CE 712 Project

High Resolution Image Generation for Remote Sensing

Chaitanya Langde
Saurabh Mahra

Need for Generation

- Scarcity of quality and labelled Data
- Required for variety of applications in Remote Sensing

Available Solutions

1. EEGAN
2. SRGAN
3. **DCGAN**

DCGAN

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

Alec Radford & Luke Metz
indico Research
Boston, MA
{alec,luke}@indico.io

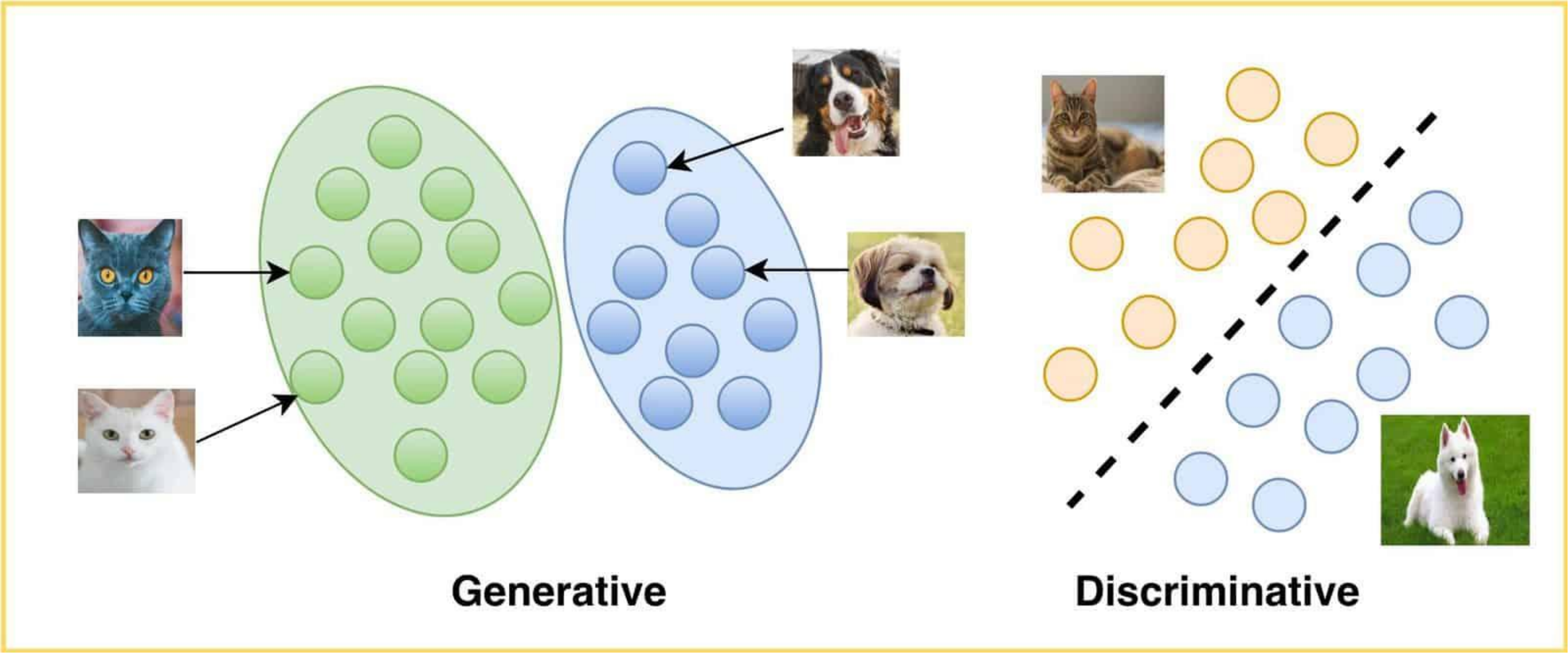
Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com

