Structurally and Semantically Coherent Deep Image Inpainting

Mirza Tanzim Sami - 17141019

Ehsanul Amin Khan - 14101118

Ishrat Naiyer Rhidita - 14310008

Supervised by:

Dr. Jia Uddin

Department of Computer Science and Engineering

December 2018

BRAC University, Dhaka, Bangladesh

Declaration

We, hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole or in part, has been previously submitted for any degree.

_____

Signature of Supervisor
(Dr. Jia Uddin)

_____

Signature of Author
(Mirza Tanzim Sami)

_____

Signature of Author
(Ehsanul Amin Khan)

_____

Signature of Author
(Ishrat Naiyer Rhidita)

Acknowledgement

First, we want to thank the all-powerful to empower us to start our thesis, to put our earnest attempts and effectively finish it up.

Secondly, we offer our authentic and heartiest gratefulness to our respected Supervisor Dr. Jia Uddin sir for his commitment and support in driving the thesis and arrangement of the report. His contribution, incorporation and supervision have enlivened us and gone about as a gigantic motivator all through our exploration.

Last but not the least; we are appreciative to the assets, seniors, friends who have been involved in an indirect way yet effectively helping with our thesis. We, likewise, want to recognize the assistance we got from different resources over the web; especially from fellow researchers' work. We take this opportunity to thank our parents for their unceasing encouragement, support and attention. We are also grateful to each other for the support throughout this venture.

Table of Contents

List of Figures

Abstract

This thesis proposes an augmented method for image completion, particularly for images of human faces by leveraging on deep learning based inpainting techniques. Face completion generally tend to be a daunting task because of the relatively low uniformity of a face attributed to structures like eyes, nose, etc. Here, understanding the top level context is paramount for proper semantic completion. Our method improves upon existing inpainting techniques that reduces context difference by locating the closest encoding of the damaged image in the latent space of a pretrained deep generator. However, these existing methods fail to consider key facial structures (eyes, nose, jawline, etc) and their respective locations. We mitigate this by introducing a face landmark detector and a corresponding landmark loss. We add this landmark loss to the construction loss between the damaged and generated image and the adversarial loss of the generative model. After several experimentation, we concluded that the added landmark loss attributes to better understanding of top level context and hence more visually appealing inpainted images.

Chapter 1

Introduction

1.1 Motivation

Bearing synthetic data in mind, we have built our thesis and hope to progress in the same direction in the upcoming days. As our research is about generating obscured parts in images to give the most realistic result, synthetic data production is something that is well within our grasp. The digital world is producing data at an exponential level. Therefore, methods like 'Big Data', 'Data Mining' is serving as a foundation for both artificial intelligence and machine learning in recent times. We all, believe that Synthetic data is about to become a major competitive advantage in the field of artificial intelligence. All three of us have come forward with the idea because we all agree that Synthetic data is getting more and more valuable as the days go by. Synthetic Data is basically any production data applicable to a given situation that are not obtained by direct measurement. As artificial data is now able to give the same results as the real data, we, as a team, want to bank on this opportunity. Our vision is to provide synthetic data to train their artificial intelligence. Big companies develop specific software and scripts that compute relevant metrics on raw data. However, in order to run tests, specific software needs proper data. One solution could be the provision of actual data but unfortunately, by doing so companies become vulnerable to the privacy laws which are becoming increasingly strict all over the world. For example, we have all observed what Facebook has gone through in recent times. Facing these lawsuits, big companies are in need of Synthetic data. So, by doing this, we have a chance of giving relief to the Big data companies from confrontation with legal hurdles. Not only that, our idea opens up a whole new era ahead where our realistic datasets might be used by companies to make products that meet the public demands without compromising the privacy concerns.

1.2 Objectives

Image completion is the process of taking any damaged or corrupted image and filling up the missing spaces with relevant information. Often times the completion task demands that the missing region synthesized by an algorithm is semantically accurate. Semantic inpainting of corrupted regions thus require a high level understanding of the surrounding regions [15]. Understanding the top level context of the human face is quite difficult given the immense variation that any face has due to structures like eyes, nose, lips, etc. A robust approach for such techniques could be to rely on external databases [16] or looking up from the internet. Such databases may help the algorithm to identify similar looking patches but they fail to maintain proper relevance of the patch with the given (corrupted) image. Hence, our objective in this thesis is to rely on cutting edge technologies like deep learning to do a significantly better on the immensely challenging task of image inpainting.

1.3 Contribution Summary

Therefore, this thesis improves upon recent inpainting methods that rely on deep learning generative models to accomplish this task much more effectively. Unsupervised deep learning parametric models can learn the feature representation of a given dataset, and once learnt, it can be utilized for inference tasks. The inference is done based on the information it had learnt from the surrounding regions of the corrupted place. Due to the recent development of powerful deep generative models like the DCGAN, it is now possible to regenerate large portions of missing regions faithfully. Our model is closest to the work done with semantic image inpainting, however in contrast, our model utilizes a face landmark detector to identify the locations of key facial structures like eyes, noses, lips and the jawline to ensure a more comprehensive understanding of high level feature rather than just pixel values. We therefore introduced a landmark loss that measures the distance between facial points on the input and the inpainted image. Our model has been primarily tested on the CelebA dataset, where it has been successful at regenerating any shape and large chunks of missing portions. The results we have obtained are both visually pleasing and quantitatively compare to the standard models available today.

1.4 Thesis Outline

- Chapter 2: Elaborations of all the algorithms and main architectural components.
- Chapter 3: Presents the proposed model of this research.
- Chapter 4: Demonstrates the results found in our research.
- Chapter 5: Concludes the thesis and states the future research directions.

Chapter 2

2.1 Literature Review

A handful of algorithms [20, 21] exists that tries to fill up missing regions by performing patch matching with known regions of the images but these solutions are only limited to a low level understanding of the entire image [17]. Past methods like PatchMatch [20] or total variation approaches [22] are strictly constrained to the input images for inferring the missing region which makes it completely unsuitable for inpainting human faces where instances like deducing the patch of a missing nose from the patch containing the eye is quite impossible. One the most recent contributions made on deep learning based image inpainting is by Yeh et al. [18]. In that model, given a masked input image, the algorithm tries to find the closest match of the masked image in the latent space of a deep generative model like the DCGAN [2]. The searching in the latent space is performed based on a weighted context loss [18] and regularized by an adversarial loss [19]. Our work builds upon these past research and in some ways improves on it.

2.2 Algorithms and Architectural Components

The following sections will go into greater details on the workings of the major components of this proposed model, chiefly the DCGAN, Facial Landmark Detector and the Backpropagation method to the input space.

2.3 DCGAN

The DCGAN architecture gains it ability to generate images from the Generative Adversarial training method in conjunction with the Convolutional Neural Network (CNN) architecture. Previously, attempts to integrate CNN with GAN has not been fruitful. A significant improvement to this effort came with the development of the LAPGAN [1] which proposed a different take of iteratively upscaling low resolution generated images. This process proved to be more stable than previous approaches. The authors of the DCGAN architecture iterated through numerous approaches until they decided to settle for a particular group of architecture that provided a stable enough environment for training on a broad range of datasets and one that could output high resolution images and work with deep generative models. The key to their success has been three modifications to the CNN architecture (Figure 1).

