CS736

Semantic Image Inpainting with Deep Generative Models

Prajval Nakrani, 17D070014
Parth Shettiwar, 170070021
Harekrissna Rathod, 17D070001

Problem Definition

- Semantic Image Inpainting is a challenging task where in, it is required to fill large missing regions, in an image, with meaningful content based on the remaining available visual data.
- This is a difficult problem, mainly in the cases where there is a large amount of missing data in the image.
- Previous algorithms mainly directly rely on the information in pixel domain in the image around the missing region.
- Hence, a majority of previous algorithms require appropriate information to be contained in the input image like similar pixels, structures or patches etc.

Why Deep Semantic Image Inpainting?

- As explained in the previous slides, most of the algorithms rely directly on the pixel domain information and hence perform very poorly in cases when there is a large amount of missing information in the image.
- In this method, the authors try to search for the closest encoding of the given corrupted image in the latent space image manifold using a certain method.
- Once such an encoding is obtained, it can be passed through the trained generator to reconstruct the inpainted image or rather infer the missing data.
- Basically, the idea here is to search for similarity in the latent space manifold rather than in the pixel domain. This leads to very high performance even in the cases when a large portion of the image is missing.

Generative Adversarial Network (GAN)

- GANs are a framework for training generative models.
- We require GAN in order to learn a semantic latent space encoding (z).
- The generator (G) maps a vector **z** from a prior distribution p_z to the original pixel domain image distribution, p_data.
- The discriminator (D) plays an adversarial role by mapping the input image to a likelihood estimation.
- The GAN framework is trained by optimizing the following loss function:

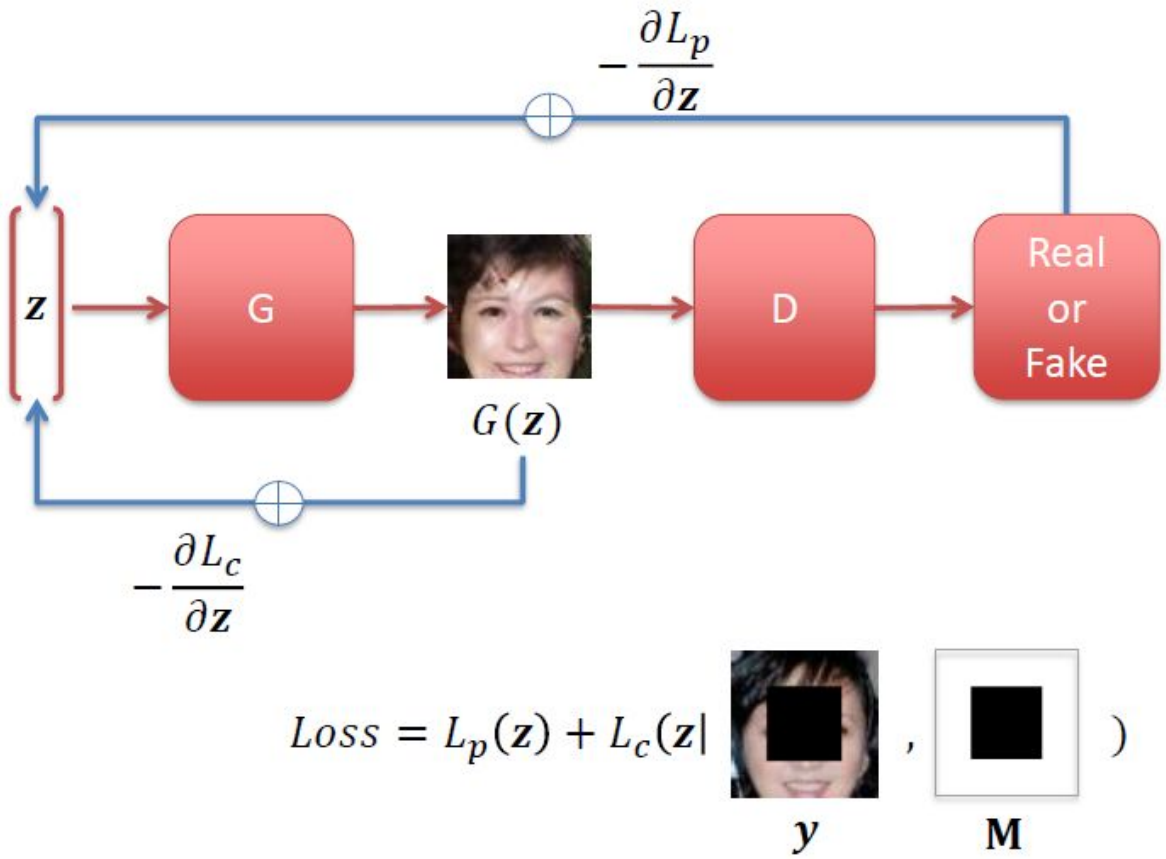
$$\min_G \max_D V(G,D) = \mathbb{E}_{\mathbf{h}\sim p_{data}(\mathbf{h})}[\log(D(\mathbf{h}))] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{Z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))]$$

Semantic Inpainting by Constrained Image Generation

- We assume that we have a generator that takes a point z drawn from distribution p_z and generates an image mimicking the samples from the original image distribution p_data.
- Given that G is efficient in its representation, we aim to recover the encoding z^ “closest” to the corrupted image while being constrained to the manifold.
- Given that y is the corrupted image and M is the mask, the authors formulate the above process of finding z^ as an optimization problem as follows:

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} \{\mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) + \mathcal{L}_p(\mathbf{z})\}$$

