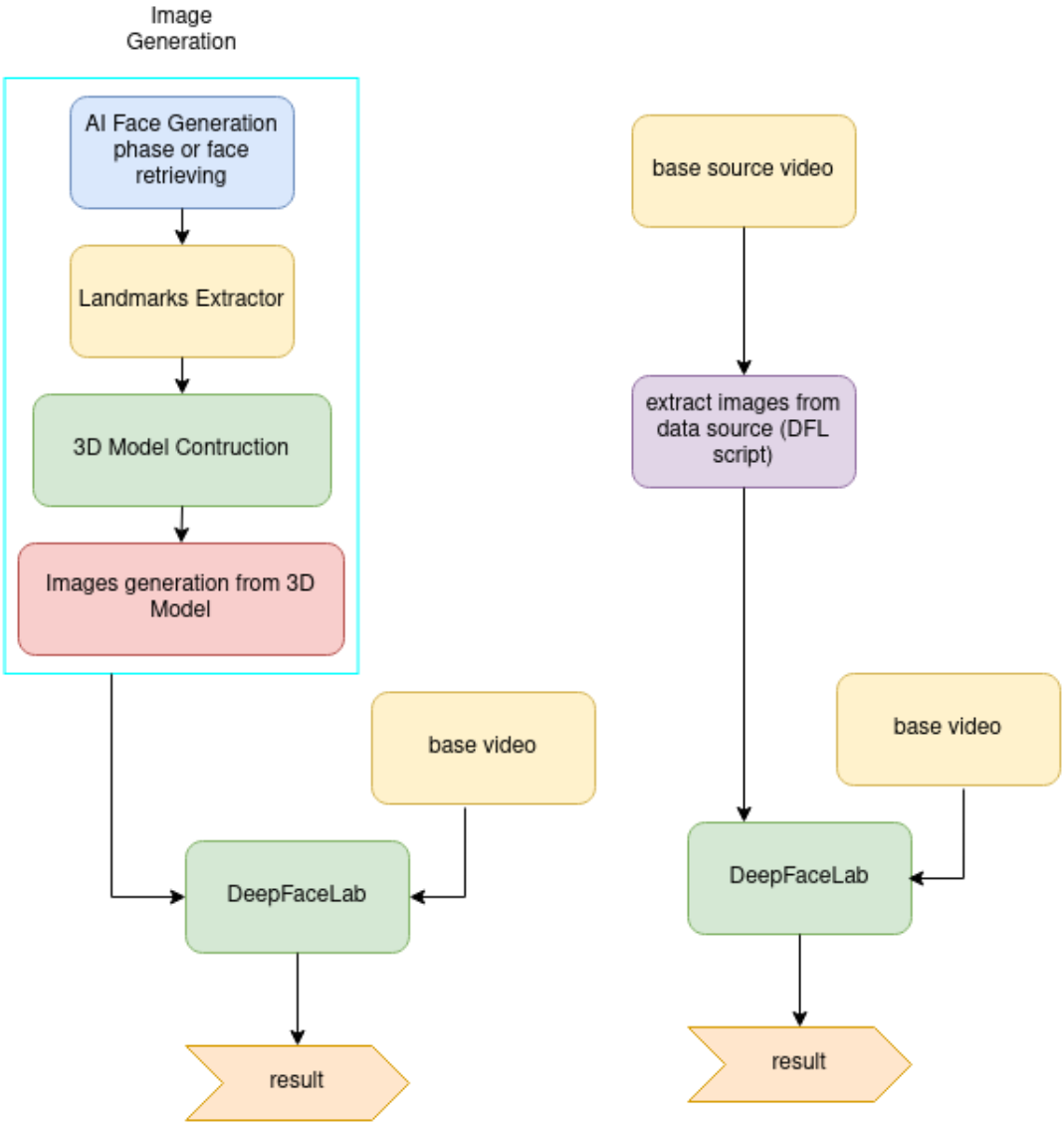

Figure 3.6: Comparison between the usage that involves the image generation, and the usage with the source's video given as input. In this case, it's required to run an additional script of DeepFaceLab. (script 2), in order to break the given video into frames.

Chapter 4

FaceSwapAugmentation Evaluation

Since FSA is build for research purpouses, rather than a single way of use, it provides several workflows. This was done both to make an analysis between different images generation or retrieving, and also to publish a more versatile tool. 5 different workflows are available, and in this chapter is presented a qualitative and quantitative comparison between them, in order establish which one outputs the best resulting video.

4.1 Flows analysis

4.1.1 Flow 1 - StyleGAN 2

The first way of use generates the source image from scratch and requires only the base video as input. As shown in figure 3.3, the image generation step of figure 3.1 is composed itself by several other steps, first of them being the AI Face Generation or Retrieving. This steps are more deeply represented in the 3.4 diagram, which shows that the starting images can be AI generated, using ThisPersonDoesNotExist or StyleGAN 2, but can also be directly provided as inputs.

The flow explained in this section will start from generating a face using StyleGAN 2. This tool creates a `.png` image representing a portrait, based upon the MetFaces dataset. Once the image generation is complete, we need - as previously explained - to add some angle variance to it. Since the generated image is 2-dimensional, this task is anything but trivial.

The main idea is to use a Morphable 3D Face Model [6], in this case the Basel Face Model, and to use *Deep3DFaceReconstructor* [5] in order to manipulate the Face Model such that it replicates the albedo and facial traits of the image generated by StyleGAN, but in a three-dimensional environment.

Before directly feeding the `.png` image into *Deep3DFaceReconstructor*, another step is required: we need to extract some landmarks from the starting image since they're required in order to generate the 3D model. To do so, we wrote a script based upon [3]'s python library, which allowed the extraction of facial landmarks from a given 2D image, just specifying which facial elements were relevant.

Both the AI generated image, and the extracted landmarks were then sufficient as input for *Deep3DFaceReconstruction*, in order to build a 3D model of the input picture.

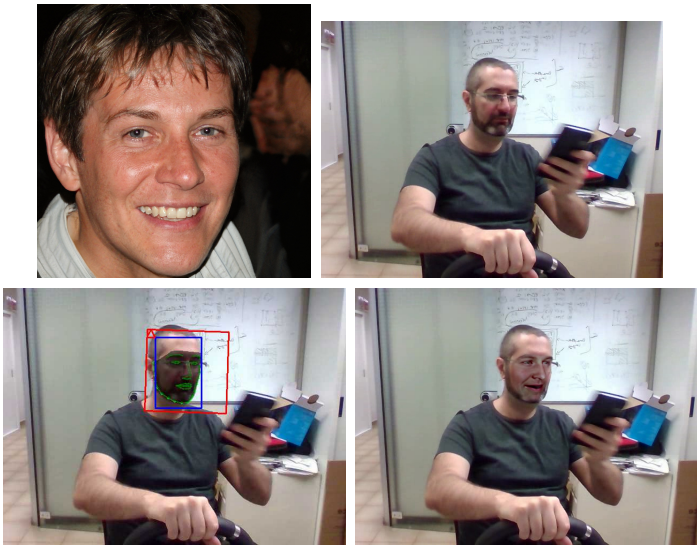

Figure 4.1: Face swap done with ThisPersonDoesNotExists, flow 2.

Once generated the 3D face model, we use Blender, and, in particular, its APIs in order to extract and save snapshots of the model from different prospective. It's important to remember that we need as many angled pictures of the model in order to achive a good face swap.

Once the pool of images is generated from the 3D model, it can be directly fed into DeepFaceLab, along with the base video, and the previously described scripts of the tool will be run. This will ouput the desired deepfake video.

4.1.2 Flow 2 and 3 - ThisPersonDoesNotExist and Provided 2D Image

These two workflows are very similar to the first workflow, the only difference being the source of the image. Using ThisPersonDoesNotExist as AI face generator is the suggested choice when using FSA without providing an input image, since it generates more realistic people, but with the downside of occasionally creating visual artifacts. Starting from a given 2D image can be the best choice when the deepfake is requested but no source videos are available. The steps involved in this scenario are similar to the previously explained workflows,with the only change being the source of the image. All the steps of *Landmarks extraction* and *3D model construction* are still required, since no 2D image can have angle variance.

In figure 4.1 is presented the comparison between the frames in the different stages of the flow, also showing how the landmarks are represented on the frame, and how the alignment squares are set. In this case is clear how the color affects the results of the swap.

Figure 4.2 shows a visual representation of the `.obj` file generated from the source image given in the flows 1, 2 and 3. It's a series of snapshots - in total are 77, there are only 3 reported here - that will then be fed into DeepFaceLab as source images.

