INRIA Report

UnmaskMe

Marine CHARRA, Nicolas COTONI, Amaury DELPRAT
Adrien DUOT et Gireg ROUSSEL
SANTE 2022

January 25th 2022

Contents

1 Abstract **2**

2 Introduction **2**

3 Mask detection **3**

3.1 Objectives . . . 3

3.2 DarKnet . . . 3

3.2.1 Dataset . . . 3

3.2.2 Implementation . . . 3

3.2.3 Results . . . 4

3.3 YOLOv5 . . . 4

3.3.1 Dataset . . . 4

3.3.2 Implementation . . . 4

3.3.3 Results . . . 5

4 Mask segmentation **6**

4.1 Objectives . . . 6

4.2 Data . . . 6

4.2.1 MaskTheFace . . . 6

4.2.2 Augmentation . . . 6

4.3 Model . . . 7

4.3.1 UNET . . . 7

4.3.2 Loss and metrics . . . 8

4.4 Output . . . 8

5 Context conditional GAN **9**

5.1 Model . . . 9

5.2 Dataset . . . 10

5.3 Implementation . . . 10

5.4 Experiences . . . 10

5.5 Limitations . . . 11

6 StyleGAN **12**

6.1 Theory . . . 12

6.2 Experiences . . . 13

7 Real Time **14**

8 Try it **14**

9 Conclusion **15**

1 Abstract

State of the art deep learning based image processing methods have convincing results for removing small object in an simple image but fail to generate believable results for removing large objects in complex images. This holds particularly true when it come to facial images, for which we would immediately notice any abnormalities. This project aims to remove face masks, that have become omnipresent today, in facial images.

2 Introduction

Wearing masks in public has been growing in recent years all over the world, but with the Covid 19 pandemic, the number of people wearing face masks in public spaces has grown exponentially. Removing mask from people who wear it in photos and videos is a problem we face nowadays.

This problem is challenging because face mask cover a significant part of the face but also because a dataset of pair mask/maskless image doesn't exists. We break the problem into three stages: masked face detection, segmentation and in-painting of the segmented mask region.

The first stage of our pipeline uses a YOLOv5 to detected masked faces. Then the second stage uses a UNet to do the segmentation and we finish with a context conditional GAN to do the inpainting.

To train our model in a supervised learning, we create a paired synthetic dataset using publicly available CelebA dataset [1] and a MaskTheFace [2] but also small datasets from Kaggle [3].

3 Mask detection

3.1 Objectives

This first part of the project is used to locate hidden faces in an image and extract them from it to send them to the segmentation. This one was built in two parts, in two networks. The first one was found quickly after some research to allow the application to have a first functional version. The second network is up to date with the state of the art, it required additional researches to understand its operation that it is at the level of the network or the dataset that it uses.

3.2 DarKnet

3.2.1 Dataset

During our researches to find the perfect dataset, we came across different approaches.

- real datasets but not much provided in terms of images [3].
- possibilities to create artificial datasets [2].

Firstly, in order to have the possibility to train our model quickly, we started with the first solution. Then, we switched to a much larger dataset; the artificial dataset.

3.2.2 Implementation

This first version was based on the Caffe model [5]. This one allowed us to have a first iteration very quickly. We were inspired by an existing Github project [4].

Figure 1: DarkNet result

3.2.3 Results

The network used was not very convincing when we started to test it on real cases. This was surely due to the low performance of the network and the dataset containing a number of fictive images.

3.3 YOLOv5

3.3.1 Dataset

To train this new network it was necessary to go back to dataset hunting. This model works with datasets made of images and annotations, which are xml files. Our choice was drastically reduced to a dataset of 853 images and annotations [6].

3.3.2 Implementation

For the implementation of this model we relied on the Github repository of Ultralytics which is the company that created the YOLOv5; the model that we used [7]. The first part of the work was to pre-process the dataset to divide it into different sub-folders of test, train and validation.

YOLO is a Convolutional Neural Network (CNN) for performing object detection in real-time. CNNs are classifier-based systems that can process input images as structured arrays of data and identify patterns between them (view image below). YOLO has the advantage of being much faster than other networks and still maintains accuracy.

