Skin Lesion Synthesis using GANs

By

Harsh Lunagariya

17BCE049

Maharsh Patel

17BCE050

DEPARTMENT OF COMPUTER ENGINEERING

Ahmedabad 382481

Skin Lesion Synthesis using GANs

Mini Project - III

Submitted in fulfillment of the requirements
For the degree of
Bachelor of Technology in Computer Engineering

By

Harsh Lunagariya(17BCE049)

Maharsh Patel (17BCE049)

Guided By
Assistant Prof. Kavita Tewani

DEPARTMENT OF COMPUTER ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING
Ahmedabad 382481

Certificate

This is to certify that the project entitled “Skin Lesion Synthesis using GANs” submitted by Harsh Lunagariya (17BCE049) and Maharsh Patel (17BCE050) towards the partial fulfilment of the requirements for the degree of Bachelor of Technology in Computer Engineering of Nirma University is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Kavita Tewani
Assistant Professor
Department of Computer Engineering
Institute of Technology
Nirma University, Ahmedabad

Dr. Madhuri Bhavsar
Professor and Head of Department
Department of Computer Engineering
Institute of Technology
Nirma University, Ahmedabad

Acknowledgment

I am thankful to and fortunate enough to get constant encouragement, support, and guidance from Prof. Kavita Tewani, who helped me in completing my work on “Skin Lesion Synthesis using GAN”.

Also, I would like to extend my sincere esteem to all the helpful and useful suggestions given by my friends and seniors to enhance the content of my mini-project.

I also express my sincere thanks to all the teaching and non-teaching staff of my department i.e. Department of Computer Engineering, Nirma University, Ahmedabad.

Abstract

The report describes the project, “Skin Lesion Synthesis using GAN” in which the images of Skin Lesion are synthesized using the Generative Adversarial Network. By generating such images, we can expand our dataset and use it in Machine Learning Applications.

1 Table of Contents

2 Table of Figures

3 Introduction

1.1 General

Generative Adversarial Networks (GANs) have attracted the deep learning community because of its ability to generate data by using the same statistics as training data. The GAN is used for data augmentation, data generation, image-to-image translation, etc. Researchers in the field of medical imaging have also incorporated the use of GANs for the generation of scare and privacy protected medical data of patients.

1.2 Objective

GANs can generate more data for the neural networks which require a huge amount of data for training. Medical Researchers have employed GANs for medical image synthesis in medical domains like skin melanoma, benign lesions, retinal fundi, PET images, brain segmentation etc.

1.3 Scope of work

GANs will allow us to generate more image data for the training of data-hungry neural networks.

4 Literature survey

4.1 General

The literature survey covered books, research papers, and journal articles that explain the topic GANs and discuss maths behind their work. Papers and articles which covers techniques for generation of images for skin lesion are covered. Those papers and useful articles are added in the references section at last.

5 Introduction of GAN

This architecture generally consists of two neural networks that compete against each other in adversarial training to produce generative models.

5.1 Generative and discriminative models

It consists of two simultaneously trained models:

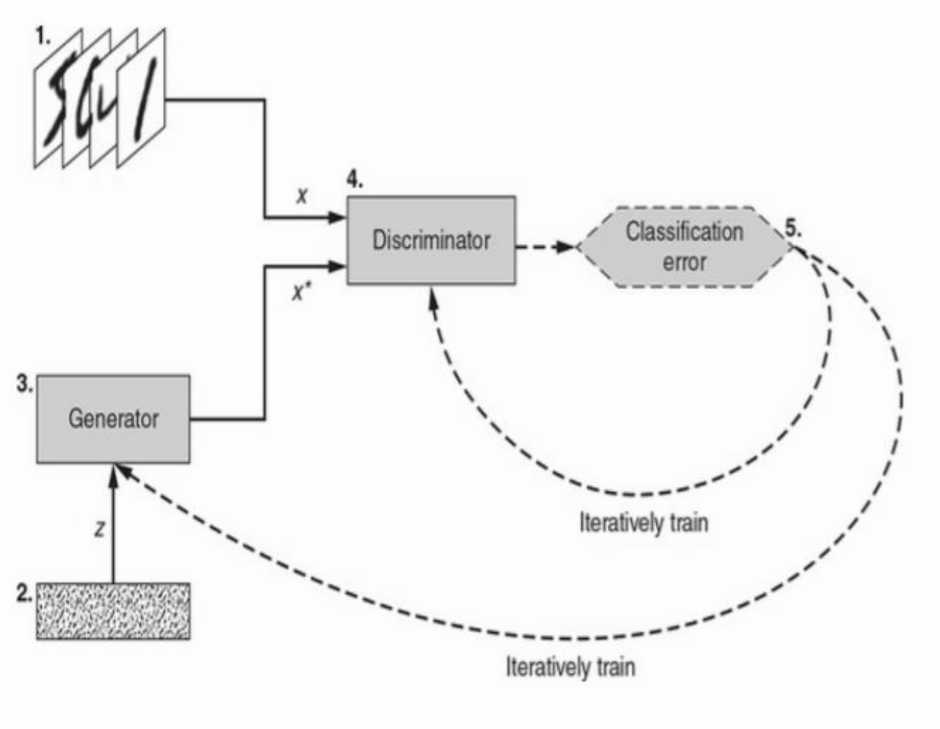
- The Generator: trained to generate fake data,
- The Discriminator: trained to distinguish the fake data from real data.

The term adversarial means involving opposition, competitive nature between the two models that pits in GAN architecture.

5.2 Working of GANs

In simple words, the Generator tries to fool the Discriminator. Another metaphor from Ian Goodfellow uses – is that of a criminal (the Generator) who forges money, and a detective (the Discriminator) who tries to catch him. The more original-looking the fake cash becomes, the better the detective must be at detecting at them, and the other way around.

In technical words, the Generator's motive is to produce samples that captures the features of the training dataset, so that the sample generated look indistinguishable. The Generator learns to generate them from latent noise (a random vector of a specific size). The Generator learns from the classifications of the Discriminator through the feedback it receives. The goal of the Discriminator is to decide whether a particular example is true (coming from the training dataset) or false (created by the Generator). Accordingly, if the Discriminator is tricked into classifying a fake image as true, the Generator knows that it has done something right. Conversely, whenever the Discriminator correctly rejects an image generated by the Generator as false, the Generator receives the feedback that it needs to improve. The Discriminator continues to improve.



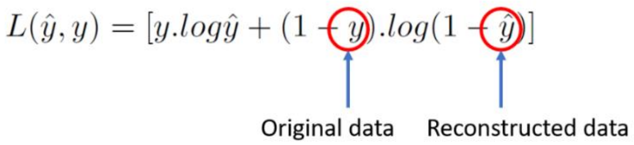
5.1 GAN subnetworks and training process

5.3 Mathematics behind GAN

LOSS FUNCTION OF GAN

- DISCRIMINATOR: Role is to distinguish between actual data and fake data.
- GENERATOR: Role is to create data in such a way so that it can fool the generator.