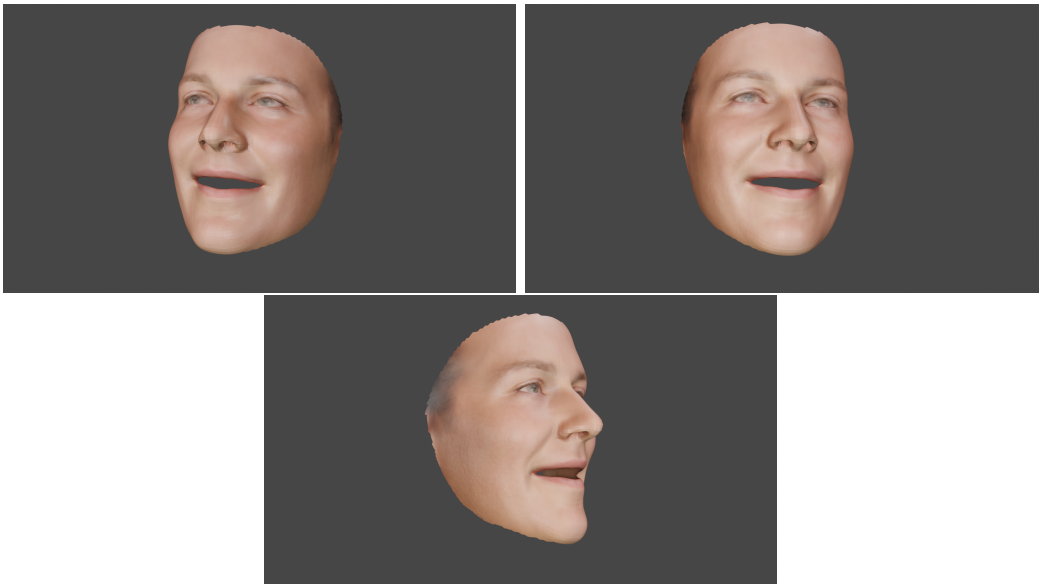

Figure 4.2: Snapshots of the obj file created from the AI-generated source image at different angles. This is the way in which angle variance is generated when starting from a 2D image.

4.1.3 Flow 4 - KDEF

As an alternative to face generation or retrieving phase, KDEF [4] can be used. The dataset of pictures contains 70 individuals, displaying 7 different emotional expressions. Each expression is viewed from 5 different angles. Thus, one folder of the dataset contains 35 images of one single individual, so one folder contains enough data to create a face swap. More importantly, each folder contains enough angle and expression variation to make a *good* face swap.

KDEF allows us to skip all the processes in which, starting from a 2D image, we created a 3D model of the face, and then we captured snapshots of it from several angles.

As shown and said before, when the flow starts by taking as input one KDEF folder, all the previous steps can be skipped, and feed the 35 images of the folder directly into DeepFaceLab. It's important to notice that starting from KDEF generally produces a better result video: this is due to the fact that, when starting from a 2D image, we managed to successfully create angle variance to it, but not any kind of emotional variance or facial expression. As previously said, the latter element is just as important as the former one in the quality of the produced deepfake.

These two qualities make `KDEF` a good choice for this paper's purpouse, but any other dataset with the same characteristics can proficiently be used.

Figure 4.3 shows an example of face substitution done starting with a KDEF subject - BM29 in this case. What's clear is the fact that the emotional variance plays a big role in the outcoming quality of the swap.

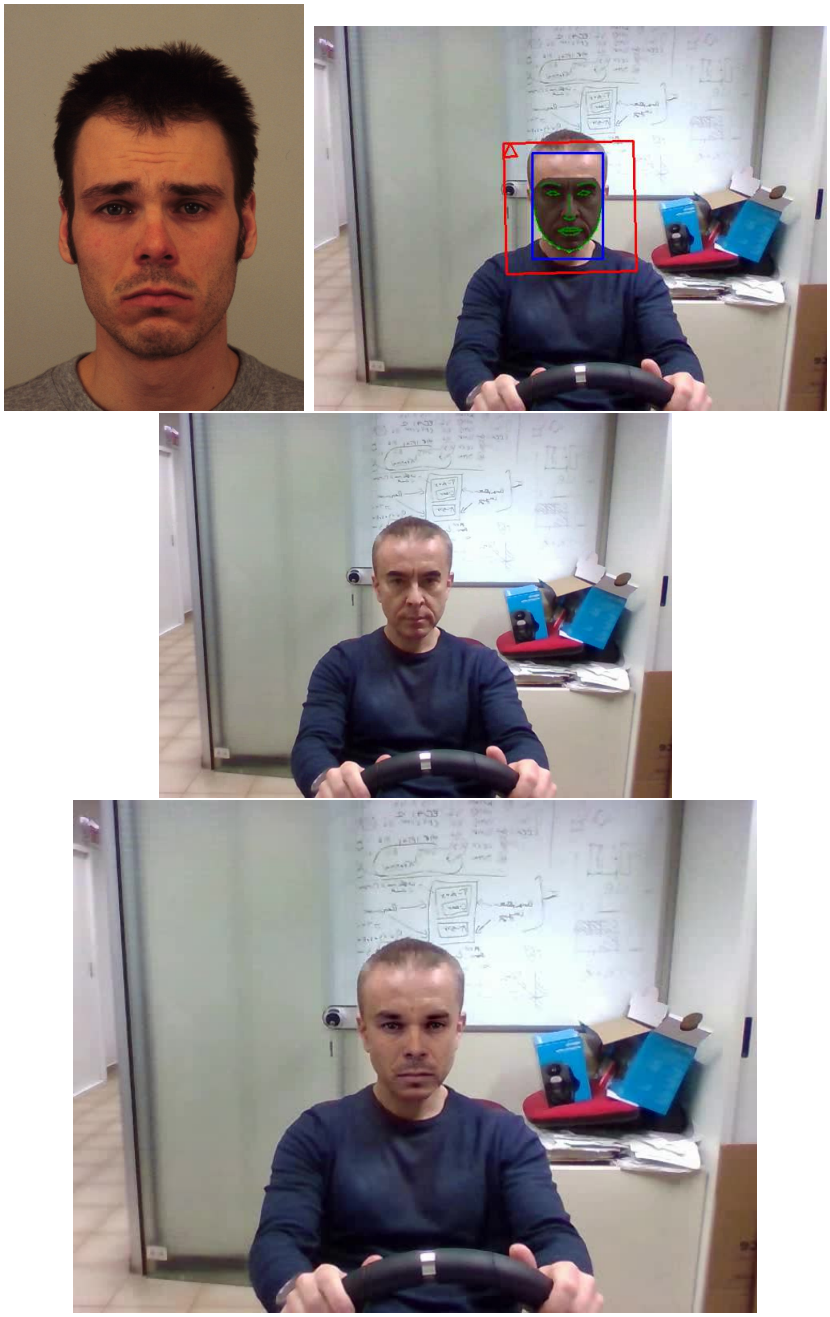

Figure 4.3: An example of KDEF swap

4.1.4 Flow 5 - Input Source Video

It is possible to give as input to FSA two videos, and directly swap the face of the `data_src.mp4` video onto the `data_dat.mp4`, skipping all the pipeline steps but the last one.

In this case, is required to run the following scripts of DeepFaceLab:

- `2_extract_image_from_data_src.sh`, which wasn't needed when source was directly fed as images.
- `3_extract_image_from_data_dst.sh`
- `4_data_src_extract_faces_S3FD.sh`
- `5_data_dst_extract_faces_S3FD.sh`
- `6_train_SAEHD_no_preview.sh`
- `7_merge_SAEHD.sh`
- `8_merged_to_mp4.sh`

In particular, the script `2_extract_image_from_data_src.sh` is needed in order to break down the input source footage into frames, that will then be used as previously done, starting with the face recognition phase, followed by the alignment and merging.

The only difference between using FSA and directly DeepFaceLab, in this workflow, is the automation, which is key for FSA, but not for DeepFaceLab, that requires human interaction several times.

4.2 Evaluation Methodology

In order to have a better grasp on the performance, each augmentation method (*stylegan2*, *ThisPersonDoesNotExist*, *KDEF DataSet* or directly from a source video) is evaluated, with a metric that takes account of the similarity between the input frames belonging to destination video and the source video.

This evaluation is necessary to understand and better classify each one of the generation flows, not only by judging the resulting videos qualitatively, but also by assigning to them an objective score. It's important to underline that the applied scoring paradigm is not unique, and arguably not the best.

