Mask inpainting with a GAN network

Luca Lumetti
244577@studenti.unimore.it

Federico Silvestri
243938@studenti.unimore.it

Matteo Di Bartolomeo
241469@studenti.unimore.it

Abstract

Our project aims to remove the surgical mask over a person's face by reconstructing the covered region. To reach our goal, we used a deep network to detect landmarks over the face, a classical pipeline to segment the surgical mask and produce as output a binary mask. Both original image and mask are then fed through the network which will output the reconstructed face. It is also possible to have a second photo of the same person without the surgical mask that can be used as a reference. From this photo, a small region of the cheek and mouth is extracted and warped over the original image, the mask is then adjusted to match the new image. This reference photo should provide to the network more information about the face thus leading to a more loyal reconstruction. The source code of the whole project is available at https://github.com/LucaLumetti/CVProject/

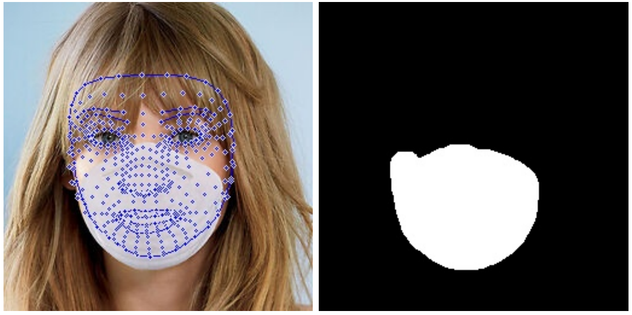
1. Related works

A lot of works on surgical mask segmentation have been made over the last year, since the start of the Covid-19 pandemic, but all of them were completely based only on end-to-end deep neural networks. About the task of image inpainting, a lot of research has been done. Most of the works were about natural images inpainting, [15] [7], and facial inpainting [2] [10] [9]. The mask can be restricted to a fixed form (usually a square) or be a free-form without any particular constraint.

2. Mask Segmentation

We made use of MediaPipe's FaceMesh [5] library to find facial landmarks over the face covered with the surgical mask and the reference photo. Facial landmarks are important to have an initial approximation of the region where to search the surgical mask and to warp the reference photo over the first one. To perform the segmentation of the mask we apply a k-means with k=3 over the polygon we created using specific face landmarks and pick the bigger region between the 3. The k has been chosen to be 3 as in the polygonal region we expect to find the mask, the background and the skin of the person's face. In the end, a binary image is created, with a 1 where the mask is present and 0 elsewhere, while in the original image, the mask area is filled with 0s.

Figure 1. Frontal image with landmarks on the left, resulting mask of the surgical mask on the right.



3. Warping the reference photo

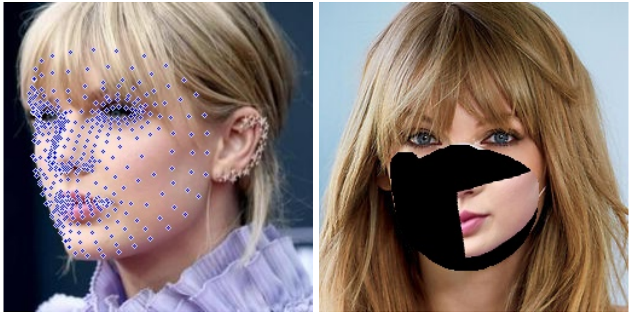
The objective of the reference photo is to guide the network to a more loyal reconstruction, by giving some information on how the mouth and close parts should look. As we allow the reference to have a yawn different than frontal, we apply a thin-plate spline transformation to adjust it. We use 30 specific landmarks as parameters as using more of them lead to distortions given by the errors in the landmarks detection while less lead to an imperfect warping. The same polygon region of Mask Segmentation is cut from the reference photo, then by applying the TPS it is stuck to main photo leading to a (partial) reconstruction. The image will then be fed to the network which will reconstruct the missing parts.

4. Image inpainting

Image inpainting is the task to fill a missing region in an image by predicting the value of the missing pixels in order to have a realistic image which is semantically close to the original one and visually correct. There are two different approaches to achieve this task:

1. Low-level feature patch-matching which does not work pretty well with non-stationary use-cases (e.g. faces, objects or complicate scenes);

Figure 2. Reference photo with landmarks found by mediapipe on the left, resulting image on the right.



2. Feed-forward models with deep convolutional networks which overcome the problem of previous case exploiting semantics learned on large scale dataset.

