Semi-Adversarial Networks: Convolutional Autoencoders for Imparting Privacy to Face Images

Neural Network Project

Bruno Marino Gianluca Pepe

May 26, 2019

1 Introduction

Nowadays, people's privacy is an hot topic. For years, global authorities have been moving to try to protect the privacy of individuals. The Cookie Law and the GDPR (General Data Protection Regulation) are just examples of recent measures.

Within the context of privacy on the images of people held by companies, it can be very interesting to apply de-identification techniques to these. De-identifications techniques usually allow to selectively confuse or preserve a set of attributes from images of human faces.

In this report we are going to discuss how we managed to implement the *Semi-Adversarial Network* proposed by *Mirjalili et al.* [1]. The addressed problem is to preserve the privacy of a subject depicted in a photo perturbing such image preserving face recognition while denying gender classification.

The novel technique proposed consist of an autoencoder whose training is influenced by two modules operating in an adversarial way. In particular the autoencoder is designed to generate image perturbations based on the feedback of two classifiers, where one classifier acts as an adversary of the other.

1.1 Problem Formulation

The main objective is to build an autoencoder able to perturb an image of a person in such a way that some characteristics are preserved whereas some others are suppressed, respectively we want to retain face verification and suppress gender recognition. In order to do this,the method proposed rely on a loss function composed by three terms:

$J_D(X, X')$: measuring the dissimilarity between the input image X and the perturbed image X' obtained as output from our autoencoder.

$J_G(y, X'; f_G)$: measuring the loss associated with correctly predicting the gender y of the perturbed image X' from an arbitrary gender classifier f_G.

$J_M(X, X'; f_M)$: measuring the loss associated with the match score between the input and the output of the autoencoder computed by an arbitrary face matcher f_M.

We have therefore a total loss (1) composed by these three terms in which the last two will influence the autoencoder's training outcome.

$$J_{total}(X, y, X'; f_G, f_M) = J_D(X, X') + J_G(y, X'; f_G) + J_M(X, X'; f_M) \quad (1)$$

We will see later in a dedicated section, once the internal structure of the model we implemented will be clarified, how each term of (1) is calculated and in which phases of the training process.

2 Model Architecture

The model optimizing the loss function previously explained, as already introduced, is composed by three different modules: the autoencoder, a gender classifier and a face matcher. The way in which these modules are combined together and used during the training phase will result in a final model able to fulfill the intended objectives.

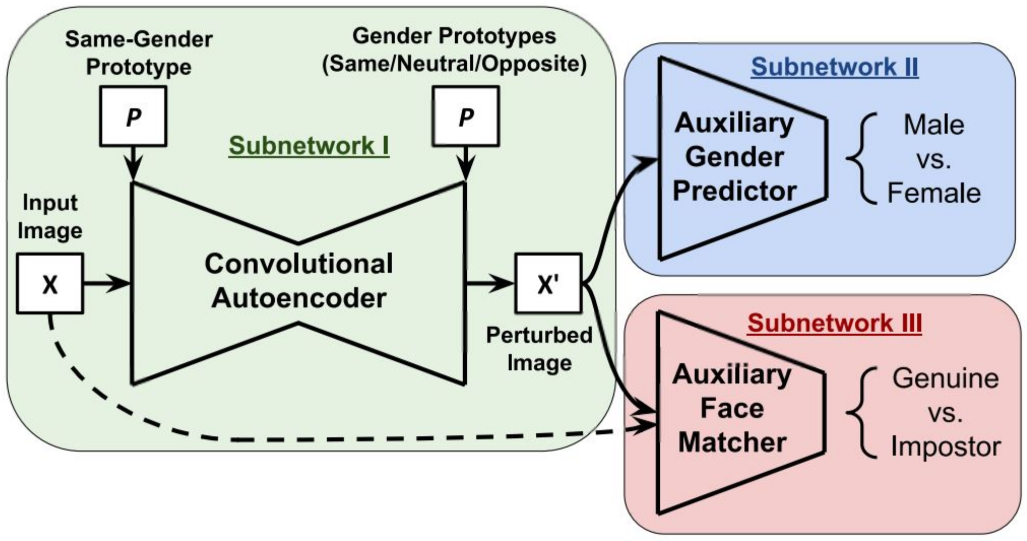


Figure 1: Model Architecture

As we can see in figure 1, the autoencoder uses a method based on *gender prototypes* for perturbing the input images. These *gender prototypes* are basically images depicting the average appearance of women, men and both genders together, i.e., the neutral gender prototype. Concatenating the same gender prototype to the input image and one of three possible prototypes to the autoencoder output, we can control the way images are perturbed and more precisely how the information about the gender are perturbed.

Once the perturbed image is obtained, the auxiliaries modules come in to play providing their contribution computing on it respectively the loss metric of the gender classification and the one referred to the matching score against the unperturbed version.

2.1 AutoEncoder

The autoencoder can be considered as the real kernel of this model, in the end indeed, other modules will be detached and the whole process of generating gender perturbed images will rely exclusively on it. According to the definition of autoencoders, the internal structure consist of two part: an encoder and a decoder. In the first part the goal is to reduce the amount of information and extract a representation of the input image in a latent-space, in the second part instead, the system try to reconstruct and infer those lost information and hence to reconstruct a complete image.

The encoder internal structure is composed, as we can see in figure 2, by two convolutional layer, both followed by a Leaky ReLU activation function and an average pooling layer. The decoder instead is made up again by two convolutional layer with Leaky ReLU activation functions, but in this case an upsampling layer using a two dimensional nearest neighbor interpolation is used instead of the average pooling.