4.2.1 Frames Evaluation Method

In addition to the creation of the face-swapped video, this pipeline also provides an evaluation method for the face substitution. The score assigned to each face-swapped frame is the evaluation of the outcoming quality, based on how close the destination and the source frames are.

To do so, I took advantage of the images in the `data_dst/aligned` and

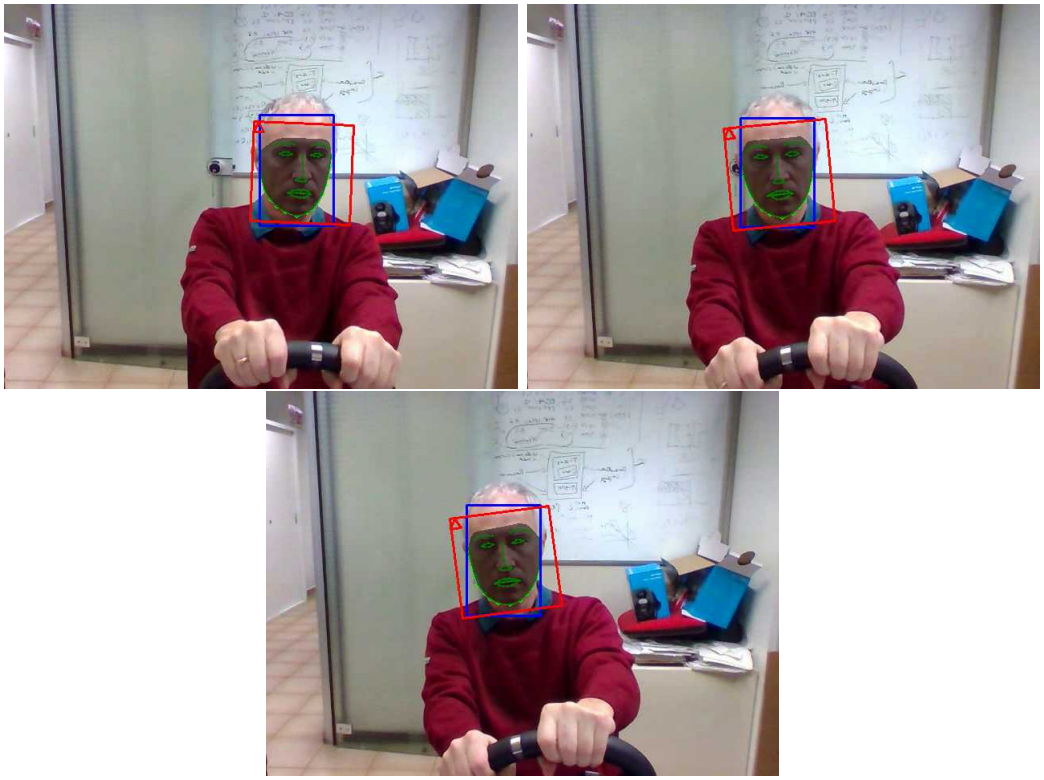

Figure 4.4: Landmarks detection in aligned images

`data_src/aligned` folders, both containing the input frames in which the face detector - `S3FD` in this case - has recognised faces.

The process of face detection is performed in the steps

`4_data_src_extract_faces_S3FD.sh` and `5_data_dst_extract_faces_S3FD.sh` of DeepFaceLab. For each face detection, the landmarks of the face are written in the metadata of the corrisponding image (that will be then put inside the `aligned` folder). To do so, a proprietary library called `DFLIMG` is used, which provides methods for reading and writing metadata in the frames `pngs`. Figure 4.4 shows images contained in the `aligned` folder of *DeepFaceLab*, that present themselves with landmarks upon specific facial elements. Both of the squares in the picture show the alignment criteria of the images. This is necessary in order to examine all the faces with the same yaw angle. In order to evaluate the face swap, the facial landmarks were extracted for each source and destination image.

The landmarks are structured and saved as a two dimensional array of floating point numbers. Each corresponding pair of values in the two dimensional array, represents respectively an x and a y value of a single landmark point.

For each destination-image's landmark, the distance with all the source-images' landmarks is calculated. The evaluation score of each pair is the sum of the distances between corresponding landmarks' points. We compute the Euclidean distance of each destination's image landmark point, with the corrisponding source's image landmark point. Then we sum up the distances of each image pair (source and dest), resulting

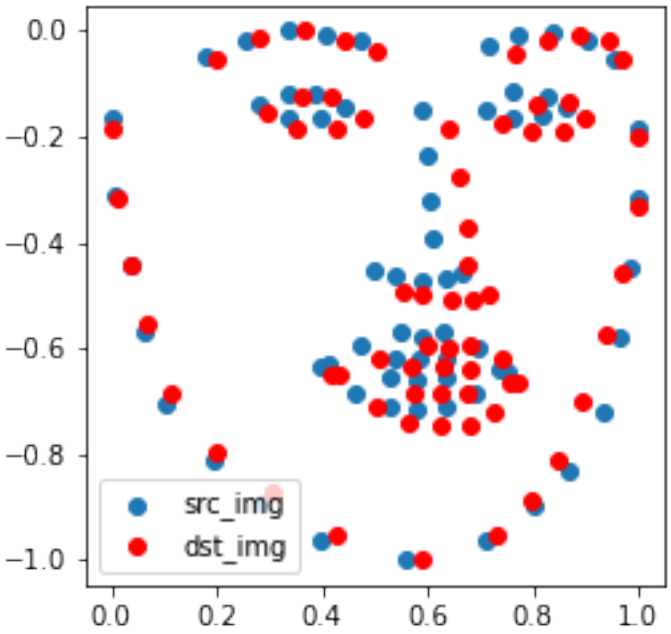

Figure 4.5: Landmarks normalization and overlapping

in the score of the swapped frame.

$$\sum_{i=0}^{n} euclidean_{distance}(p_{src}, p_{dst}) \tag{4.1}$$

Where n is the number of landmark points of a single image, and p_{src} and p_{dst} are two corresponding landmark points of the two images.

Figure 4.5 shows a visual representation of the landmarks in a plot. As shown, in the same plot are present both the source image's landmarks and the destination image's landmarks. Qualitatively speaking, the ideal scenario would be to have all the dots of the plot to perfectly overlap each other, meaning that the two images are perfectly similar, and thus ideal to be swapped. This would correspond to a score of 0 - since it's the distance we're talking, the smaller the better.

The 4.6 figure shows a graphical representation of the distances considered in the metric - the red segments - that will be summed together in order to calculate the score for this specific frames pair.

As a quality threshold, we take account only of the frames which score falls inside the 95th percentile.

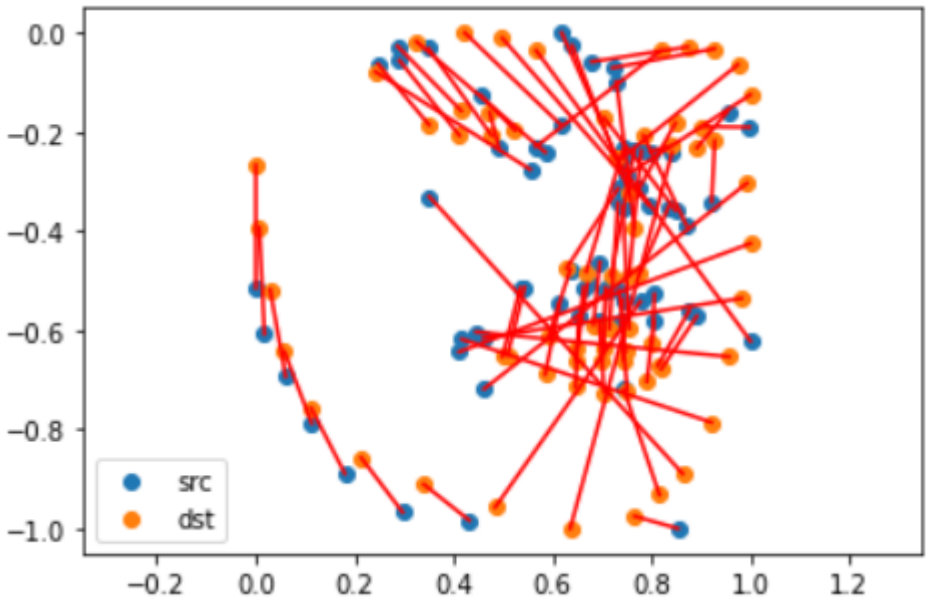

Figure 4.6: Visual representation of the distances between corresponding landmarks' dots