Abstract

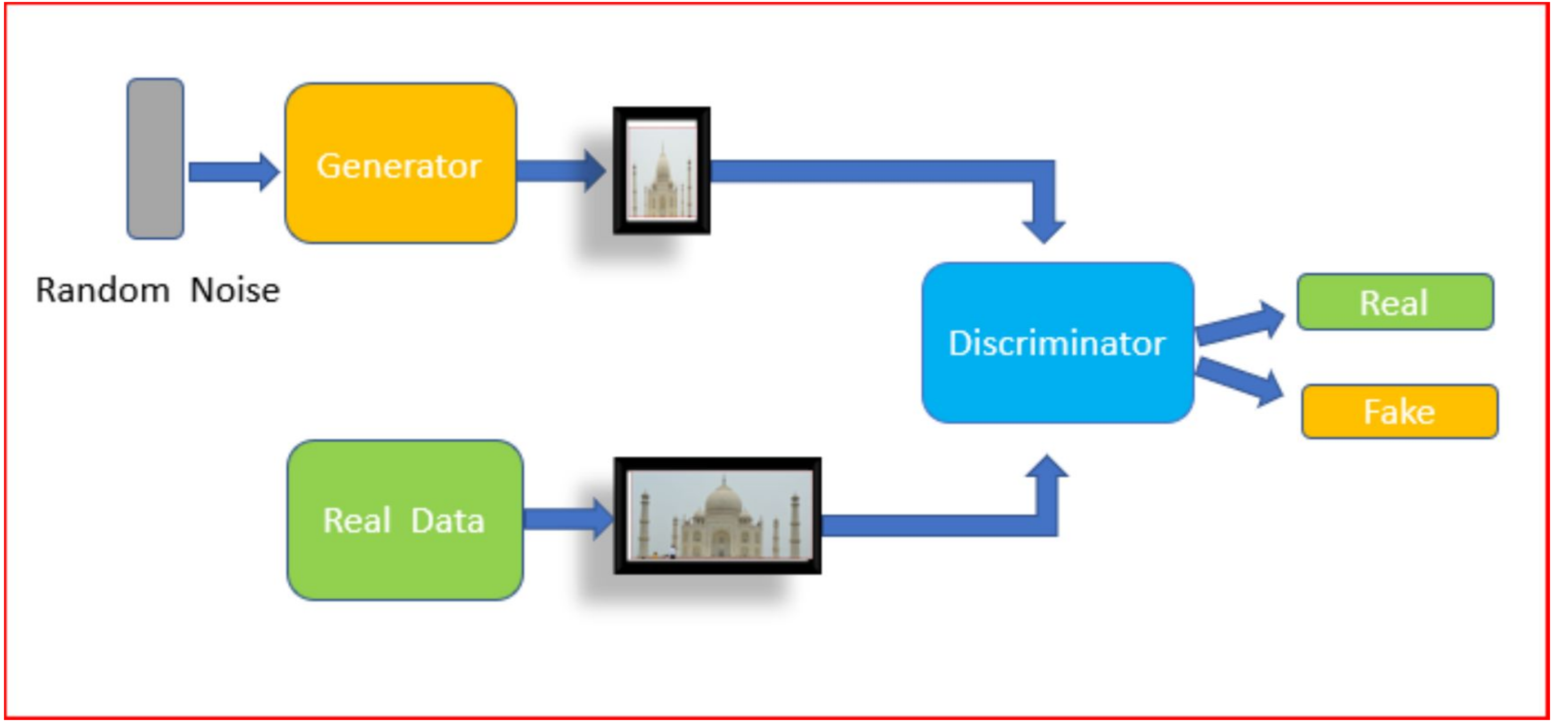
In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. Comparatively, unsupervised learning with CNNs has received less attention. In this work we hope to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. We introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning. Training on various image datasets, we show convincing evidence that our deep convolutional adversarial pair learns a hierarchy of representations from object parts to scenes in both the generator and discriminator. Additionally, we use the learned features for novel tasks - demonstrating their applicability as general image representations.