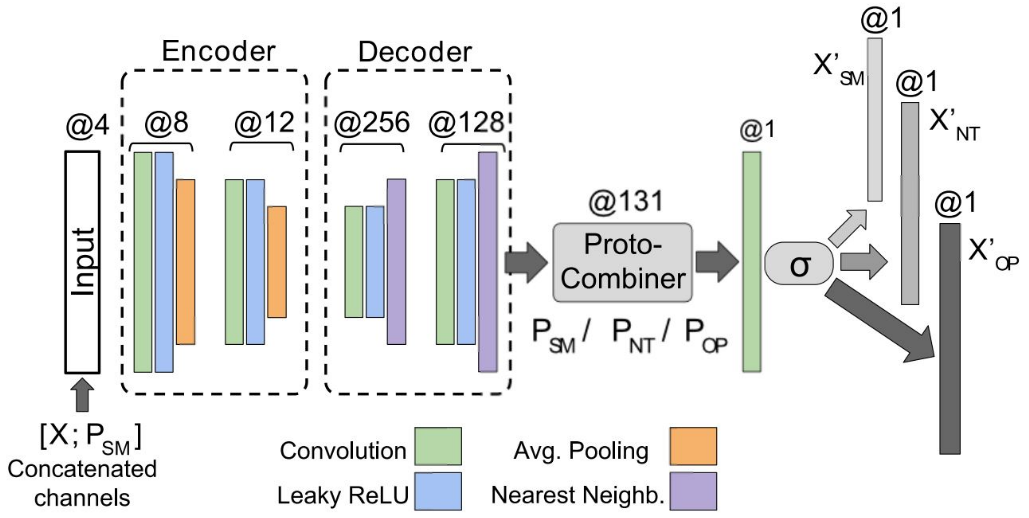


Figure 2: AutoEncoder Architecture

2.1.1 Mode of Operation

Following the schema depicted in figure 2, the autoencoder input is the concatenation of the grayscale image to be perturbed and the rgb same gender prototype P_{SM}. This means that female prototype will be used when the gender of the person is female and male prototype will be used in case of a male person in the grayscale input image. The encoder then apply the two convolutional steps resulting in a feature map of size $56 \times 56 \times 12$ and the same goes the decoder resulting in a feature map of size $224 \times 224 \times 128$. The final step in the generation of perturbed images is the proto-combiner that actually operate a concatenation of the decoder output with one of the three gender prototypes. The result then goes through one last convolutional layer and a sigmoid activation function yielding a perturbed image according to the gender prototype used by the proto-combiner.

2.2 Gender Classifier

The subnetwork II in figure 1 is again a convolutional neural network and its purpose is to predict the gender of an input image. In the model we are presenting, this classifier will operate on the perturbed images produced by the autoencoder and will provide him his loss metric in order to influence its weights during the training phase.

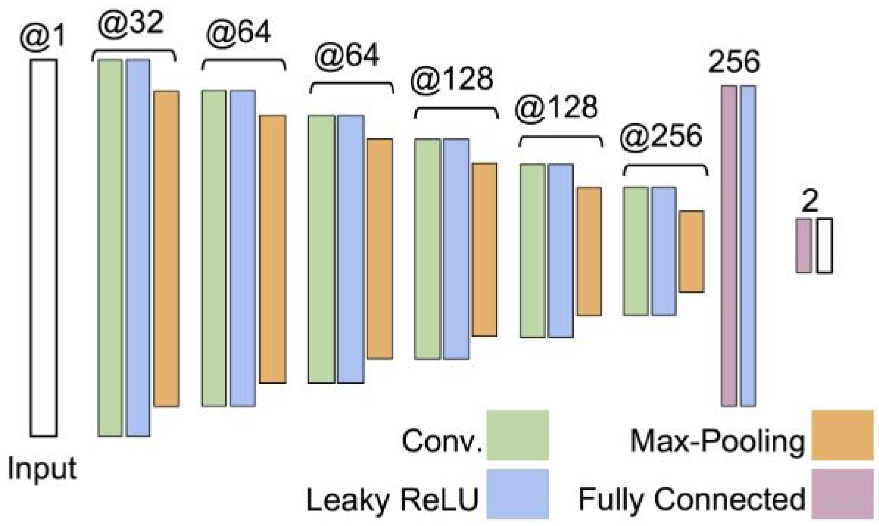


Figure 3: Gender Classifier Architecture

As we can see in figure 3, this network takes as input a grayscale image that goes then through six convolutional layers, each one followed by a Leaky ReLU activation function and a max-pooling layer. The max-pooling operation downscale the dimensions of the sample by a factor of 2, meaning that a filter matrix of size 2×2 will be considered for computing the max value in the pool. A stride of the same size dimension of the pool size will then be used for moving the filter matrix across the image avoiding overlapping with other pools.

2.2.1 Mode of Operation

The sequence of convolutional layers is used to derive a feature map of size $4 \times 4 \times 256$ and it's then sent through a fully connected layer followed by the

LeakyReLU activation function, then through another fully connected layer, this time with an output space of size 2 resulting therefore in a feature map of size $1 \times 1 \times 2$. Finally we have the sigmoid activation function that yields a probability distribution over the two labels: 0: "Female" and 1: "Male". As output from this network we obtain therefore the likelihood for which a given image is classified either male or female.

2.3 Face Matcher

The final module involved in the model we introduced so far is the face matcher and its main purpose is to compute the distance, i.e., the dissimilarity between two images in order to establish whether the person depicted in them share the same identity or not. In this model we will use it for giving a feedback to the autoencoder in order to help him to generate perturbed images preserving the subjects identities.

The model used for this purpose is the VGG16-face described by Parkhi et al. [2] consisting of 16 weight layers. Its output is the input embedding in a space of 2622 dimensions, i.e, a vector representation of the input image. We are then able to compare the matching score between two images obtaining their vector representations from the VGG network and computing then the L2 norm between them.

We will not analyze the internal structure of this module since its publicly available and we used it as it is.

3 Datasets and Prototypes

In any Machine Learning problem, the dataset represents the starting point. The choice and preparation of the dataset is fundamental: it contains the data on which perform training and evaluation. In addition to this, the generation of prototypes is also a crucial point for our work.

3.1 Datasets

To reimplement the work of the paper on which we are based, for our project of Neural Networks exam, we have chosen 2 datasets:

- CelebA Dataset [3]
- LFW Face Database [4, 5]

CelebA is a large-scale face attributes dataset with thousands of celebrity images, each with many attribute annotations. The only attribute we use in our work is the gender, basing on it, we divide the dataset in male and female folders. We organize this dataset in the following folders, numbers in brackets indicates the number of images each folder contains:

- dataset
  - prototype (3)
  - test
    - female (11511)
    - male (7929)
  - train
    - female (88207)
    - male (62517)
  - validation
    - female (15565)
    - male (11032)

We originally split the dataset into train and test set and then we randomly move 15% of sample from the training se to the validation set. All CelebA images are 224 x 224 pixels.