LOSS FUNCTION (BINARY CROSS ENTROPY)



$$L(\hat{y}, y) = [y.log\hat{y} + (1 - y).log(1 - \hat{y})]$$

Discriminator loss:

While training discriminator, the label of data coming from $P_{data}(x)$ is $y = 1$ (real data) and $\hat{y} = D(x)$. Substituting this in above loss function we get,

$$L(D(x), 1) = log(D(x)) \tag{1}$$

and for data coming from generator, the label is $y = 0$ (fake data) and $\hat{y} = D(G(z))$. So in this case,

$$L(D(G(z)), 0) = log(1 - D(G(z))) \tag{2}$$

Now, the objective of the discriminator is to correctly classify the fake and real dataset. For this, equations (1) and (2) should be maximized and final loss function for the discriminator can be given as,

$$L^{(D)} = \max[log(D(x)) + log(1 - D(G(z)))] \tag{3}$$

GENERATOR LOSS:

Here, the generator is competing against discriminator. So, it will try to minimize the equation (3) and loss function is given as,

$$L^{(G)} = \min[log(D(x)) + log(1 - D(G(z)))] \tag{4}$$

Combining these both conditions of generator and discriminator we get

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

5.4 Types of GAN

- Deep Convolutional GAN (DCGAN): Mostly used for images. Consist of convolutional networks. They can be used for style transfer.

- Conditional GAN (CGAN): similar to DCGAN. The only difference is, in this model we provide a one-hot vector for class classification. They are not necessarily unsupervised.
- Wasserstein GAN (WGAN): This type of generative adversarial network uses Wasserstein distance in its cost function.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \big[\, \|x - y\| \,\big] \ ,$$

- And many more.

6 Skin Lesion Synthesis using GAN

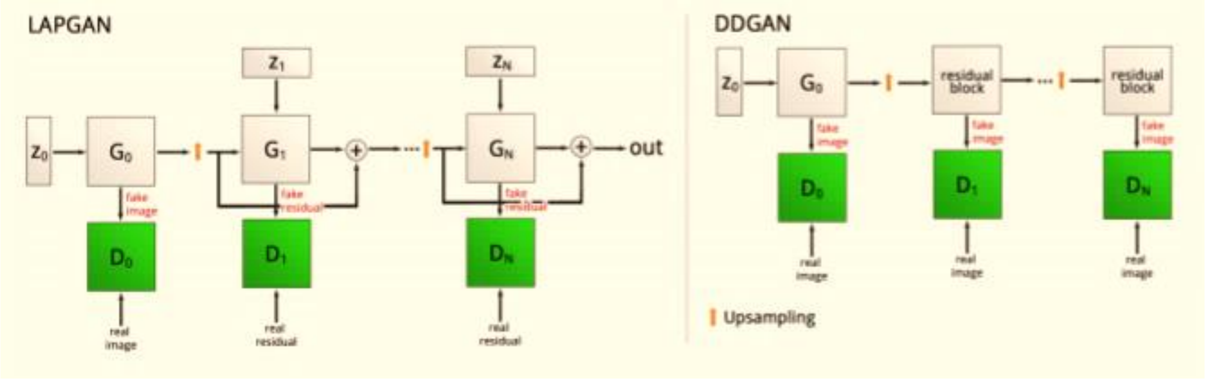
6.1 MelanoGANs [1]

The proposed model generates skin lesion images using Deeply Discriminated GAN (DDGAN).

The model is trained using the ISIC 2017 dataset containing 2000 256x256 pixels dermoscopic images of both benign and malignant skin lesions. (1372 benign lesions, 374 melanoma images and 254 images of seborrheic keratosis)

6.1.1 Working:

The DDGAN architecture for the generator is proposed which maps the probability distribution function of the training data to produce low-resolution image samples. The low-resolution discriminator then pro-vides the lowest resolution generator with gradients. This process is continued by upsampling the images and feeding them into another generator.



6.1 Architecture of LAPGAN and DDGAN

6.1.2 Methods of Evaluation:

1) Comparison of normalized colour histograms of generated images with the histogram of original images from the training set by using JS Divergence.

2) Wasserstein distance

6.1.3 Results:

- DCGANs can work well for image synthesis at resolutions of 64x64 px.

- LAPGAN can generate images at resolutions of 96x96 px.

6.2 Skin Lesion Synthesis with Generative Adversarial Networks [2]

The main objective of this paper is to generate high-resolution (1024x512) synthetic images of skin lesions without compromising on the fine-grained details. Focus on malignancy markers and their coherent placement and sharpness results in the generation of visually appealing images. pix2pixHD GAN is introduced for the accomplishment of this task.

6.2.1 Dataset:

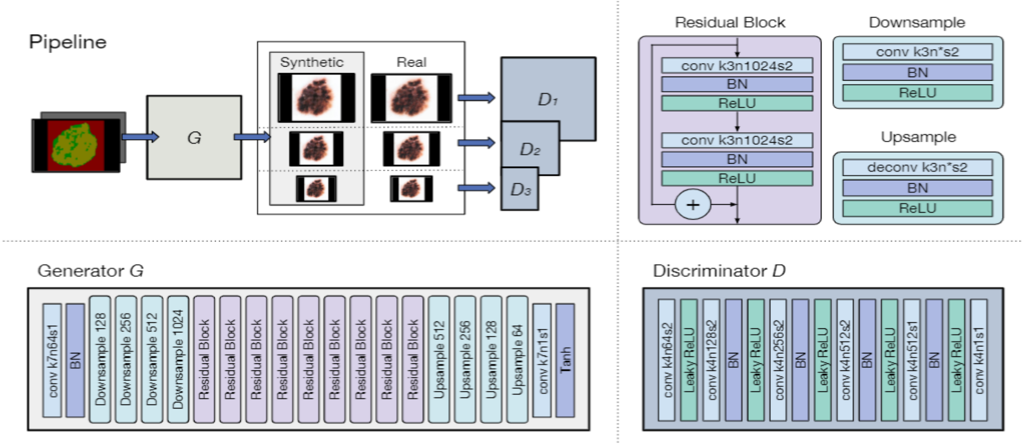
The model is trained using the ISIC 2017 database challenge with 2,000 dermoscopic images, ISIC Archive with 13,000 dermoscopic images, Dermofit Image Library with 1,300 images, and PH2dataset with 200 dermoscopic images.

For testing, the Interactive Atlas of Dermoscopy with 900 dermoscopic images (270 melanomas) are used.

6.2.2 Working:

The proposed pix2pix GAN model is enhanced by employing a coarse-to-fine generator, a multi-scale discriminator architecture, and a robust adversarial learning objective function. The Inception-v4 network is employed for the synthesis of skin lesion images.

Here, instead of generating the images from noise (usual procedure with GANs), semantic label map (an image where each pixel value represents the object class) is synthesized and an instance map (an image where the pixels combine information from its object class and its instance) is generated.