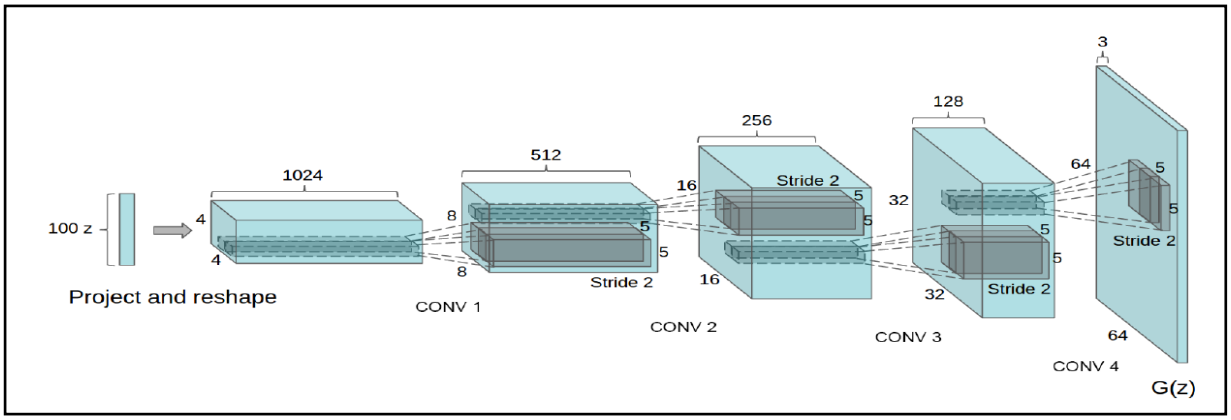

Figure 1. DCGAN Generator Architecture

First, is the use of the use of the all convolutional net [3] which gets rid of maxpooling and strided convolutions, giving the network the capability to learn its own spatial downsampling. This approach allows a DCGAN to learn its own spatial upsampling and discriminator [2]. The first is the all convolutional net [3] which replaces deterministic spatial pooling functions (such as maxpooling) with strided convolutions, allowing the network to learn its own spatial downsampling. We use this approach in our generator, allowing it to learn its own spatial upsampling, and discriminator. The Second change was the decision to get rid of fully connected layers that were present over the convolutional features. This step is important because global mean pooling can yield greater stability for the model but can reduce the speed of convergence of the network. (Radford et al., 2014) states that an intermediary state of directly connecting the highest convolutional features to the input and output respectively of the generator and discriminator seemed to do a good job. In GANs the first layer is a fully connected layer, which takes in the random noise, applies matrix multiplication and reshapes the data into a four dimensional tenor, from this point onwards the convolution part begins. In the discriminator side, the final convolution layer is flattened and then put through as a single sigmoid output. The architecture can be seen in Fig. 1.The Third and final change made in the DCGAN architecture was the use of Batch Normalization [8]. This technique can make learning more stable since it normalizes the input to have zero mean and unit variance, before being fed to every node in the neural network. It is particularly effective at mitigating training issues, especially the very common issue of GANs where the generator collapses all samples into one point. Another point to note for DCGANs is that direct application of batch normalization to all layers can cause some level of instability. This can be easily resolved by not applying batch normalization to the generator output

layer and the discriminator input layer. For our purpose, the DCGAN we used applied all the aforementioned changes and was quite effective at generating human faces. Our DCGAN was trained on the celebA dataset [5]. Training of the model employed stochastic gradient descent (SGD) and data was taken in mini-batches. The weights of the model were initially sampled from a Normal distribution with standard deviation 0.02 and centred at zero. The activation function used in the nodes were the LeakyReLU [9], where the slope of the leak was set to 0.2. The optimizer used for the DCGAN was the Adam optimizer [4]. It does not use the momentum to accelerate training unlike typical GANs. We used a learning rate of 0.001 as suggested by (Radford et al., 2014). Only the output layer of the generator uses the tanh activation function, the rest uses the LeakyReLU as mentioned before. Once training was complete, we saved the model weights and create a loadable model file (dcgan.pb) using tensorflow [11].

2.4 Facial Landmark Detector

In this section we will describe the model for the facial landmark detection. The model we used was proposed by (Kazemi et al, 2014). They introduced a method that used multiple layers of regressors. The following part of this section will describe the major components of the model and how it is trained. To understand the working principles of this model, we must get familiar with the following notation as used in the original paper [23]. We will consider $x_i \in \mathbb{R}^2$, which will be the x,y coordinates of the *i*th landmark point in a given input image *I*. The vector $S = (x^T_1, x^T_2, \ldots x^T_p,)^T \in \mathbb{R}^{2p}$ marks all the *p* facial landmark points in the image *I*. Furthermore, $\hat{S}^{(t)}$ represents the current estimate of $\hat{S}^{(t)}$. Every regressor in the cascade can be represented by $r_t(.,.)$.The regressors will take in *I* and the current estimate $\hat{S}^{(t)}$ and predict and updated vector, which will then be added to $\hat{S}^{(t)}$ to get the improved estimate $\hat{S}^{(t+1)}$.

$$\hat{S}^{(t+1)} = \hat{S}^t + r_t\left(I, \hat{S}^{(t)}\right) \ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

The predictions made by the regressor r_t are dependent on attributes such as pixel intensity values of the input image *I* and it is indexed relative to $\hat{S}^{(t)}$. A result of this process is the formation of geometric invariances that can give us the assurance that as the cascade progresses, the correct locations on the face are getting semantically indexed. We should keep in mind that the range of outputs expanded by the cascade is only guaranteed to be in a linear subspace of training data if the first estimate $\hat{S}^{(0)}$ is a member of the same space. Thus, no further constraints on the predictions

has to be applied. This means that the initial shape can be taken to be the mean shape centered around the training data and furthermore it can be scaled with the bounding box of any regular face detector. The regressors are trained according to the techniques in [24] which utilizes gradient tree boosting algorithms [12] and calculates loss as the sum of square. The following section will go into further details as to how each r_t in the cascade is learnt.

2.4.1 Learning each regressor in the cascade

Let us consider the training data (I_1, S_1), . . . ,(I_n, S_n). Here, every I_i is a face image and S_i is the corresponding shape vector of the image. The first regression function r_0 in the cascade can be learnt by creating three copies on an image from the training dataset, an initial shape estimate and the target update step, which follows:

$$\left(I_{\pi_i}, \hat{S}_i^{(0)}, \Delta S_i^{(0)}\right) where$$

$$\pi_i \in \{1, .., n\}$$

$$\hat{S}_i^{(0)} \in \{S_1, .., S_n\}/S_{\pi_i}$$

$$\Delta S_i^{(0)} = S_{\pi_i} - \hat{S}_i^{(0)} \quad \text{.....(2)}$$

Algorithm 1 Learning r_t in the cascade

Have training data $\{(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)}, \Delta\mathbf{S}_i^{(t)})\}_{i=1}^N$ and the learning rate (shrinkage factor) $0 < \nu < 1$