We use CelebA dataset during all phases of our project. Specifically, we use train and validation folder during pre-training and further-training phases and we use the test folder during the evaluation. In this last phase we generate perturbed images of CelebA, starting from the ones of the test-set, in a new folder that we called 'generated'. This last step will be described in the evaluation chapter.

Figure 4: Sample images from CelebA dataset. First row: female samples. Second row: male samples.

LFW stands for Labeled Faces in the Wild, it is a public available dataset of face photographs. The original data set contains more than 13.000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. For some people more pictures are available and for this reason, this dataset is used during the evaluation phase where we need a second original picture for the same person. That's why we took from the dataset only people who had at least two images.

We organize this dataset in the following folders, numbers in brackets indicates the number of images each folder contains:

- LFW
    - female
        - OR_1 (442)
        - OR_2 (442)
    - male
        - OR_1 (1237)
        - OR_2 (1237)

Also in this case, during the evaluation phase we created the folders containing the perturbed images, which were obtained by perturbing the faces contained in the folders OR_1. LFW images are 250 x 250 pixels so we had to scale it to 224 x 224 to perform the perturbation.

Figure 5: Sample images from LFW dataset. First two column: female samples. Second two columns: male samples. In the second row is reported another image of the same person in first row.

3.2 Gender prototypes

Prototypes are used to confound gender classifier while mantaining biometric matching during the semi-adversarial training (further-training) of the autoencoder.

Basically, prototypes are RGB images of size 224 x 224 pixels. We obtain them as average of images. To be precise, we have three prototypes: male (P_{male}), female (P_{female}) e neutral. Male prototype is obtained as the average of all the males in training set (88207 samples), female prototype is obtained as the average of all females in training set (62512) and finally, neutral prototype is obtain as the weighted average among male and female prototype.

From here on out, we'll talk about these three prototypes:

- **Same-gender prototype (SM):** is the prototype of the same gender of the considered picture. If the picture represent a male, SM prototype is the male prototype, if the picture represent a female, SM prototype is the female prototype. Formally: $SM = yP_{male} + (1 - y)P_{female}$.

- **Opposite-gender prototype (OP):** is the prototype of the opposite gender of the considered picture. If the picture represent a male, OP prototype is the female prototype, if the picture represent a female, OP prototype is the male prototype. Formally: $OP = (1 - y)P_{male} + yP_{female}$.
- **Neutral-gender prototype (NT):** is the already defined neutral gender prototype obtained as the weighted average (based on the number of samples) among P_{male} and P_{female}.

Here we have $y = 1$ if we consider an image representing a male and $y = 0$ if we consider an image representing a female.

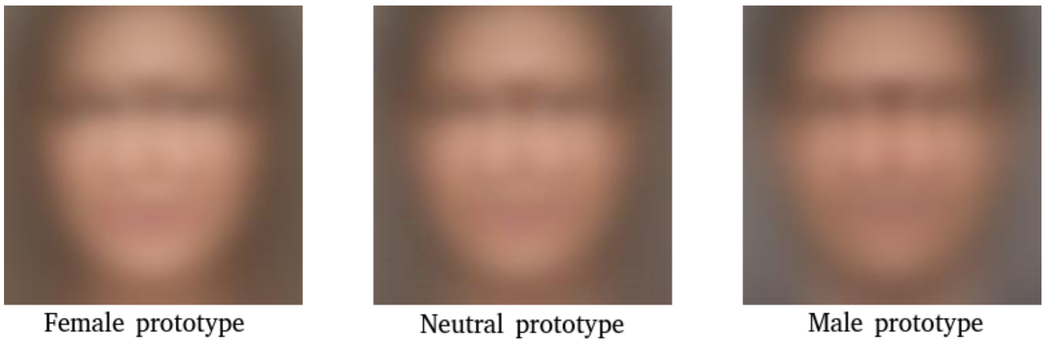


Figure 6: Prototypes obtained from CelebA training-set

The convolutional autoencoder (figure 1 and figure 2) is provided with same-gender prototype images, which are concatenated with the input image before being transmitted to the encoder module, this allows to derive a compressed representation of the original image along with the same-gender prototype information. After the decoder reconstructs the original images, the three different gender-prototypes are added as additional channels via the proto-combiner (always figure 2). Finally, the last convolutional layer of the autoencoder produces three different perturbed images: X'_{SM} (obtained when the same-gender prototype is used), X'_{NT} (when the neutral prototype is used), and X'_{OP} (when the opposite gender prototype is used).

4 Training Phase

The system, like any neural network, should be able to search the set of trainable weights for all the layers we introduced in the previous section in order to find the weights performing good enough predictions. In this context the loss function comes in to play, its purpose is to calculate a score value summarizing all features of the model. The loss function is therefore a function used for evaluating a given set of learned weights and lowering it(or maximizing it, depending on the addressed problem) allow us to get a better model.

This process takes place in the so called training phase of the model and in this project two distinct training phases have been used: *pre train and further train.*

4.1 Pre Train

In this phase the autoencoder and the two sub-networks are initially trained as independent and standalone modules. We want indeed to initially obtain an autoencoder able to produce images as much as possible similar to the input images, a gender classifier able to predict genders correctly and a face matcher able to produce good embeddings.

Given that the autoencoder should be able to generate similar to the input, in this phase only *same gender prototypes* will be provided.

The loss function (2) implemented for pre training the autoencoder is the pixel wise cross entropy S between input and output images and it represents the first term in the total loss discussed in the introduction.

