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 a face mask over a person's face, by reconstructing the covered part of the face. To have a more precise reconstruction of the missing parts (mouth and nose) behind the mask, we plan to use a second photo of the same person without the mask as a reference during the facial reconstruction process. There are no constraints on the features of the reference photo, for instance the face can have a different yaw than the first one. To sum up the pipeline, given as input an image containing a person's face partially covered by a medical mask and another photo of the same person without any occlusion, the output will be the first image with the mask-covered parts, mouth and nose, reconstructed. Future development could lead to generalizing the occlusion caused by the mask to any type of occlusion possible.

1. Mask Segmentation

We made use of MediaPipe's FaceMesh [4] 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 choosen 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.

2. Warping the reference photo

The objective of the reference photo is to guide the network to a more loyal reconstruction, by giving some informations 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

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

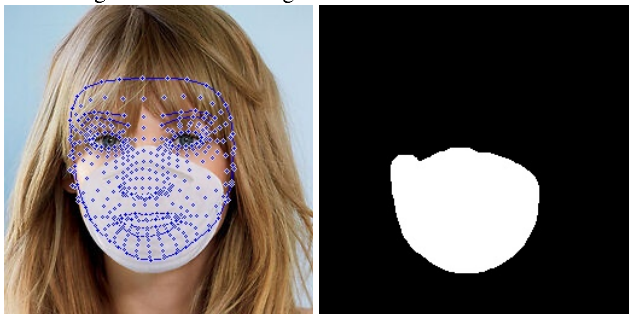
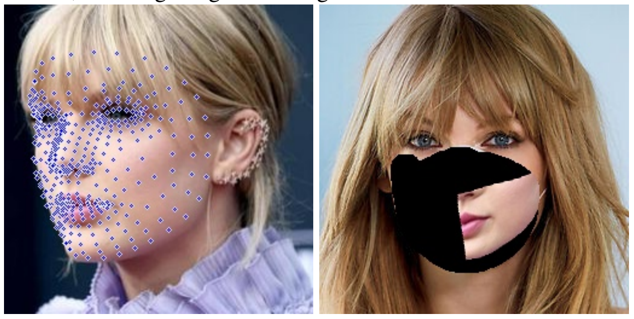


Figure 2. Reference photo with landmarks found by mediapipe on the left, resulting image on the right.



detection while less lead to an imperfect warping. The same polygon region of Mask Segmentation is cut from the reference photo, the by applying the TPS it is sticked to to main photo leading to a (partial) reconstruction. The image will then be feeded to the network which will reconstruct the missing parts.

3. Image inpainting

Image inpainting (a.k.a. image completion) 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 approches 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);

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

4. Our approach

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

- Generator:
  - Coarse network whose aim is to provide a rough estimation of missing pixels;
  - Refinement network which takes the output of the previous network as input and takes care of its detailed decoration.
- 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 weigths, 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 went for Adam as the generator and discriminator optimizer using a 0.5 momentum and different learning rates, respectively 0.0001 and 0.0004 for the first 10 epochs and then they will linearly decrease for 5 more epochs.

4.1. 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 $\mathbf{I}_{out}$.

4.1.1 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 $f(x) = \max(0, x)$.
It is the 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.

4.1.2 Reconstruction Loss.

$$\mathcal{L}_{recon} = \lambda_{hole}\mathcal{L}_{hole} + \mathcal{L}_{valid} \tag{4}$$

where $\mathcal{L}_{hole}$ is the sums of the distances calculated only from the missing pixels, $\mathcal{L}_{valid}$ is like $\mathcal{L}_{hole}$ but for valid pixels. λ_{hole} is a weight to the pixel-wise loss withing the missing regions.

4.1.3 Total variation (TV) Loss.

$$\begin{aligned}\mathcal{L}_{tv} = \sum_{x,y}^{H-1,W} \frac{\left\|\mathbf{I}_{com}^{x+1,y} - \mathbf{I}_{com}^{x,y}\right\|_1}{N_{\mathbf{I}_{com}}^{row}} \\ + \sum_{x,y}^{H,W-1} \frac{\left\|\mathbf{I}_{com}^{x,y+1} - \mathbf{I}_{com}^{x,y}\right\|_1}{N_{\mathbf{I}_{com}}^{col}}\end{aligned} \tag{5}$$

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 of the regularization of the image to improve the smoothness of the output image.

4.1.4 Perceptual Loss.

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

where ϕ is the well-trained loss network, VGG-19[10], and $\phi_l^{\mathbf{I}}$ the actiovation 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. Its weight is set to 0.05.