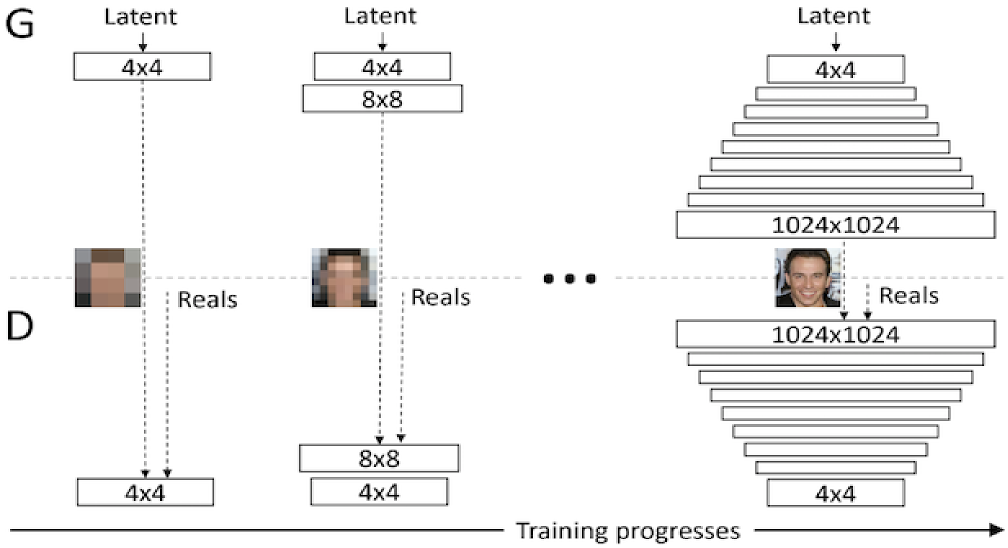


6.2.3 Methods of Evaluation:

To evaluate the relevance of synthetic images, a skin cancer classification network with synthetic and real images is trained.

6.3 High-resolution medical image synthesis using progressively grown generative adversarial networks [3]

The main purpose of this paper was to generate high resolution images using iterative training phase in which we train for small to high resolution iteratively. It provides improved quality, variation and constancy.



More details are covered in implementation section.

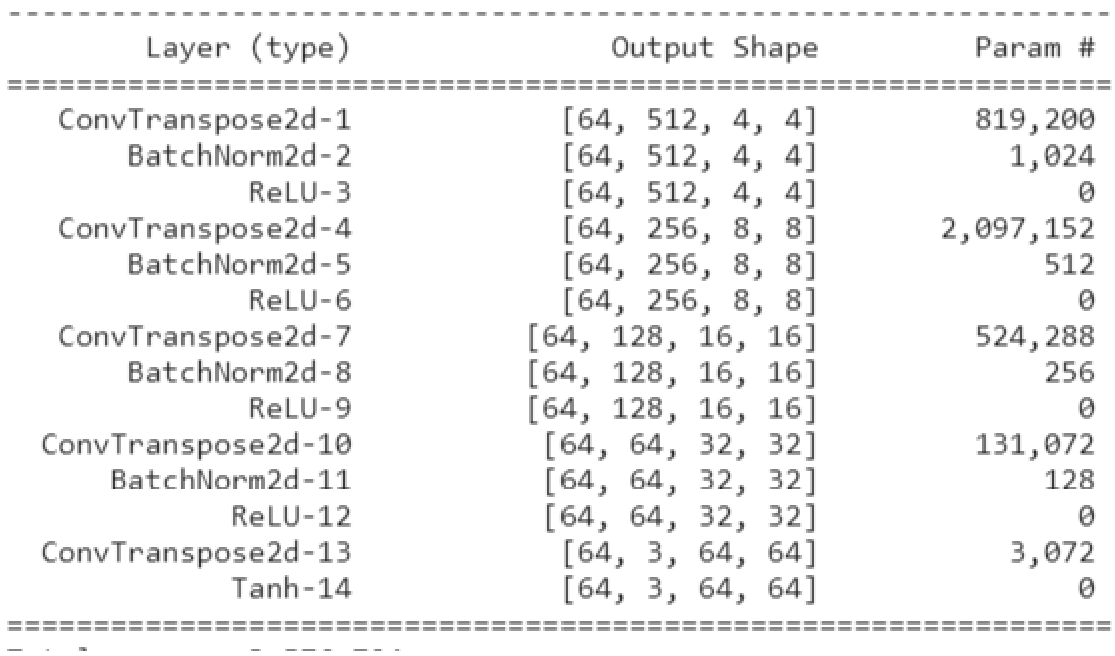
7 Implementation and Results

7.1 DCGAN

7.1.1 Architecture

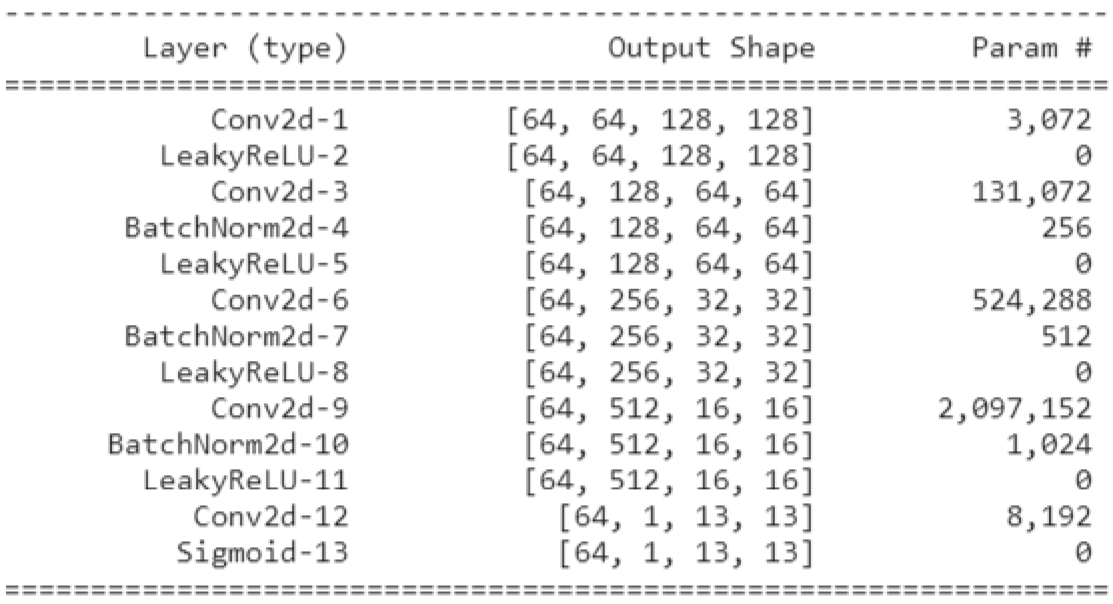
Implemented the following architecture.