5. Our approach

We decided to follow the latter one designing a coarse-to-fine Generative Adversarial Network (GAN) characterized by:

- Generator is made of a coarse network, whose aim is to provide a rough estimation of the missing region, and a refinement network which takes the output of the previous network and the binary mask as input and improve the coarse output by making it more detailed.
- Discriminator which is responsible of distinguishing real samples from the one created by the generator.

The input of the network is a pre-processed RGB image so that its values are in range $[-1, +1]$ and its binary mask. For initializing the starting values of the network weights, we opted for Kaiming initialization method. We provided different methods to initialize the starting values of the network weights, such as normal, Xavier, orthogonal and, the default one, Kaiming. We used Adam as the generator and discriminator optimizer using a 0.5 momentum and different learning rates, respectively 0.0001 and 0.0004 for 10 epochs.

5.1. Losses

Our final loss function is given by the sum of six different losses:

$$\begin{aligned}\mathcal{L}_{tot} = \mathcal{L}_{adv} + \mathcal{L}_{recon} + \mathcal{L}_{tv} + \mathcal{L}_{contra} \\ + \lambda_{perc} \cdot \mathcal{L}_{perc} + \lambda_{style} \cdot \mathcal{L}_{style}\end{aligned} \tag{1}$$

Specifically, the weights λ used are: $\lambda_{perc} = 0.05$, $\lambda_{style} = 80$ and $\lambda_{adv} = 0.1$.
In the following formulas we will use symbols to refer to specific elements of the loss function, $\mathbf{I}_{in}$ is the masked image, $\mathbf{I}_{gt}$ is the reference ground truth image, $\mathbf{I}_{out}$ is the output of the refinement stage, $\mathbf{I}_{com}$ is the masked image where masked pixels are replaced with $\mathbf{I}_{out}$.

5.1.1 Adversarial Loss.

$$\mathcal{L}_{gen} = -\mathbb{E}_{\mathbf{I}_{in} \sim \mathbb{P}_i}[D(\mathbf{I}_{in}, \mathbf{I}_{com})] \tag{2}$$

$$\begin{aligned}\mathcal{L}_{discr} = \mathbb{E}_{\mathbf{I}_{in} \sim \mathbb{P}_i}[\mathrm{ReLU}(1 - D(\mathbf{I}_{in}, \mathbf{I}_{gt}) \\ + \mathrm{ReLU}(1 + D(\mathbf{I}_{in}, \mathbf{I}_{com})]\end{aligned} \tag{3}$$

where $\mathbb{P}_i$ is the data distribution of $\mathbf{I}_{in}$, *D* and *G* are, respectively, the discriminator and the generator and ReLU is the rectified linear unit defined as $f(x) = \max(0, x)$.
This is also know as GAN Hinge loss for generative adversarial learning, where discriminator is trained to distinguish $\mathbf{I}_{com}$ from $\mathbf{I}_{gt}$ and generator has the aim of cheating the classification of the discriminator.

5.1.2 Reconstruction Loss.

$$\mathcal{L}_{recon} = |\mathbf{I}_{gt} - \mathbf{I}_{coarse}| + |\mathbf{I}_{gt} - \mathbf{I}_{com}| \tag{4}$$

Is the standard L1 loss applied on both coarse and refined images.

5.1.3 Total variation (TV) Loss.

$$\begin{aligned}\mathcal{L}_{tv} = \sum_{x,y}^{H-1,W} \frac{\left\|\mathbf{I}_{com}^{x+1,y} - \mathbf{I}_{com}^{x,y}\right\|^2}{N_{\mathbf{I}_{com}}^{row}} \\ + \sum_{x,y}^{H,W-1} \frac{\left\|\mathbf{I}_{com}^{x,y+1} - \mathbf{I}_{com}^{x,y}\right\|^2}{N_{\mathbf{I}_{com}}^{col}}\end{aligned} \tag{5}$$

where *H* and *W* are the height and width of $\mathbf{I}_{com}$ and $N_{\mathbf{I}_{com}}^{row}$ and $N_{\mathbf{I}_{com}}^{col}$ are the number of pixels in $\mathbf{I}_{com}$ without the last row and the last column.
It is responsible for the regularization of the image to improve the smoothness of the output image.