4.1.5 Style Loss.

$$\begin{aligned}\mathcal{L}_{style} = \sum_{l=1}^{\mathbf{I}_{out},\mathbf{I}_{com}} \sum_{l=1}^{L} \frac{1}{C_l C_l} \Bigg\| \frac{1}{C_l H_l W_l}((\phi_l^{\mathbf{I}})^{\top}(\phi_l^{\mathbf{I}}) \\ - (\phi_l^{\mathbf{I}_{gt}})^{\top}(\phi_l^{\mathbf{I}_{gt}}))\Bigg\|\end{aligned} \tag{7}$$

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. Its weigth is set to 40.

4.1.6 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 [9]. $(z_i, z_i^{'})$ are the encoding version of the images (x_q, x_k), where $x_q$is the original image and x_k is the trasformated image. τ is an hyper-parameter that control the sensitivity of the product and it's called temperature. The dot product between the encoding vector of the original image and the traformated image measure the similarity between rappresentations. In our network the positive images are created by the original images with some transformations and we don't use negative examples. This loss is helpful to get more information during the training because the network learns from the similarity between images. This methods try to maximise similarity between representations of positive similar pairs and minimises the similarity with the feature extracted from negative images[5]. 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 preserve the information of the original image with average pooling so we use a vector with dimension 1x1x512. The transformations used are inspired by StyleGAN[3] 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

These transformations are been proven to be effective in our case as the network was overfitting the skin color and the mouth position; some image in the FFHQ dataset are from people with painted face (blue and red), when the network was feeded with these images, the inpainted region wasn't of the same color of the face. After implementing augmentation and contrastive loss, these problems disappeared.

4.2. Datasets

GAN networks are data-hungry and needs a lot of diverse training examples in order to generate quality images, for this reason we used the FFHQ 1024x1024 images [2], 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 weared using facial landmarks. To test our network, instead, we use CelebA256 dataset in order to compare our results with DeepGIN.

4.3. Architecture

Our architecture is highly inspired by Free Form Image Inpainting with Gated Convolution [11] and DeepGIN [6]. 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.

4.3.1 Coarse Network

In this stage we decided to use the gated convolution 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. Furthermore using gated convolutive layers we can avoid the inner drawbacks of vanilla and partial convolution. In fact taking a look to the vanilla convolution formula:

$$O_{y,x} = \sum_{i=-k_h'}^{k_h'} \sum_{j=-k_w'}^{k_w'} W_{k_h'+i,k_w'+j} \cdot I_{y+i,x+j} \tag{9}$$

where $O_{y,x}$ is the output, x, y represent x-axis and y-axis of the output map, k_h and k_w is the kernel size, $k_h' = \frac{k_h-1}{2}$,$k_w' = \frac{k_w-1}{2}$, $W \in \mathbb{R}^{k_h \times k_w \times C' \times C}$ represents the convolutional filters and $I_{y+i,x+j}$ is the input image, we can notice that it considers all pixels valid and it is applied to the entire input image. This cause color discrepancy and blurriness in final output image.
In partial convolution, thanks to a masking and renormalization step, the operation depends only on valid pixels:

$$O_{y,x} = \begin{cases} \sum\sum W \cdot (I \odot \frac{M}{sum(M)}), & \text{if sum(M)} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

M is the binary mask where a pixel with a value of 1 is considered valid and invalid with a value of 0 and $\odot$ is the element-wise multiplication. There is a mask-update step based on the rule: $m'_{y,x} = 1, \iff sum(M) > 0$
This operation, however, has some problems:

- It will set to one a pixel in next layer no matter the number of 1-value-pixels covered by the filter range in the previous layer;
- Invalid pixels will progressively fade out from the mask going deeper in the network layers;
- All channel in each layer shares the same mask limiting the flexibity of the model.