Ref: https://arxiv.org/pdf/1511.06434.pdf

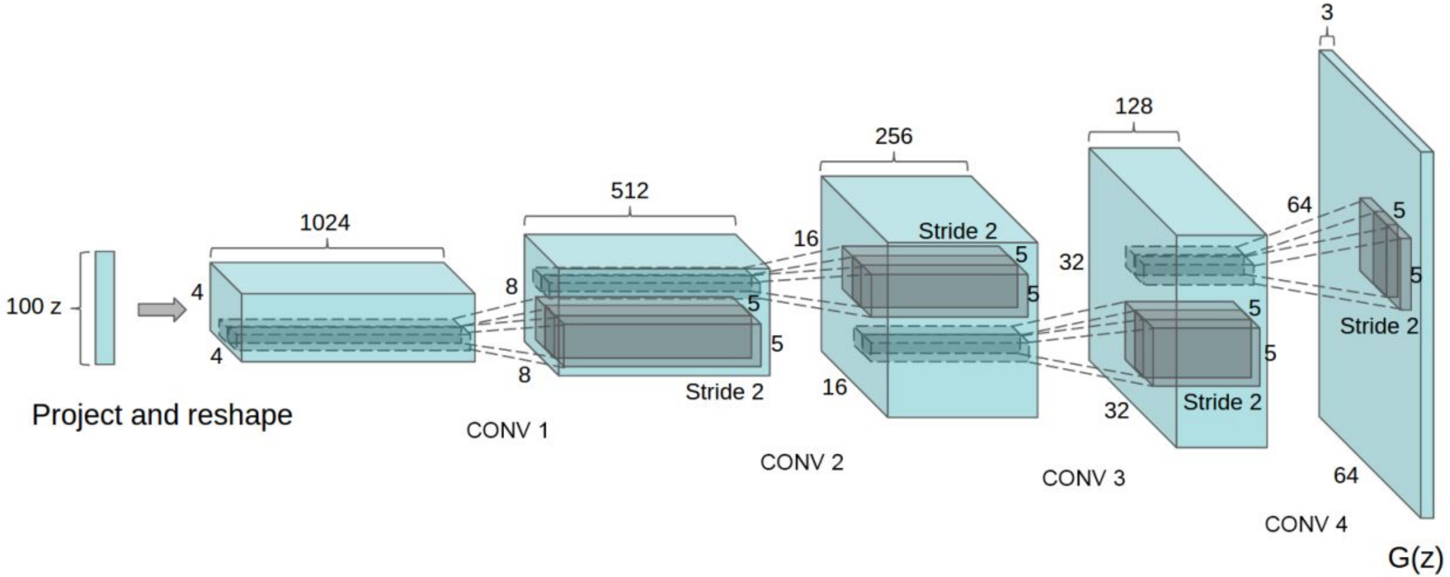
What is a GAN?



How it Works?

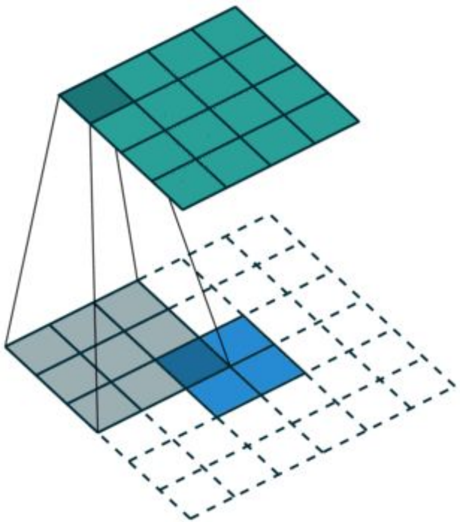


DCGAN?