Generator



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [64, 512, 4, 4]         819,200
       BatchNorm2d-2            [64, 512, 4, 4]           1,024
              ReLU-3            [64, 512, 4, 4]               0
   ConvTranspose2d-4            [64, 256, 8, 8]       2,097,152
       BatchNorm2d-5            [64, 256, 8, 8]             512
              ReLU-6            [64, 256, 8, 8]               0
   ConvTranspose2d-7          [64, 128, 16, 16]         524,288
       BatchNorm2d-8          [64, 128, 16, 16]             256
              ReLU-9          [64, 128, 16, 16]               0
  ConvTranspose2d-10           [64, 64, 32, 32]         131,072
      BatchNorm2d-11           [64, 64, 32, 32]             128
             ReLU-12           [64, 64, 32, 32]               0
  ConvTranspose2d-13            [64, 3, 64, 64]           3,072
             Tanh-14            [64, 3, 64, 64]               0
================================================================
```

Discriminator

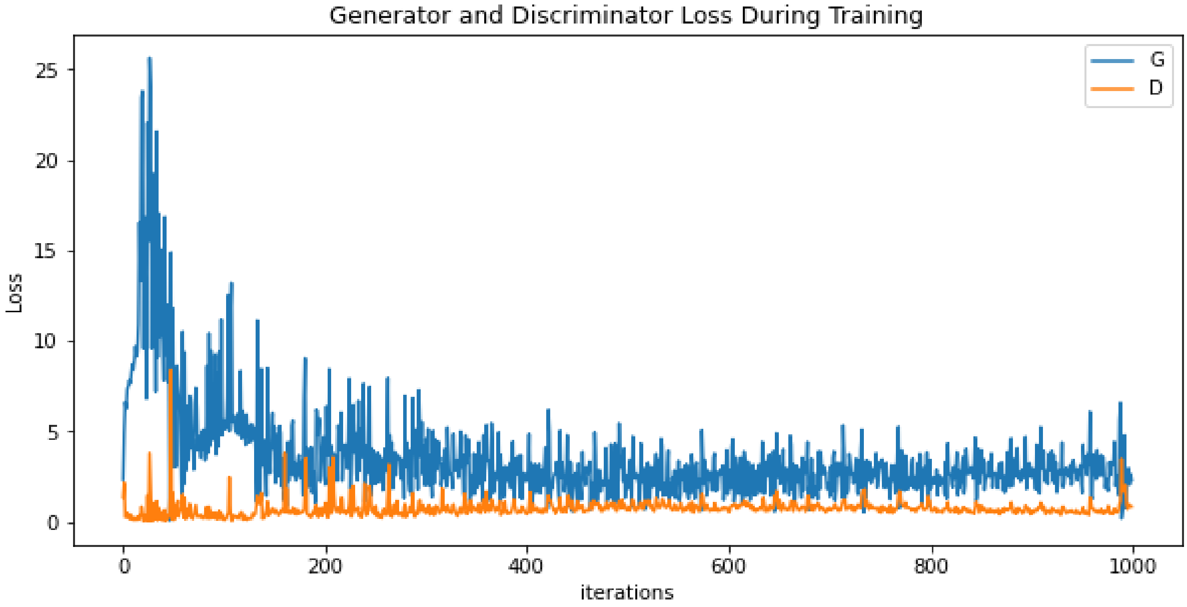


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [64, 64, 128, 128]           3,072
         LeakyReLU-2        [64, 64, 128, 128]               0
            Conv2d-3          [64, 128, 64, 64]         131,072
       BatchNorm2d-4          [64, 128, 64, 64]             256
         LeakyReLU-5          [64, 128, 64, 64]               0
            Conv2d-6          [64, 256, 32, 32]         524,288
       BatchNorm2d-7          [64, 256, 32, 32]             512
         LeakyReLU-8          [64, 256, 32, 32]               0
            Conv2d-9          [64, 512, 16, 16]       2,097,152
      BatchNorm2d-10          [64, 512, 16, 16]           1,024
        LeakyReLU-11          [64, 512, 16, 16]               0
           Conv2d-12            [64, 1, 13, 13]           8,192
          Sigmoid-13            [64, 1, 13, 13]               0
================================================================
```

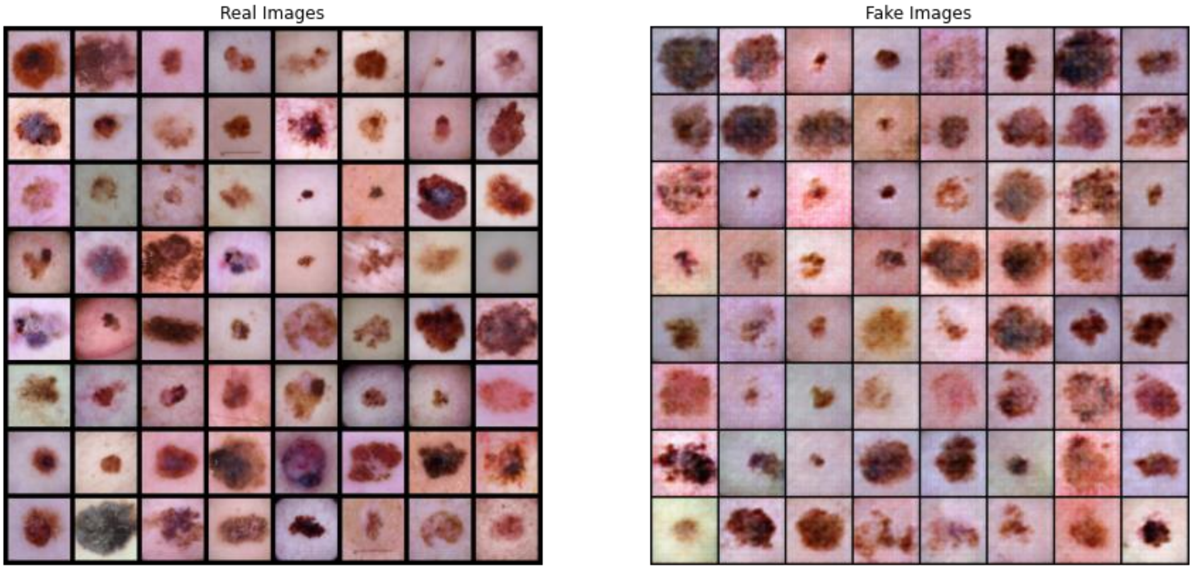
7.1.2 Training

Started with randomly initialized weight in the network with a mean of 0 or 1. Used binary-cross entropy as loss function and Adam optimizer. Trained for 500 epochs with the batch size of 64. Trained for ISIC 2017 dermoscopic images of skin lesions.

