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 quality of the reference photo, for instance the face can be taken from a different point of view than the first one. To sum up, 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 occlusions, 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 [**?**] 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 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. As we allow the reference to have [avere un'angolazione diversa da quella frontale], we apply a thin-plate spline transformation to make it frontal [meh che traduzione brutta]. We use 30 specific landmarks as parameters as using more parameters lead to distortion given by the error in the landmarks detection and less lead to an imperfect warping. The same polygon region of Mask Segmentation is cut from the reference photo, the by applying the TPS is sticked to to main photo leading to a (partial) reconstruction.

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 preprocessed RGB image so that its values are in range $[-1, +1]$. We provided different methods to initialize the strating 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.

4.1. Network learning

Our loss function is the sum of six different loss functions:

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

In the following formulas we will use symbols to refer to specific elements such:

$\mathbf{I}_{in}$	masked input image
$\mathbf{I}_{gt}$	reference ground truth image
$\mathbf{I}_{out}$	output image (after refinement stage)
$\mathbf{I}_{completed}$	$\mathbf{I}_{out}$ with valid pixels replaced by ground truth

Adversarial Loss.

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

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

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 classical loss for generative adversarial learning where discriminator is trained to distinguish $\mathbf{I}_{completed}$ from $\mathbf{I}_{gt}$ and generator has the aim of cheating the classification of the discriminator.

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.

Total variation (TV) Loss.

$$\mathcal{L}_{tv} = \sum_{x,y}^{H-1,W} \frac{\|\mathbf{I}^{x+1,y}_{completed} - \mathbf{I}^{x,y}_{completed}\|_1}{N^{row}_{\mathbf{I}_{completed}}} + \sum_{x,y}^{H,W-1} \frac{\|\mathbf{I}^{x,y+1}_{completed} - \mathbf{I}^{x,y}_{completed}\|_1}{N^{col}_{\mathbf{I}_{completed}}} \tag{5}$$

where *H* and *W* are the height and width of $\mathbf{I}_{completed}$ and $N^{row}_{\mathbf{I}_{completed}}$ and $N^{col}_{\mathbf{I}_{completed}}$ are the number of pixels in $\mathbf{I}_{completed}$ 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.

Perceptual Loss.

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

where ϕ is the well-trained loss network, VGG-19[**?**], 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.
Style Loss.

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

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[**?**] 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.

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[**?**]. $(z_i, z_i^{'})$ 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[**?**]. 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.

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

[**?**], rescaled to 256x256, during training. In other GAN inpainting architectures, the mask region to reconstruct is usually calcuated during the training in a randomized way. We do not have this randomization process, so for each image of FFHQ we precalculated the face region where the mask is weared using facial landmarks. For testing we used CelebA256.

4.3. Architecture

Our architecture is highly inspired by Free Form Image Inpainting with Gated Convolution [**?**] and DeepGIN [**?**]. Our net is composed of two generators: 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: $m'_{y,x} = 1, if f sum(M) > 0$
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:

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

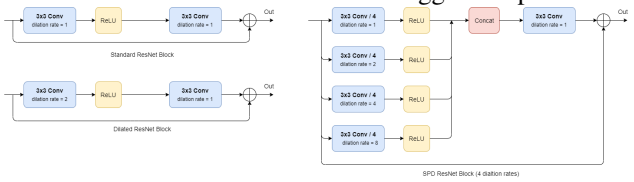
$$Feature_{y,x} = \sum\sum W_f \cdot I \tag{12}$$

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

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 ResNet blocks with different dilation and some gated convolutional layers. This modified ResNet blocks are called Spatial Pyramid Dilation(SPD). 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 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. This convolutional methods allow to captures the Markovian patches that rappresents better contextual feature[**?**]. We add a spectral normalization to improve the training stabilization[**?**]. The input of this network is the image (real or fake), the mask and the guidance channels. To understand if the input image is real or if it's fake, we use the hinge loss as objective function of the discriminator.

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

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

4.4. 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**