Semester Project Report

Detection of digital face beautification via social media filters

DASGUPTA Anasuya
dasgupta@eurecom.fr

RAMON Matthieu
matthramon@gmail.com

EURECOM

February 17th, 2023

Abstract

Detecting digital face beautification is important in a variety of applications such as in public administration or to counter cybercrime. Previous works studied the impact of face beautification on weight and gender identification, or studied source social media indentification from unfiltered face pictures. Inspired by that, we propose two models based on the 18-layers ResNet-architecture. The first one to classify between unfiltered and filtered face pictures, the second one to do source social media identification on unfiltered and filtered face pictures. To do so we augmented an already existing dataset, providing more than 14 000 unfiltered and filtered pictures coming from the CALFW and VIP datasets.

Keywords: Dataset Augmentation, Filtered/Unfiltered classification, Source Social Network Identification, ResNet18.

1. Introduction

TODO

1. Dataset Augmentation.
2. Building a classification network for filtered images.
3. Social Network Identification with filtered images.

2. Related Works

This project is based on two previous papers:

1. The first reference is "Impact of Digital Face Beautification in Biometrics" by Nelida Mirabet-Herranz, Chiara Galdi and Jean-Luc Dugelay. They created a dataset of about 8000 face pictures coming from the CALFW and VIP datasets. Those pictures were either uploaded on

Instagram and not filtered, in order to have a base reference, or filtered with three different Instagram filters. They then used this database to study the impact of filters on weight and gender classification.

2. The second reference is "Two-stream Convolutional Neural Network for Image Source Social Network Identification" by Alexandre Berthet, Francesco Tescari, Chiara Galdi and Jean-Luc Dugelay. Their goal was to do source social network identification for unfiltered face pictures. To do so they computed two different domains for the pictures: Discrete Cosine Transform and Photo-Response Non-Uniformity. They then used both of these to separately feed neural networks for feature extraction. Finally they concatenated the obtained features and used it for classification.

3. Database Description and Augmentation

The idea was to expand the dataset provided by the first reference paper. It contained about 8000 face pictures from the CALFW and VIP datasets, both unfiltered and filtered on Instagram. The VIP dataset contained images of 513 male and 513 female while the CALFW datset had 500 male and 500 female image which need to passed through the social media filters. So, the process was further divided into three sub-steps.

1. Choosing the two social networks.
2. Choosing filters to apply on the images for each SN to build the 'filtered' dataset
3. Making an 'unfiltered' dataset by making the images pass through each of the social networks without filter.

The last step is done so all the images go through the same amount compression according to the Social networks. So, the classifier does not learn the compression rates instead.

3.1. Choosing the Two Social Networks:

We were to choose two more popular social networks for the 'filtered' dataset. Although based on a survey, we found out that Facebook, YouTube and Whatsapp were one of the most popular ones. However, we could not proceed with them since they use the filters only in the ' Messenger ' or messages form. So we chose the following:

1. SnapChat which works on both images and videos.
2. TikTok which work only on videos.

3.2. Choosing the Filters from each Social Network:

The filters chosen should not be fancy which adds extra objects likes hearts and animation into the picture. Ideally, the fiters should modify the facial features slightly and change the skin tone , eye color and such. Finally the filters chosen were:

1. SnapChat:
   a. Face and body mellow glow which smoothens skin, adds color change/makeup and modify features and background.
   b. Fresh light: similar as above.
   c. Fresh vibes: adds luminosity and brightness

2. Tikok: (we did not have many options on TikTok for uploaded videos , we had to choose only two relevant)
   a. Belle: smoothens the skin tone ,adds blush, gives an airbrush effect, changes the eye tint a bit .
   b. Spring Glow: changes the skin tint by adding a yellowish shade, changes shade of skin and adds a glow

A glimpse of how the filtered images should look:

Figure 1. Snapchat filters

Figure 2. TikTok filters

3.3. Methodology for Database Augmentation

During the database augmentation, all the pictures went through the same process. This process is important because filtering thousands of pictures one by one manually would have been too much time consuming.

- We start with the original images from the CALFW and VIP dataset.
- Pictures are concatenated together by group of 45, in videos. It is like a video of a slide show of the 45 pictures one by one.
- These videos were then uploaded on both TikTok and Snapchat, with and without filters. Note that every video was uploaded 7 times: once on Snapchat without filter, once on TikTok without filter, then the other 5 times for the 5 filters.
- The videos are then downloaded from the social networks and re-converted to 45 images each.