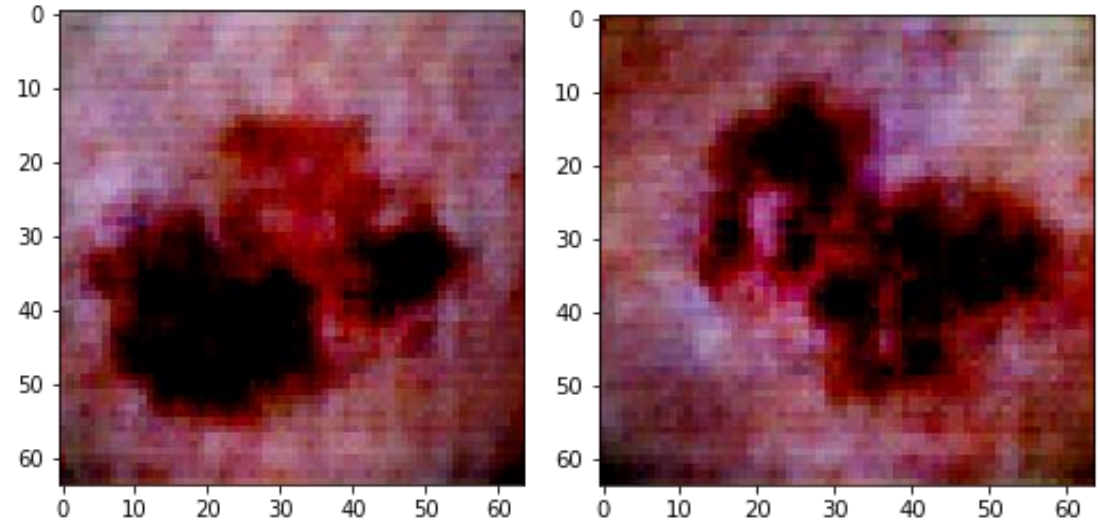
7.1.3 Result



7.1 Generator and Discriminator loss during training



7.2 Real and Generated Images



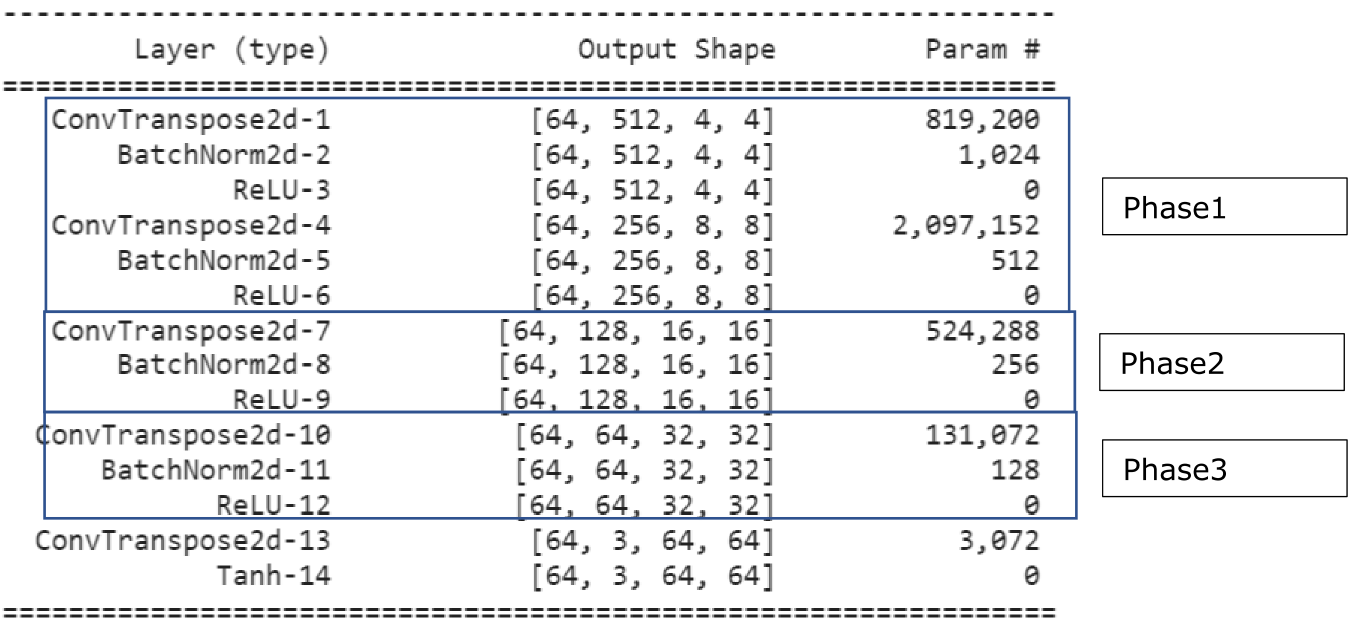
7.3 Generated (Fake) Images

7.2 Progressively Growing GAN

7.2.1 Architecture

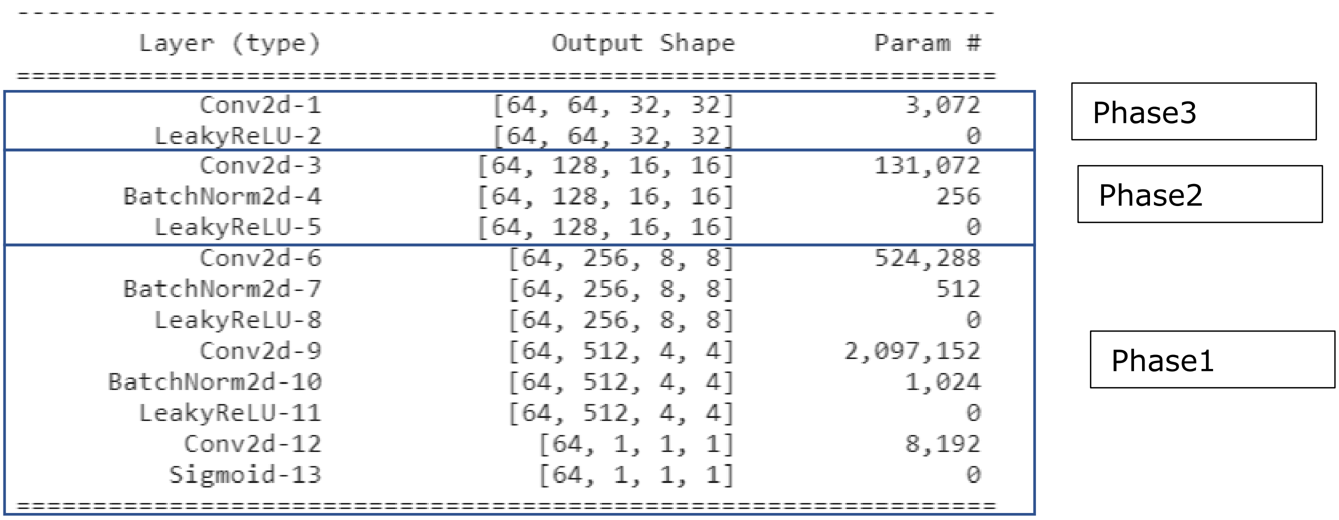
Implemented the following architecture:

Generator



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [64, 512, 4, 4]         819,200
       BatchNorm2d-2            [64, 512, 4, 4]           1,024
              ReLU-3            [64, 512, 4, 4]               0
   ConvTranspose2d-4            [64, 256, 8, 8]       2,097,152
       BatchNorm2d-5            [64, 256, 8, 8]             512
              ReLU-6            [64, 256, 8, 8]               0
   ConvTranspose2d-7          [64, 128, 16, 16]         524,288
       BatchNorm2d-8          [64, 128, 16, 16]             256
              ReLU-9          [64, 128, 16, 16]               0
  ConvTranspose2d-10           [64, 64, 32, 32]         131,072
      BatchNorm2d-11           [64, 64, 32, 32]             128
             ReLU-12           [64, 64, 32, 32]               0
  ConvTranspose2d-13            [64, 3, 64, 64]           3,072
             Tanh-14            [64, 3, 64, 64]               0
================================================================
```

Discriminator



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [64, 64, 32, 32]           3,072
         LeakyReLU-2           [64, 64, 32, 32]               0
            Conv2d-3          [64, 128, 16, 16]         131,072
       BatchNorm2d-4          [64, 128, 16, 16]             256
         LeakyReLU-5          [64, 128, 16, 16]               0
            Conv2d-6            [64, 256, 8, 8]         524,288
       BatchNorm2d-7            [64, 256, 8, 8]             512
         LeakyReLU-8            [64, 256, 8, 8]               0
            Conv2d-9            [64, 512, 4, 4]       2,097,152
      BatchNorm2d-10            [64, 512, 4, 4]           1,024
        LeakyReLU-11            [64, 512, 4, 4]               0
           Conv2d-12              [64, 1, 1, 1]           8,192
          Sigmoid-13              [64, 1, 1, 1]               0
================================================================
```

7.2.2 Training

Started with randomly initialized weight in the network with a mean of 0 or 1. Used binary-cross entropy as loss function and Adam optimizer. Trained for 500 epochs with the batch size of 64. Trained for ISIC 2017 dermoscopic images of skin lesions.

Trained the model phase wise:

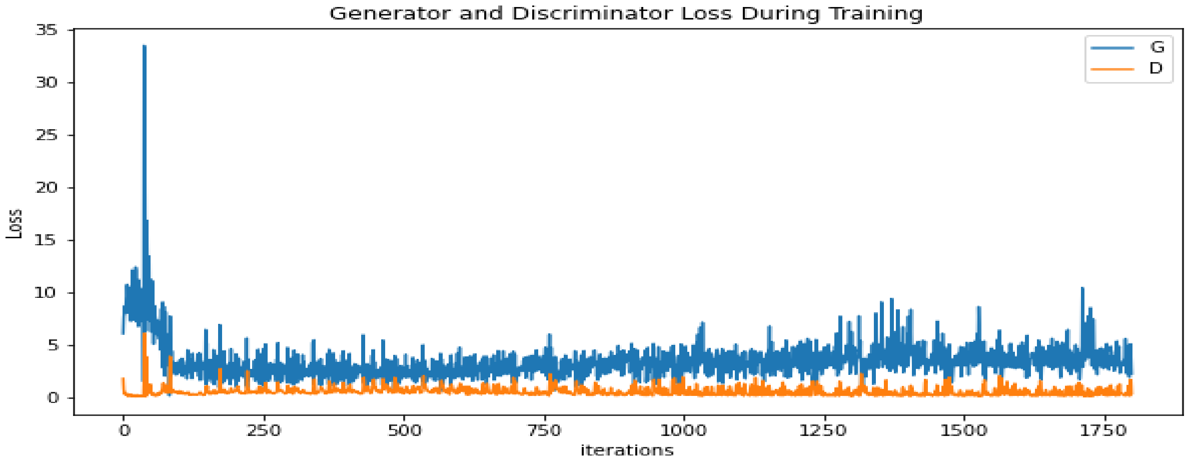
In 1st phase we will train it for 16 x 16 resolution.
We will freeze the phase 1 layers.

In 2nd phase we will add additional layers and train it for 32 x 32 resolution.
We will freeze the phase 2 layers. (At this time phase 1 layers will be frozen.)

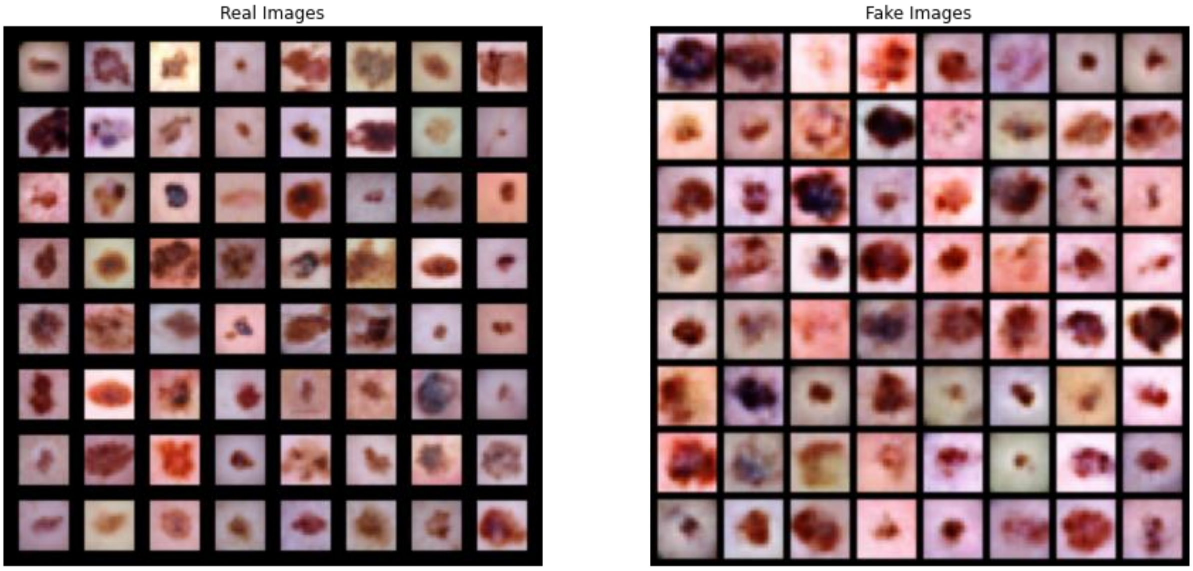
In 3rd phase we will add additional layers and train it for 64 x 64 resolution. We will freeze the phase 3 layers. (At this time phase 1 and 2 layers will be frozen.)