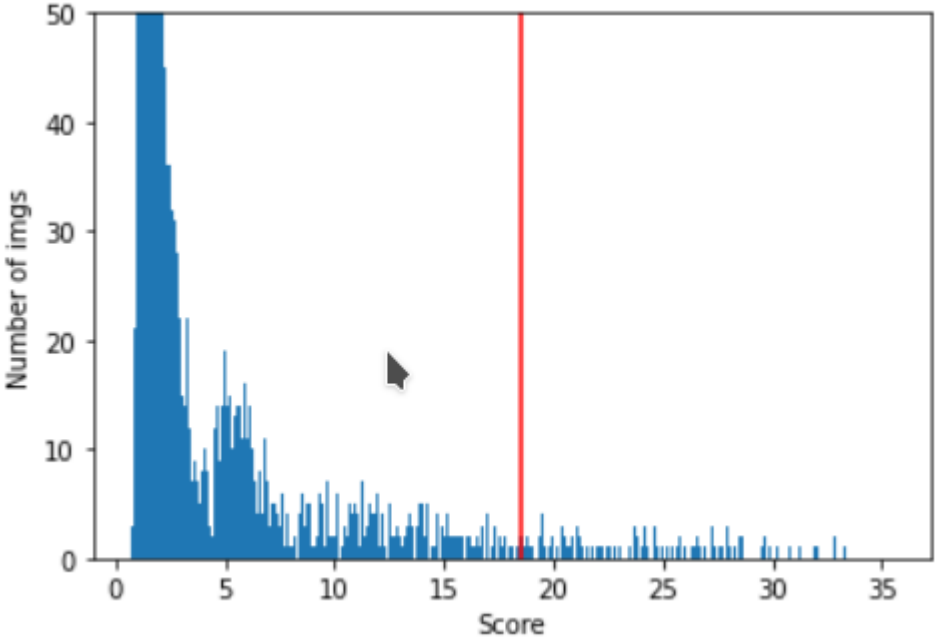

Figure 4.7: Distances distribution among swapped frames

4.2.2 Flows evaluation and comparison

As explained in the FaceSwapAugmentation chapter, since this project is developed for research purpouses, it is designed to provide several different workflows. It's reasonable to compare the different results generated in each flow, by giving them a score, similarly as done for the frames. First of all, in order to compare the different flows, it's necessary to decide which method to use in order to see which one creates the best results. This by itself is a non-trivial task, since ideally we want an objective score that suits good the qualitative results. A good evaluation method for one single workflow should be based upon the score described in the previous section, since it's strongly related to the overall score given to the frames of a deepfake. For each face face swap, the value of the 95th percentile is saved, representing a good metric of the overall deepfake: the smaller it is, the bigger would be the number of frames within a small score. For each flow, we measured the 95th percentile of several face swaps, the idea being to use the distributions of the percentiles among different the methods, in order to compare the different workflows. A first, but imprecise solution, would be to calculate the average scoring of each flow, and then use that for the comparison between workflows. This, however, would cause some information loss and give spurious results, due to the fact that the substitutions done with each method are not enough to give a realistic average. From one face swap to another - made with the same flow - the scoring can variate up to 10-12 points, meaning that one single swap can strongly shift the average value.

A better idea would be to plot out the distrubution of each flow, then generate a Cumulative Distribution Function (CDF), and directly compare the CDFs of the workflows. This would show in an intuitive and graphical way how each flow behaves, and whether there are or not anomalous situations, in which some method works better than another, but isn't better in general.

Before doing so, it's good practice to check whether the empirical distributions of frames-scoring - belonging to each deepfake - follows a theoretical distribution. In this case, the two most likely theoretical distributions to fit, were the lognormal and the bimodal distribution. Figure 4.8 shows an example of a lognormal distribution, characterized by a high peak right after the x axis zero, and then asimptotically converging towards the y axis zero.

Figure 4.9 shows an example of bimodal theoretical distribution, similar to a lognormal, but characterized by two peaks instead of one.

Both were chosen as plausible theoretical distribution functions because of two main reasons:

- In the best case scenario, the distance between source image's landmarks and destination image's landmarks is zero. In the worst case, the distance goes towards the $+\infty$. No cases could have gotten a negative scoring whatsoever, so a normal distribution was not plausible.

- In addition to that, was reasonable to think that most of the scores would have fallen close to the zero, and a exponential diminishing number of frames would have gotten a worse score.