1. Initialise

$$f_0(I, \hat{\mathbf{S}}^{(t)}) = \arg\min_{\gamma \in \mathbb{R}^{2p}} \sum_{i=1}^{N} \|\Delta\mathbf{S}_i^{(t)} - \gamma\|^2$$

2. for $k = 1, \ldots, K$:
   (a) Set for $i = 1, \ldots, N$

$$\mathbf{r}_{ik} = \Delta\mathbf{S}_i^{(t)} - f_{k-1}(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)})$$

   (b) Fit a regression tree to the targets $\mathbf{r}_{ik}$ giving a weak regression function $g_k(I, \hat{\mathbf{S}}^{(t)})$.
   (c) Update

$$f_k(I, \hat{\mathbf{S}}^{(t)}) = f_{k-1}(I, \hat{\mathbf{S}}^{(t)}) + \nu\, g_k(I, \hat{\mathbf{S}}^{(t)})$$

3. Output $r_t(I, \hat{\mathbf{S}}^{(t)}) = f_K(I, \hat{\mathbf{S}}^{(t)})$

The model requires setting the total number of the aforementioned triplets to N = nR, here R is the number of initializations that is used for every image I_i. From the set $\{S_1, \ldots, S_n\}$ every initial shape estimate of an image I_i has to be sample uniformly. It is from this data that the regression function r0 can be learnt by leveraging on the gradient tree boosting algorithm [12] with a square error loss. Please refer to Algorithm 1 for more concrete understanding. The next regressor r_1 is learnt when *t* is set to zero and when the training data,$(I_{\Pi i,}, \hat{S}_i^{(1)}, \Delta S_i^{(1)})$ is given. The training data is received by updating the set of training triplets.

$$\hat{S}_i^{(t+1)} = \hat{S}_i^{(t)} + r_t\left(I_{\pi_i}, \hat{S}_i^{(t)}\right) \ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

$$\Delta S_i^{(t+1)} = S_{\pi_i} - \hat{S}_i^{(t+1)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$

The aforementioned process will be iterated until a cascade of *T* regressors $r_0, r_1, \ldots, r_{T-1}$ are learnt, it is then complied together to achieve a desired degree of accuracy. One important thing to note is that the values calculated inside the innermost loop is correspondent to the gradient of the square error loss function as measure for every training sample. In the algorithm above, a learning rate parameter can be observed. It is called the shrinkage factor, $0 < v \leq 1$. v is set to be less than 1, it helps to prevent overfitting and for the most part results in regressors which can generalize better as opposed to instances where learning is done with v set to 1.

2.4.2 Regressor tree

Tree based regressors are at the heart of all regression function r_t. It needs to fit to the residual targets when the gradient boosting algorithm is run. For every split node in the regression tree, a decision is made depending upon the difference in intensity among two pixels. The pixels are ascribed to u and v coordinates according to the mean shape. Since face image are arbitrarily shaped, indexing the point with the same relative position to the shape of the face as u and v coordinates does to the mean shape requires warping the image to the mean shape depending on the current shape estimate before the features are extracted. Warping the location points of the image are a more efficient method as opposed to the whole image as only sparse representation of the image are used anyway. A more detailed explanation follows below. Consider k_u be the index of the facial landmark in the mean shape that is closest to u. Then its offset from u is defined as:

$$\delta x_u = u - \bar{x}_{k_u} \ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)$$

Additionally, let's consider a shape S_i defined in image I_i. The position in I_i which has qualitatively similarity with u in the mean shape image can be represented as follows:

$$u' = x_{i,k_u} + \frac{1}{S_i} R_i^T \delta x_u \text{ ……..(6)}$$

In the above equation S_i and R_i are the scale and rotation matrix of the similarity transform which transforms S_i to the mean shape $\bar{S}$. S_i and R_i can minimize the sum of squares between the mean shape's facial landmark points, $\bar{x}_j$'s, and those of the warped shape. v′ is defined in a similar manner.

$$\sum_{j=1}^{p} \left\| \bar{x}_j - (s_i R_i x_{i,j} + t_i) \right\|^2 \text{ ……..(7)}$$

2.4.3 Feature selection

The thresholding of the difference in intensity value of two pixel determines the decision at each node. However, using pixel differences, introduces a problem where the number of potential split (feature) candidates is quadratic in the number of pixels in the mean image. This in turn makes finding a good value for θ very difficult unless a huge number of it is searched. This problem was partially mitigated by taking the structure of the image data into account. Hence, an exponential prior was introduced over the distance between the pixels used in a split, which

$$P(u, v) \propto e^{-\lambda \|u - v\|} \text{ ……..(8)}$$

promoted closer pixel pairs to be chosen. The prior above significantly reduced the prediction error on numerous face datasets.

For the training of this model, the authors had used the following parameter settings. The number of strong regressors, r_t, in the cascade was set as $T = 10$. Every r_t was set up with K = 500 weak regressors g_k. The depth of the trees that were used to represent g_k was set to F = 5. For every layer of the cascade P = 400 pixel locations are sampled from the image. In order to train the weak regressors, a pair of randomly sampled P pixel locations are picked according to the prior and a random threshold. The best split as noted by (Kazemi et al, 2014) was with S = 20, and this value optimized the objective. The landmark detector that was deployed in our model come pre-trained with the dlib library [10]. Dlib's landmark detector was trained with the iBUG 300-W dataset [7]. Each image in this dataset are annotated with 68 facial landmarks points. Although we are

primarily using the celebA dataset and dlib's landmark detector are not trained on this dataset, the detector work well for most cases and has been effective at detecting facial landmark on the images in the celebA dataset and the images generated by the DCGAN.

2.5 Backpropagation to input

The focal issue that back-propagation works with is the assessment of the impact of a parameter on a function whose calculation includes a few ele-mentary steps. The chain rule is the answer to this issue; however back-propagation exploits the specific type of the capacities utilized at each progression (or layer) to give an exquisite and local system [25]. In this section, we will discuss some past research [13] that used 'backpropagation to input' and from the equation used by them, we will derive our process of back propagating. In order to create another texture based on a given picture, [26] suggested that Gradient descent is utilized from a white noise image to discover another picture that coordinates the Gram-matrix portrayal of the first picture. By minimizing the mean-squared distance between the entries of the gram matrix of the original image and gram matrix of the generated image, the optimization is achieved.

Let $\vec{x}$ and $\hat{\vec{x}}$ be the original image and the image that is generated, and G^l and $\hat{G}^l$ their respective Gram-matrix representations in layer *l*. The contribution of layer *l* to the total loss is then:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - \hat{G}_{ij}^l)^2 \quad \text{.....(9)}$$

where w_l are weighting factors of the contribution of each layer to the total loss. The derivative of E_l with respect to the activations in layer l can be computed analytically:

$$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l \quad \text{.....(10)}$$

The gradients of E_l, and thus the gradient of $L(\vec{x};\hat{\vec{x}})$, with respect to the pixels $\hat{\vec{x}}$ can be promptly figured utilizing standard error back-propagation [27]. The gradient $\partial L / \partial \hat{\vec{x}}$ can be utilized as input for some numerical advancement procedure. L-BFGS [14] is utilized is our work, which appeared a sensible decision for the high-dimensional enhancement issue at hand. The whole technique depends primarily on the standard forward-backward pass that is utilized to prepare the convolutional network. In this manner, despite the expansive intricacy of the model, texture

$$\frac{\partial E_l}{\partial \hat{F}_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} \left(\left(\hat{F}^l \right)^T (G^l - \hat{G}^l) \right)_{ji} & if\ \hat{F}_{ij}^l > 0 \\ 0 & if\ \hat{F}_{ij}^l < 0 \end{cases} \quad \text{.....(11)}$$

generation is possible in a sensible time utilizing GPUs and performance-enhanced toolboxes for training deep neural systems [28].