It allows the model to look at the whole image at test time, so its predictions are informed by the global context in the image. YOLO and other convolutional neural network algorithms "score" regions based on their similarities to predefined classes.

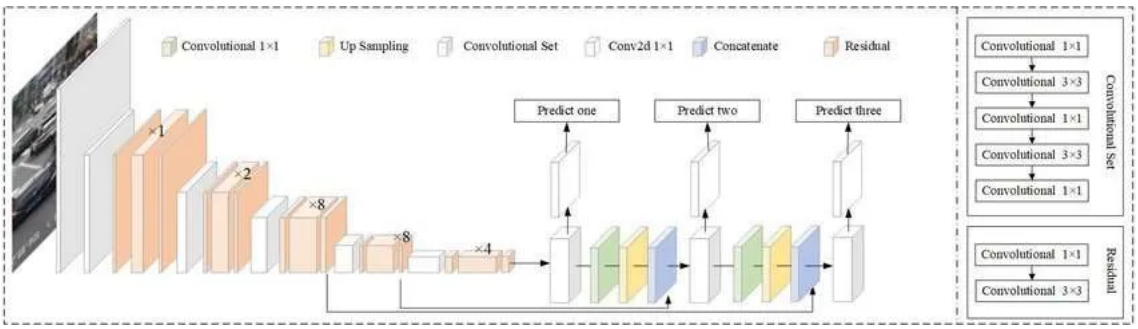


Figure 2: Yolo implementation

3.3.3 Results

In spite of the few images that our dataset contains, the results are very convincing. On Figure 3, the YOLO results allow to detect many more faces than the previous model.

Figure 3: YOLOv5 result

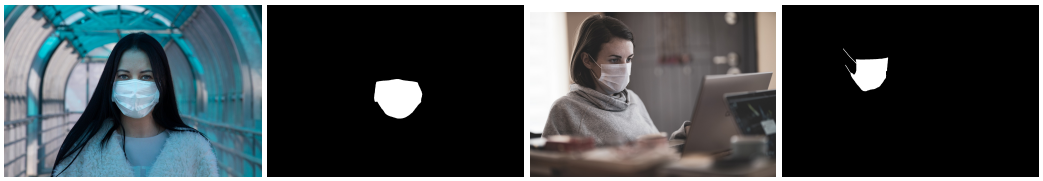
4 Mask segmentation

4.1 Objectives

The goal of this part is to detect for each pixel of the image if it is part of a mask or not.

4.2 Data

To do so, we need a dataset of persons wearing mask with its ground truth associated to locate the mask.



However, a dataset like this does not exist or has too few images and a lot of variance between them.

4.2.1 MaskTheFace

In order to replicate the desired dataset we had to create an artificial dataset using MaskTheFace a Github project that will add a covid mask on all detected faces, by detecting the key points of the face. So, by using the CelebA dataset [1], a face dataset of about 200 000 images of celebrities, we could create a large annotated dataset.

However, since the dataset is artificial, it has a bias: there is a brightness problem between the mask and the face of the person, so to increase the efficiency of our model we must increase the variance of by removing the bias through images augmentation techniques.

4.2.2 Augmentation

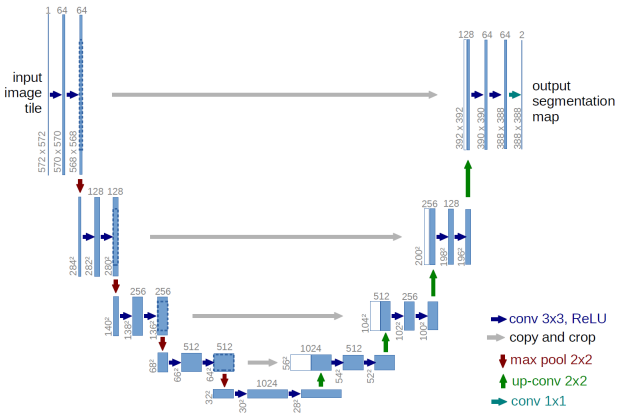
To improve our data we used several techniques, first adding a light spot.