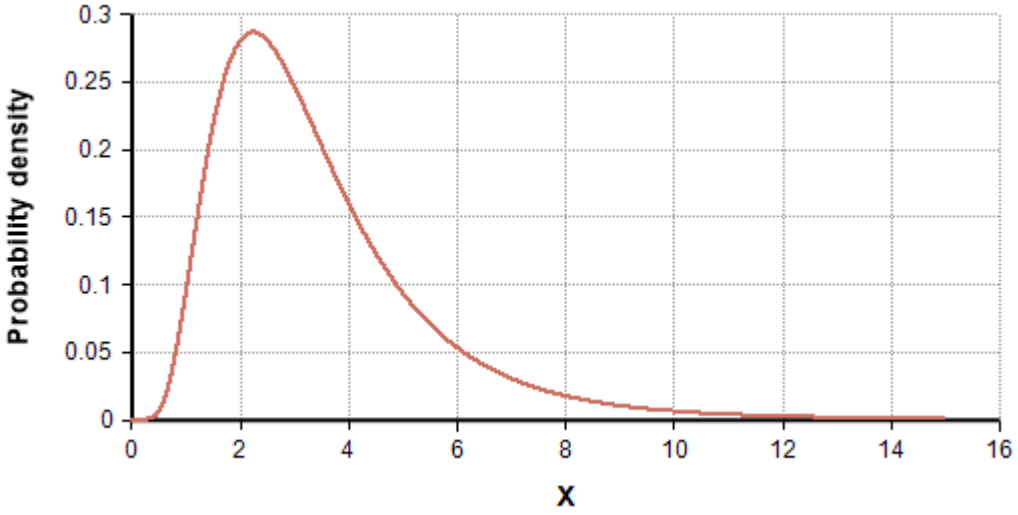

Figure 4.8: Example of lognormal distribution

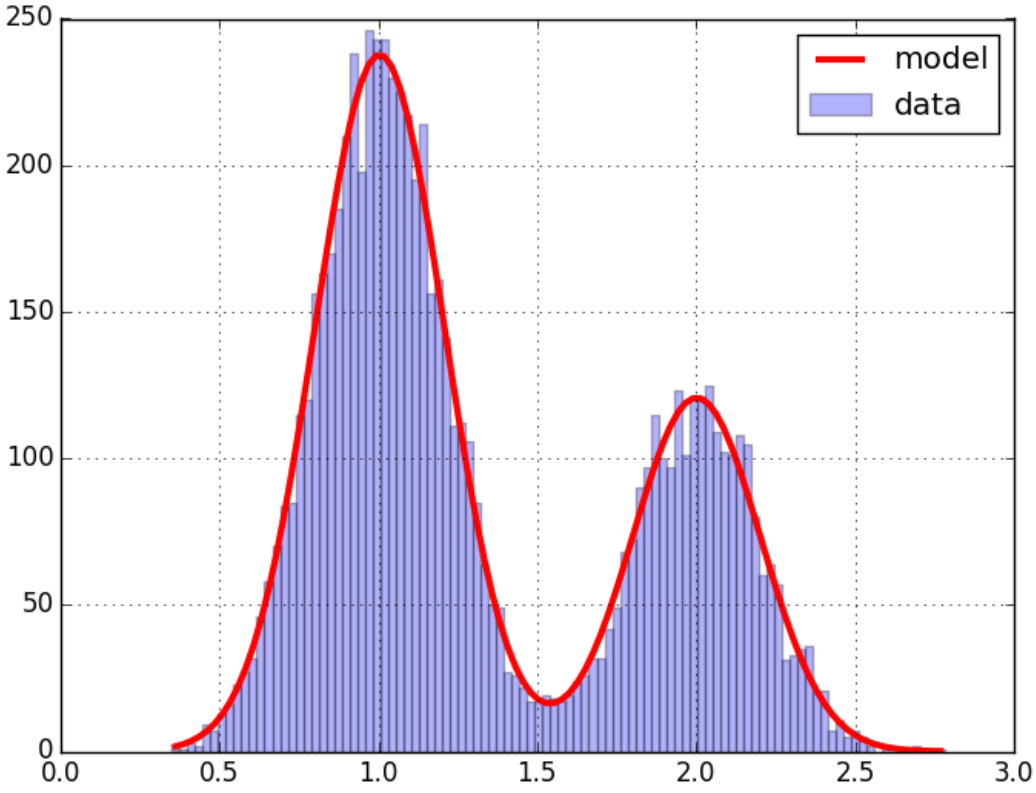

Figure 4.9: Example of bimodal distribution

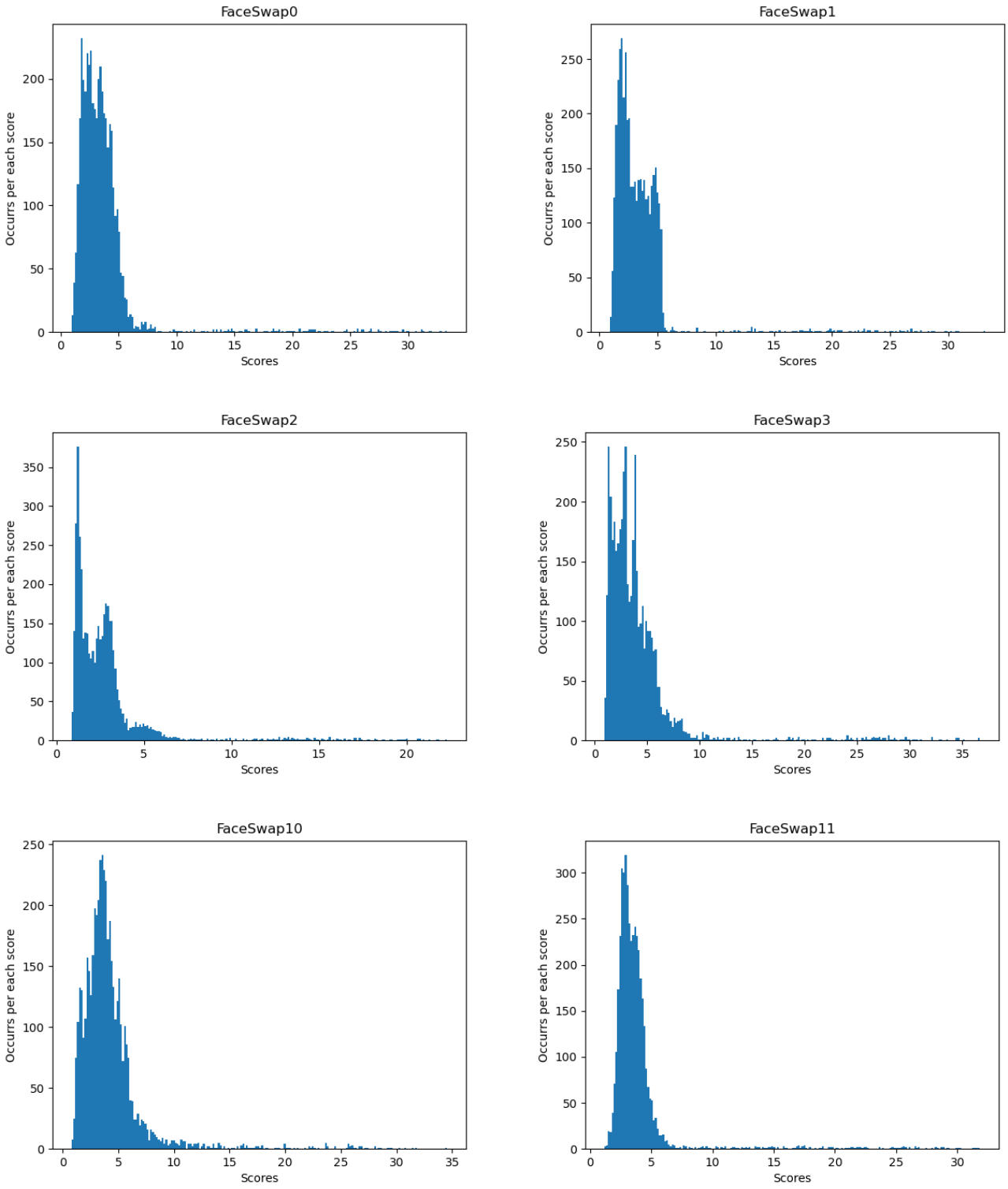

Figure 4.10: Some empirical distributions

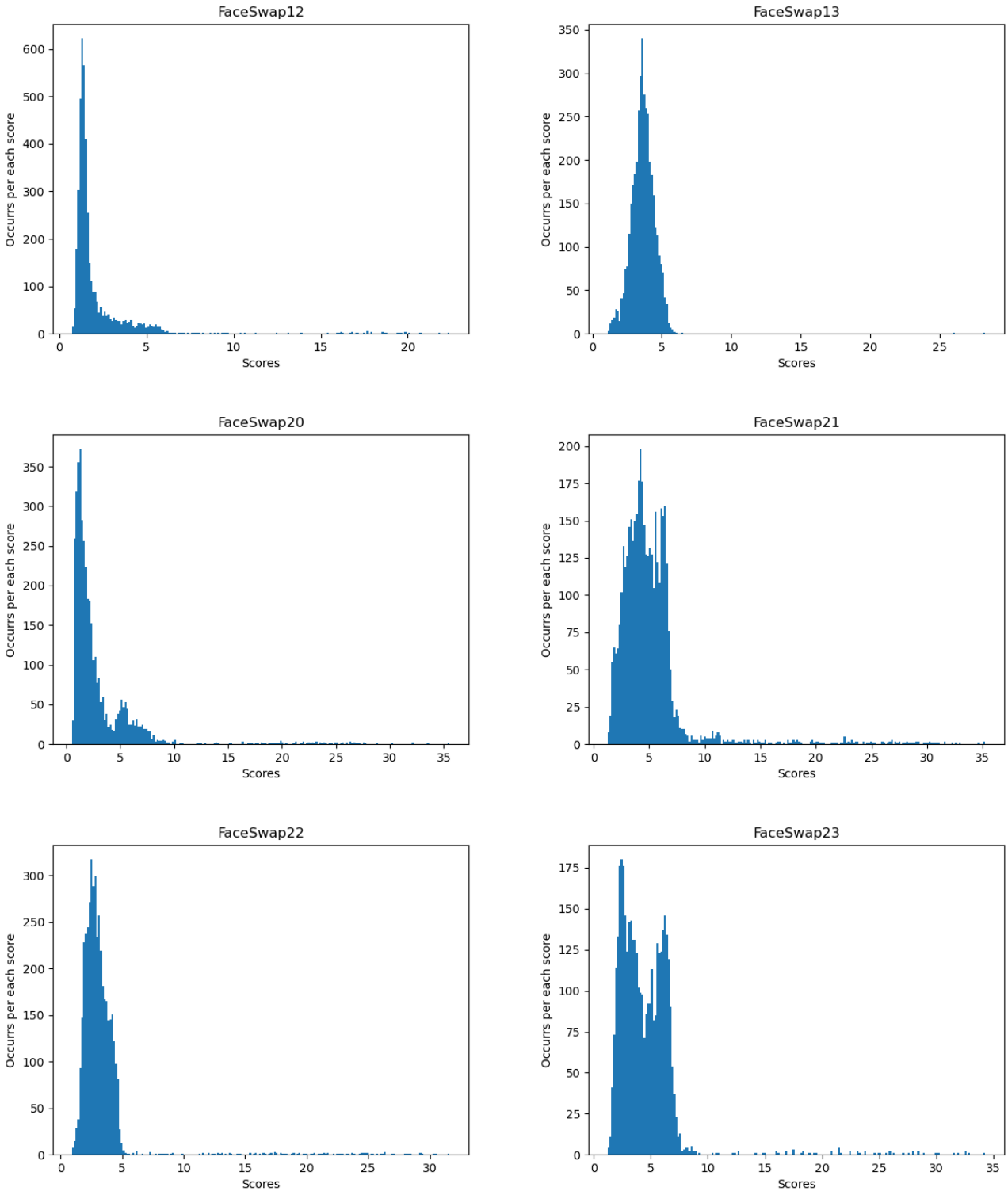

Figure 4.11: Some empirical distributions

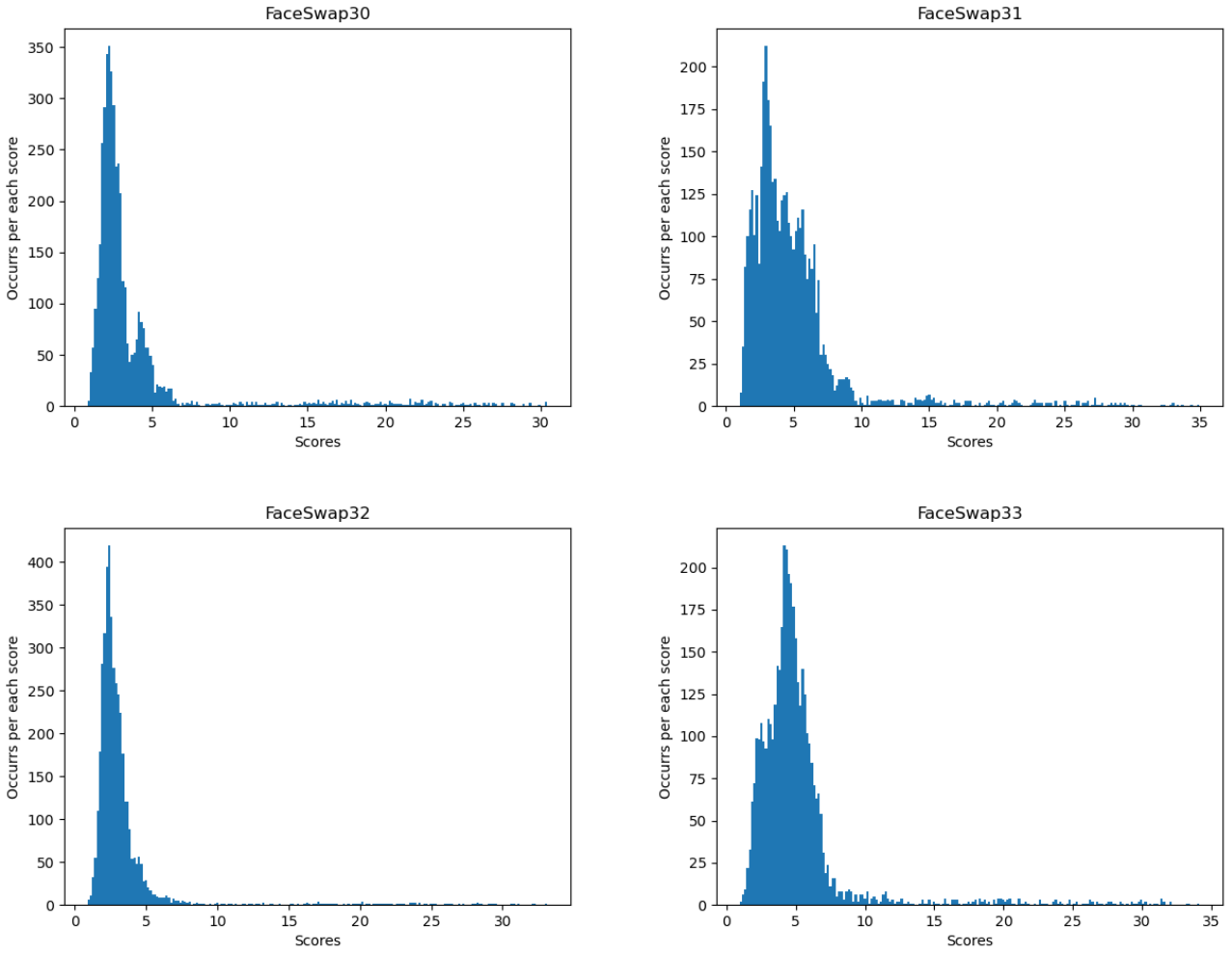

Figure 4.12: Some empirical distributions

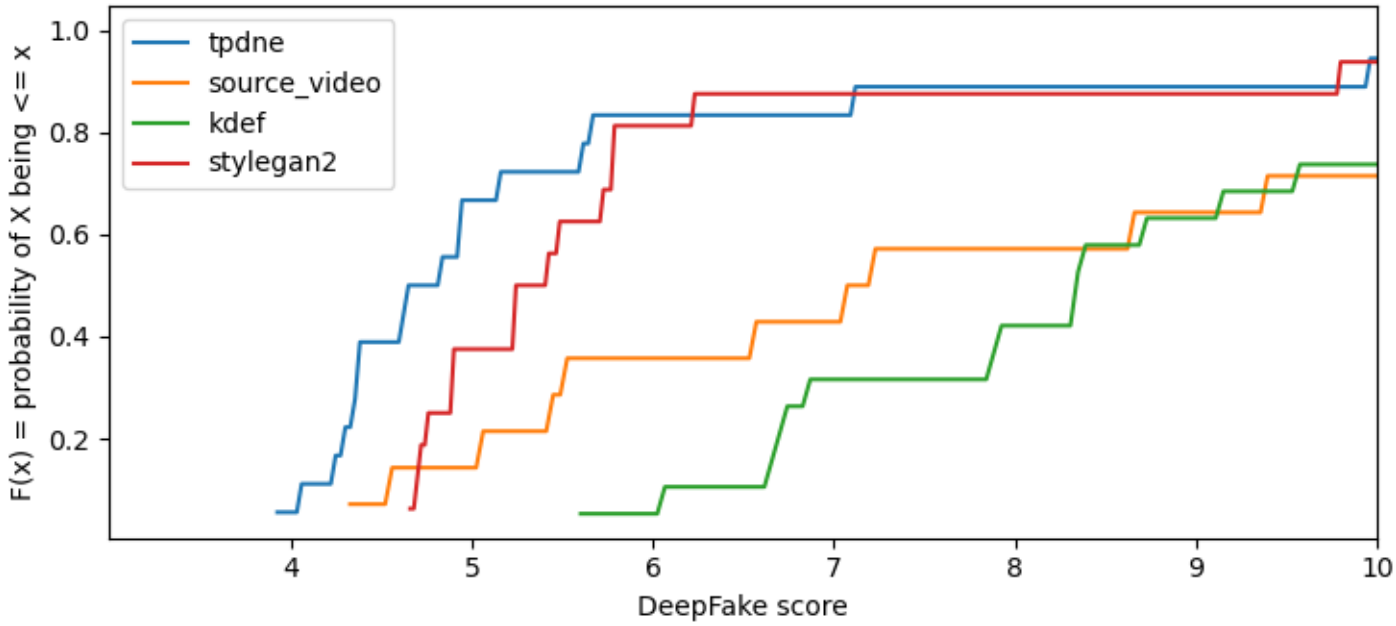

Figure 4.13: Cumulative Density Function of the distributions of the given score, for each faceSwapAugmentation usage flow

Figure 4.10 shows some empirical distributions, resulting of the workflows evaluation process. It would seem that the assumptions previously made were close to the results, in particular the second assumption upon the distribution structure. To what concerns the first assumption, instead, most of the distribution fit quite good a lognormal or a bimodal, but some of them are closer to a trimodal or spurious distributions.

4.2.3 Comparison Outcomes

For each of the proposed flows, 30 deepfakes were created. For each one of them, we saved the scoring corresponding to the 95th percentile, considering it a possible general evaluation for the considered swap.

Every single flow has then a corresponding distribution of scores, depending on the quality of the generated flows. A cumulative density function was plotted for each flow, in order to qualitatively see which one performed the best.

Figure 4.13 shows the CDF distribution of the different flows.

Cumulative Density Function

The Cumulative Density Function of a distribution plots out on the x axis the possible scores of the distribution, and on the y axis the probability of a random value being less or equal than the value on the x axis.

The CDF distributions show that that, using the evaluation method based upon the Euclidean distance, the best results are produced by ThisPersonDoesNotExist (Style-

gan), closely followed by Sytlegan 2.

Chapter 5

Outcomes on the model performance

In addition to build an augmentation tool - that is FaceSwapAugmentation - this thesis also tries to investigate on the possible effects of using such tool on the performance of classifying models. The strategy idea used to judge the quality of the generated dataset is to employ it to train the classificator model, and then compare the new performance with:

- The the accuracy of the model before the new phase of training (the performance described in [1]), and
- the performance of the model trained on the original dataset, but using 20 images per-subject rather than just 10, as done in the first training phase.