As suggested by [29], this segment will depict a system for picturing the class models, learnt by the picture characterization ConvNets. Given an educated grouping ConvNet and a class of intrigue, the perception technique comprises in numerically creating a picture [30], which is illustrative of the class as far as the ConvNet class scoring model.. More formally, let $S_c(I)$ be the score of the class c, processed by the arrangement layer of the ConvNet for a picture I. The L2-regularised image can be found, such that the score S_c is high with:

$$\arg\max S_c(I) - \lambda \|I\|_2^2 \text{.....(12)}$$

where λ is the regularization parameter. The back-propagation is utilized to optimize the layer weights to identify the ConvNet training strategy. The thing that matters is that for our situation the optimization is performed as for the input, while the weights are settled to those discovered amid the training.

The optimization as initialized with the zero picture (for our situation, the ConvNet was prepared on the zero-focused picture information), and afterward included the preparation set mean picture to the outcome. It ought to be noticed that we utilized the (unformulated) class scores S_c, rather than the class posteriors, returned by the soft-max layer: $Pc = \frac{\exp Sc}{\sum_c \exp Sc}$. The reason why it was done is because by minimizing the scores of different classes, the boost of the class posterior can be accomplished. In this manner, we improve S_c to guarantee that the optimization focuses just on the class being referred to c. Different things were tried with advancing the propagation P_c, yet the outcomes were not outwardly better.

Chapter 3

3.1 Proposed Model

We are keenly focused on designing a model that work specifically for human faces and one that can generate synthetic images for filling in holes/noise in the input image. We propose an augmented model (Figure 2) that improves upon the state of the art algorithm [18] by using a facial landmark detector [6]. Our model utilizes a pre-trained DCGAN [2]. The DCGAN architecture will be trained on the celebA image dataset [5]. The dataset contains thousands of images human faces in different angles and lighting conditions. A trained DCGAN can sample random noise from a normal distribution, apply its newly learnt weights and biases to the noise data and convert the noise into an image. This in turn, gives the DCGAN the ability to sample anywhere from the normal distribution and generate completely new images of human faces (based on the images it had learnt from the dataset). This ability is crucial for our task at hand. Once a masked input image is given to our model as input, it calls the pre-trained DCGAN to randomly generate a batch of 64 images which has been sampled as 'z' from the normal distribution. The generated images are then applied with the same mask used for the input image. Initially the generated images look nothing like the input image, thus incurring a high loss. The loss is calculated based on three parameters. First the l1 distance between the masked input image and generated images, second the distance of the 68 landmark facial points and third the adversarial loss from the discriminator of the

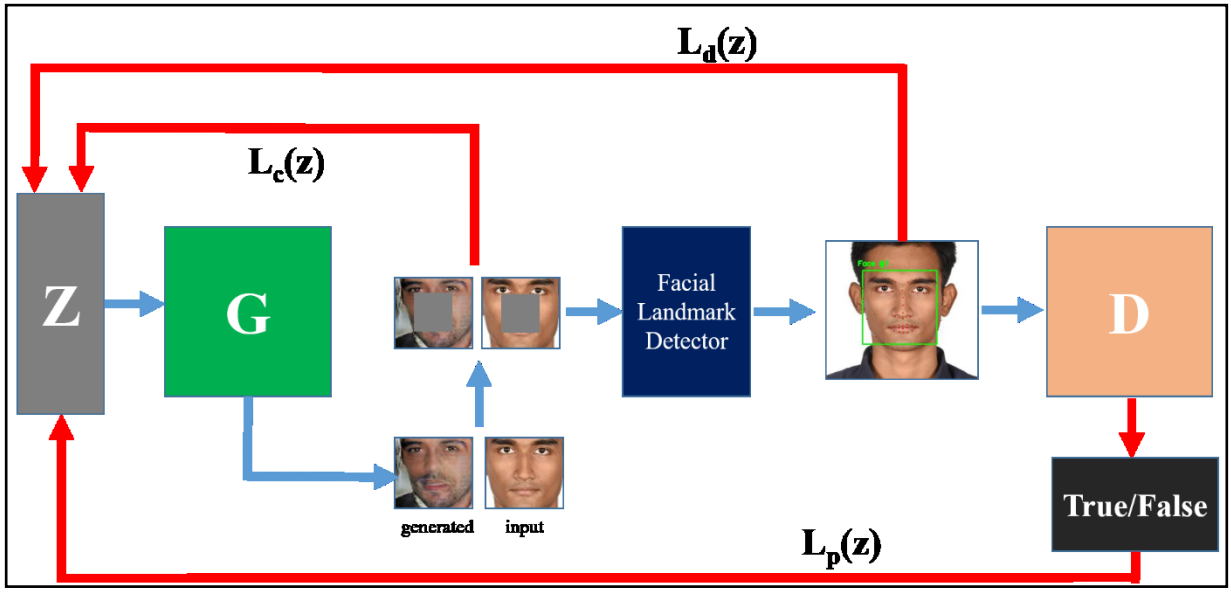

Figure 2. Proposed Architecture. Z is latent space, G is the generator, D is the discriminator. L_c is context loss, L_p is prior loss,L_d is landmark loss.

DCGAN. The total loss calculated is then used to find the gradient in terms of the input noise z (random samplings form the normal distribution) that is feed to the generator of the DCGAN. This gradient is used to back-propagate to the latent input space and traverse across it.

3.2 Loss Function

Generative Adversarial Networks (GANs) are state of the art deep learning techniques for training parametric models. Deep learning architecture deploying this technique has shown immense success at synthesizing images that are both high quality and visually appealing [19, 1, 2]. At the core of this framework are a pair of neural networks. a generator, G, and a discriminator D. The generator has the objective of mapping a random vector z, which can be sampled from a prior distribution p_Z, to the image space. On the other hand the Discriminator, D has the objective of finding out the likelihood of the generated image being from the image dataset. Hence, G aims to generate realistic images, while D takes in the role of an adversary, always trying to discriminate between the image generated from G and the real image that originated from the dataset's distribution p_{data}. The G and D networks can be trained with their combined objective function as below:

$$\underbrace{\min}_{G}\ \underbrace{\max}_{D}\ V(G,D)$$
$$= \mathbb{E}_{h\sim p_{data}(h)}\big[\log\big(D(h)\big)\big]$$
$$+ \mathbb{E}_{z\sim p_z(z)}\big[\log\big(1-D(G(z))\big)\big] \quad \text{.....(13)}$$