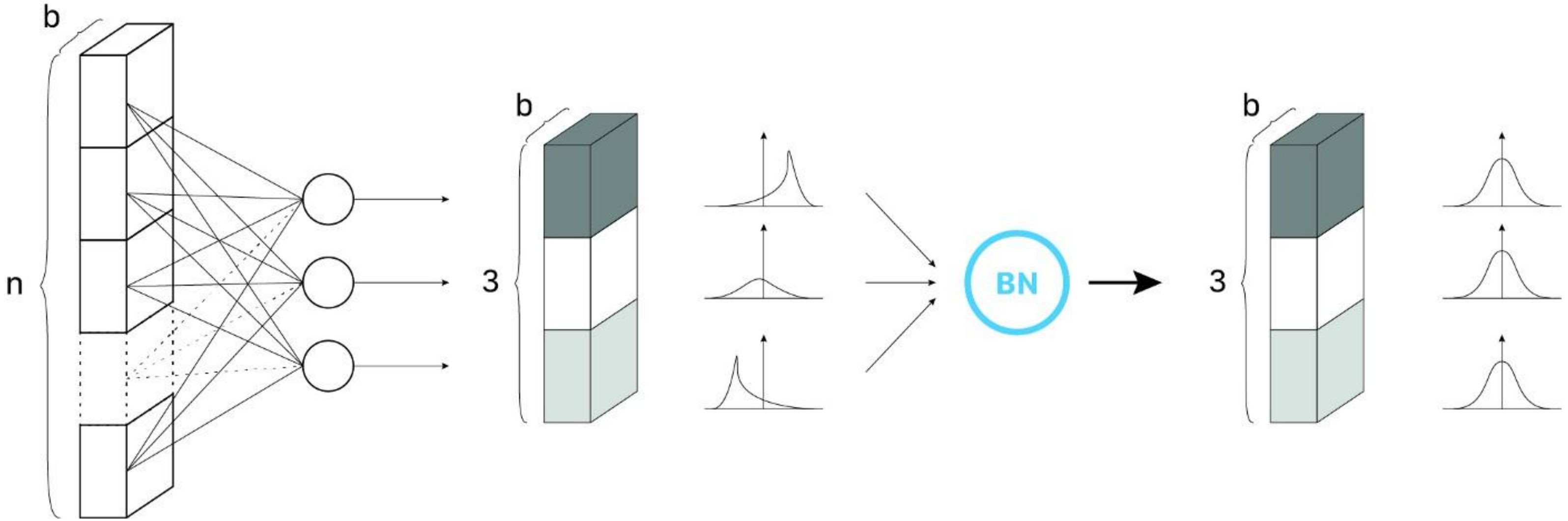

100 z
Project and reshape
1024
4
4
CONV 1
512
8
8
5
5
Stride 2
CONV 2
256
16
16
Stride 2
5
5
CONV 3
128
32
32
5
5
Stride 2
CONV 4
3
64
64
5
5
Stride 2
G(z)

Model Architecture

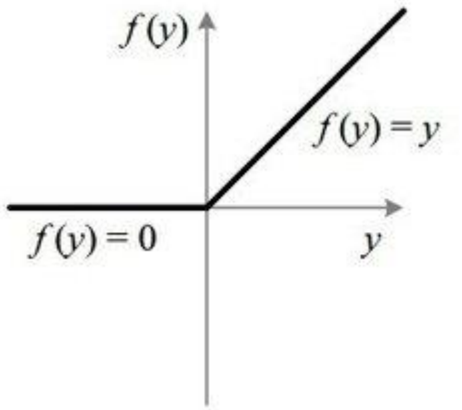
- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).

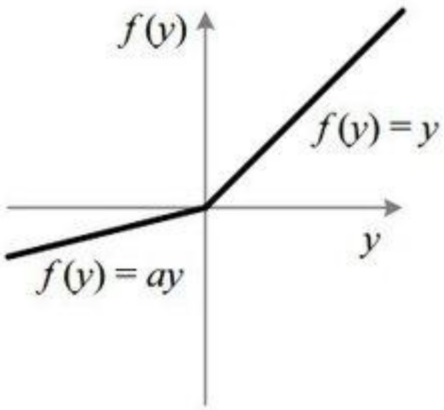


- Use batch-normalisation in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.