Adding Gaussian noise so the images and masks added are less homogeneous and therefore more "natural".

4.3 Model

4.3.1 UNET

The current state of the art for segmentation is the Unet.



The network consists of a contracting part and an expansive path, which gives it a "U" shaped architecture. The contracting part is a typical convolution network which consists of a repeated application of convolutions. During contraction, spatial information is reduced while feature information is increased. The expansive path combines geographic and spatial feature information through a sequence of upward convolutions and concatenations with high resolution features from the contracting path.

4.3.2 Loss and metrics

We use two combined loss functions: the BCE (Binary cross entropy), a loss function widely used in segmentation, added to a dice function that looks at the superposition of the result with respect to the expected labels.

4.4 Output

The different steps of the segmentation from the original image to the prediction and the expected prediction are shown below.

We got a good results on artificial data (DICE : 98%) but on real data the results mostly depends on the light you have on your face.

5 Context conditional GAN

The purpose of this section is to generate the lower part of the face from the already segmented mask.

5.1 Model

This model is based on the paper Semi-supervised learning with context-conditional generative adversarial networks by Emily Denton, Sam Gross, Rob Fergus [9].

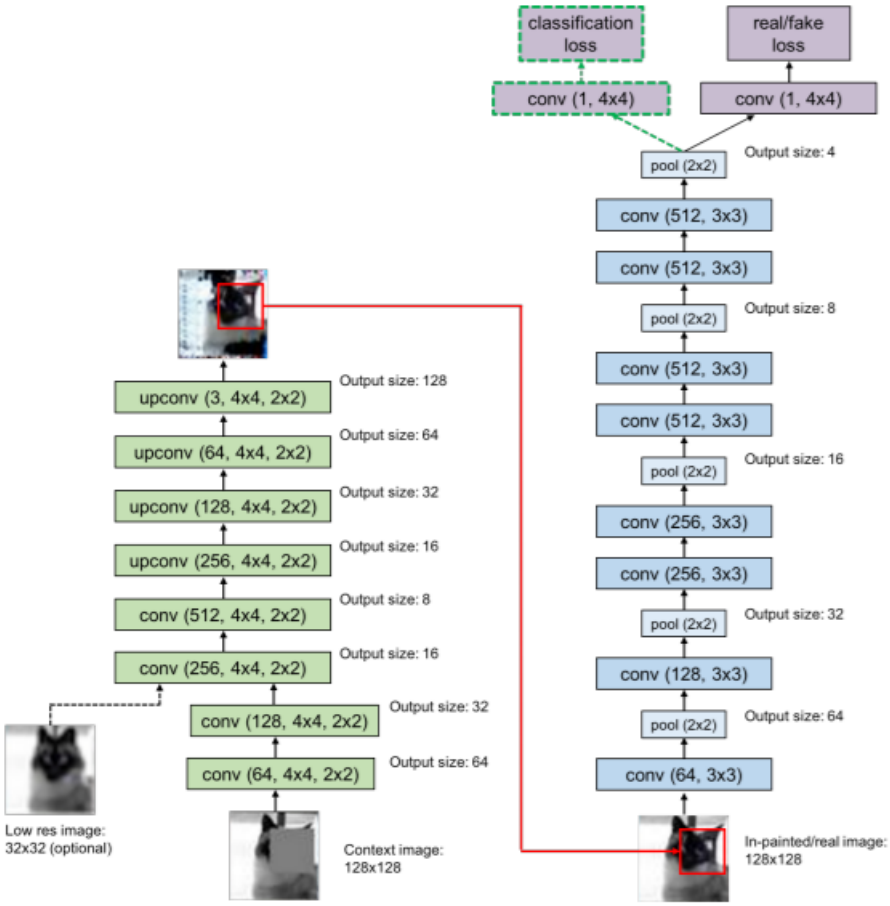


Figure 4: CCGAN Schema

On the figure 4 we see the Context conditional GAN (CCGAN) is composed of a generator and a discriminator. The generator takes an image with a context (being the hidden area) and a low resolution image. The discriminator takes the image generated and tries to determine if it is a real or fake image.