This method was a huge time-saver as it allowed us to go 40 time faster than filtering the pictures one by one. Note that every picture (filtered or unfiltered) went through the same process.

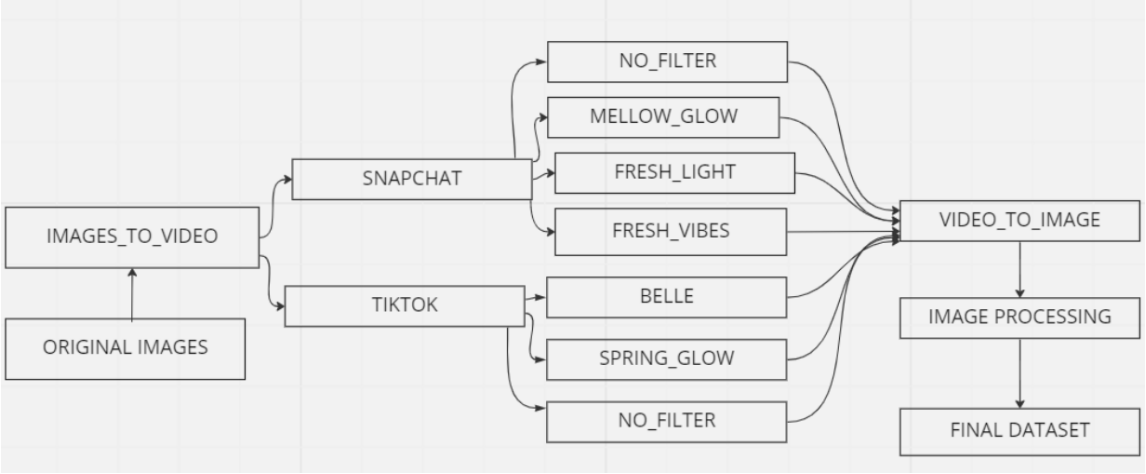

Figure 3. Data Augmentation Methodology

4. Preprocessing

For the TikTok videos we faced an issue after downloading the videos. It always had the Tiktok logo saying Tiktok and account identity in it which would cause a bias.We had to remove the logo. The logo is on every image in different position , so it was not feasible to select a particular area and crop it for all. So we decided to detect the face and crop out the part.
Note: This process was applied to all the images including the 'non filtered' images in the dataset

Figure 4. Problem with Tiktok logos

4.1. Face Detection

We chose the Haar Cascade Algorithm for face detection. It is widely used for object detection.

- The algorithm works by using a set of pre-trained Haar-like features, which are simple **rectangular filters** that are applied to an image at different scales and positions.

- Each feature is designed to detect a specific aspect of the object being searched for, such as edges, lines, or corners.

- By applying these features at different scales and positions, the algorithm can identify the object regardless of its size or location within the image.

- The Haar Cascade algorithm uses a cascade of weak classifiers (like AdaBoost) to improve the detection accuracy while minimizing computation time.

- The cascade is a hierarchical series of stages, where the 'more complex features' are evaluated later.
  each stage consists of a set of weak classifiers that make binary decisions about whether a particular feature is present or not. .

- If an image region fails a stage, it is immediately rejected, thus reducing the number of false positives.

In our case, We had quite a number of false positives, ie. two 'faces' being detected from single image, one of which is not a face at all ie. False positives. We removed experimenting with the parameters called:
a. *'min_neighbours'*: (specifies the number of neighbors a detected object candidate must have in order to be considered a true positive detection. In other words, it is the minimum number of overlapping rectangles required for a region to be considered as an object) Higher min_neighbour could be set to lower the false positives and too high would lead to reducing the true positives. For us, the optimal was 'min_neighbor' equal to 5 across all sub datasets. A glimpse of the effect of parameter on the sausets of data (these are same across different filters too)

Sub-dataset	Min_Neighbor	False_Positive	Skipped True Negative
CALFW_Female	4	24	7
CALFW_male	4	26	9
VIP	4	13	5
CALFW_Female	5	9	5
CALFW_male	4	7	5
VIP	4	5	3

b. *'min_size'*: used to fix the minimum size of the window or rectangle the object has to be inside to be detected. Our intention was to filter out the images tiny areas, covering only the eye for example. Many of the dimensions False positive rectangles were no more than (106,106), thus we chose this as our optimal value for 'min_size'.