In order to inpaint significant portions of missing data both the generator and the discriminator will have to trained with undamaged, normal data. The GAN based generator for our purpose is the aforementioned DCGAN. Once training is complete, the generator G of the DCGAN can draw points z drawn from a uniform distribution p_Z and synthesize images that looks similar to the samples from the dataset distribution p_{data}. Although GANs are capable are generating new images, they alone are not enough for the inpainting task as the images produced are random and irrelevant to the corrupted image at hand. According to the work done in [18], it known that given a G, which has an efficient representation of the data, must not have a representation of the foreign (corrupted) image in its encoded latent space representation p_{data}. Thus, our goal should be to locate the encoding $\hat{z}$ that has the highest similarity to the damaged image at hand, all the while maintaining a constraint to the latent space. As the algorithm traverses through the latent space,

the search for the closest encoding gets optimized with each step of finding ẑ. Once ẑ is found, the generator G can be called upon to convert the encoding of the latent space to an image. Concretely, the authors of [18] has formulated the procedure of locating ẑ as an optimization problem. Thus the variable y represent the damaged image to recover and a binary mask which is the same size as the image and denoted by M is used to mark the missing parts. Follow through with these notation, the nearest encoding ẑ can be represented as:

$$\hat{z} = \arg\min\{\mathcal{L}_c(z|y, M) + \mathcal{L}_c(z)\} \quad \text{.....(14)}$$

Where Lc denotes the context loss, its purpose is to constraint the generated image provided the input damaged image y and the mask M; Lp denotes the prior loss, its purpose is to penalizes images that does not look visually appealing.

3.2.1. Weighted Context Loss

To fulfill their desired task with utmost coherence they use available information from the uncorrupted regions of the images. (Chen et al) used a context loss to frame such information by calculating a ℓ2 norm between the generated image G(z) and the uncorrupted portion of the input image. They also stated how this proposal is not very efficient as it treats every pixel equally. According to them pixels that are far away from the hole should be treated with lesser precedence compared to that of pixels that are closer to the hole. They proposed that to achieve this goal they asserted that the importance of an uncorrupted pixel is directly correlated with the number corrupted pixels surrounding it. A pixel that is situated very far from the corrupted image has a very little significance. They framed their intuition with the importance weighting term, **W**,

$$\mathbf{W}_i = \begin{cases} \sum_{j \in N(i)} \frac{(1 - M_j)}{|N(i)|} & if\ M_j \neq 0 \\ 0 & if\ M_j = 0 \end{cases} \quad \text{.....(15)}$$

where i is the pixel index, Wi denotes the importance weight at pixel location i, N(i) refers to the set of neighbors of pixel i in a local window, and i, |N(i)| denotes the cardinality of N(i). They used a window size of 7 and stated that l1-norm has worked better compared to l2-norm in their experiments. To sum up they defined contextual loss to be a weighted l1-norm difference between the recovered image and the uncorrupted portions.

$$\mathcal{L}_c(z|y, M) = \|\mathbf{W} \odot (G(z) - y)\|_1 \quad \text{.....(16)}$$

3.2.2 Landmark Loss

The landmark loss can be calculated because of the dlib [10] landmark detector, which takes in an input image and finds the (x,y) coordinates of 68 points on the face. We have tested this detector on normal image and masked image and it worked well on both. For calculating loss, we initially obtained the (x, y) coordinated for the damaged image we want to inpaint, this becomes our ground truth. Then the DCGAN is asked to generate a batch of 64 images. These images are then applied with the same mask and feed to the landmark detector in order to obtain their (x,y) coordinates. For every facial point of the generated image, if the x-coordinate is greater or less than by three, corresponding to x-coordinate of the same facial point of the ground image, we increase the land mark loss $L_{d,x}$ by 0.5. We settled on measuring the loss on a distance of three because this parameter yielded the best results in our experimentation.

$$\mathcal{L}_{d,x}(x) = \begin{cases} 0, & |x - \hat{x}| \leq 3 \\ 0.5, & |x - \hat{x}| > 3 \end{cases} \quad \text{.....(17)}$$

Similarly, we add another 0.5 to $L_{d,y}$ for the y-coordinated for a distance greater than 3 in either direction.

$$\mathcal{L}_{d,y}(y) = \begin{cases} 0, & |y - \hat{y}| \leq 3 \\ 0.5, & |y - \hat{y}| > 3 \end{cases} \quad \text{.....(18)}$$

We then add the total loss for all 68 points for all 64 images in our batch and that yields our final landmark loss $\mathcal{L}_d$ for that batch.

3.2.3. Prior Loss

The prior loss refers to a class of penalties based on high-level image feature representations instead of pixelwise differences. They used prior loss to ensure that the recovered image is highly close to the training set. Their prior loss is different than that of [31] which uses features from pre-trained neural networks. In GANs, the discriminator, D, is trained to differentiate generated images from real images. Therefore, they chose the prior loss to be identical to the GAN loss for training the discriminator D, i.e.

$$\mathcal{L}_p(z) = \lambda \log(1 - D(G(z))) \quad \text{.....(19)}$$

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. Without L(p) the mapping of y to z would have been highly difficult to achieve.

3.2.4 Inpainting

The usage of prior loss, context loss and the landmark loss allowed us to achieve a desirable mapping of the corrupted image from the latent space representation which is denoted by $\hat{z}$. After generating $G(\hat{z})$, the inpainting result can be easily obtained by overlaying the uncorrupted pixels from the input. Poisson Blending [32] to sharpen the final results was applied in [18] but we avoided it. Thus our final loss, L, is defined as:

$$L = \lambda_1 \mathcal{L}_c + \lambda_1 \mathcal{L}_p + \lambda_1 \mathcal{L}_d \text{ ……..(20)}$$

3.3 Training

The entire operation of our model begins by taking in a normal image. We can then apply binary masks to it. The masks can be a centered box, left or right aligned or completely randomly generated. Once the image been processed, we start working with DCGAN. The generative model, G, takes a random 100 dimensional vector sampled from a uniform distribution and generates a

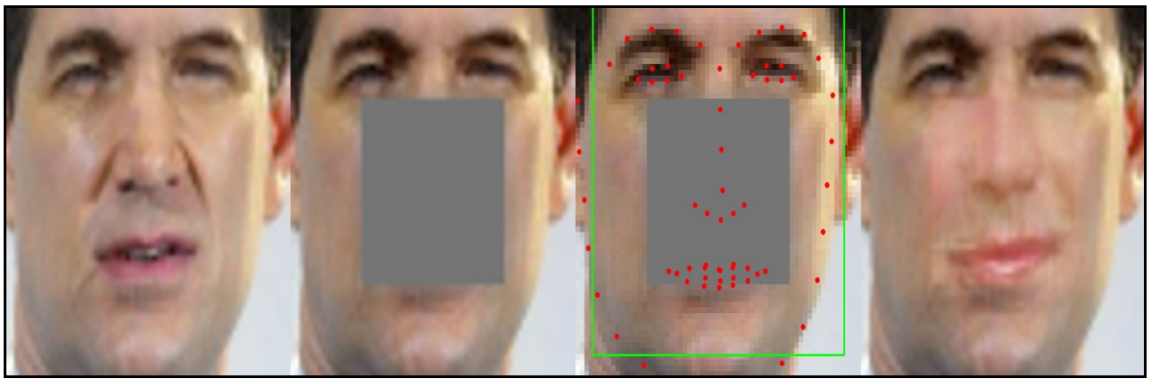

Figure 3. Inpainting at each step.