5.2 Dataset

For this section we used the CelebA dataset [1] with the segmentation of a mask on top of it to get the image with context.

Figure 5: Creation of CCGAN input

5.3 Implementation

The code for the discriminator and generator models, some datasets and the training are based on the CCGAN implementation of the paper above [9] on the GitHub Pytorch GAN by Erik Linder-Norén [10]. Instead of the reference paper, the low resolution image is not optional.

5.4 Experiences

During training we faced multiple problems. Technicals ones such as the lack of GPU and the time it takes to train the model but also issues with the result. Because of the technical problems we stopped at 110 epochs to demonstrate with something working instead of taking too much time training the model. First, the color of the output tend to become more gray. The texture of the image changes too. To solve that, we use the original image and just change pixels inside the mask with our prediction. Also, the low resolution image used in training is mandatory in our model so we had to find an average skin color to put inplace of the mask for the low resolution image.

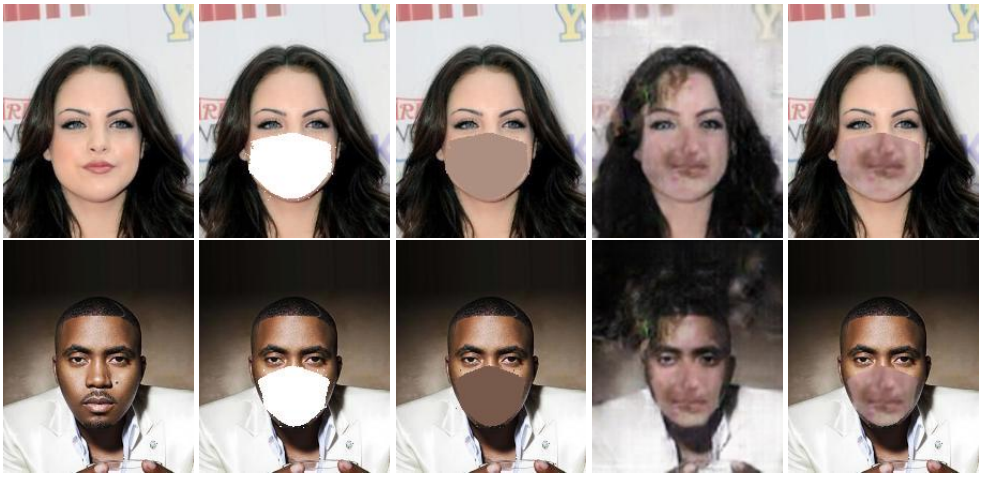


Figure 6: Images (left to right): Ground truth, input, low resolution input, prediction, output

We had other issues while training such as phosphorescent spots, face deformation and change of color. We can observe theses issues on figure 7.