4.2. Face Cropping

The rectangles with faces were then selected (explained in the previous section) and the image was cropped out according to the dimension of the rectangle with face. Further the image was resized to 250 X 250 and saved. A glimpse of how the final images after cropping look:

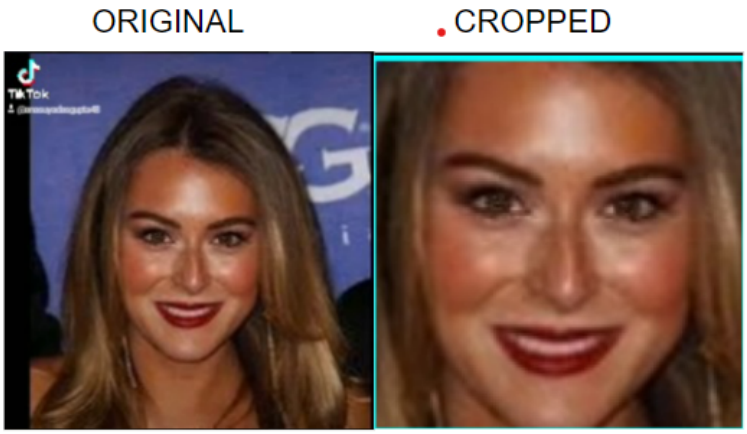

Figure 5. Original vs Cropped

5. Analysing the Difference in Images:

to do

6. First Model Tests

6.1. Different model possibilities

Our first objective was to compare different model possibilities. Indeed pytorch models can be loaded pre-trained on the ImageNet dataset. This dataset contains millions of images for multiclass object identification. We also wanted to see the difference between a finetuned model (all the weights are retrained with our dataset) and simple feature-extraction (we only retrain the classification layer). We thus tried these three different possibilities:

- A pretrained model that we finetuned.
- A pretrained model with only feature extraction.
- A model with the default pytorch weights, that we trained from scratch, entirely on our dataset.

For these tests we used the ResNet18 architecture, and as we just wanted to have a rough overview, we only used a part of our dataset for the sake of time. The results are presented in Figure 3.

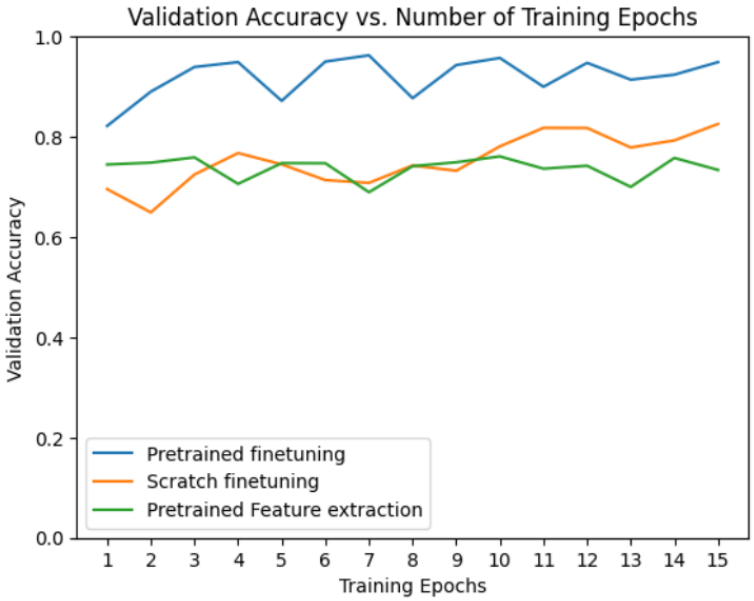

Figure 6. Comparing 3 Model Possibilities

Seeing those results we chose to use models that would be pretrained on ImageNet and then finetuned with our dataset.

6.2. Choosing a model architecture

The next step was to choose an architecture for our model. For this we compared 3 neural network architectures that are usually used and successful in image recognition tasks:

- ResNet18
- AlexaNet
- VGG (Visual Geometry Group)

VGG got less than 1% better accuracy than ResNet18 but was 2 times longer to train. We chose to use a ResNet18 architecture for the rest of the tests.