64x64x3 image. On the other side of the DCGAN generator is the discriminator model, D, which is in reverse order. The input layer of the generator is an image of dimension 64x64x3, it is then followed by a number of convolution layers. In these layers the image dimension is reduced to half, and the channel size is doubled the size of the previous layer. The output layer has the softmax activation function. The training the DCGAN model is done according to [2] and Adam Optimizer is used [4]. We used $\lambda = 0.003$ for all our testing purpose. Once the generator is trained, we output a batch of 64 images, which will naturally be 64x64 in size. The entire batch of image is then masked with the same mask used for the ground truth image.

Once masking is complete, we find the weighted context loss L_c as per equation(16) between the ground truth image and all the masked generated image in our batch. Then the 68 facial point coordinates for the ground image is first obtained with the landmark detector. We then find the facial points for each image in the batch and with that calculate the landmark loss L_d with equations(17,18). The last loss, prior loss, L_p is calculated as per equation(19). Finally, all three losses are calculated and the total loss L is found according to equation(20). This loss can now be used to backpropagate to the input latent dimension z for optimization (Figure 3). In the inpainting stage, we need to find $\hat{z}$ in the latent space z using back-propagation. We have used Adam Optimizer for optimization and restricted z to [-1; 1] for all iterations, in order to achieve the best

possible result. We ran our model for 1000 iterations using the tensorflow [11] framework on a NVDIA 1050Ti GPU enable machine.

Chapter 4

4.1 Experimental Results

In this section of our report, we will discuss the experimentations that we have done with our model. We will delve into details on its performs in hole filling tasks and how it compares to the latest models in use today.

4.2 Datasets and Masks

We have evaluated our method primarily on the CelebFaces Attributes Dataset (CelebA) [5]. This dataset is composed of 202, 599 face images (Figure 4) with coarse alignment [5]. For our experimentation purpose, we have sample around 500 images from the dataset and applied masking the images for semantic hole-filling. All images used for training and hole-filling were initially cropped at the centre (focused primarily at the face) to 64x64, The images captures faces

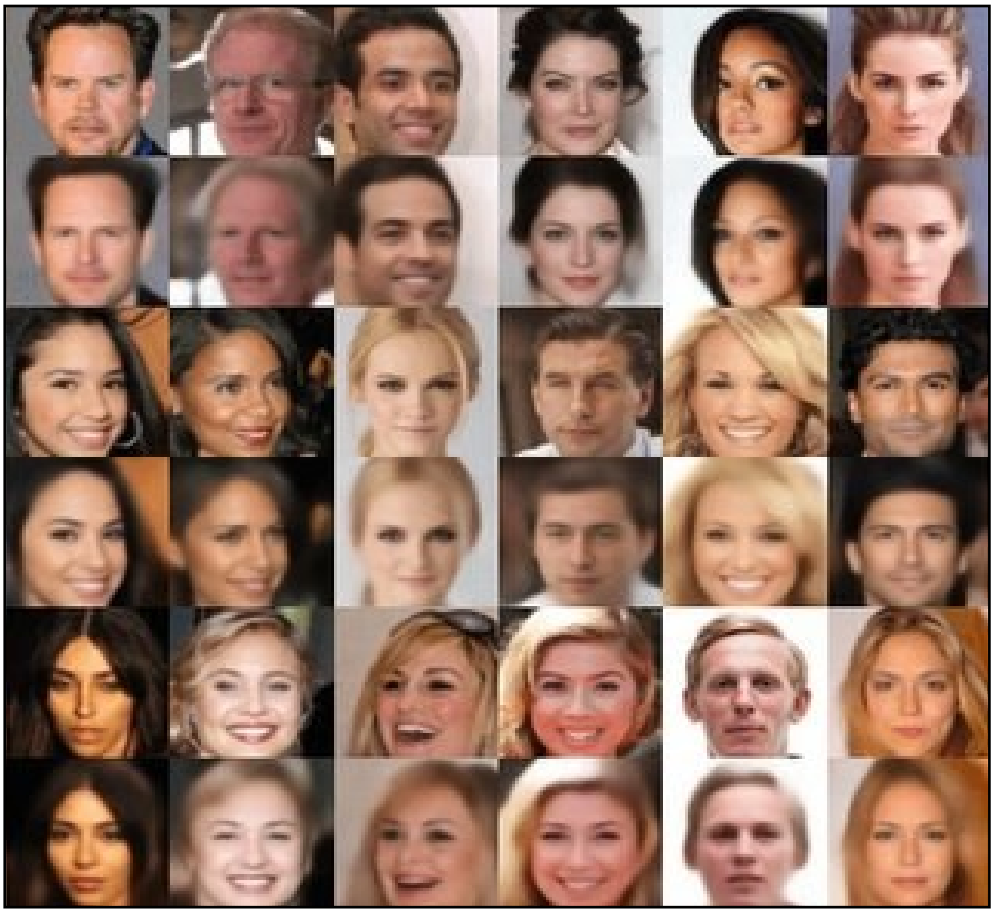

Figure 4. The CelebA Dataset

at different angles, lightning condition and a wide range of skin tone. Once the images were processed, they were ready to be fed into the DCGAN model architecture. The DCGAN was trained and tested according to the instruction previously mention in this report and as intended by the authors [2]. We have tested different types of masks on the image to be tested on, the main variant of the masks used was the central square mask as, in these images, as much as 25% of the face was missing and the model was still capable of generating the missing parts of the face.

4.3 Face Parsing

Although the primary dataset for our model is the CelebA [5] dataset, it not useful for training the facial landmark detector we are using. This dataset do not have segment labels which can guide the algorithm to learn the facial features. Instead, we used a landmark detector that was trained on the iBUG 300-W dataset [7]. This dataset has numerous facial images and each face has 68 segment labels that takes into account all major facial components like eyes, nose, lips, etc.

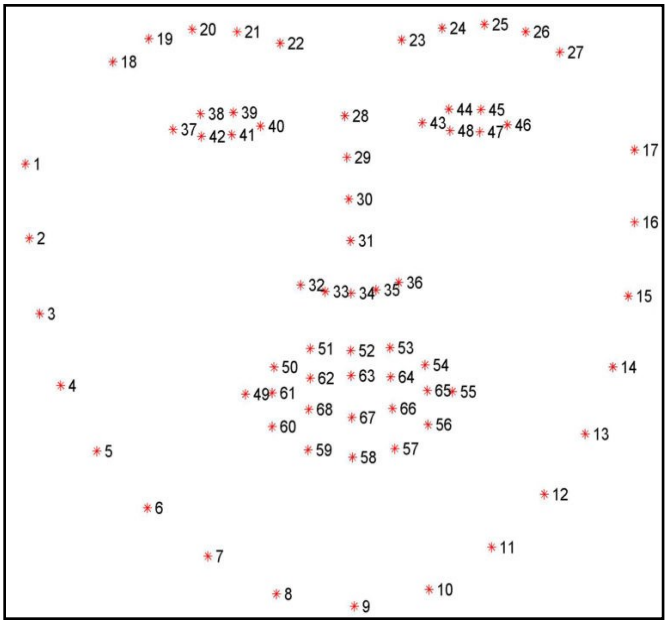

Figure 5. Face Landmark points

Since we used a pre-trained landmark detector directly from dlib's library, it does not undergo any changes during our training and only does detection. We used the landmark detector on masked images. It was surprisingly good at estimating the locations of the facial point even in parts of the face that were completely removed by the binary mask. We first use the landmark detector on the damaged input image, by parsing this face we get the (x,y) coordinated of the 68 facial points

(Figure 5) which becomes our ground truth. We can then compare it with the generated images to regularize the landmark loss.

