[Team 34] Image In-painting using DCGANs

1st Rajan Anbazhagan
Unity Id:(ranbazh)

2nd Sreenidhi Ganapati Raman
Unity Id: (sganapa4)

3rd Sai Krishna Wupadrashta
Unity Id: (swupadr)

Abstract

In this project, we aspire to accomplish Image in-painting using Deep Learning, which reconstructs missing parts of an image.

The motivation for this project stems from the fact that there aren't many robust algorithms that reproduce complete images from blurred out or incomplete images. The end goal of our project is that the implementation is robust and effective enough such that an observer perceives the generated image as a completely plausible picture of a person.

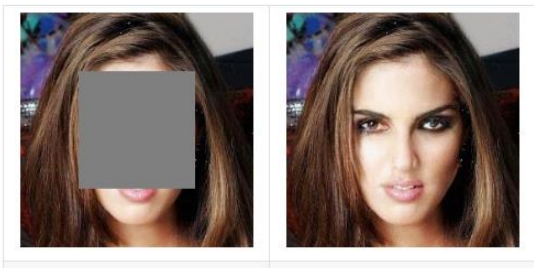


Image 1

Existing methods which extract information from only a single image generally produce unsatisfactory results due to the lack of high level context. In this project, we attempted a novel method for semantic image inpainting using Deep Convolutional Generative Adversarial Networks (DCGANs), which generates the missing content by conditioning on the available data. Given a trained generative model, we search for the closest encoding of the corrupted image in the latent image manifold using our context and prior losses. This encoding is then passed through the generative model to infer the missing content. In our method, inference is possible irrespective of how the missing content is structured.

This project has the potential to gain relevance in various sectors like image processing -where we might need to reconstruct certain parts of an image to improve the quality. This could also be used in the domain of intelligence and security to potentially sketch out portraits of wated criminals whose photos have not been captured.

We use the Large-scale CelebFaces Attributes (CelebA) Dataset which contains more than 200k celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including 10,177 number of identities, 202,599 number of face images, and 5 landmark locations, 40 binary attributes annotations per image.

The DCGAN algorithm has been used to get variations in the generated images by growing both the generator and discriminator progressively. The sensing modalities we can obtain are mainly visual as the output obtained is a generated image.

I. Model training and selection

The basic workflow is to feed the neural network an input image with patches that need to be filled. To enable the neural network to understand about the parts of the image that needs to be filled in, a separate layer called the mask is used. The mask contains pixel information for the missing data. We actually don't need any set shape or mask to cover portions of an image, but we are using it because it is easier to generate and check the results. The network produces a new complete synthetic image- by overlaying the region covered by the mask from the new generated image on top of the remaining portion of the old image.

A. *The Model*

We are using a DCGAN to build the deep learning network model for image in-painting that consists of a completion network (for convolution and deconvolution) . It mainly composes of convolution layers without max pooling or fully connected layers. It uses convolutional stride and transposed convolution for the downsampling and the upsampling. The DCGAN replaces all the max pooling with convolutional strides and uses transposed convolution for upsampling. Batch normalization is used for the majority part except for the outer layer of the generator and the input layer of the discriminator. ReLU activation function is used in the generator, except for the output which uses tanH. This is because these images are normalized between [-1,1] rather than [0,1], and thus tanH is preferred over sigmoid.