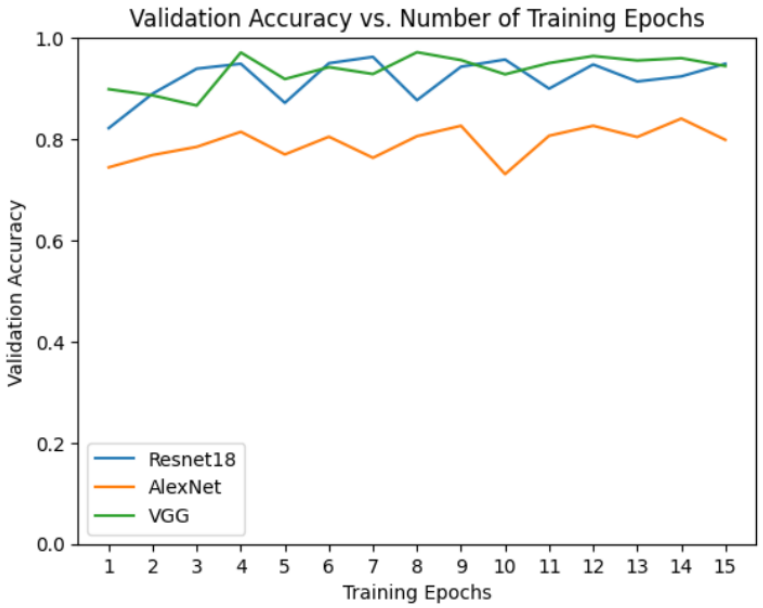

Figure 7. Comparing Architectures

6.3. Final Model Chosen

After these preliminary tests, we chose a ResNet18 architecture, pretrained on ImageNet and fine-tuned with our dataset.
We kept the default Pytorch hyperparameters as it already gave us meaningful results and we did not have time to tune them:

- Batch size: 8 (due to computational limitations)
- Training epochs: 15
- Optimizer: Stochastic gradient descent with a learning rate of 0.001 and a momentum of 0.9
- Loss: CrossEntropy

6.4. Inputs

We split our dataset for training and validation (80% / 20%), the class "unfiltered" was oversampled in the training set in order to achieve balanced class distribution.
The inputs were the cropped filtered and unfiltered pictures, as described in parts 3) and 4). They were resized to 224x224, about half were flipped horizontally, and they were normalized as per required by the pytorch model's documentation.

7. Classifying filtered and unfiltered face images

The first task was a binary classification between filtered and unfiltered pictures.

7.1. Results

Classification: not_filtered / filtered

	Overall	Male	Female
Accuracy (± 0.05) not_filtered / filtered	0.95 0.97 / 0.93	0.94 0.97 / 0.93	0.96 0.98 / 0.94

Figure 8. Classifying filtered and unfiltered face images: Results

We can see that there is no big bias between male and female which seems logical because our dataset is balanced in term of genders.
We can also highlight that filtered pictures seem harder to classify than unfiltered.

7.2. Failure to generalize

The problem with this model is that it fails to generalize on unknown filters. In order to test this, we trained the model on 7 out of our 8 filters and we tested on the last one. (ie. we took our whole dataset and removed all the pictures with one type of filter, and used them to test).

We ran this 3 times, each time trying to generalize on a filter from a different social network. The results were inconclusive as we got on average a 65% accuracy, independently of which filter was left out.
In conclusion; this model cannot generalize on unknown filters.

8. Source Social Network Identification

The second task was source social network identification with filtered pictures. This is a 3 class classification problem: Instagram, Snapchat and TikTok.

8.1. Results

First, the model was trained with ONLY filtered pictures (we removed the unfiltered pictures from the training and validation datasets).

	Overall	Insta	Snap	TikTok
Accuracy (± 0.05)	0.99	0.99	0.98	0.99

Figure 9. Source Social Network Identification with only filtered pictures: Results

The results were immediately really conclusive. The model seem to be easily able to differentiate pictures filtered from different social media.
We then tried the same experiment but with a training and validation datasets each containing about a third of unfiltered pictures: all accuracies dropped by 1 to 2%. It thus seems a bit harder for the model to classify unfiltered pictures.

8.2. Successfully generalizing

This time the model generalises successfully on unseen filters.
The same test as in 6.2 were made and this time the accuracy was between 96 and 98% on unseen filters.