If the model trained with the new augmented dataset does worse than the model trained with 20 images per subject, this would mean that training with real people is still a better solution for the performance and accuracy matter, but may not be the best solution overall. It's possible to better evaluate the outcome based upon the additional following elements:

- How much worse is the model trained on the augmented data in comparison to the 20-images-trained one
- How much money were saved using the augmented dataset
- How much time was saved using the augmented dataset
- How close is the model trained on the augmented dataset to achieve a better performance than the 20-images-trained one
- How much more scalable is FSA than real data generation

5.1 Methodology

To make a comparison between the real-data-trained and the deepfakes-trained classificators, we used the 15-classes configuration (of the three that are studied and described in [1]). The 15 classes are reported in table 5.1. The analyzed data to make a comparison is gathered from the following two setups:

Table 5.1: Action class definitions for the [1] dataset

Class	Name	Description
C00	Safe driving	The subject simulates a normal, safe driving, looking straight at the road or at the mirrors.
C01	Texting right	The subject holds the cellphone in her/his right hand, and is writing a text message.
C02	Talk phone right	The subject holds the cell phone in her/his right hand, and is performing a phone call.
C03	Texting left	Same as C01 with the left hand.
C04	Talk phone left	Same as C02 with the left hand.
C05	Operate radio	The subject gaze is directed at the car radio, and she/he plays with it using the right hand.
C06	Drinking	The subject is drinking (or about to drink) from a bottle/glass using one hand.
C07	Look behind	The driver is stretching to grasp some object on the back seat.
C08	Hair & makeup	The subject is adjusting her/his hair or doing some makeup.
C09	Talk passenger right	The driver is looking at her/his right and talking.
C10	Talk passenger left	Same as C09 on the left.
C11	Look below	The driver is stretching to grasp something on the car floor.
C12	Look right	The driver is looking at her/his right, but she/he is not talking.
C13	Look left	Same as C12 on the right.
C14	Yawn	The driver is yawning, perhaps inattentive to the driving task.

1. Training done with real data, and tested on real data. This corresponds to the result produced in [1] - T1 setup and configuration 15, that consists in training and testing the model with unmasked subjects

2. Training done with deepfakes, tested on deepfakes.

The model chosen for this task is a classificator that is trained in order to label the actions of a driving person. To this purpouse, the CNN model used to perform the classification task [1] is based upon the state-of-the-art EfficientNet-B1 architecture.

The classificator gains its knowledge through transfer learning from another model pre-trained upon the ImageNet dataset. After that, the classificator gets fine tuned against the MaskDar dataset.

This approach allows a fast model convergence and high classification accuracy [1]. The so called *leave-one-out* process is employed in the cross-validation, in order to avoid biased results. Each set is composed of 30 individuals: with the *leave one out* technique, 29 instances are used as training set, and the left-out one is used for testing. This process is iterated multiple times, taking into account the performance average.

Each test is done after a 50 training epochs, using batches of 32 images.

For each experiment we generate a confusion matrix, that reports the number of istances classified for each action.

Confusion Matrix

It consists of a squared matrix in which each row corresponds to the real class of an instance, and each column is the model-predicted class of the instance. In each cell it's reported che number of instances whom's real class is the row of the cell, and the predicted class is the column of the cell.
The ideal outcome would be to have all the diagonal cells as the most populated.

The performance metrics employed are:

$$Precision = \frac{TP}{TP + FP}, \qquad Recall = \frac{TP}{TP + FN}$$

$$F1\text{-}score = 2 \cdot \frac{Precision * Recall}{Precision + Recall}$$

The score of the model on the single experiment is given by $\mathbb{E}$[F1], i.e. the mean F1-score value over all the classes.

5.2 Augmented Dataset structure and construction

A dataset of 30 videos was produced by using FaceSwapAugmentation:

- 15 videos were generated using the pipeline starting from *ThisPersonDoesNotExist*
- 15 videos were generated using existing videos as sources

On average, each deepfake requires around two to three hours to be generated on an istance of the HPC4AI cluster, belonging to the Università degli Studi di Torino. The istance is equipped with: 32GB of memory, an `Nvidia Tesla T4` and an `Intel Xeon Skylake IBRS`

The dataset is then used to train and test the classificator model, organizing the experiment in the three previously described setups, to see whether the artificial dataset is useful, other than feasible.

5.3 Outcomes

5.3.1 Setup 1 - Real data only

In this setup, the results are the one described as baseline in [1], and are reported in the 5.1 figure.

The classificator's performance in the 15 classes configuration resulted in an average F1-score of 0.76. The overall performance is quite good, the model correctly classifies most of the samples. The confusion matrix also shows that some classes are hard to distinguish, such as C10 (*Talk passenger left*) and C13 (*Look left*), since the data feed consists of separated images rather than a sequence - so most of the *talk passenger left* images are very similar to *look left* images [1].

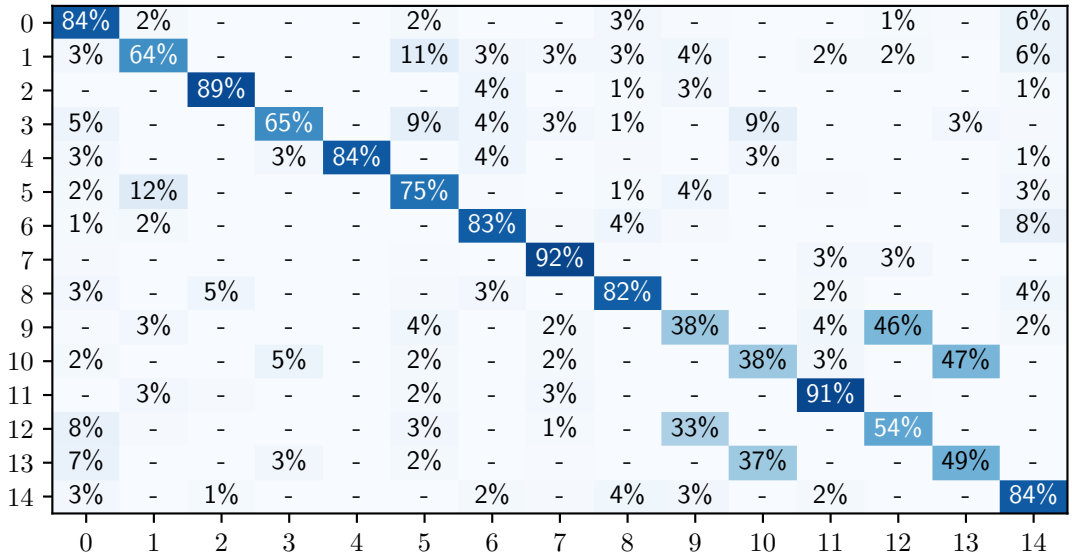

Figure 5.1: Confusion Matrix of the first experimental setup, in which the model is trained upon real data, and tested on real data.

5.3.2 Setup 2 - Deepfakes only

The second setup included only artificial generated data in the training and in the testing phases. The results in this case show an F1-score of 0.70, and an accuracy of 0.74. As reported in table 5.2, the results are not very far from the setup 1. It has a pretty decent overall score too, but not as good as the setup 1. As inferable from the 5.2 confusion matrix, the model has some trouble distinuguish C03 (*texting left*) from C10 (*talk to passenger left*), but also C04 (*talk phone left*) from C11 (*Look Below*).

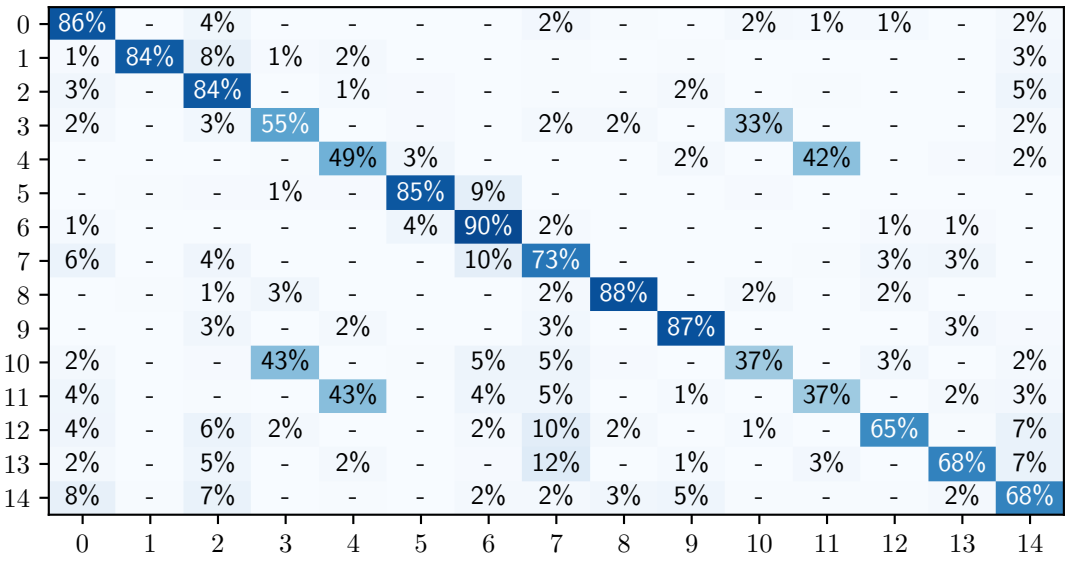

Figure 5.2: Confusion Matrix of the second experimental setup, in which the model is trained upon deepfakes, and tested upon deepfakes.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

6.2 Future Work

Notwithstanding the performance of FaceSwapAugmentation, there are number of aspects of it that can be improved. For some of them, a fix is already possible with the current state of the art, and were not done in this thesis due to a matter of time.