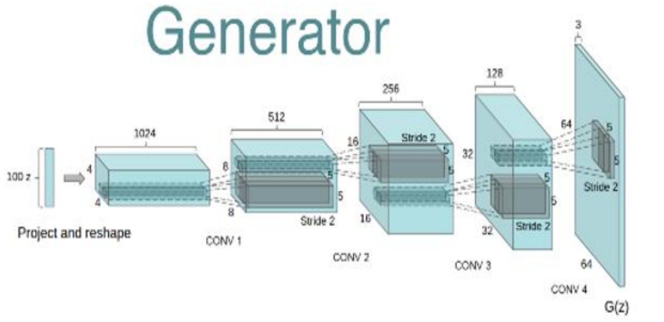


Image 2

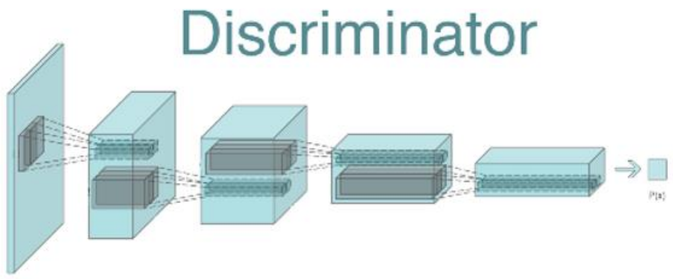


Image 3

The images 2 and 3 shown above have been split into two halves- and shown in the interest of space constraints, whereas originally they are present on the side of each other. Image 2 and Image 3 show the respective decomposition of the DCGAN model into the generator and discriminator respectively. Now, the major points to note between the discriminator and the generator are shown below in Table 1.

	GENERATOR	**DISCRIMINATOR**
CONVOLUTION	*Up convolutions*	*Standard Convolution*
POOLING LAYERS	*Fractional Strided Convolutions*	*Strided Convolutions*
BATCH NORMALIZATION	*Batch Normalization*	*Batch Normalization*
ACTIVATION	*RelU and tanH (for last layer)*	*Leaky RelU*

Table -1

B. *Baseline for Comparasion*

The original image or the ground truth if present before any distortion will serve as the baseline for comparison. In case of any genuinely distorted or damaged image being fed as the input, then a baseline doesn't exist cause our model literally generates a plausible version of how the distorted version of the image would have been.

Firstly, the missing features are filled in the right position and orientation. Secondly, the features seem plausible and finally, each plausible variation has a difference when compared to the other outputs.

However, for the sake of checking whether or not the model performs satisfactorily, we have used some models from an online dataset, and have stored the input image as the ground truth.

This input image is then masked or distorted on purpose to study how well the image is then reconstructed. For this, we are using the CelebA images dataset available online as the baseline for comparison.

Link : http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

The images obtained from this dataset are pre-processed to make the image data compatible to our need. The following steps are done to ensure that it's taken care of properly:

i) Sematic Alignment of training images is ensured and images are cropped for further processing.

ii) Resizing of each image to dimensions of 64x64.

iii) Normalizing the data- to be a distribution between [-1,1] in order to make it a zero-centric distribution to facilitate easier calculation of L1 and L2 norms.

Some sample images taken from the dataset to represent the ground truth are shown below:

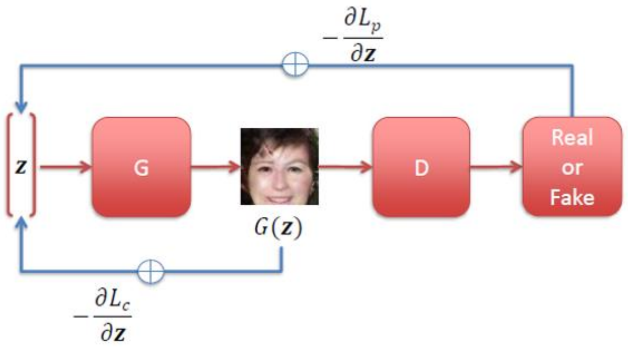
Image 4 (Sample input images)

II. Experimental Section

A. *Metrics*

The original image serves as the ground truth. To emulate the in-painting problem, artificial distortions like noise, text, scratch, objects or masks are added to the original image. The method of evaluating the obtained results for inpainting algorithms depends on the technique used.

The following image shows the architecture of the generator- discriminator model- and one loss term associated with each model. G represents the generator and D represents the Discriminator. Z is the random vector that is mapped from the prior distribution Pz to the image space by G.


$-\frac{\partial L_p}{\partial z}$
z
G
$G(z)$
D
Real or Fake
$-\frac{\partial L_c}{\partial z}$
Image 5 (D & G network architecture)

The main underlying metric used is the loss function used by the model.

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

Where, y is the corrupted image, M is the binary mask with size equal to the image. We use a combination of two loss models to build our cumulative loss function: Weighted context Loss (Lc) and Prior Loss (Lp).