Gated convolution, instead, is based on the following formula:

$$Gating_{y,x} = \sum\sum W_g \cdot I \tag{11}$$

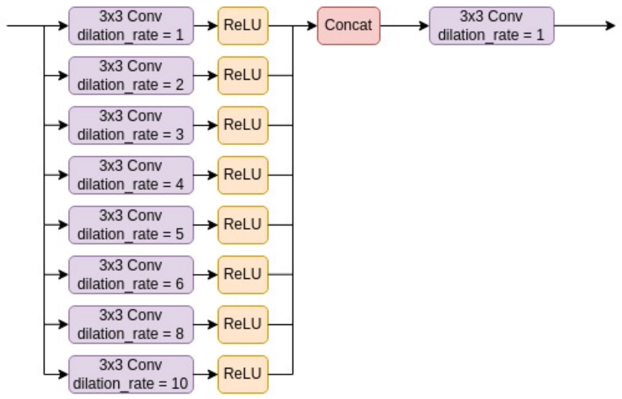
$$Feature_{y,x} = \sum\sum W_f \cdot I \tag{12}$$

$$O_{y,x} = \phi(Feature_{y,x}) \odot \sigma(Gating_{y,x}) \tag{13}$$

where there is sigmoid function σ to have the output in the range $[0, 1]$, while ϕ represents an activation function such ReLU, ELU and LeakyReLU (we used the latter). W_g and W_f are two different convolutional layers.
As said before the benefits of using this operation is that the network is able to learn a mechanism to select feature dynamically for each channel and each spatial location considering also the semantic segmentation in some channels.
As shown in **[Inserire riferimento alla figura]** the coarse net is charaterized by an initial downsampling phase, followed by a convolutional one and at the end there is and 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 coarse net will pass through an activation function (we chose a $\tanh$) and the result will be given as input to the refinement network.

4.3.2 Refine Network

The second generator take the output of the coarse net and the mask and it's useful for refinement the image. In this stage there are 4 custom ResNet blocks with different dilation and some gated convolutional layers. This modified ResNet blocks are called Spatial Pyramid Dilation (SPD) (Figure 3). 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 using the self-similarity between different layers and it's helpful


3x3 Conv dilation_rate = 1
ReLU
Concat
3x3 Conv dilation_rate = 1
3x3 Conv dilation_rate = 2
ReLU
3x3 Conv dilation_rate = 3
ReLU
3x3 Conv dilation_rate = 4
ReLU
3x3 Conv dilation_rate = 5
ReLU
3x3 Conv dilation_rate = 6
ReLU
3x3 Conv dilation_rate = 8
ReLU
3x3 Conv dilation_rate = 10
ReLU
Figure 3. ResNetSPD block used in the coarse network

to have a better coherence in the final image. We are control the self-similarity with three different scale: 16x16, 32x32, 64x64. The central layer are composed of self attention block. We use 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 feature and we have a better reconstruction.

4.3.3 Discriminator

Our discriminator is composed by 6 convolutional blocks with kernel size 5 and stride 2. These convolutional methods allow to captures the Markovian patches that represents better contextual feature[7]. We add a spectral normalization to improve the training stabilization[8]. The input of this network is the image (real or fake) and the relative mask.

$$\begin{aligned}\mathcal{L}_{D^{sn}} &= \mathbb{E}_{x\sim\mathbb{P}_{data(x)}}\left[ReLU(1 - D^{sn}(x))\right] \\ &+ \mathbb{E}_{z\sim\mathbb{P}_{data(x)}}\left[ReLU(1 + D^{sn}(G(z)))\right]\end{aligned} \tag{14}$$

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

4.4. Multi-GPU and Multi-Node Training

The network has been designed to be trained on multi-GPUs and multi-nodes, as during the development we have understood that a single Nvidia K80 we could use a maximum batch size of 1 (which became 2 after the data augmentation). Our first idea to overcome this problem was to use 2 Nvidia K80 as these are the resource given by AImagelab server for this project. This brings our maximum batch size to 3 (which became 6 after data augmentation). Still we weren't happy as in 24 hours we could't complete a full epoch. Second idea was to exploit the fact that we had 3 accounts and each of them could use 2 GPUs and the same time, so we basically had 6 GPUs in total, could we use all of them at the sime time?

We started to develop multi-node training and each account was seen as a single node. Every account could start 2 processes, one for each gpus, and communicate with a master process to share and upadte gradients and weights. After discussing this with L. Baraldi, he gave us 2 more GPUs for a single account and then proceeded with only the multi-gpu training over 4 GPUs, but still the multi-node support was terminated them it can be used if needed.

4.5. Results

To evaluate the results we use different metrics: L1 error, PSNR, SSIM, LPIPS and FID. We conducted the test with CelebA-HQ Dataset so we can compare our results with DeepGin results. **TODO: TABELLA CON RISULTATI**

Method	PSNR	SSIM	L1	FID	LPIPS
DeepFillv2	22.52	0.845	12.029	19.336	0.128
DeepGIN	24.36	0.867	9.797	37.577	0.142
Ours	0	0	0	0	0

References

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

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

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

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

[11] 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.