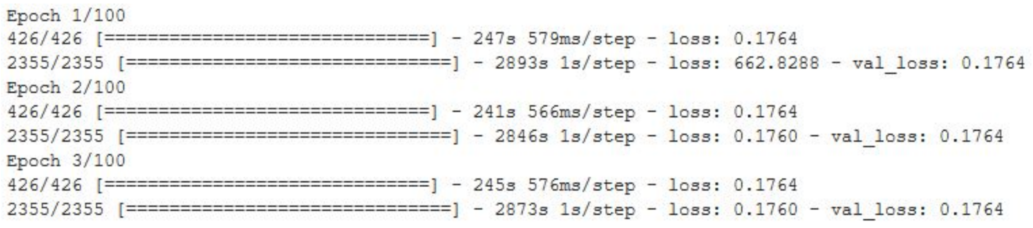
$$J_D(X, X'_{SM}) \ = \sum_{k=1}^{224^2} S\left(X^{(k)}, X_{SM}^{(k)}\right) \tag{2}$$

For what concerns the two sub-networks, a cross entropy loss function have been used for the gender classifier while pre-trained weights have been used for the face matcher.

4.1.1 Training Results

The autoencoder has been pre-trained for 3 epochs on the whole CelebA [3] train dataset using a batch size of 64 resulting in a quite long process lasting more than 2 hours. The history of this initial training is shown in Fig. 7.

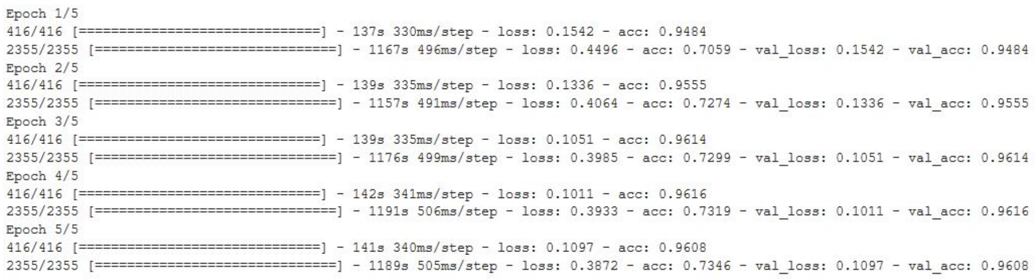
For the gender classifier, instead, we followed the same method presented by our reference paper [1] consisting in five epochs train using again batches



```
Epoch 1/100
426/426 [==============================] - 247s 579ms/step - loss: 0.1764
2355/2355 [==============================] - 2893s 1s/step - loss: 662.8288 - val_loss: 0.1764
Epoch 2/100
426/426 [==============================] - 241s 566ms/step - loss: 0.1764
2355/2355 [==============================] - 2846s 1s/step - loss: 0.1760 - val_loss: 0.1764
Epoch 3/100
426/426 [==============================] - 245s 576ms/step - loss: 0.1764
2355/2355 [==============================] - 2873s 1s/step - loss: 0.1760 - val_loss: 0.1764
```

Figure 7: Autoencoder pre-train history

of 64 samples. We also obtained the same result with a test accuracy reaching 96% as shown in Fig. 8



```
Epoch 1/5
416/416 [==============================] - 137s 330ms/step - loss: 0.1542 - acc: 0.9484
2355/2355 [==============================] - 1167s 496ms/step - loss: 0.4496 - acc: 0.7059 - val_loss: 0.1542 - val_acc: 0.9484
Epoch 2/5
416/416 [==============================] - 139s 335ms/step - loss: 0.1336 - acc: 0.9555
2355/2355 [==============================] - 1157s 491ms/step - loss: 0.4064 - acc: 0.7274 - val_loss: 0.1336 - val_acc: 0.9555
Epoch 3/5
416/416 [==============================] - 139s 335ms/step - loss: 0.1051 - acc: 0.9614
2355/2355 [==============================] - 1176s 499ms/step - loss: 0.3985 - acc: 0.7299 - val_loss: 0.1051 - val_acc: 0.9614
Epoch 4/5
416/416 [==============================] - 142s 341ms/step - loss: 0.1011 - acc: 0.9616
2355/2355 [==============================] - 1191s 506ms/step - loss: 0.3933 - acc: 0.7319 - val_loss: 0.1011 - val_acc: 0.9616
Epoch 5/5
416/416 [==============================] - 141s 340ms/step - loss: 0.1097 - acc: 0.9608
2355/2355 [==============================] - 1189s 505ms/step - loss: 0.3872 - acc: 0.7346 - val_loss: 0.1097 - val_acc: 0.9608
```

Figure 8: Gender Classifier pre-train history

For the face matcher, as already said, we just loaded public available pretrained weights based on the work done in [2], therefore we didn't perform any kind of training on this sub-network.

4.2 Further Train

In this phase all modules are combined together and the autoencoder will be trained using the information coming from the gender classifier and the face matcher. The two sub-networks will be used as fixed items, they will be hence set as non-trainable. Both same and opposite prototypes were used this time in order to preserve gender information in case of same gender(SM) and to suppress those information in case of opposite gender (OP).

The term associated to the gender classifier's feedback (3) is the cross-entropy between the true gender label y and the predicted gender. When predicting the gender over the image perturbed with the opposite gender prototype we use the opposite true gender label as well, in this way we get a measure of how well the autoencoder is suppressing the gender informations.

$$J_G(y, X'_{SM}, X'_{OP}; f_G) \;=\; S(y, f_G(X'_{SM})) \;+\; S(1-y, f_G(X'_{OP})) \qquad (3)$$

The loss term coming from the face matcher (4) consist instead of the squared euclidean distance between the two vector representation: respectively the perturbed image using same gender prototype and the original image. As already explained, the face matcher is used to derive those image representation.

$$J_M(X, X'_{SM}; R_{vgg}) \;=\; \|R_{vgg}(X'_{SM}) - R_{vgg}(X)\|_2^2 \tag{4}$$

The further train phase will use hence a loss function (5) made up by the sum of these two terms:

$$J_{FT}(X, y, X'_{SM}, X'_{OP}; f_G, R_{vgg}) \;=\; J_G(y, X'_{SM}, X'_{OP}; f_G) + J_M(X, X'_{SM}; R_{vgg}) \tag{5}$$

4.2.1 Training Details and Results

For combining together the three modules and in order to obtain the autoencoder to be influenced by the sub-network's feedback, we built a wrapping model containing the other three. From now on we will refer to it as the *SemiAdversarial* model or as the *SAN* (SemiAdversarial Network). This model receive as input the following items:

- original grayscale image,
- same gender prototype,
- opposite gender prototype,
- label (y) and opposite label ($1 - y$),
- original image embedding computed by the face matcher (Vgg).