5.1.4 Perceptual Loss.

$$\mathcal{L}_{perceptual} = \sum_{l=1}^{L} \frac{\left\|\phi_l^{\mathbf{I}_{com}} - \phi_l^{\mathbf{I}_{gt}}\right\|_1}{N_{\phi_l^{\mathbf{I}_{gt}}}} \tag{6}$$

where ϕ is the well-trained loss network, VGG-19[13], and $\phi_l^{\mathbf{I}}$ the activation maps of the l^{th} layer of ϕ given an image $\mathbf{I}$. $N_{\phi_l^{\mathbf{I}_{gt}}}$ denotes the number of elements in $\phi_l^{\mathbf{I}_{gt}}$ and *L* is the number of layers used. This loss represents the L1-norm distance between high-level feature representations in 5 different convolutive layers.

5.1.5 Style Loss.

$$\mathcal{L}_{style} = \sum_{l=1}^{L} \left\| \frac{1}{C_l H_l W_l} ((\phi_l^{\mathbf{I}_{com}})^\top (\phi_l^{\mathbf{I}_{com}}) - (\phi_l^{\mathbf{I}_{gt}})^\top (\phi_l^{\mathbf{I}_{gt}})) \right\| \tag{7}$$

where C_l refers to the number of activation maps of the l^{th} layer of ϕ and H_l and W_l are its height and width respectively. With $(\phi_l^{\mathbf{I}})^\top (\phi_l^{\mathbf{I}})$ we represented the auto-correlation matrix, the Gram matrix[1] which computes the features correlations between each activation map of the l^{th} layer of ϕ given the image $\mathbf{I}$.
Using the same 5 levels of the previous loss, it is the sum of the distances of the auto-correlation matrixes between the output and the ground truth multiplied by a factor that depends on the size and number of the activation maps in those layers.

5.1.6 Contrastive loss.

$$\mathcal{L}_{contrastive} = -\log \frac{exp(z_i^\intercal z_i^{'}/\tau)}{\sum_{j=0}^{K} exp(z_i z_j^{'}/\tau)} \tag{8}$$

The equation (8) is the categorical cross-entropy of classifying the positive sample correctly [12]. z_i is the encoded version of the image, τ is a hyper-parameter that controls the sensitivity of the product and it's called temperature. The dot product between the encoding vector of the original image and the transformated image measure the similarity between representations. In our network the positive images are created by the original images with some transformations. This methods try to maximize the similarity between representations of positive similar pairs and minimize the similarity with the feature extracted from negative images[6]. To calculate this loss we use the feature vector with the biggest number of channels in our network (512). We squeeze this vector in a way that preserves the information of the original image with average pooling so we use a vector with dimension 1x1x512. The transformations used are inspired by StyleGAN[4] and they are:

- horizontal flip with probability 1.0
- change in brightness with probability 0.75
- change in contrast with probability 0.75
- change in saturation with probability 0.75
- hue rotation with probability 1.0

This loss and these transformations are been proven to be really effective in our case as the network was overfitting the skin color, the mouth type and its position.

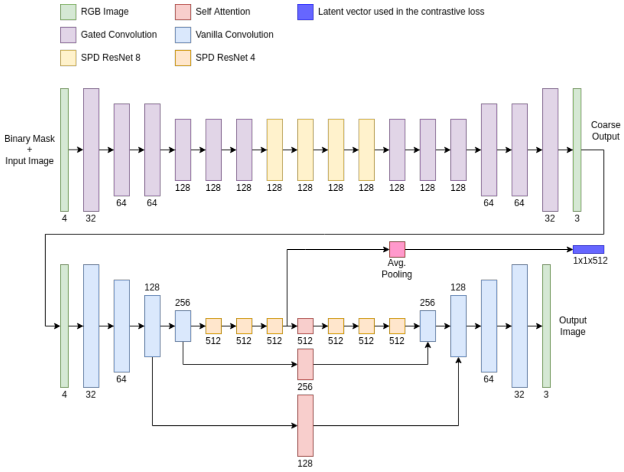


Figure 3. Generator network

5.2. Datasets

GAN networks are data-hungry and need a lot of diverse training examples in order to generate quality images, for this reason we used the FFHQ 1024x1024 images [3], rescaled to 256x256. In other GAN inpainting architectures, the mask region to reconstruct is usually calculated during the training in a randomized way. As we do not need this randomization process, for each image of FFHQ we precalculated the face region where the mask is worn using facial landmarks. To test our network, instead, we use CelebA256 dataset.

5.3. Architecture

Our architecture is inspired by Free Form Image Inpainting with Gated Convolution [15] and DeepGIN [7]. A mixed-precision training is used to train the network, to be more precise, **O2** option of nvidia-apex (https://nvidia.github.io/apex/amp.html#o2-almost-fp16-mixed-precision) has been used on both discriminator and generator. Our generator net is composed of two stages: Coarse Network and Refine Network.