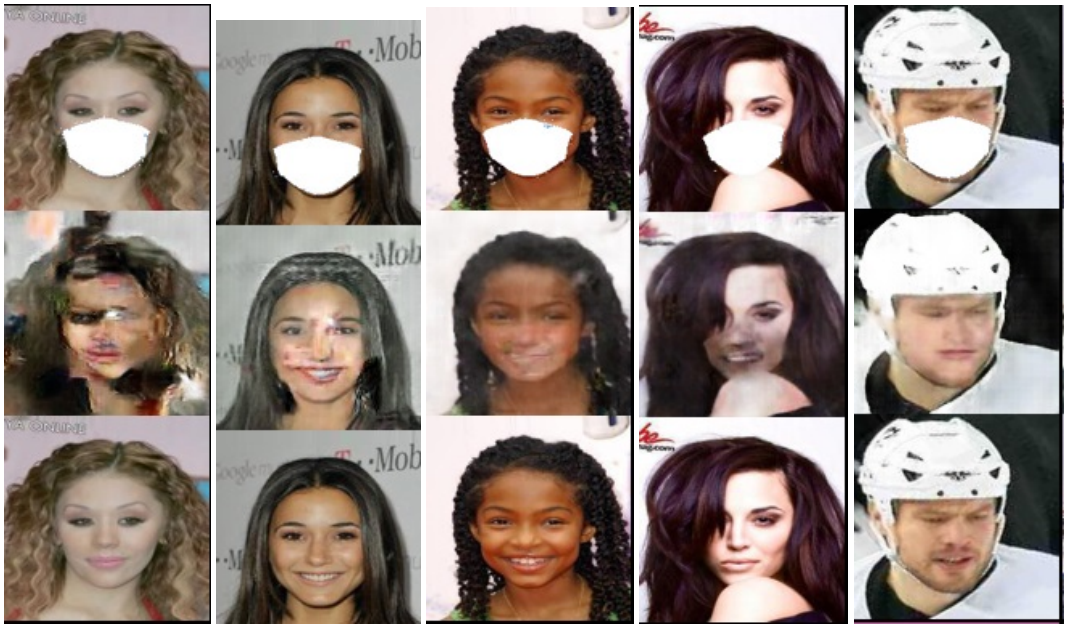


Figure 7: Images (up to down): input, prediction, ground truth, Issues (left to right): phosphorescent spots, noise, color change, color change, good prediction

5.5 Limitations

A problem with the CCGAN is that it is from a paper issued in 2016, which is quite old for GAN technology. Also, the resolution of the input of our GAN (128x128) is too small. If we had the time to train it again with powerful computers we would had take a bigger size.

6 StyleGAN

A way to improve our result in matter of generating the lower part of the face from an already segmented mask was to use the state of the art concerning GANs.

6.1 Theory

The model used is Intermediate Layer Optimization, a novel optimization algorithm for solving inverse problems with deep generative models. This implementation was made by Giannis Daras, Joseph Dean, Ajil Jalal and Alexandros G. Dimakis [11].

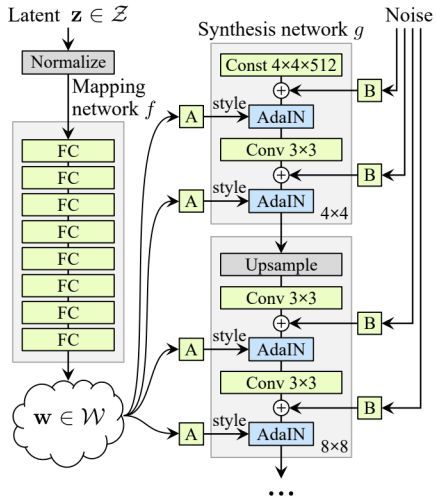


Figure 8: Summary of the StyleGAN Generator Model Architecture

The StyleGAN is described as a progressive growing GAN architecture with five modifications:

- Baseline Progressive GAN

StyleGAN's generator and discriminator models are trained using the progressive growing GAN training method. Both models start with 4x4 images, then these models are fit until stabilization, then the models are expanded to double the sizes of the image.

A block is added to the generator and the discriminator to support the new images' sizes, which is faded over training. It's then trained until the desired target image size is met.

- Addition of tuning and bilinear upsampling

Instead of using nearest-neighbor up and downsampling for the generator and discriminator the StyleGan uses bilinear sampling. The implementation is possible by lowpass filtering the activations with a separable 2nd order binomial filter after each upsampling layer and before each downsampling layer.

- Addition of mapping network and AdaIN (styles)

The next step is using a standalone mapping network that takes a randomly sampled point from the latent space as input and generates a style vector. This mapping network is composed of 8 fully connected layers.

The style vector is transformed and incorporated into each generator model's block after the convolutional layers via an operation called adaptive instance normalization or AdaIN. The AdaIN layers first standardize the output of the feature map to a standard Gaussian and then add the style vector as a bias term.

- Removal of latent vector input to generator

This change has the purpose of increasing the number of input point.