4.4 Visual Comparisons

We have compare the results of our proposed model with some of the older and newer available models. At first we did our comparisons with inpainting methods like TV [22] and LR inpainting [5, 33]. The methods are local inpainting based. As can be seen in Figure 6, local methods perform quite poorly when it comes to regenerating large regions of damage in an image. However, our output images prove that our method is better at this task as seen in Figure 6. Because the local methods cannot cope with the large number of missing data, the results it produces are generally blurry and noisy. Due to a large number of missing points, TV and LR based methods cannot recover enough image details, resulting in very blurry and noisy images. Other method like PM [20] cannot be used at all because face images do not have similar patches, especially when

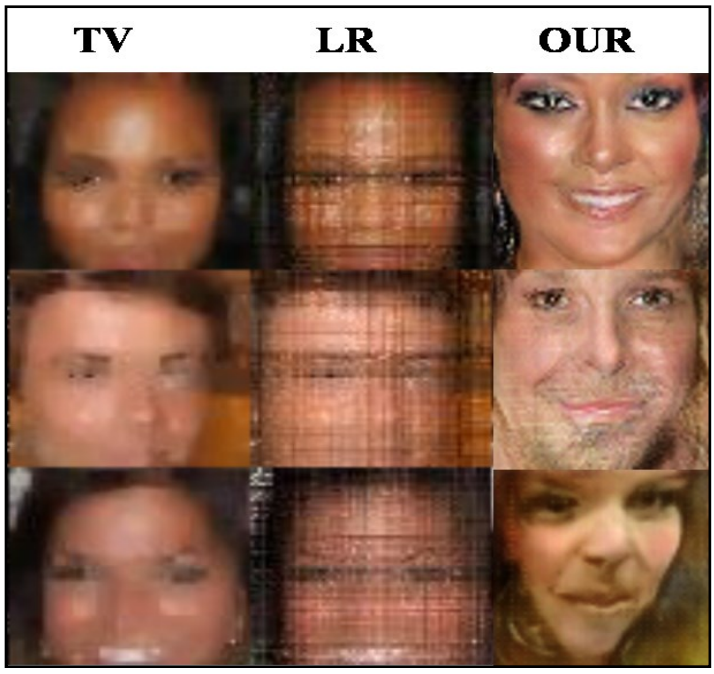

Figure 6. Comparison with older, local models

masked. When comparing with nearest neighbor (NN) filling [16, 34] from the training dataset we observed a lot of misalignment of the main facial features such as eyes, noses, etc. This further enforced the need for our model and the importance of using landmark loss in the overall objective function.

Next we will discuss our results with the latest available methods (Figure. 7). Namely with the context encoder CE [15], semantic inpainting [18] and generative face completion [17]. It is important to note that the masks is required to train the CE and in the generative face completion method. Compared to which our method does not need any specific mask for training. If the mask size changes, the whole network has to be retrained for those model, making it difficult to fill in

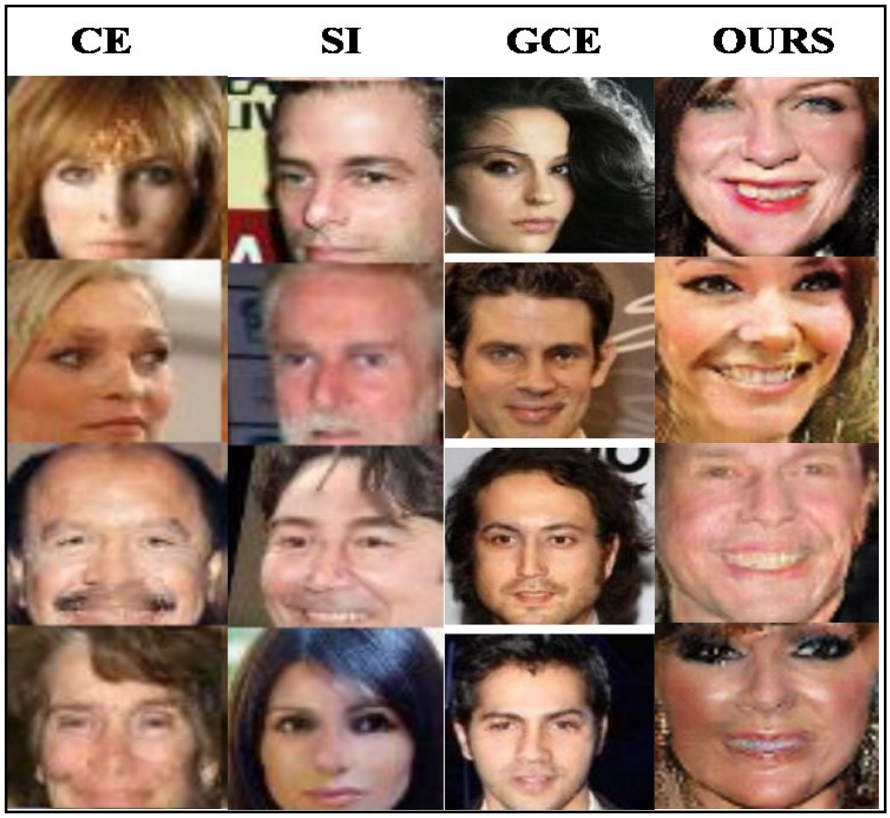

Figure 7. Comparison with latest models

random holes and performing real life inpainting applications. In contrast, our method can be applied to masks of any shape or form without ever needing to retrain the whole network.

One problem with CE particularly is that it does not use the mask for inference but to pre-fill the missing region with an average color. That is why it is prone to mistake undamaged pixels with similar color as missing pixels. Thus CE results in blurry images with ghosting effects. Although the method in [18] overcomes most of the aforementioned issues, it still does not address the problem of lack of structural integrity of generated image and the ground truth image. While the method in [17] does use their own parsing network, they are limited in real life application because

they need a fixed mask for training and needs retraining every time the mask is changed. This makes our model the best of both and quite effective at what it does.

4.5 Quantitative Comparisons

We would like to clarify again that our model is not made to recreate the ground truth image, rather it tries to fill in the damaged place with the most similar possible content. Now, we may get lucky and the true image may lie in the encoding of the latent space of the generator but cannot always be guaranteed. To give a more concrete understanding of the performance of our model, we will be comparing it with three cutting edge models, namely Context encoder (CE) [15], Semantic inpainting (SI) [18] Generaitve face completion (GCE) [17].