The first three inputs are sent to the autoencoder in order to get two perturbed images as output, one perturbed retaining gender information (SM prototype) and the other one suppressing those information (OP prototype). Once those perturbed images are generated, the model will use the gender classifier to get gender predictions over both those images and the face matcher to get the vector representation over the same gender perturbed one.

Now the model has all the necessary elements for computing the two terms expressed by equations (3) and (4) in order to get the total loss (5). This loss function (5) has been implemented as a Keras custom layer getting as input the gender predictions computed by the gender classifier, the true gender labels and the original image and same gender perturbed image vector

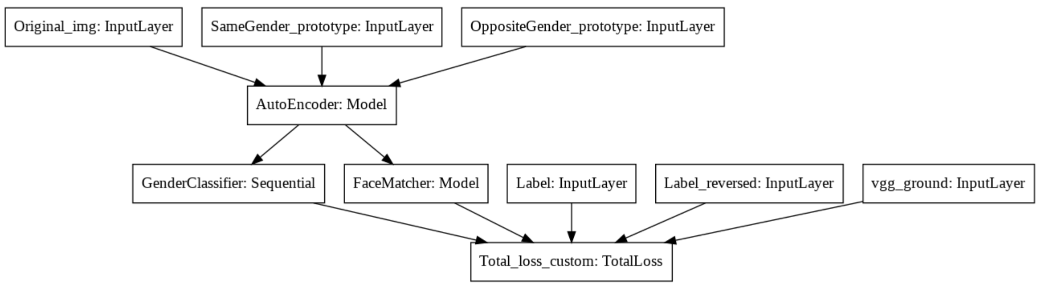

Original_img: InputLayer
SameGender_prototype: InputLayer
OppositeGender_prototype: InputLayer
AutoEncoder: Model
GenderClassifier: Sequential
FaceMatcher: Model
Label: InputLayer
Label_reversed: InputLayer
vgg_ground: InputLayer
Total_loss_custom: TotalLoss

Figure 9: Further train custom loss function, calculation scheme.

representations. The whole process described so far in order to get such custom loss function is depicted in Fig. 9.

As already said, the only trainable module in this phase is the autoencoder, the other two modules are set as non trainable because we just need their feedback to refine the set of pre-trained weights on the autoencoder. Given the amount of input we reduced the dataset to 10000 samples and the batch size to 32. The model has then been trained for 10 epochs in which we observed the loss value decreasing over time as expected, train history and loss plot are shown respectively in Fig. 10 and 11.

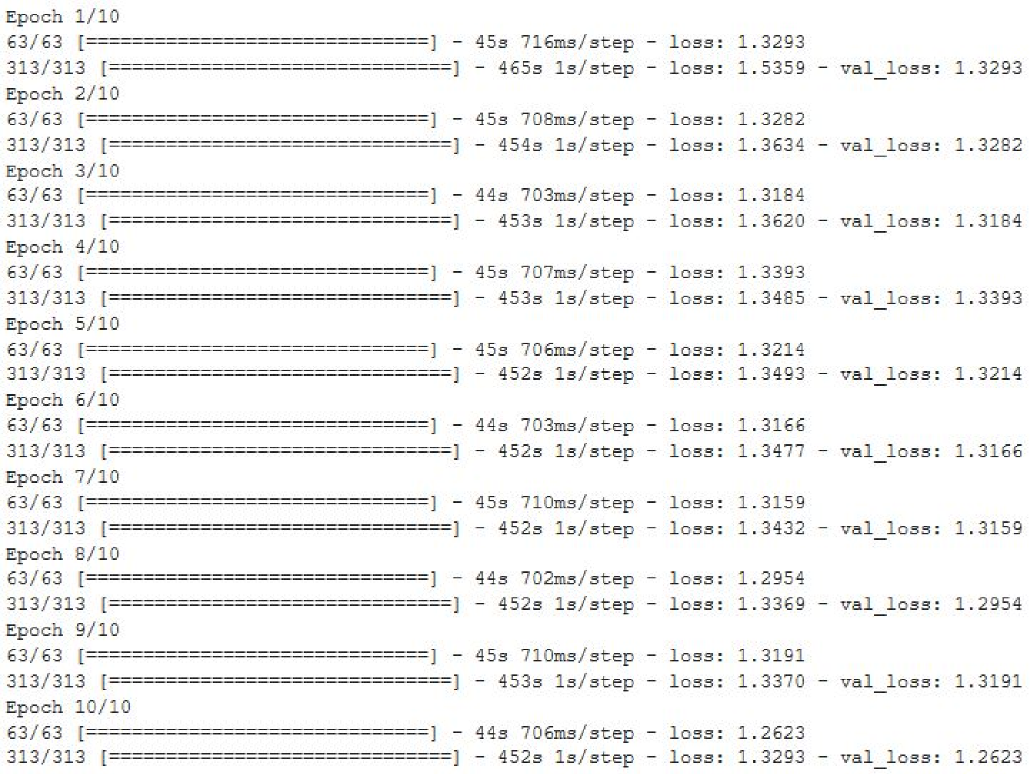
4.3 Implementation details and software

The neural networks described in this work, are implemented in TensorFlow [6] and the high-level neural networks API Keras [7]. We have used Keras sequential API to implement the genderclassifier and Keras functional API to implement: the autoencoder, the facematcher and the semi-adversarial network. The Keras functional API gives the possibility to define complex models w.r.t. the Keras sequential API, such as multi-output or multi-inputs models and also to define custom layers.

Given that we have found the pretrained facematcher implemented with Keras sequential API and requiring RGB images as input we have had to re-implemented it. We transformed the model using Keras functional API and we defined a custom layer performing the concatenation of the input gray scale image to itself (three times), in order to obtain images with 3 channels (RGB) starting from images with one channel (grayscale).