Pictorial Representation of the Algorithm



Importance Weighted Context Loss (L_c)

- A convenient choice for the context loss is simply the L-2 norm between the generated sample G(z) and uncorrupted portion of the input image **y**.
- However, that will give equal importance to all the pixels in the image.
- We require pixels that are surrounded by corrupted pixels to have more weight since they are actually in neighbourhood of the missing pixels.
- Inspired from above idea, following weighting is used:

$$\mathbf{W}_i = \begin{cases} \sum_{j \in N(i)} \frac{(1-\mathbf{M}_j)}{|N(i)|} & \text{if } \mathbf{M}_i \neq 0 \\ 0 & \text{if } \mathbf{M}_i = 0 \end{cases}$$

$$\mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) = \|\mathbf{W} \odot (G(\mathbf{z}) - \mathbf{y})\|_1$$

Prior Loss (L_p)

- Broadly speaking, prior loss is used to penalize high-level image feature representations instead of pixel-wise differences.
- In our case specifically, prior loss is used to ensure that the generate image is similar to the samples drawn from the training set.
- Basically, it ensures that the estimated images fall close to the original image data distribution.
- To aid this, standard GAN adversarial loss is used as follows:

$$\mathcal{L}_p(\mathbf{z}) = \lambda \log(1 - D(G(\mathbf{z})))$$

Inpainting

- With the help of previously defined context loss and prior loss, the given corrupted image can be mapped to closest z in the latent space representation denoted by z^.
- However, the issue is that the generated image might not preserve the overall intensity values of the missing pixels, even though the the content might be correct and well aligned.
- To mitigate this problem, Poisson blending is used that minimizes the following objective function to achieve the final inpainted image:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\nabla \mathbf{x} - \nabla G(\hat{\mathbf{z}})\|_2^2$$
$$\text{s.t. } \mathbf{x}_i = \mathbf{y}_i \text{ for } \mathbf{M}_i = 1$$

Final Algorithm

1. Train a GAN, to learn the mapping between a latent distribution p_z and the image distribution p_data.
2. Give G, we need to find the optimal latent vector z^ that lies on the latent image manifold and is closest to the given corrupted image.
    a. In order to perform this step, we need to use the combination of context loss and prior loss as explained in the previous set of slides.
3. Finally, to level out the generated missing data intensities with the original image intensities, Poisson blending is applied.

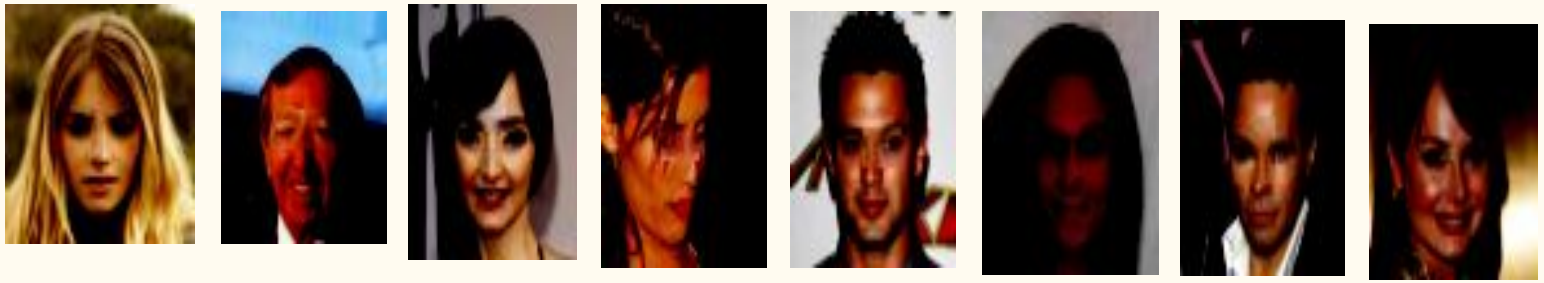
Implementation details

- We implemented this algorithm on CelebA dataset which is dataset of 202,599 celebrity images.
- The standard DCGAN has been used to get the uncorrupted image given a latent vector. We trained a DCGAN using pytorch tutorial for 50 epochs on a subset of dataset of celebA “https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html”.
- The images are resized to (64,64,3) in size(Since the DCGAN trained is on images of that size).
- We randomly blacked out a box from centre, max size of whose can be 12 x 12 and pass it as corrupted image.
- The optimal latent vector computation is done for 1500 iterations as mentioned in paper and followed by poisson blending for 500 iterations.
- Window size = 7 has been used for calculating the weights as mentioned in paper.
- For Poisson blending, we calculated gradients in x and y direction using a standard sobel filter.

Qualitative Results

- We show 2 results, one before poisson blending and after poisson blending. It was observed that after poisson blending, the image develops some sort of dotted texture. We predict this is due to sobel filters used across 3 channels independently leading to independent optimization.
- Overall results are good after optimal latent vector is found. The colours of the prediction for in-paint region and surrounding region are little different in shades.

Input



Optimal Latent vector

After Poisson Blending

Results for Modified Poisson Blending using Convex Optimization

Future Work

- We plan to improve upon the poisson blending process to incorporate the dependence of channels during optimization.
- We plan to apply this algorithm on various other variations of this inpainting task where we remove random pixels from image and try to reconstruct,
- This technique for inpainting could be used on medical image datasets too.
- Nowadays many new GAN models have arrived which are very robust and give variety of generations. The DCGAN model we have used was one of the first GANs developed in 2016 and is quite old.
- Do quantitative analysis (using metrics like PSNR, NRQM etc.)

Thank You