Weighted context loss is proposed with the hypothesis that the importance of an uncorrupted pixel is positively correlated with the number of corrupted pixels surrounding it. We express Lc in the form of L1 loss representation:

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

W is a weighting term here.

The prior loss focuses more on high-level image feature representations instead of pixelwise differences. The prior loss encourages the recovered image to be similar to the samples drawn from the training set.

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

Here, λ is a parameter to balance between the two losses. z is updated to fool D and make the corresponding generated image more realistic.

The generative model, G, takes a random 100 dimensional vector drawn from a uniform distribution between [−1, 1] and generates a 64×64×3 image. For the discriminator model D, the input layer is an image of dimension 64×64×3, followed by a series of convolution layers where the image dimension is half, and the number of channels is double the size of the previous layer, and the output layer is a two class softmax.

We use Adam for optimization during the training stage with λ = 0.003 In the inpainting stage, ˆz is found by using Adam and restricting z to [-1,1] in each iteration.

The algorithm takes the approach of back-propagation and tries to identify which image in the latent space corresponds to the input masked image, and using this the generator tries to find the prediction G(z), while the discriminator tries it's best to fool the generator saying that the generated image is a fake by giving false predictions in order to make the model as robust as possible.

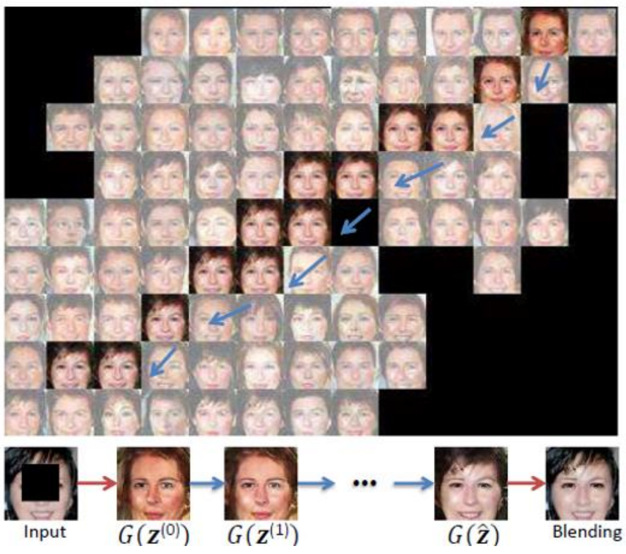


Image 6 (Back-propagative approach to reconstruct image)

B. *Model Selection*

From this project, we expect proper filling of patches in images by maintaining continuity with the rest of the image and plausible variations in the filled patches in order to obtain pluralistic image in-painting.

In-order to do thus satisfactorily, the modelled network has to have a good enough performance and accuracy- which is improved by tuning the hyper-parameters accordingly.

The choice of hyper-parameters is done after a detailed analysis of the kind of network we want to build, what activation functions to use and what optimizer is the best suited for this purpose. Now, the following two images depict how the loss was minimized after hyperparameter tuning.