For the implementation of the loss functions we have used custom code as the required loss functions are not present natively in Keras or TensorFlow.

For the SAN model we have included the autoencoder, the genderclassifier and the facematcher as layers, giving each layer the correct inputs. Then, we have built the total loss of the SAN as a final custom layer.



```
Epoch 1/10
63/63 [==============================] - 45s 716ms/step - loss: 1.3293
313/313 [==============================] - 465s 1s/step - loss: 1.5359 - val_loss: 1.3293
Epoch 2/10
63/63 [==============================] - 45s 708ms/step - loss: 1.3282
313/313 [==============================] - 454s 1s/step - loss: 1.3634 - val_loss: 1.3282
Epoch 3/10
63/63 [==============================] - 44s 703ms/step - loss: 1.3184
313/313 [==============================] - 453s 1s/step - loss: 1.3620 - val_loss: 1.3184
Epoch 4/10
63/63 [==============================] - 45s 707ms/step - loss: 1.3393
313/313 [==============================] - 453s 1s/step - loss: 1.3485 - val_loss: 1.3393
Epoch 5/10
63/63 [==============================] - 45s 706ms/step - loss: 1.3214
313/313 [==============================] - 452s 1s/step - loss: 1.3493 - val_loss: 1.3214
Epoch 6/10
63/63 [==============================] - 44s 703ms/step - loss: 1.3166
313/313 [==============================] - 452s 1s/step - loss: 1.3477 - val_loss: 1.3166
Epoch 7/10
63/63 [==============================] - 45s 710ms/step - loss: 1.3159
313/313 [==============================] - 452s 1s/step - loss: 1.3432 - val_loss: 1.3159
Epoch 8/10
63/63 [==============================] - 44s 702ms/step - loss: 1.2954
313/313 [==============================] - 452s 1s/step - loss: 1.3369 - val_loss: 1.2954
Epoch 9/10
63/63 [==============================] - 45s 710ms/step - loss: 1.3191
313/313 [==============================] - 453s 1s/step - loss: 1.3370 - val_loss: 1.3191
Epoch 10/10
63/63 [==============================] - 44s 706ms/step - loss: 1.2623
313/313 [==============================] - 452s 1s/step - loss: 1.3293 - val_loss: 1.2623
```

Figure 10: Further train history

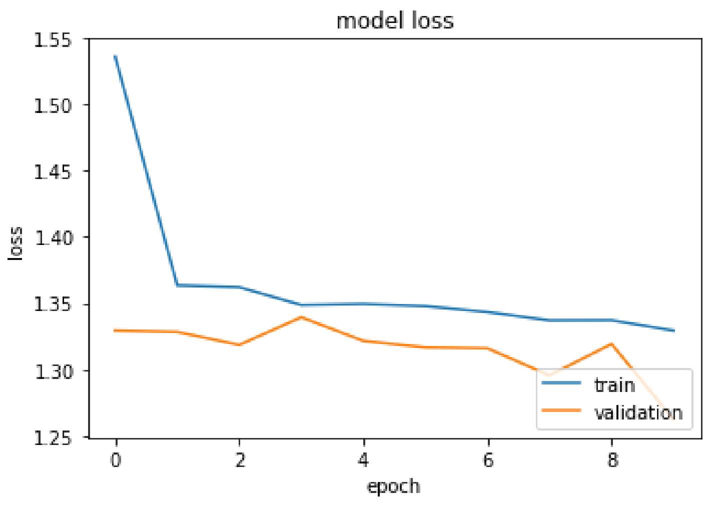


Figure 11: Loss plot in further training phase

5 Results and Evaluation

After we have trained the autoencoder using the SAN during the further training phase, we have detached from it the two semi-adversarial models. At this point, we used the autoencoder, togher with its obtained weights, to perturb images from Celeb-A-test and LFW dataset. For each pictures, we have obtained one output for each prototypes. To be clear, we have obtained three 'new' faces: the orginal one perturbed with the same-gender prototype (SM), the neutral-gender prototype (NT) and the opposite-gender prototype (OP).

Given that pictures from LFW dataset are different in size with respect to the ones we have for the three prototypes, we have had to rescale them in order to obtain the perturbation with the autoencoder, that accepts input images of the size of 224 × 224 pixels.

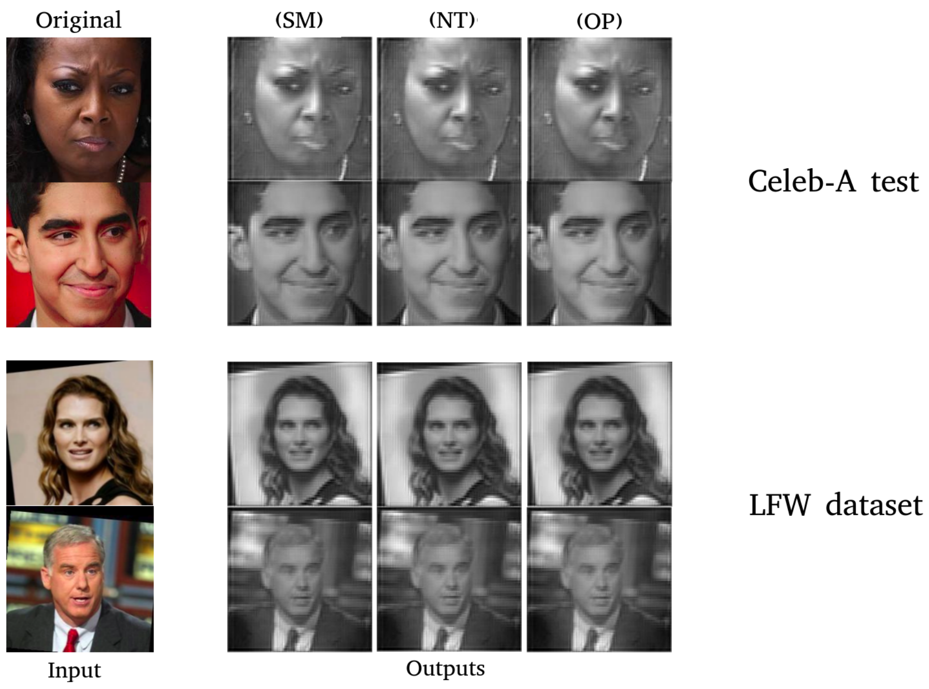


Figure 12: Example of obtained perturbed images starting from the original from Celeb-A test and LFW dataset.