5.3.1 Coarse Network

In this stage we decided to use the gated convolution [15] so that the generator is able to learn a dynamic feature selection mechanism for each channel and for each spatial location. The feature selection mechanism takes into account not only the background and the mask given in input, but also the semantic segmentation in some channels.
As shown in figure 3 the coarse net is characterized by an initial downsampling phase, followed by a residual one and at the end there is an upsampling phase using the dilated gated convolution that could be seen as a gated convolution operation preceded by a resize operation. The output of the

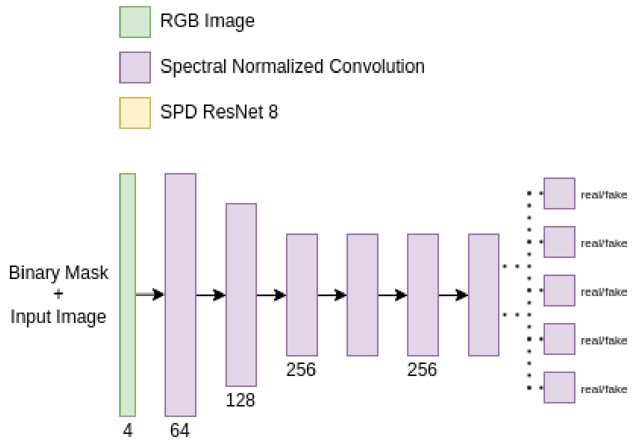


Figure 4. Discriminator network, GAN loss is calculated on every patch of the last vector

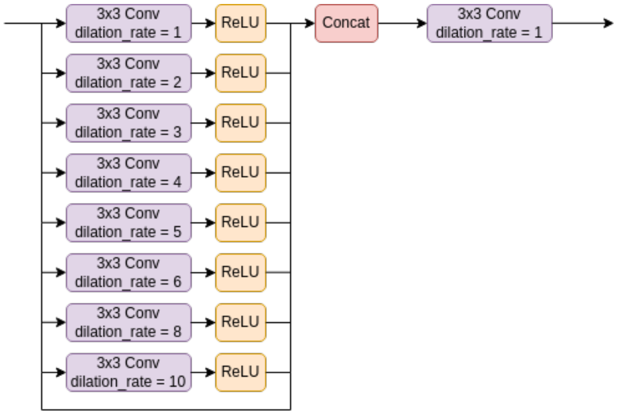

3x3 Conv dilation_rate = 1
3x3 Conv dilation_rate = 2
3x3 Conv dilation_rate = 3
3x3 Conv dilation_rate = 4
3x3 Conv dilation_rate = 5
3x3 Conv dilation_rate = 6
3x3 Conv dilation_rate = 8
3x3 Conv dilation_rate = 10
ReLU
ReLU
ReLU
ReLU
ReLU
ReLU
ReLU
ReLU
Concat
3x3 Conv dilation_rate = 1

Figure 5. ResNetSPD block with 8 different dilation rate used in the coarse network

coarse net will be multiplied with the mask and added to the original image, then fed in the refine net.

5.3.2 Refine Network

The refine network takes the output of the coarse net and the mask as input. In this stage there are 6 custom ResNet blocks with four different dilation rates and some gated convolutional layers. This modified ResNet blocks are called Spatial Pyramid Dilation (SPD) (Figure 5). This layer is composed of different convolutional blocks with different dilation, and the output of these blocks is concatenated together. With different value of dilation rate we can take information from a bigger receptive field. Another useful feature that is implemented in this net is the Multi Scale Self Attention (MSSA). The MSSA uses the self-similarity between different layers and it's helpful to have better coherence in the final image. We control the self-similarity with three different scales: 16x16, 32x32, 64x64. The central layers are composed of self attention block. We use a standard convolutional layer to reduce the size before the self attention. In this manner we avoid an excessive increase of the parameters. With self attention we can find a better correlation between features and have a better reconstruction.

5.3.3 Discriminator

Our discriminator is composed of 6 convolutional blocks with kernel size 5 and stride 2. These convolutional methods allow to capture the Markovian patches that represents better contextual feature[8]. We add a spectral normalization to improve the training stabilization[11]. The input of this network is the image (real or fake) and the relative mask.

$$\begin{aligned}\mathcal{L}_{D^{sn}} &= \mathbb{E}_{x\sim\mathbb{P}_{data(x)}}\left[ReLU(1 - D^{sn}(x))\right] \\ &+ \mathbb{E}_{z\sim\mathbb{P}_{data(x)}}\left[ReLU(1 + D^{sn}(G(z)))\right]\end{aligned} \tag{9}$$

where D^{sn} is the spectral-normalized discriminator and G is the generator that create the image z.