7.2.3 Result

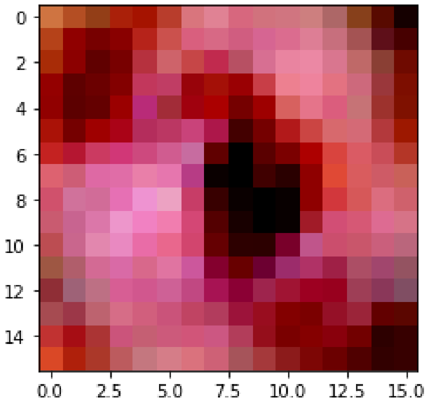
Phase 1:



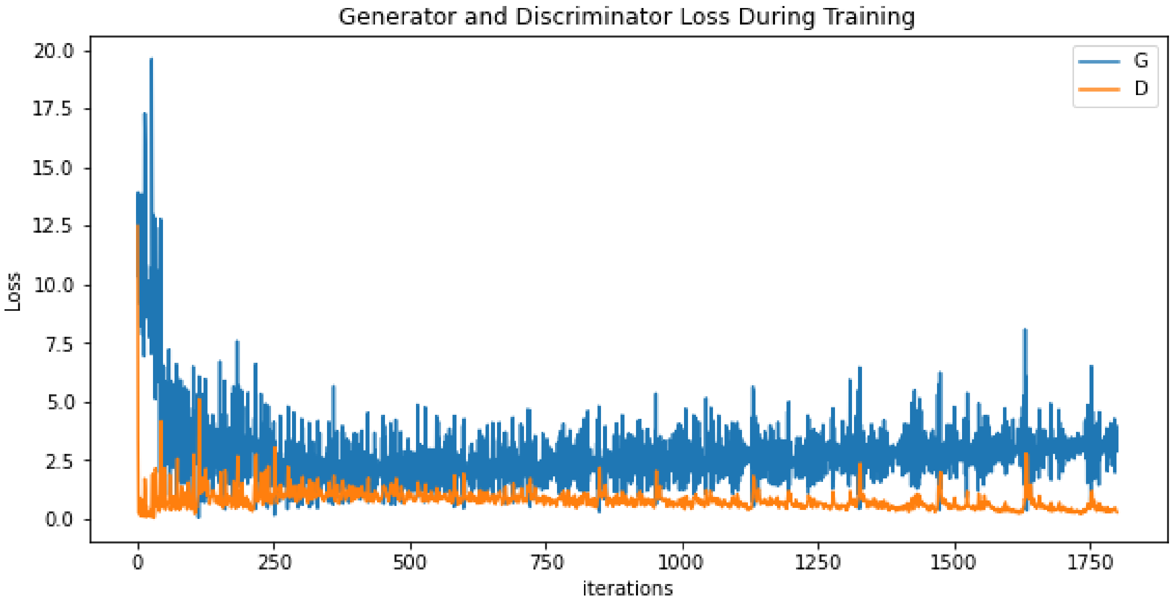
7.4 Generator and Discriminator loss during training



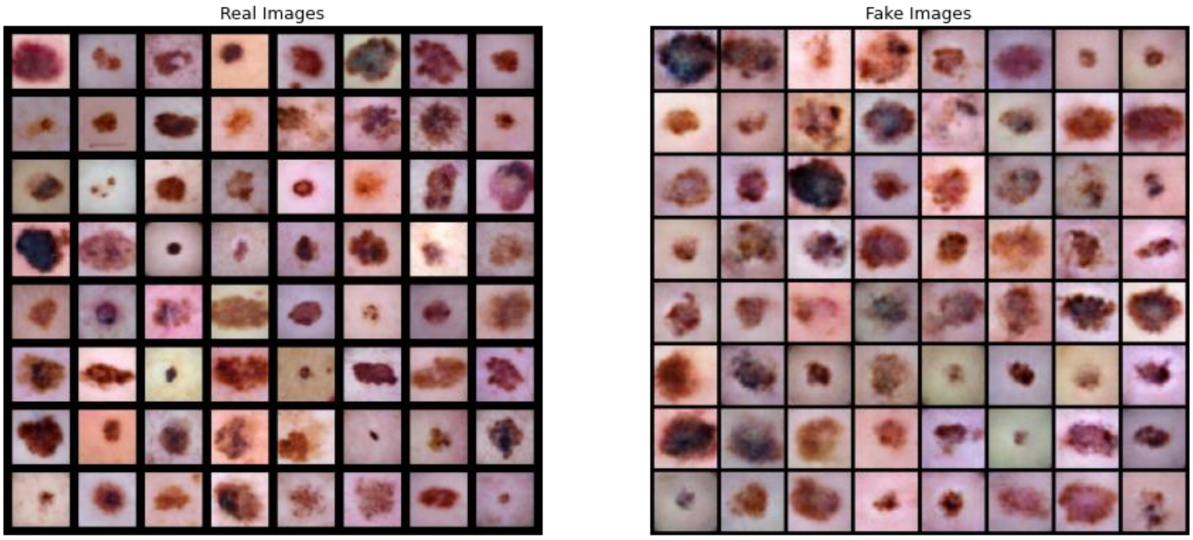
7.5 Real and Generated Images



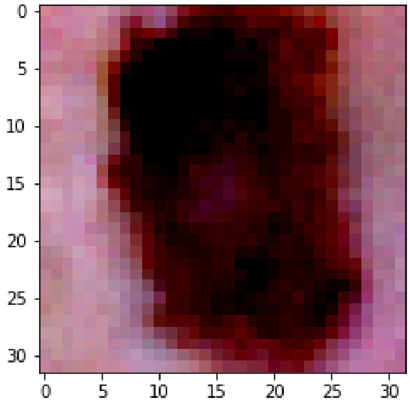
Phase 2:



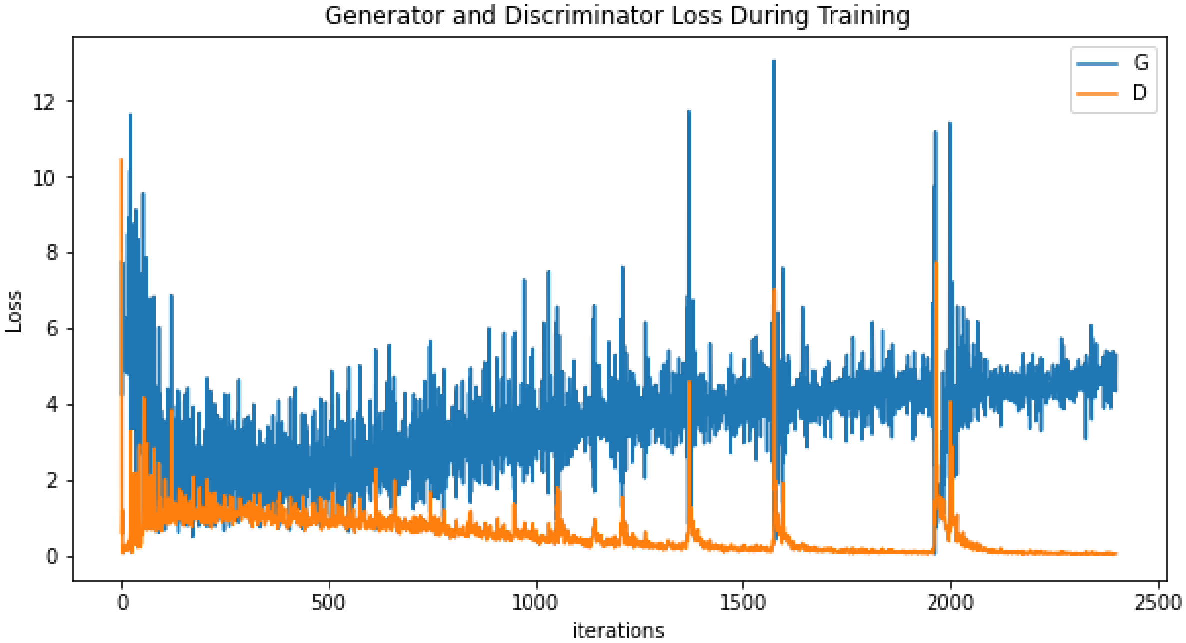
7.6 Generator and Discriminator loss during training