9. To go further: Histograms

An idea to have better results and maybe make the model generalize for the unfiltered/filtered classification would be to do like in the second reference: providing additional inputs for each picture.
For this we explored histograms but we did not have the time to implement an architecture to test it.
The goal is to see if histograms could provide useful features to identify a filtered image from an unfiltered one. The experiment is the following:

- We computed the histogram for all the unfiltered pictures of the Snapchat dataset.
- 3 histograms were computed for each picture, one for each color channel.
- We averaged all these histogram to get an "average histogram" for each color channel.
- We did the same thing for the pictures filtered by each of the 3 Snapchat filters.
- We normalized every "average histograms" to get "normalized average histograms" for each color channel.

Here are the results for the 3 channels. "Normal" is the histogram for the unfiltered pictures. "Light", "Glow" and "Vibes" are the three filters from Snapchat:

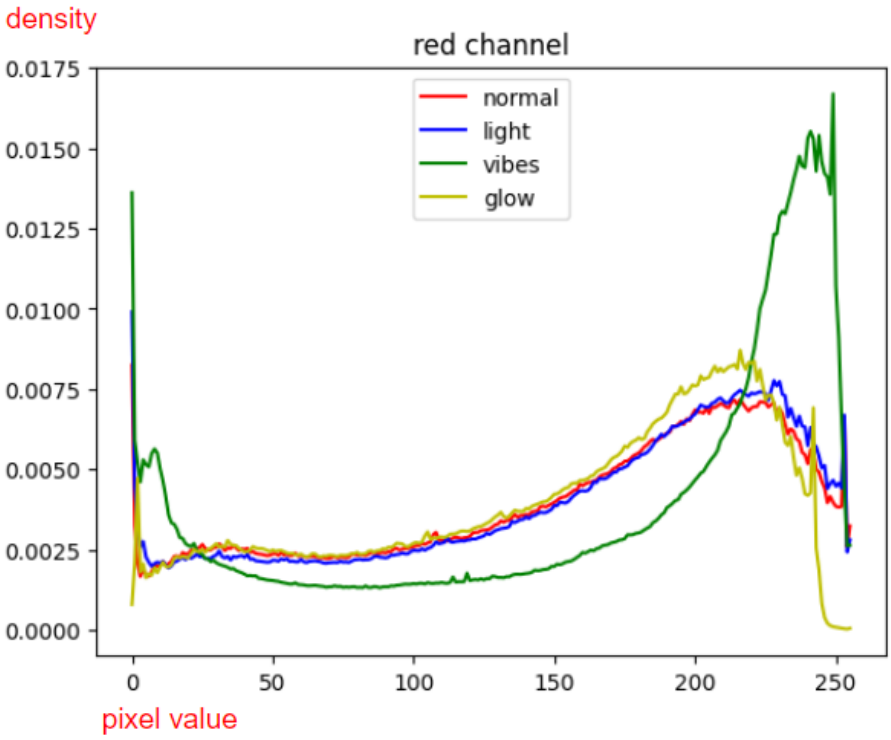

Figure 10. Normalized Average Histogram, Red channel

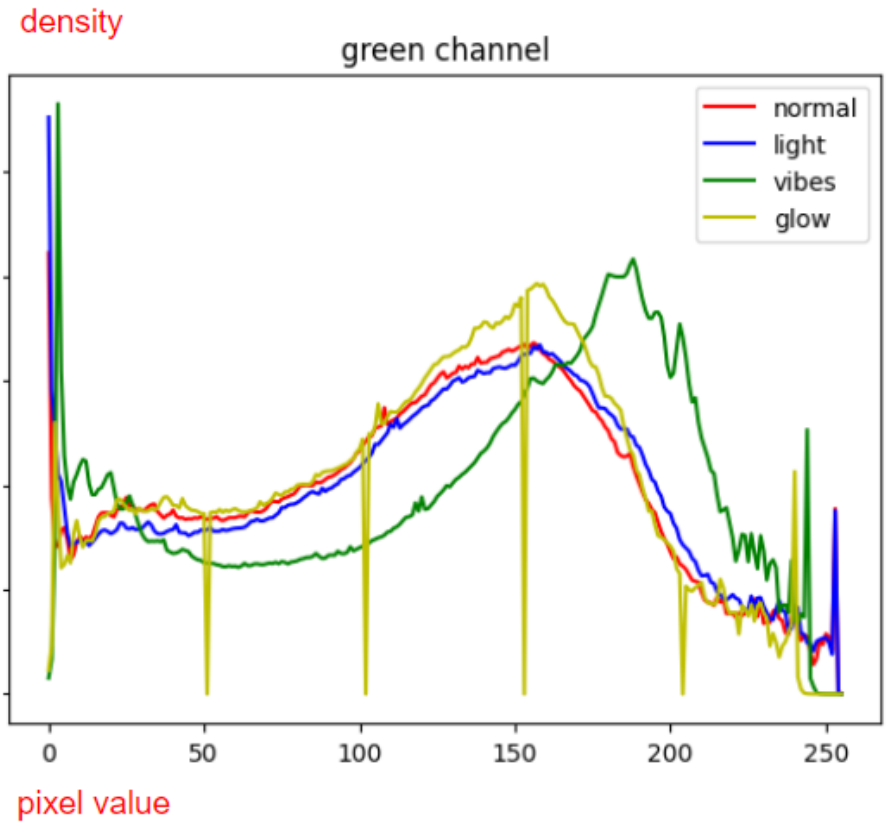

