GENERATIVE NEURAL NETWORK BASED IMAGE COMPRESSION

Tarun Mohandas - 2018201008
Ankit Pant - 2018201035
K.A.Meghashree - 2018201055

Contents

List of Figures

1 Introduction

Generator Adversarial models (or GANs) are learning models which consist of a combination of the generator model and a Discriminator model working in tandem to achieve better learning. This model makes use of the concepts of Game Theory and the objective (loss) function is generally modelled as a two player zero sum game. The generator model takes in random noise as input as attempts to faithfully recreate the image it is training again, while the discriminator is tasked with identifying which images were created by the generator and which were the original images. Hence, these models have a capacity to learn together and affect each others' learned parameters.
The Standard lossy image compression techniques such as JPEG and WebP are not data specific. They are not designed specifically for the data being compressed. Hence, they do not achieve the best possible compression rates for images. We construct a deep neural network based compression architecture using a generative adversarial model which is pre-trained with several datasets, which consists of similar images (belonging to a particular class). Our architecture compresses related images by firstly using the learnt weights of the GAN model to generate an image based on the latent vectors learnt of the datasets. After the image is generated, another generator is used to fit the generated image to the original image to be compressed. Three data-sets have been used to train the model on. The training-validation split is done in the standard (80-20) ratio
In this project, we have implemented the following:

1. Extracting Latent vectors from Images using Generative adversarial networks
2. Compressing the image using the extracted latent vectors
3. Compare the performance of the model on several datasets

The various details of the project such as architecture, functionality and performance are described in the following sections.

2 Literature Survey

2.1 Image compression using Neural Networks

2.1.1 Architecture Overview

A deep neural network is generally used to compression architectures. A generative model is used [1] to achieve image compression. The dataset used to train such networks generally contain semantically related images so that particular features specific to particular datasets may be learnt better. An implementation suggested consists of reversing the generator of a GAN to compress images [1], omitting the encoder altogether.

2.1.2 GANs

A GAN consists of two networks called the generator and the discriminator. The training phase can be summarised as follows

- The network tries to minimize an adversarial loss function
- Generator tries to create images that cannot be differentiated from true images
- Discriminator tries to correctly classify images as real or generated

- Discriminator is constructed just for the training purposes and is discarded after training
- The remaining generator network maps from a low dimensional latent space to a higher dimensional image space
- Perceptual similarity metrics (Structural Similarity Index (SSIM)) used for training

The Metric Equation for Structural Similarity Index (SSIM) used to compare two images is given by:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

A suggested Loss Function to be minimised is given by:

$$\frac{1}{N}\sum_{x,y} 1 - SSIM(x; y)$$

2.1.3 Image Compression

The Standard Techniques used for image compression are generally lossy (e.g. JPEG) and are rarely data specific. These techniques do not make use of the semantic relations among the images that they attempt to compress. Hence they is quite some scope to improve on these standard techniques.
Compression using deep neural networks can be achieved by training a vector in the latent space. In general the generated images can be further compressed using bzip2 (standard lossless compression scheme) and the decompression can be performed with a forward propagation of the latent vector through the GAN generator [1]

2.1.4 Generative adversarial networks for extreme learned image compression

An implementation [2] proposes a framework for extreme learned image compression based on Generative Adversarial Networks (GANs). This technique [2] is claimed to obtain visually pleasing images at significantly lower bitrates. For this approach, two modes of operation are considered:

- Generative compression (GC), preserving the overall image content while generating structure of different scales
- Selective generative compression (SC), completely generating parts of the image from a semantic label map while preserving user-defined regions with a high degree of detail

2.1.5 Losses for Generative Adversarial Learning

There exist various mathematical issues regarding the way the gradients for the generative model are computed [3]. One of the most notable is to determine the way to take into account how the discriminator itself depends on the generator parameters. Certain mathematically sound methodology has been proposed [4] that transform any generative loss expressed as some divergence between the generative and the sample distributions, and involving a discriminator D, such that the loss gradient properly accounts for Ds dependency w.r.t. θ

3 Methodology

3.1 Architecture

The architecture consists of the discriminator block and the adversarial block and can be visualised with the following diagram:

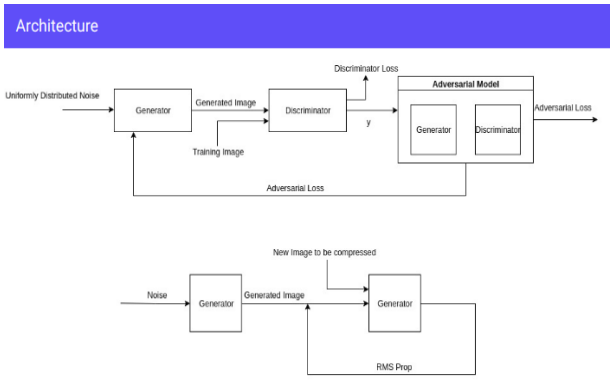

Figure 3-1: The architecture of the GAN model

3.1.1 Discriminator Block

The discriminator consists of the following layers:

1. Convolutional Layer 1
2. Maxpooling layer 1
3. Convolutional Layer 2
4. Maxpooling layer 2
5. Dense Layer
6. Output

It can be diagrammatically visualised as

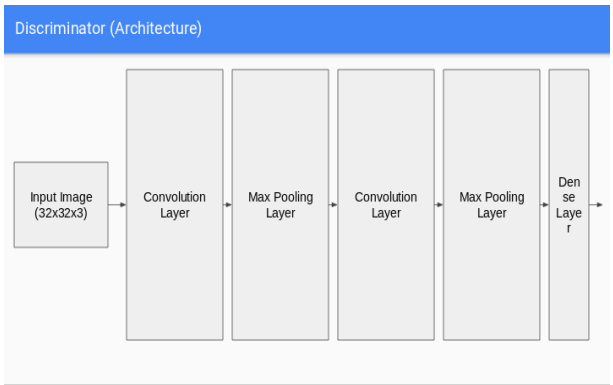

Figure 3-2: Discriminator architecture

3.1.2 Adversarial Block

The adversarial block consists of the following sub-blocks:

- **Generator:** The generator consists of the following layers:
  1. Dense Layer with LeakyReLU activation function
  2. Upsampling Layer 1
  3. Deconvolutional Layer 1 with LeakyReLU activation function
  4. Upsampling Layer 2
  5. Deconvolutional Layer 2 with LeakyReLU activation function
  6. Upsampling Layer 3
  7. Deconvolutional Layer 3 with LeakyReLU activation function
  8. Upsampling Layer 4
  9. Deconvolutional Layer 4 with LeakyReLU activation function
  10. Upsampling Layer 5
  11. Deconvolutional Layer 5

  It can be diagrammatically visualised as

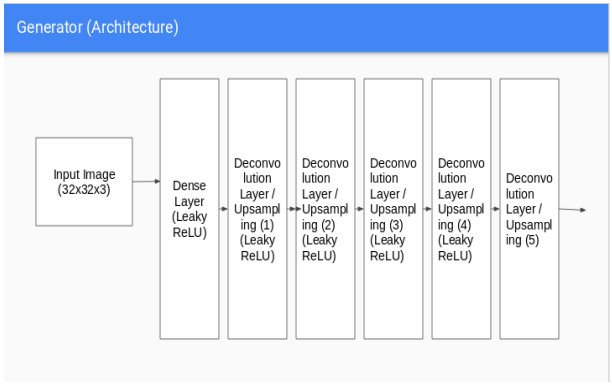

Figure 3-3: Generator architecture

- **Discriminator:** The discriminator consists of the layers as described previously in the *Discriminator Block*

3.2 Algorithm

The algorithm that was used to compress images using the GAN and can be summarised as:

1. The dataset is split into the training and validation sub-datasets. Since the input data was in the form of images, they were preprocessed and converted to numpy array following which they were stored as pickle files for easier access in the future
2. The GAN was trained for a particular data-set's training data. The training was typically done for 30000 epochs with a batch size of 256 images. The learnt parameters of both the discriminator and the generator were then saved for future use

Dataset	L1_error	MSE	PSNR	MS-SSIM
Airplane	84741.75	4189.843896484375	16.87189130603839	0.9313289045511537
Automobile	206785.75	20089.507373046876	6.993157541042008	0.6641013650823496
Flowers	104047.25	6389.831982421875	13.0600959692392	0.7960429337082997

Table 1: The average errors for various datasets

3. The specific test image is then taken (chosen randomly from the created validation data-set). A new image is also generated from random noise using the latent vectors (learned parameters) which were saved in the previous step

4. A second generator model takes this generated image and minimise the difference between the generated image and the original image so as to reproduce the original image as much as possible

5. Since the generated image was generated through the latent vectors, it reduces the number of epochs required, minimises the difference between the two images and also successfully compresses the image.

6. The output image is then the compressed version of the provided input image with low loss (difference between two images)

4 Results

The GAN model was successfully tested on three datasets. The details of the datasets are:

- **Airplane Datasets:** Images of dimension - 32*32*3
- **Automobile Datasets:** Images of dimension - 32*32*3
- **Flowers Datasets:** Images of dimension - 32*32*3 scaled from original HD images

The test images were provided to the *compress* algorithm and the output images that were compressed were then compared with the original images using various loss functions. The loss functions were L1-error (absolute error), Mean Squared Error (MSE), Peak Signal to Noise Ratio(PSNR) and Multi-scale Structural Similarity Index (MS-SSIM). The following were the average results obtained:

Some of the sample images that were obtained are illustrated in the following figures:

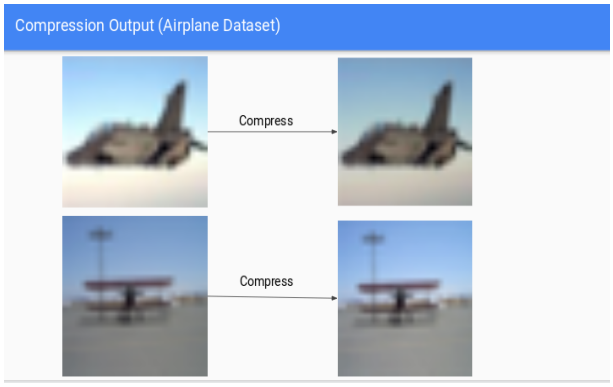

Figure 4-4: Compression of Airplane dataset

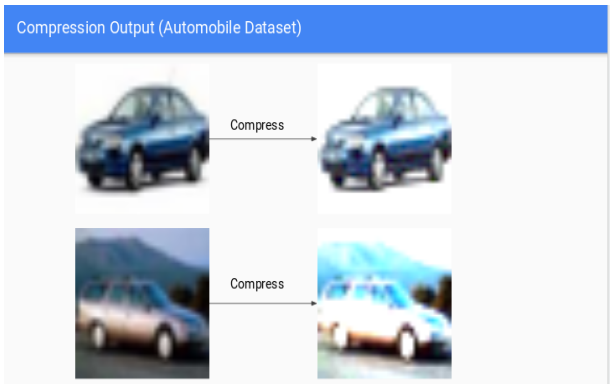

Figure 4-5: Compression of Automobile dataset

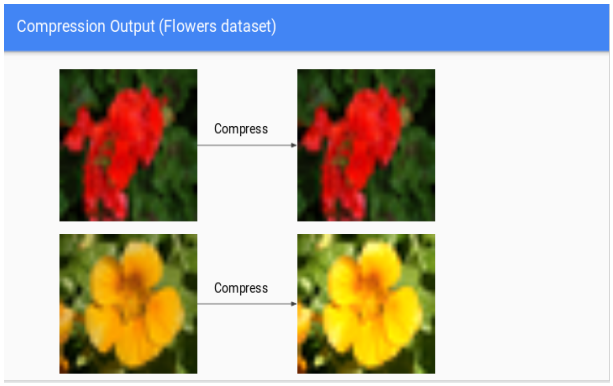

Figure 4-6: Compression of Flower dataset

4.1 Performance

The performance of the model can be visualised with the following graphs:

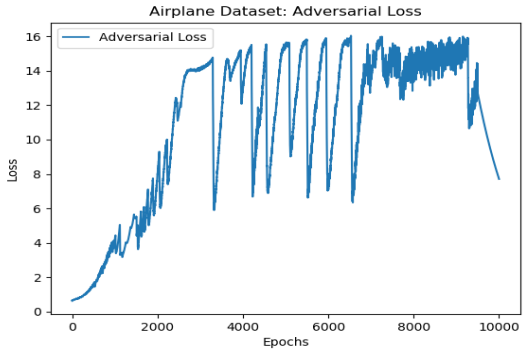

Figure 4-7: Airplane adversarial performance

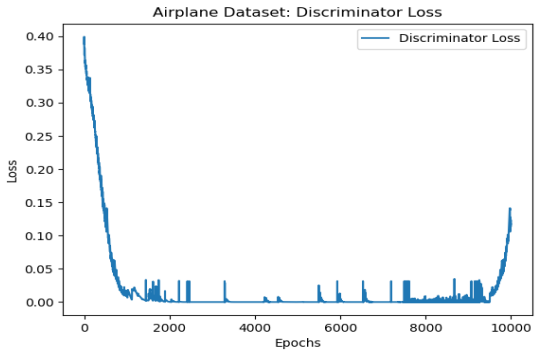

Figure 4-8: Airplane discriminator performance

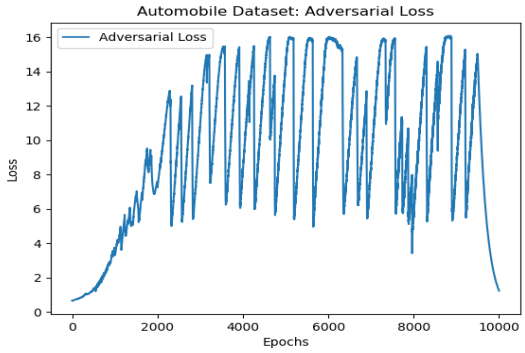

Figure 4-9: Automobile adversarial performance

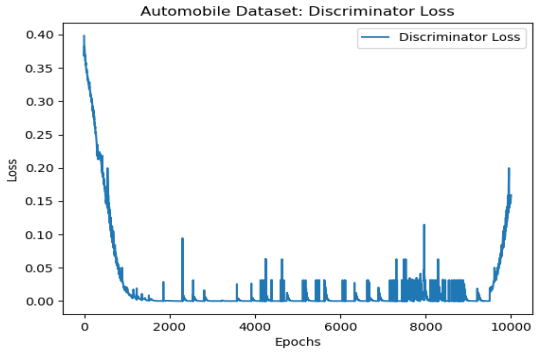

Figure 4-10: Automobile discriminator performance

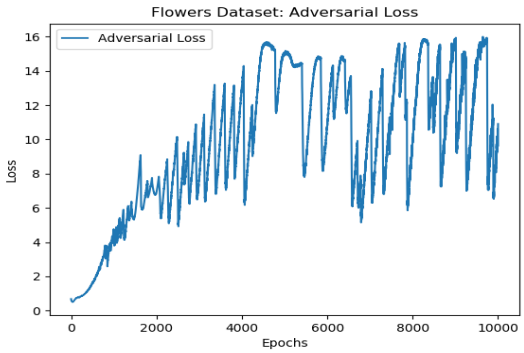

Figure 4-11: Flowers adversarial performance

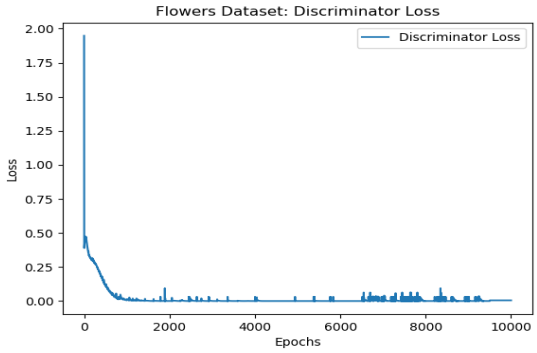

Figure 4-12: Flowers adversarial performance

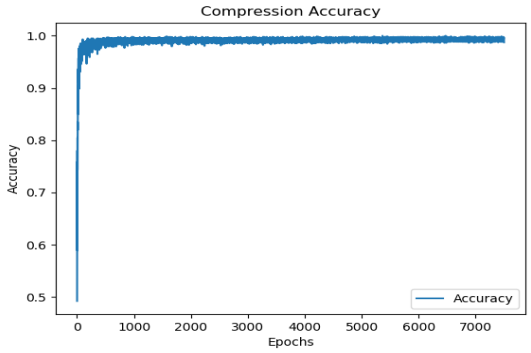

Figure 4-13: Compression Accuracy

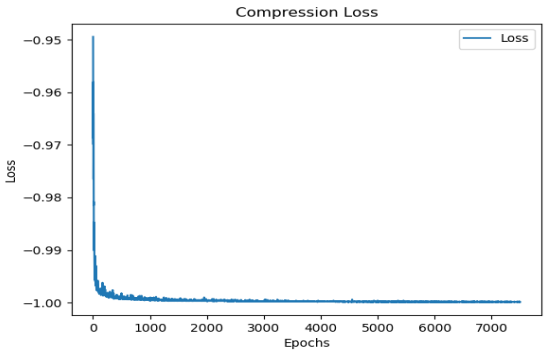

Figure 4-14: Compression Loss

5 Future Scope

The project can be extended to find latent vectors better and thus compress the input images more effectively by training the current model for larger epochs in more powerful production machines. A GUI based application mat also be created to give a user friendly interface to compress desired images. Further the model can be extended by training it on other data-sets.

6 Individual Contributions

- Tarun Mohandas
  - Design
  - Generator model
  - Train models for latent vectors
- Ankit Pant
  - Design

  - Discriminator model
  - Test models to optimize compression

- Megha K A
  - Data Pre-processing and manipulation
  - Loss functions

7 Conclusion

The result of the project is an image compression program that, if given images similar to the data-sets it has been trained for, can compress the image to a tolerable noise(error) level. The model can also be extended to train on other data-sets so that it can learn the latent vectors and later be used to compress images that are quite varied. The result also contain a comparative study of various loss functions used by compression algorithms. The study also includes the performance of the model on multiple datasets.

References

[1] Generative Neural Network Based Image Compression, E. Meltem Tolu- nay, Ahmad Ghalayini,
http://cs229.stanford.edu/proj2018/report/44.pdf

[2] Generative adversarial networks for extreme learned image compression, Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool, 2018,
http://arxiv.org/abs/1804.02958

[3] Deep Generative Models for Distribution-Preserving Lossy Compression, Eirikur Agustsson, Michael Tschannen and Mario Lucic, 2018,
https://arxiv.org/abs/1805.11057

[4] New Losses for Generative Adversarial Learning, Victor Berger and Mich‘ele Sebag, 2018,
https://arxiv.org/abs/1807.01290