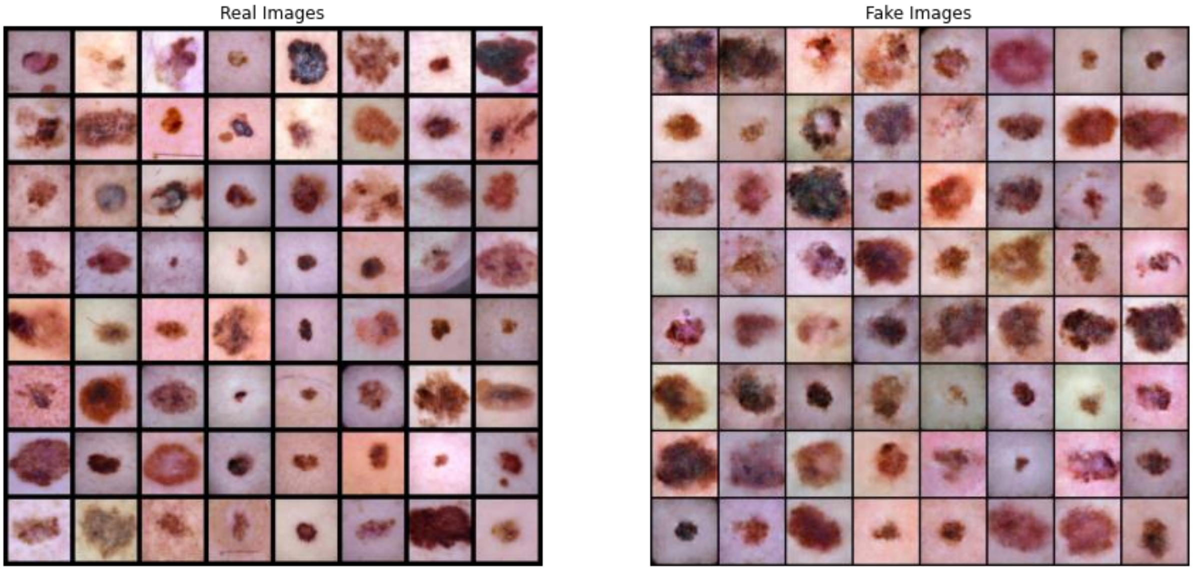
7.7 Real and Generated Images



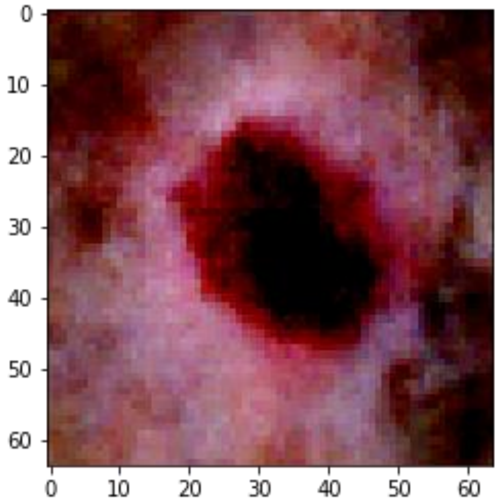
Phase 3:

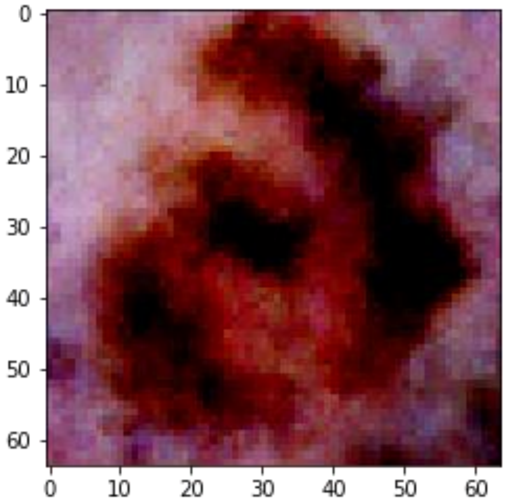


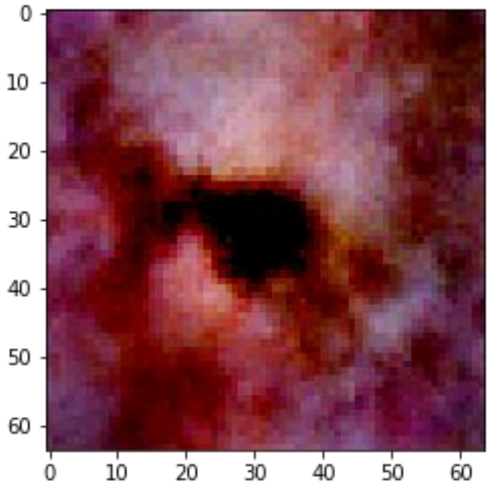
7.8 Generator and Discriminator loss during training



7.9 Real and Generated Images







7.3 Comparison:

- Learn fast and new patterns compared to DCGAN.
- Solution to mode collapse problem, which occurs in GAN
- Better resolution with equal iterations
- More stable image generation

Loss	DCGAN	PGGAN
Discriminator	0.57	0.05
Generator	3.1	3.98

8 Conclusion

GAN can be used to generate training data with high resolution, vast variation and stability for training purpose or anything else.

Bibliography

[1] C. Baur, S. Albarqouni and N. Navab, “MelanoGANs: High Resolution Skin Lesion Synthesis with GANs,” *CoRR,* vol. abs/1804.04338, 2018.

[2] A. Bissoto, F. Perez, E. Valle and S. Avila, “Skin lesion synthesis with generative adversarial networks,” in *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, Springer, 2018, pp. 294-302.

[3] A. Beers, J. M. Brown, K. Chang, J. P. Campbell, S. Ostmo, M. F. Chiang and J. Kalpathy-Cramer, “High-resolution medical image synthesis using progressively grown generative adversarial networks,” *CoRR,* vol. abs/1805.03144, 2018.

Appendix A – List of Useful Websites

1. A Gentle Introduction to the Progressive Growing GAN
   https://machinelearningmastery.com/introduction-to-progressive-growing-generative-adversarial-networks/
2. Deep Convolutional Generative Adversarial Network
   https://www.tensorflow.org/tutorials/generative/dcgan
3. DCGAN Tutorial
   https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html