The real images from the dataset are used as ground truth reference. Table. 1 and Table. 2 provides the results on the celebA dataset for two tests, The PSNR and SSIM [35]. Compared to our model, the PSNR (Table. 2) and SSIM values (Table. 1) of the latest models are generally similar or slightly better most cases. The main exception is for the random masks, where our model and SI seems to do much better. It is a bit surprising but not shocking because images generate by GANs are commonly known to look visually realistic but not numerically accurate. Thus such conflicts in the result with the visual images, and numerical scoring must be perceived cautiously, since the visual results are significantly better as seen in Figure 7

Table. 1 SSIM score for each level of loss

SSIM	SI (L_1+L_a)	Ours (L_1+L_a+L_d)	CE	GFC
Centre	0.808	0.826	0.818	0.841
Left	0.759	0.741	0.772	0.824
Right	0.763	0.708	0.774	0.826
Random	0.808	0.789	0.675	0.708

Table. 2 PSNR score for each level of loss

PSNR	SI (L_1+L_a)	Ours (L_1+L_a+L_d)	CE	GFC
Centre	19.4	21.1	21.3	20.0

Left	17.9	14.1	15.5	19.8
Right	18.9	15.3	16.0	20.0
Random	22.8	19.7	15.5	14.6

This further enforces the fact that quantitative result do not represent well the real performance of our method especially when the ground truth image can be variable depending on the dataset. Similar observations can be noted in [31, 36], where better visual results corresponds to lower PSNR values. Nevertheless, for random holes, our method performed better in most cases both in PSNR and SSIM scoring. Our method could outperform model like CE and GFC because the undamaged pixels were spread more widely throughout the image, and PSNR is a much more meaningful scoring technique in this case.

4.6 Limitations

Although our model has done very well with the task at hand, t still suffer from some limitations. First, the addition of the landmark detector makes our training process significantly slower. Dlib's landmark detector in itself is not deep learning based so it does not enjoy the benefits of GPU accelerated processing. We found each learning iteration to be noticeably slower than the latest models. Another limitation we have noticed is the lack of understanding of scale by the model. The model is good at locating the structures of the face but it does not understand the size of the facial landmark with respect to the size of the whole face. This mean, in cases of failure, the network may inpaint faces with eyes or nose that are relatively bigger or smaller with respect to the face being inpainted. Finally, we have also noticed that the network cannot distinguish between men and women. It sometimes inpainted male faces with female features as longs as the loss criteria was met. We provide a few samples of our proposed inpainting model in Figure 8.

Figure 8. Sample Outputs

Chapter 5

5.1 Conclusion

In this thesis we propose a deep generative network for structural and semantically coherent image inpainting. The system comprises of a pre-trained GAN, which has two neural networks, a generator and a discriminator, a parser network for landmark detection and adversarial loss for generative model. The proposed model is comparatively more effective at orchestrating semantically legitimate and realistic images for the key missing facial parts from arbitrary clamor. Our strategy enhances existing inpainting methods that reduce context difference by finding the nearest encoding of the obscured picture in the latent space of a pre-trained GAN. Furthermore, we reasoned that the additional landmark loss credits to better comprehension of best dimension setting and thus more outwardly engaging inpainted pictures.

5.2 Future Works

The rise of technological advancements have enabled the world to move at a faster pace and provided the space for additions in the fields of technology. Therefore, focusing on different industries, it leaves a lot of space for improvements. This system can be improved in the future by building a better parser network algorithm for more effective and faster landmark detection. As the system takes most of its' training time in parser network and isn't always able to recognize the orientation of the face and corresponding components, which can be alleviated with 3D data augmentation, it's definitely a portion that we can focus on to improve in the future. Besides, our search for better architecture for the whole system will be in full motion throughout this time. These future plans will definitely help our primary motivation for doing this thesis which is synthetic data.

Reference

[1] Denton, E. L., Chintala, S., & Fergus, R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In Advances in neural information processing systems (pp. 1486-1494).

[2] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional gene Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014).

[3] Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.

[4] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[5] Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. In Proceedings of the IEEE International Conference on Computer Vision (pp. 3730-3738).

[6] Kazemi, V., & Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1867-1874).

[7] Sagonas, C., Tzimiropoulos, G., Zafeiriou, S., & Pantic, M. (2013). 300 faces in-the-wild challenge: The first facial landmark localization challenge. In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 397-403).

[8] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[9] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

[10] King, D. E. (2009). Dlib-ml: A machine learning toolkit. Journal of Machine Learning Research, 10(Jul), 1755-1758.

[11] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). Tensorflow: a system for large-scale machine learning. In OSDI (Vol. 16, pp. 265-283).

[12] Meir, R., & Rätsch, G. (2003). An introduction to boosting and leveraging. In Advanced lectures on machine learning (pp. 118-183). Springer, Berlin, Heidelberg.

[13] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. M¨uller. Efficient backprop. In Neural networks: Tricks of the trade, pages 9–48. Springer, 2012.

[14] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Transactions on Mathematical Software (TOMS), 23(4):550–560, 1997.

[15] D. Pathak, P. Kr¨ahenb¨uhl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. 2016.

[16] J. Hays and A. A. Efros. Scene completion using millions of photographs. ACM Transactions on Graphics, 26(3):4, 2007.

[17] Y. Li, S. Liu, J. Yang, and M.-H. Yang, "Generative face completion," inTheIEEE Conference on Computer Vision and Pattern Recognition (CVPR),vol. 1, no. 2, 2017.

[18] R. A. Yeh, C. Chen, T.-Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models." inCVPR, vol. 2, no. 3, 2017.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D.Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014.

[20] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics, 28(3):24, 2009.

[21] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf. Image completion using planar structure guidance. ACM Transactions on Graphics, 33(4):129, 2014.

[22] M. V. Afonso, J. M. Bioucas-Dias, and M. A. Figueiredo. An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems. IEEE TIP, 2011.

[23] M. Dantone, J. Gall, G. Fanelli, and L. V. Gool. Real-time facial feature detection using conditional regression forests. In CVPR, 2012.

[24] V. Le, J. Brandt, Z. Lin, L. D. Bourdev, and T. S. Huang. Interactive facial feature localization. In ECCV, pages 679–692, 2012.

[25] Y. Le Cun. A theoretical framework for back-propagation. In U. iouretzxy, G. Hinton, and T. Sejnowski, editors, Proceedings of the 1988 Connection:st Models Summer School, pages 21-28,

[26] CMU, Pittsburgh, Pa, 1988. Morgan Kaufmann. Gatys, L., Ecker, A. S., & Bethge, M. (2015). Texture synthesis using convolutional neural networks. In Advances in Neural Information Processing Systems (pp. 262-270).

[27] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. M¨uller. Efficient backprop. In Neural networks: Tricks of the trade, pages 9–48. Springer, 2012.

[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the ACM International Conference on Multimedia, pages 675–678. ACM, 2014.
[29] Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.
[30] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, Jun 2009.
[31] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In ECCV, 2016.
[32] P. P´erez, M. Gangnet, and A. Blake. Poisson image editing. In ACM TOG, 2003.
[33] Y. Hu, D. Zhang, J. Ye, X. Li, and X. He. Fast and accurate matrix completion via truncated nuclear norm regularization. IEEE PAMI, 2013.
[34] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet-based inpainting. In BMVC, 2009.
[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE TIP, 2004.
[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE TIP, 2004.