Other issues need some deep research and work, in order to attempt some kind of patch/solution to the matter.

6.2.1 Possible improving of the DeepFake scoring method

The deepfakes scoring method adopted in this thesis is certainly objective, but several improvements can still be made. The possibility to choose a better scoring criteria is explicitly made clear by the fact that the scores given to each faceswap (and thus, to each pipeline workflow) were not always coherent to the qualitative result. The deepfakes that were assigned the best score were almost always not the most realistic ones. Similarly, the deepfakes that looked the best and closer to actual humans, generally did not receive a very good score, but rather a pretty average one.

The issue of this scoring method is that it's solely based upon the distance between the landmarks, not taking into account:

- The color delta between the source image and the destination image
- The intensity of the light of the source and the destination image
- The presence or absence of any emotional variance

It's also strictly dependant on the number and the granularity of the landmarks themselves: this can result in a more or less accurate reproduction and synthesis of facial elements such as eyes, mouth and nose.

A more accurate evaluation must take into consideration, in addition to the used score, a method to evaluate each of the above-discussed points.

An objective scoring for the colors delta of two faces can be based upon the distribution of colors in a segmented face. Each color is commonly coded with an RGB number. It's possible to count how many pixels are of a certain color, and then plot out a color distribution. The distance between the distributions computed for both source and destination faces, can then be considered in addition to the landmarks distance. The same can be done for the light intensity.

Regarding the presence or absence of emotional variance, the issue will be thoroughly discussed in a following section.

6.2.2 Masks and protection devices

Whilst this thesis is being written, Covid-19 world pandemic is still taking place, meaning that in most parts of the world it's mandatory to wear some sort of face cover. This kinds of protection devices can cause, as shown in [1], some issues for the model that has to recognize and classify the driver's action, if the driver itself is wearing a mask. This issues present themselves only if the model is trained with a dataset of subjects that are not wearing masks. If the model gets trained again, this time with a dataset of masked people, the performance of the classificator gets restored [1].

To the matter of deepface's face swaps, the presence of a mask in the source subject or in the destination subject, can cause some sort of aberrations in the resulting video. This issue may be caused by two main elements:

- The face detector, in this case `S3FD`, which may have trouble to correctly identify faces, due to the lack of mouth and nose in the subject
- The merger. This is the most likely element to cause trouble, specifically if the source subject does not wear a mask, and the destination does. This will result in not only a bad face replacement to the matter of landmarks, but also a terrible color mesh, because the mask's color (blue, white or black) will be overlapped with a natural complexion color, resulting in strange face swaps.

A solution to this issue can be represented by a face detector that is also trained to spot facial masks: if a facial mask is recognized in the destination footage, the face replacement will be performed only for the section of the source face that goes from the forehead to the cheekbones, keeping the mask in place.

6.2.3 Accessories and Hair

A similar issue is caused by elements such as glasses, a beard and hair. To this matter, the solution seems to be less trivial compared to the masks issue. First of all, a choice has to be made, whether to keep in the result the elements listed above or remove them. After this decision-making phase, some sort of post production is required, in order to remove (or coherently keep) all the residuals of of the element.

In the current state of FaceSwapAugmentation, if the `dst` subject is wearing, for example, glasses, and the `src` subject is not, in the resulting videos will still be present the some shades of the `dst` glasses frame, especially in the face swaps performed to extreme angles.

6.2.4 Temporary Obstruction Elements

Elements that temporarely cover the face, such as water bottles or cups when the subject is drinking, can also cause some sort of aberration. A solution to this issue could be structured in a similar way to the proposed fix for the mask problem: in addition to the face recognising tool, some sort of classificator can be used to identify bottles or cups. When one of this obstructing elements is recognized, it can be ignored in the face swap phase, and then can be put back on the resulting frame.

6.2.5 Improving qualitative results

The goal of this thesis was to build an automatic data-augmentation pipeline, that could generate a dataset that can be used to train the [1] classificator. Creating realistic/human-like results wasn't one of the main goals. In addition to that, the automation aspect of the pipeline was key. This resulted in a chioce of color swapping mode that is the same of every face swap (probably the overall best mode, but not the best for each face swap). This obviously isn't the ideal when working with subject with various skin tones or lighting.

A possible fix to this problem can be a *pre-processing phase* in which both the source and destination subjects are analyzed in order to acquire data on the complexion color and albedo. This data will then be used in the merging step, in order to choose the color mode that suites the best for the current subjects.

The most interesting (and arguably the hardest) of the issues of this pipeline, is the lack of emotional variance that is present when the `src` subject is AI generated, such as when using *StyleGAN 2* or `ThisPersonDoesNotExist`. This results in a facial replacement that is very static on the facial expression and muscle movements, giving a wax-like effect on the outputted face.

There are two possible approaches to this issue:

- Adding the facial expression through a manipulation of the `.obj` file generated from the `src` face.

- *A manipulation of the Morphable 3D Face Model, using a neutral base image, and a basic facial expression or a pair of valence-arousal (VA) emotional state descriptors to be generated, or a path of affect in the 2D VA space to be generated as an image sequence*, such as reported in the [8] article.

In the second (and best) approach, *the affect synthesis is implemented by fitting a 3D Morphable Model on the neutral image, then deforming the reconstructed face and adding the inputted affect, and blending the new face with the given affect into the original image*[8].

6.3 Additional performance tests

In addition to the two tests presented in this thesis, it's also important to check whether the deepfakes-trained model does a good job in classifying images of real people, since

it's the main reason for which FSA was created. To do so, an analogue test to the ones previously described will be done, measuring the accuracy and the F1-score of the hybrid setup. The first gotten results seem to be promising, but further attention must be given to them in order to be totally certain to the matter.

List of Figures

List of Tables

Bibliography

[1] Elvio G. Amparore et al. “Challenges for Driver Action Recognition with Face Masks”. In: (In Review).

[2] Nantheera Anantrasirichai and David Bull. “Artificial intelligence in the creative industries: a review”. In: *Artificial Intelligence Review* (2021), pp. 1–68.

[3] Adrian Bulat and Georgios Tzimiropoulos. “How far are we from solving the 2D & 3D Face Alignment problem? (and a dataset of 230,000 3D facial landmarks)”. In: *International Conference on Computer Vision*. 2017.

[4] Manuel G Calvo and Daniel Lundqvist. “Facial expressions of emotion (KDEF): Identification under different display-duration conditions”. In: *Behavior research methods* 40.1 (2008), pp. 109–115.

[5] Yu Deng et al. “Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[6] Bernhard Egger et al. “3d morphable face models—past, present, and future”. In: *ACM Transactions on Graphics (TOG)* 39.5 (2020), pp. 1–38.

[7] Tero Karras et al. “Analyzing and improving the image quality of stylegan”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8110–8119.

[8] Dimitrios Kollias et al. “Deep neural network augmentation: Generating faces for affect analysis”. In: *International Journal of Computer Vision* 128.5 (2020), pp. 1455–1484.

[9] Ivan Perov et al. “DeepFaceLab: Integrated, flexible and extensible face-swapping framework”. In: *arXiv preprint arXiv:2005.05535* (2020).

[10] Tong Sha et al. “Deep Person Generation: A Survey from the Perspective of Face, Pose and Cloth Synthesis”. In: *arXiv preprint arXiv:2109.02081* (2021).

[11] Connor Shorten and Taghi M Khoshgoftaar. “A survey on image data augmentation for deep learning”. In: *Journal of big data* 6.1 (2019), pp. 1–48.

[12] Simranjeet Singh, Rajneesh Sharma, and Alan F Smeaton. “Using GANs to synthesise minimum training data for deepfake generation”. In: *arXiv preprint arXiv:2011.05421* (2020).

[13] Francesca Soro et al. “The New Abnormal: Network Anomalies in the AI Era”. In: *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning* (2021), pp. 261–288.