5.4. Multi-GPU and Multi-Node Training

As GAN networks are heavy to train, so we decided to design the whole network to be trainable on multi GPU and multi node architectures. Training it on a single K80 was infeasible as a batch size greater than 2 gives CUDA out of memory error. We end up using 4 GPU K80 in parallel that let us to complete a full epoch in under 24h (which is the time limit on AImageLab's servers). We also give support for multi node because even if we were on the same node. We initially could use at most 2 GPUs per account, so every account could be seen as a different node thus we could break the GPUs limit. Anyway as our max GPU per user was upgraded to 4, we didn't use that technique.

5.5. Results

To evaluate the results (Table 1) we use different metrics: PSNR, SSIM, LPIPS and FID. We conducted the test with CelebA 256 Dataset and compared the results with different inpainting network. It must be noted that our network is the only one who is restricted to inpaint a specific part of an image of a human face, while others generalize to the whole image.

It must be noted that these results were got by using different datasets, we just wanted to have a rough estimation of our distance from the state-of-the-art networks. Results are not that bad, FID is pretty high and we think that this is due to some artifacts that often appear in the generated regions. In the Table 5.5 we show some of the reconstructed images.

Method	PSNR	SSIM	FID	LPIPS
Iizuka et al. [2]	18.62	0.69	n.d.	0.51
SymmFCNet [9]	27.81	0.97	n.d.	0.61
DeepFillv2 [15]	29.70	0.72	75.56	0.29
DIP [14]	25.46	0.85	n.d.	n.d.
Ours	58.95	0.88	151.95	0.23

Table 1. Comparison with different network **on different datasets** on the same metrics

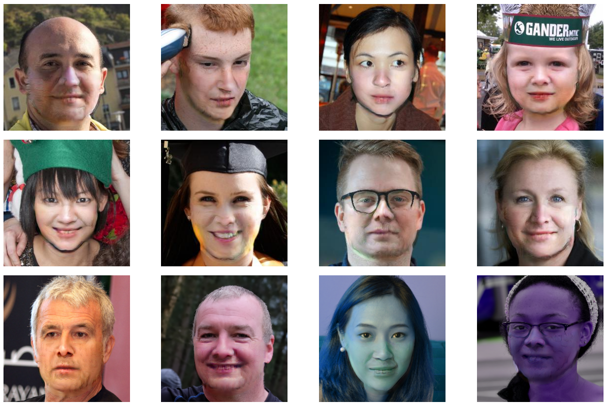


Table 2. Some of the generated images, the latter two were augmented before feeding them to the network.

5.6. Conclusion

Our results are not too far to the state-of-the-art and we think that they can be improved by training the network for more time, with more data and a larger variety of masks. Some of the networks we have compared had 1̃0x more epochs and used a lot more training data (CelebA + FFHQ + StyleGAN). We couldn't afford that as the time and computing resource were not feasible for us.

What make this network different from previous works is the specific task (inpainting the surgical mask region) and the use of a contrastive loss to prevent overfitting.

References

[1] LA Gatys, AS Ecker, and M Bethge. A neural algorithm of artistic style. arxiv preprint (2015). *arXiv preprint arXiv:1508.06576*.

[2] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017.

[3] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[4] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[5] Yury Kartynnik, Artsiom Ablavatski, Ivan Grishchenko, and Matthias Grundmann. Real-time facial surface geometry from monocular video on mobile gpus. *CoRR*, abs/1907.06724, 2019.

[6] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 2020.

[7] Chu-Tak Li, Wan-Chi Siu, Zhi-Song Liu, Li-Wen Wang, and Daniel Pak-Kong Lun. Deepgin: Deep generative inpainting network for extreme image inpainting. In *European Conference on Computer Vision*, pages 5–22. Springer, 2020.

[8] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European conference on computer vision*, pages 702–716. Springer, 2016.

[9] Xiaoming Li, Guosheng Hu, Jieru Zhu, Wangmeng Zuo, Meng Wang, and Lei Zhang. Learning symmetry consistent deep cnns for face completion. *IEEE Transactions on Image Processing*, 29:7641–7655, 2020.

[10] Yijun Li, Sifei Liu, Jimei Yang, and Ming-Hsuan Yang. Generative face completion. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3911–3919, 2017.

[11] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[12] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[15] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4471–4480, 2019.