In the picture are illustrated same samples of obtained images. From a human sight perception they may appear exactly the same but observing carefully and in sequence it's possible to see that they are indeed slightly different.

To organize the new obtained images inside the Celeb-A dataset folder we added the ‘generated’ folder which have the following structure, numbers in brackets indicates the number of images each folder contains:

- generated
  - female
    - NT (11511)
    - OP (11511)
    - SM (11511)
  - male
    - NT (7929)
    - OP (7929)
    - SM (7929)

To organize the new obtained images inside the LFW dataset folder we added the ‘NT’, ‘OP’ and ‘SM’ folders inside ‘male’ and ‘female’ folders of the LFW dataset structure. Here there are less images w.r.t. Celeb-A test: 442 for each perturbed under female folder and 1237 for each perturbed under the male folder.

Once we have obtained the perturbed images we have choose face matcher and geneder classifier networks to test our results. In order to test, we use togheter with other networks, also the gender classifier and the face matcher used during the training phases. Results with this two neural networks is expected to be optimistically biased and we have tested it.

Let’s remember that our aim is to perturb images depicting human faces to confuse a software that acts as a gender classifier while keeping the biometric of the face as unaltered as possible. So that, a human must be able to immediately connect a perturbed face picture with the original picture.

5.1 Evaluation on Gender Classification

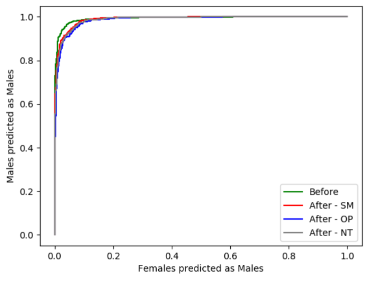
To test the autoencoder ability to suppress gender information we have selected three different networks, we expect that theese gender classifiers will be confused. The three gender classifiers are:

- **GC from the paper:** The gender classifier developed following the paper.

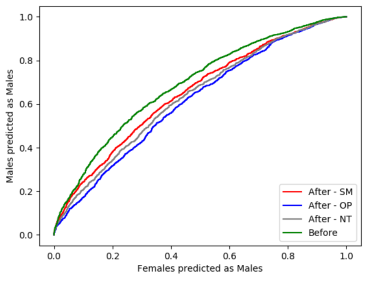
- **CvLib:** A high level and open source computer vision library for python, under MIT license. It uses a pretrained keras model. [8]

- **SimpleNN:** A pretrained network publicly available on the web, it is part of a bigger project whose aim is the real-time face detection and classification. [9, 10]

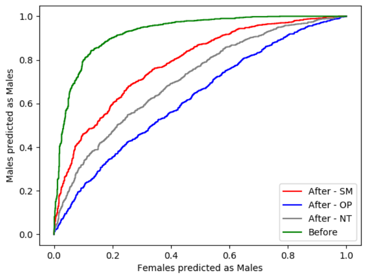
We have tested these three software both on CelebA test and LFW dataset. In the first case on 2000 samples and in the second case on all samples available (LFW has less samples). The obtained results are illustrated in Fig. 13 and 14 where 'before' represent the roc curve resulting from the gender predictions over the original images, other curves refer instead to gender predictions over perturbed images using the respective gender prototype.



(a) GC from the paper



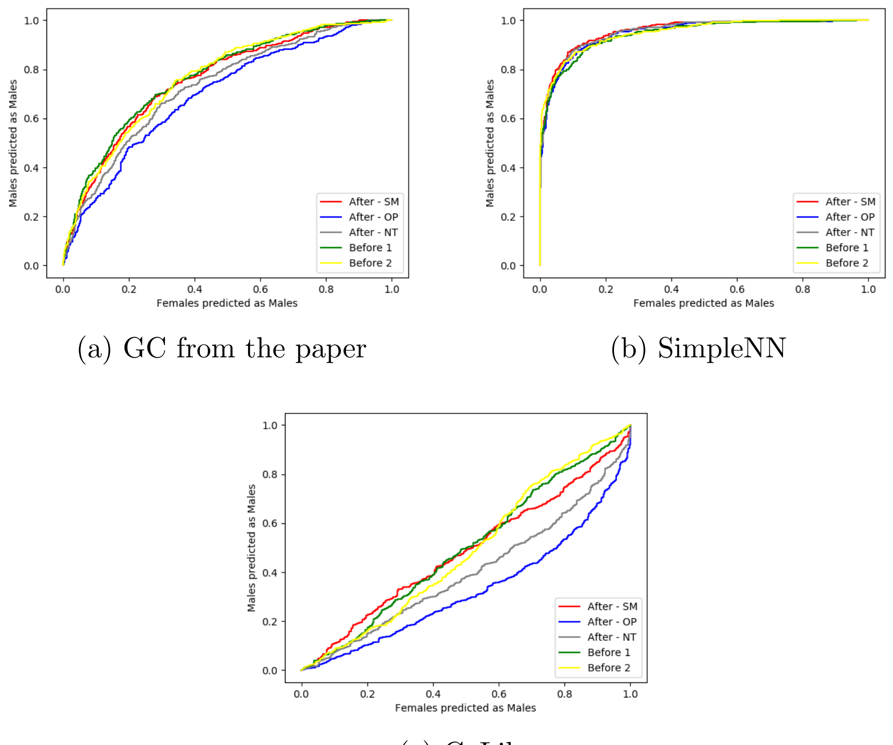
(b) SimpleNN



(c) CvLib

Figure 13: ROC curves on CelebA dataset

We can see in Fig. 13 that the three different models have very different performances on the CelebA dataset. As expected the result from the gender classifier used during the training phase (Fig. 13a) are the best due to its positive bias. The other two ROC curves (Fig. 13b and 13c) are more interesting because the models are not trained on the CelebA dataset and they both show how performances get worse perturbing images with the opposite gender prototype. This is particularly evident in the CvLib's ROC where the blue curve related to predictions over images perturbed with opposited gender prototype is almost diagonal meaning almost random predictions whereas they're fairly good on the unperturbed images (the green curve).