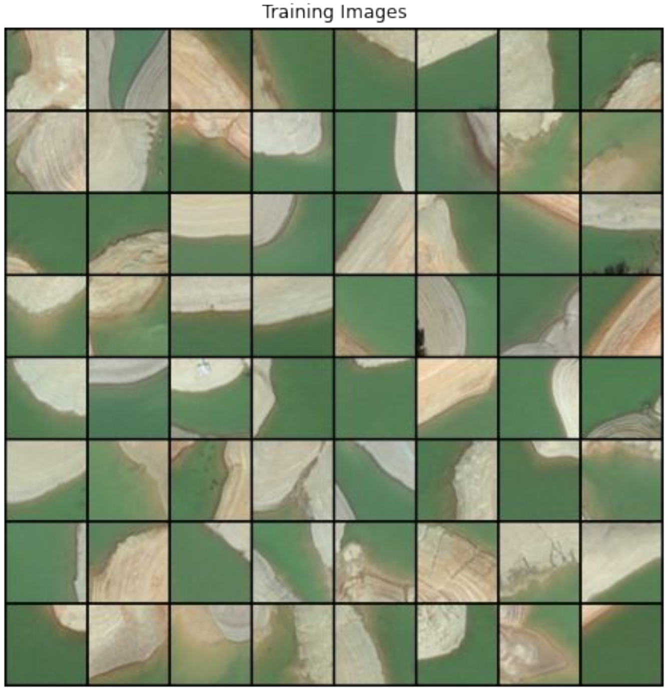


- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.





Training Dataset



RSI-CB 128

Loss Function

- Binary Cross-Entropy Loss function which can change according to the ground truth label

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

Optimiser

- A standard Adams Optimiser for a deep network

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla w_t$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * (\nabla w_t)^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$$

Hyperparameter Tuning

- According to the Goodfellow’s paper on GANs and the DCGAN paper :