- Addition of noise to each block

Each output of convolutional layer in the synthesis network is a block of activation maps. In StyleGAN Gaussian noise is added to each of the activation maps prior to the AdaIN operations. A different sample of noise is generated for each block.

- Addition Mixing regularization

Mixing regularization involves first generating two style vectors from the mapping network. This encourages the layers and blocks to localize the style to specific parts of the model and corresponding level of detail in the generated image.

6.2 Experiences

After reading a considerable number a Git repository we did not manage to use a StyleGAN to our project. Nevertheless, we started a transition to our GAN to a StyleGAN in another non merge branch of our repository. We manage to use a StyleGAN for our subject, but the results are more funny than relevant. The problem by this approach was because the StyleGAN used was pre trained and we were not able to manage to train our dataset with it.

Results for inpainting are shown below with the ILO implementation [12].

Figure 9: Inpainting approach using ILO implementation [12]

7 Real Time

The main objective of this project was to create a real time application. To do so we use a python package call Streamlit.io [13] it make the project much more user friendly. The segmentation in a real case doesn't cover the full mask, even with a good lighting. We can guess a rather blurred mouth and nose generated by the CCGAN.

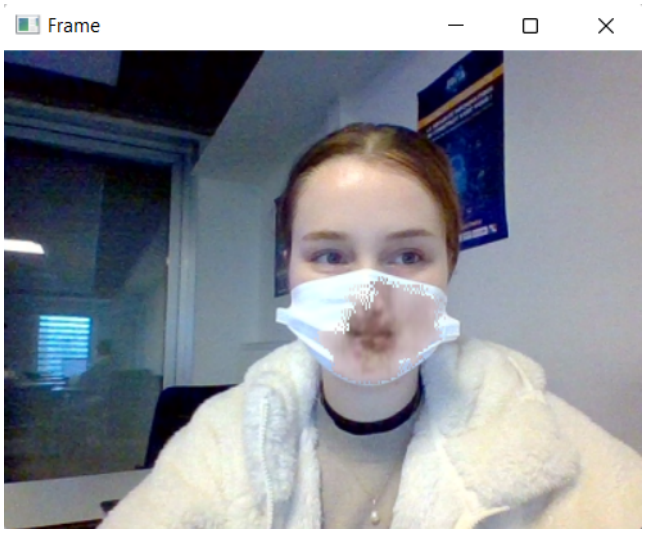


Figure 10: Real Time capture

8 Try it

You can download our code in order to try our project on GitHub : UnmaskMe.

9 Conclusion

This project proved to be very enriching as it consisted in a concrete approach of a complete AI project. Indeed, taking initiative, respecting deadlines and working in a team as well as finding documentation to answer a problem will be essential aspects of our future job.

Thanks to this project we manage to experience a few technologies as YOLOv5, Unet, CCGAN and StyleGAN.

One of the main problems we encountered was the collection of data, as it is difficult to find an annotated dataset of masked faces. Our other main problem was related to GANs, between CCGAN which was complicated to train and not up-to-date and StyleGAN which was complicated to integrate.

A way to improve our project would be to integrate the StyleGAN instead of the CCGAN and to improve our segmentation by seeking for other datasets to merge with ours.

References

[1] Large-scale CelebFaces Attributes (CelebA) Dataset

[2] Github Project to create fictive dataset

[3] Kaggle dataset to train basic model

[4] Github Project to create first project iteration

[5] Deep learning framework

[6] Dataset compatible with YOLOv5 training

[7] YOLOv5 Github Project

[8] different Kaggle projects used as examples

[9] Semi-supervised learning with context-conditional generative adversarial networks by Emily Denton, Sam Gross, Rob Fergus

[10] PyTorch GAN by eriklindernoren on GitHub

[11] Intermediate Layer Optimization for Inverse Problems using Deep Generative Models by Giannis Daras, Joseph Dean, Ajil Jalal, Alexandros G. Dimakis

[12] Official Implementation: Intermediate Layer Optimization for Inverse Problems using Deep Generative Models by giannisdaras on GitHub

[13] Python packages to create front pages

[14] UnmaskMe GitHub project