(a) GC from the paper (b) SimpleNN

(c) CvLib

Figure 14: ROC curves from gender classifiers on LFW dataset

On the LFW dataset the models still show very different results but we can still observe the main feature for which the predictions are overall worse when done on perturbed images, in particular on those perturbed with opposite gender classifier. On this dataset though, the SimpleNN have the best performance and the curves are mostly overlapped (Fig. 14b), this means that the classifier is somehow immune to the autoencoder's perturbation. The CvLib's model (Fig. 14c) instead has the worst performance: predictions are mostly random on unperturbed images (green and yellow curves) but as expected they get even worse when perturbed images with opposite gender prototype are considered.

5.2 Evaluation on Face Matching

To test the autoencoder ability retaining subject's identities we have selected two networks computing images embedding, we expect in general that these face matchers will not be confused. The two matching software are:

- **FM of the paper:** The face matcher developed following the paper.
- **FaceNet:** A convolutional pre-trained neural netowork (Keras implementation). [11]

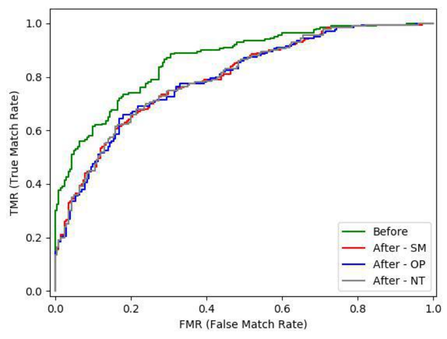
We tested these two software only on LFW dataset, given that it gave us the possibility to match two original images of the same person whereas it was not possible with CelebA dataset. In all tests, 400 samples are considered: 200 males and 200 females.

As previously explained, these networks don't provide directly the matching score between two images, they only provide us the input face-image's embedding. In other words, these networks extract features from face, they provide feature vectors as result.

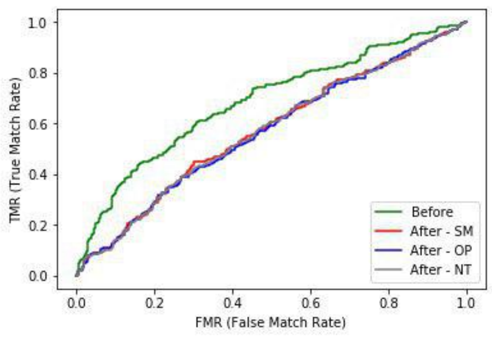
We then computed the matching score between two images using the same function used in the custom loss function in further train, i.e., the squared euclidean distance. For computing ROC curves though, class membership probabilities are needed and we obtained them inverting such distance with:

$$P(X, X') = \frac{1}{(1 \; + \; \|X - X'\|_2^2)} \tag{6}$$

where X is the embedding for the unperturbed image while X' is the embedding for the perturbed image. Results of tests carried out using such matching probability are shown in Fig. 15.



(a) FM from the paper



(b) FaceNet

Figure 15: ROC curves from face matchers on LFW dataset

The two models perform quite differently: we can see again in Fig. 15a that the face matcher used in the training phase is slightly positively biased and hence performing better than the FaceNet model (Fig. 15b) which has not a so good matching rate also on original samples.

The interesting feature we can observe from such results is that, unlike in gender classification results, this time is basically impossible to distinguish the curves related to perturbed images. This means that the type of perturbation, i.e., the gender prototype used, does not really affect the way the accuracy decrease. That is, the same image perturbed with the three different prototypes gives more or less the same matching performance.

Of course, FaceNet does not give us good results at first sight, but in general, also original images are not matched so well. Maybe another trained network can perform better, or another dataset on which perfom test can give us better results. By the way, we can consider our results not so bad, given that trends of both ROC of original images and ROCs of perturbed images are quiete similar and not so distant.

6 Conclusions

We've seen the whole development process for this work, starting from setting the main goals, formulating them in a model and then implementing a possible solution. Even though obtained result are not as good as those obtained in the reference paper [1], we've still been able to observe the expected behavior and patterns. The gender classifiers we tested are indeed significantly impacted by opposite gender perturbation and this is one of the established goals, the face matcher instead shows some performance degradation when perturbing images but, as we have seen, this is not necessarily due to perturbations given that we can observe such degradation regardless the type of perturbation applied. Perhaps on another dataset or with another facematcher, different results would have been obtained. One reason we got slightly different results (but not so much) might be that we didn't use commercial state of the art softwares in order to evaluate the obtained results, as our source did.

The whole model is just an example of how could be possible imparting privacy to face images, it would be surely possible to implement it considering other biometric information such as age, ethnicity or even emotional state in order to provide more privacy and security when dealing with user's face images.

References

[1] V. Mirjalili, S. Raschka, A. M. Namboodiri, and A. Ross, "Semi-adversarial networks: Convolutional autoencoders for imparting privacy to face images," *CoRR*, vol. abs/1712.00321, 2017.

[2] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," vol. 1, pp. 41.1–41.12, 01 2015.

[3] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[4] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Tech. Rep. 07-49, University of Massachusetts, Amherst, October 2007.

[5] G. B. Huang, V. Jain, and E. Learned-Miller, "Unsupervised joint alignment of complex images," in *ICCV*, 2007.

[6] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016.

[7] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[8] A. Ponnusamy, "cvlib: a high level, easy to use, open source Computer Vision library for Python." `https://www.cvlib.net/`, 2018. [Online; accessed 24-May-2019].

[9] O. Arriaga, M. Valdenegro-Toro, and P. Plöger, "Real-time convolutional neural networks for emotion and gender classification," *CoRR*, vol. abs/1710.07557, 2017.

[10] O. Arriaga, "Real-time face detection and emotion/gender classification." `https://github.com/oarriaga/face_classification`, 2017. [Online; accessed 24-May-2019].

[11] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015.