1. Learning rate: 0.0002
2. Batch size: 128
3. Beta1=0.5
4. Latent z: Gaussian with mean 0 and std:0.02

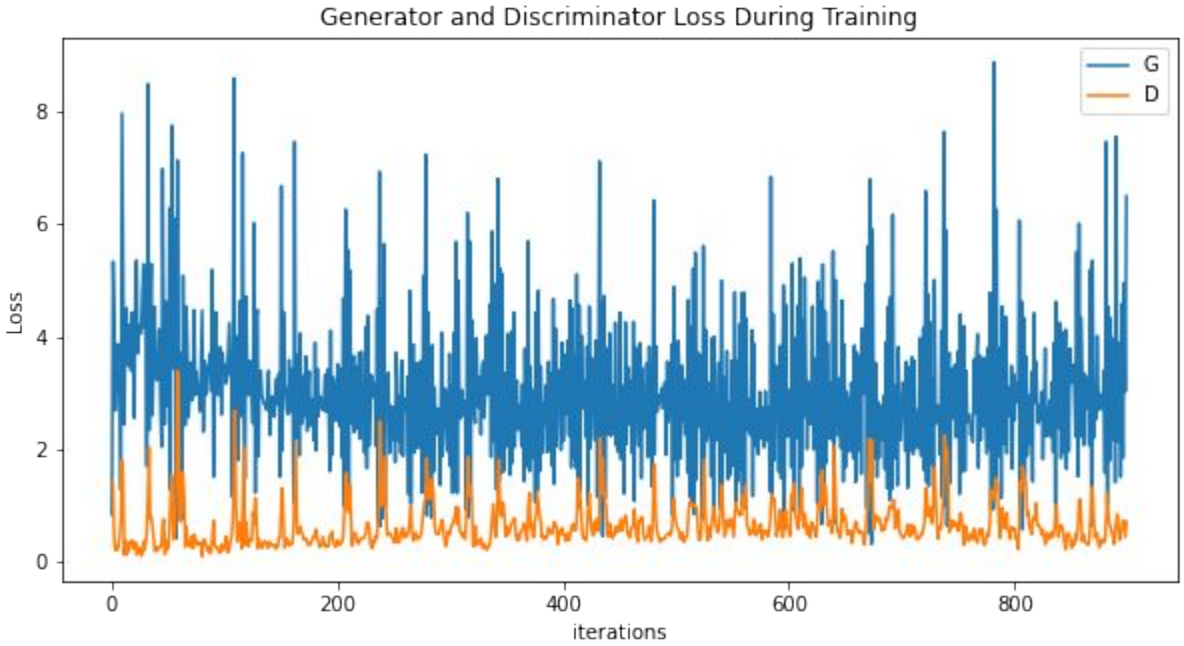
Salient Features

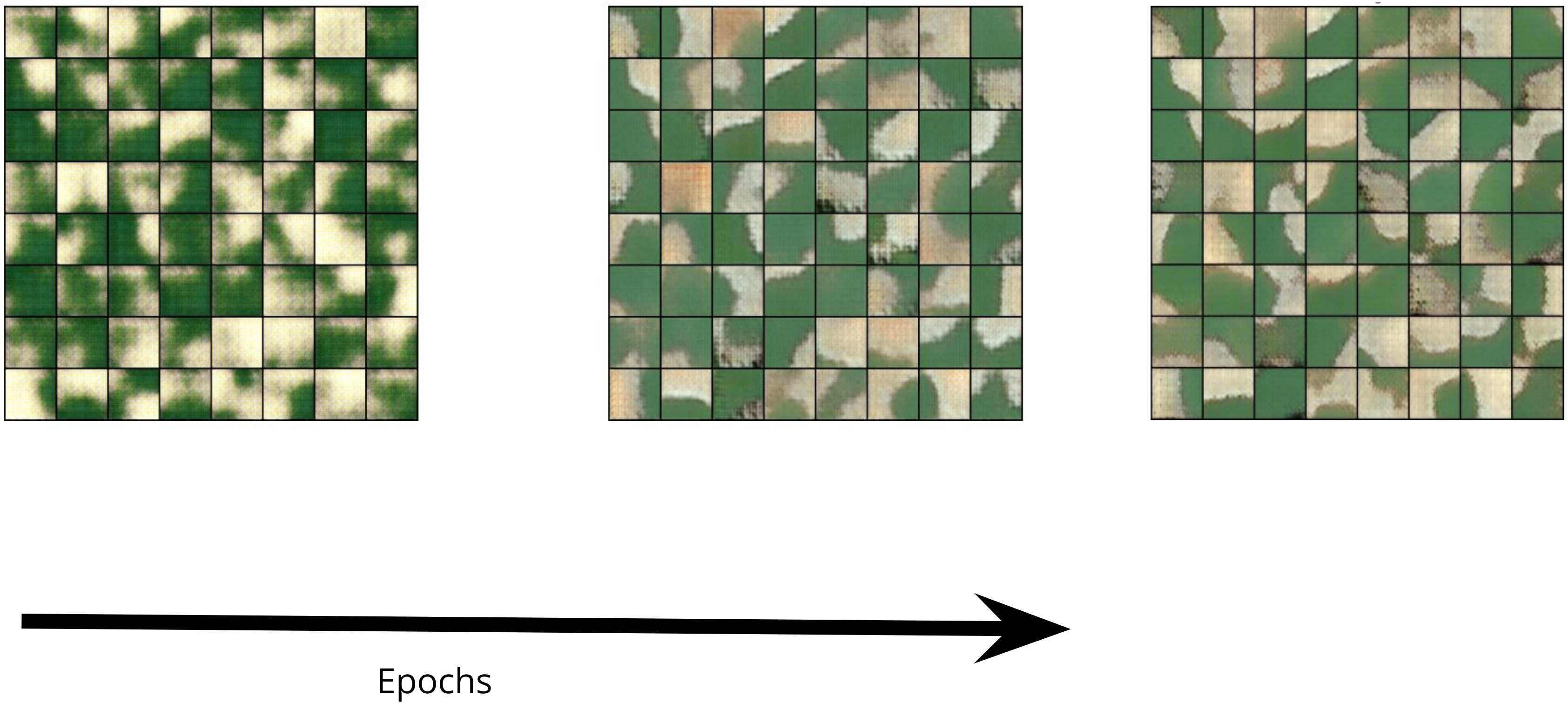
- Pytorch Dataloader
- Weights Initialisation
- 3-Channel Image