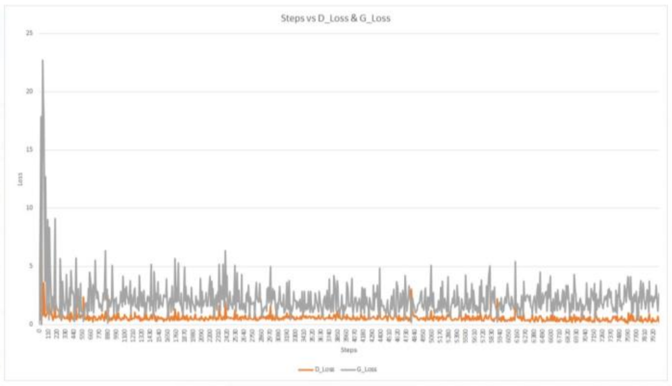


Image 7 – D and G Loss before hyperparameter tuning

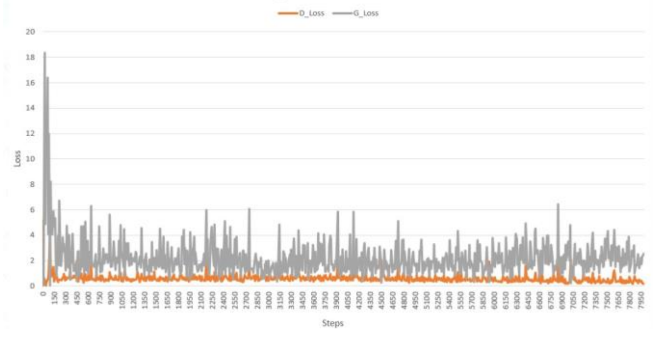


Image 8 – D and G Loss after hyperparameter tuning

We see that the values of the Loss for the discriminator and the generator have come down (Look closely at the scale, which is a +5 addition for one and +2 addition for the other plot).

Table 2 below describes the list of the optimized hyper-parameters chosen after analysis.

HYPERPARAMETER	VALUE
Input image size	64x64x3
No. of neurons in each layer for generator	[512,256,128,64] decreasing in width
No. of neurons in each layer for discriminator	[64,128,256,512] Increasing in width
Kernel size for up convolution in generator and discriminator	5X5
Stride length for convolution in generator and discriminator	2
Spectral Normalization value	'True'
Learning rate for G and D	2e-4
Lambda (λ) -balancing parameter for prior loss	1e-3
Dimension of Z vector	100x1

Table -2 (Optimized Hyperparameters)

C. *Performance in comparision to baseline*

From this project, we expect proper filling of patches in images by maintaining continuity with the rest of the image and plausible variations in the filled patches in order to obtain pluralistic image in-painting.

So, we obtain the following results after passing the images through the DCGAN network.

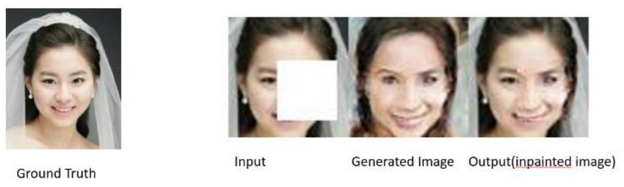


Image-9 Output example 1

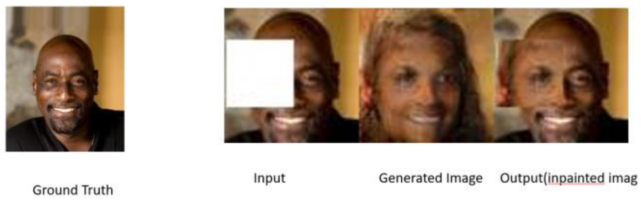


Image-10 Output example 2

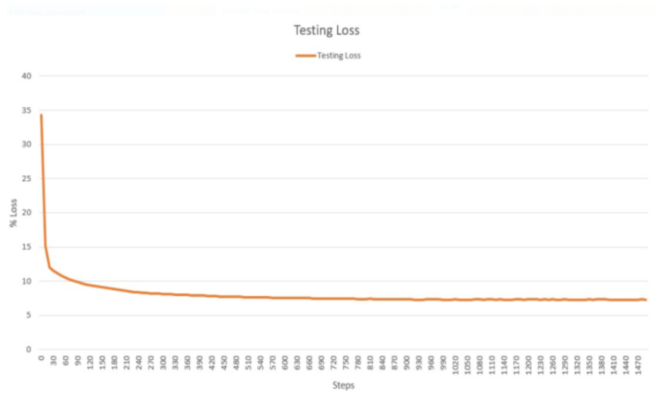


Image 11- Loss % graph for the in-painted image

Finally, the graph in image 11 depicts the % loss, as in the difference between the ground truth and the output image that has been in-painted.

This shows that the % loss for the in-painted image decreases as we go on increasing the number of iterations. This proves that the process of Image inpainting is successful.

III. References

[1] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

[2] Raymond A. Yeh∗ , Chen Chen∗ , Teck Yian Lim, Alexander G. Schwing, Mark Hasegawa-Johnson, Minh N. Do : Semantic Image Inpainting with Deep Generative Models.

[3] J. Omar Elharrouss,Noor Almaadeed,Somaya Al-Maadeed and Younes Akbaria. Image inpainting: A review.