Figure 11. Normalized Average Histogram, Green channel

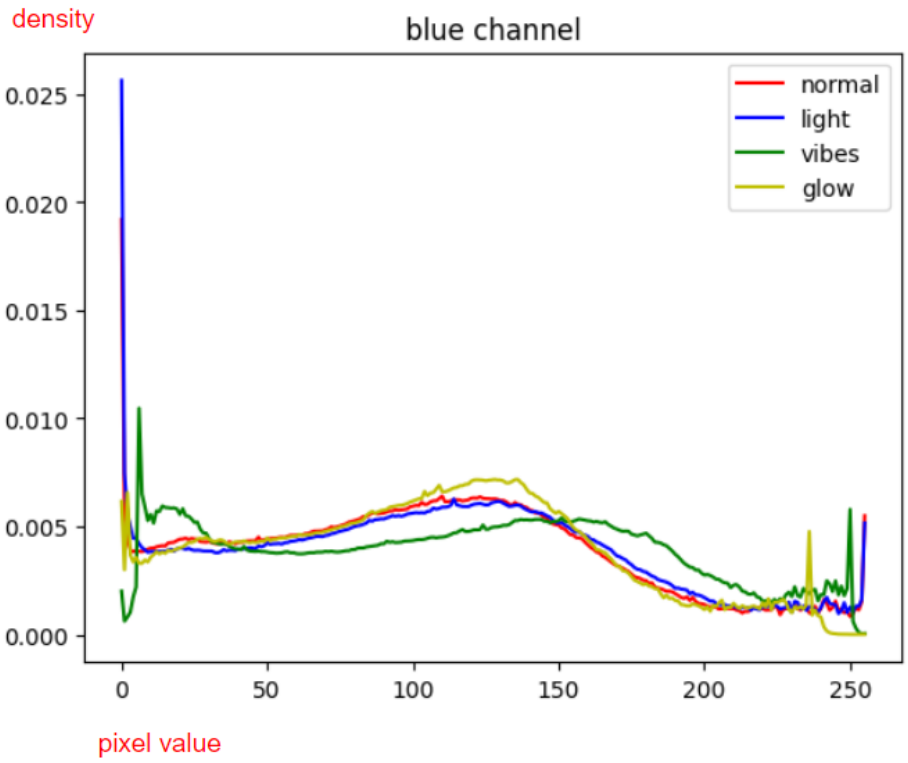

Figure 12. Normalized Average Histogram, Blue channel

We can see that each filter, on average, modifies the histogram of every channel. It could thus be interesting to use histograms as additional inputs in order to differentiate unfiltered from filtered pictures.
The idea would be similar to the model used in the second reference:

- Have 1 Neural Network with the pictures as inputs (the one we have in this project).
- Have 1 Neural Network with the histograms as inputs.
- Use these 2 Network as feature extractors.
- Concatenate the obtained features and use them as inputs for a classification Neural Network.

10. Conclusion

In conclusion, during this project:

- We augmented the dataset of the first reference by about 14 000 pictures (8000 from Snapchat, 6000 from TikTok / 10.000 filtered, 4.000 unfiltered).
- We built a model that successfully classified filtered from unfiltered pictures when it was trained on the filters, but failed to generalize on unseen filters.
- We built a model that successfully identifies the source SN of pictures and that seems to generalize on unseen filters.