Results

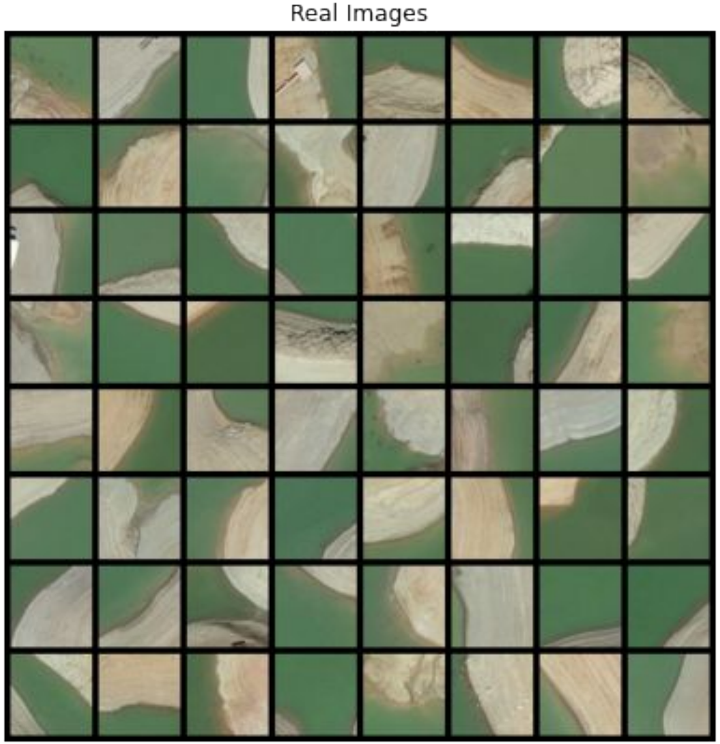
Training Results

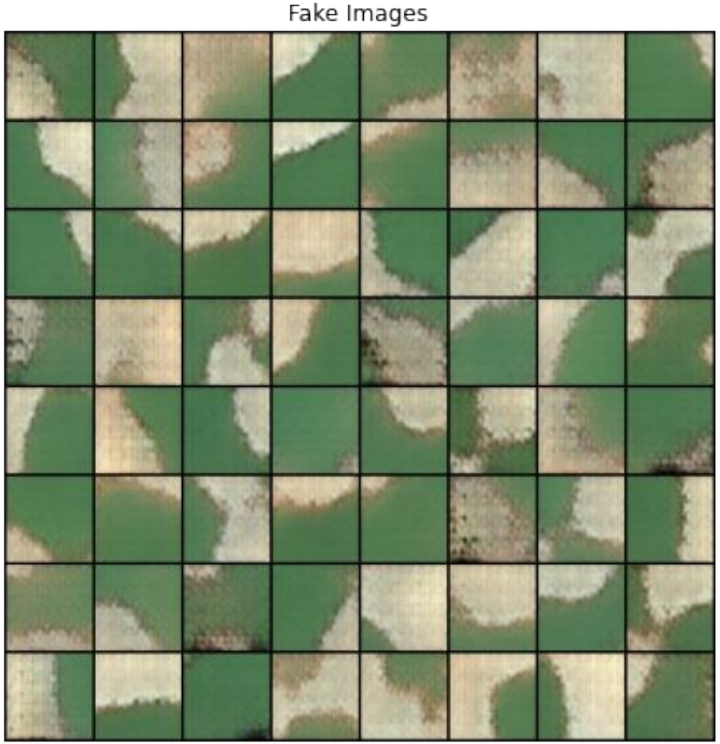
- Network trained on 100 epochs for 1h 44m 50s




Epochs

Final Results!





Expected Results

*Ref:https://github.com/aashishrai3799/Remote-Sensi
ng-Image-Generation*

Improvements

- Longer Training on sufficient GPU

- Trying out multiple classes

- Change the size of Images

- Increase the channels

References

1. Literature:

- *UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS , ICLR 2016 BY ALEC RADFORD & LUKE METZ*
- *GENERATIVE ADVERSARIAL NETS BY IAN GOODFELLOW, BING XU & DAVID-WARDE FERLEY*

2. Code

- https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
- https://github.com/aashishrai3799/Remote-Sensing-Image-Generation
- https://pytorch.org/tutorials/beginner/basics/data_tutorial.html

THANK YOU