Generative Neural Network Based Image Compression

By,
Tarun Mohandas (2018201008)
Ankit Pant (2018201035)
Meghashree K A (2018201055)

Goals

1. Train images of same kind using Generative Adversarial Networks
2. Generate a generalized image from the GAN to be used as base image for compression
3. For any image belonging to that kind, use the base image to train a generator to generate a compressed image of the original image

Literature Survey

- Deep neural network based compression architecture
- Used Generative Model
- Dataset consists of semantically related images
- Architecture compresses related images by reversing the generator of a GAN and omits the encoder altogether

Literature Survey (Contd)

Training Phase:

- 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

Literature Survey (Contd...)

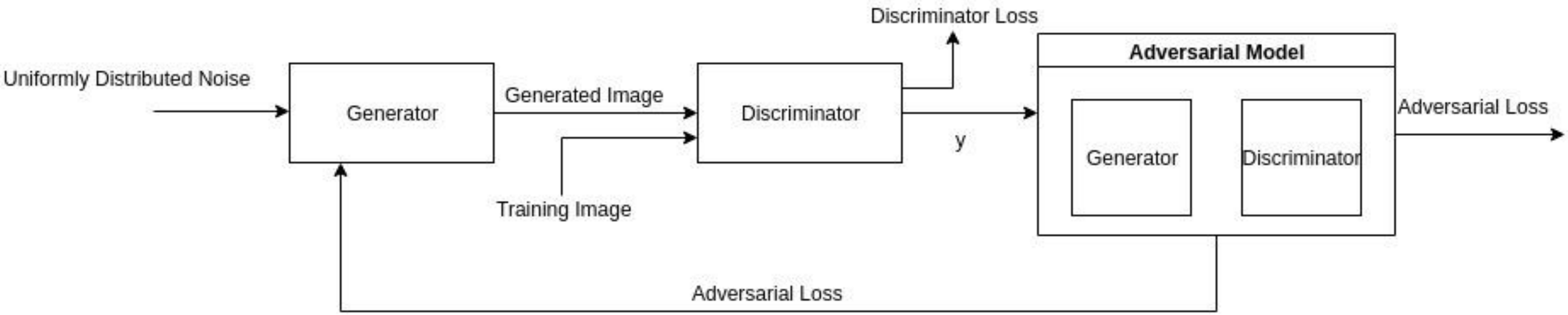
- Standard lossy image compression techniques such as JPEG and WebP are not data-specific
- Standard techniques do not make use of the semantic relations among the images
- Hence standard techniques do not achieve the best possible compression rates
- Compression is done via training a vector in the latent space
- Further compressed with bzip2 (standard lossless compression scheme)
- Decompression of images is simply done with a forward propagation of the latent vector through the GAN generator

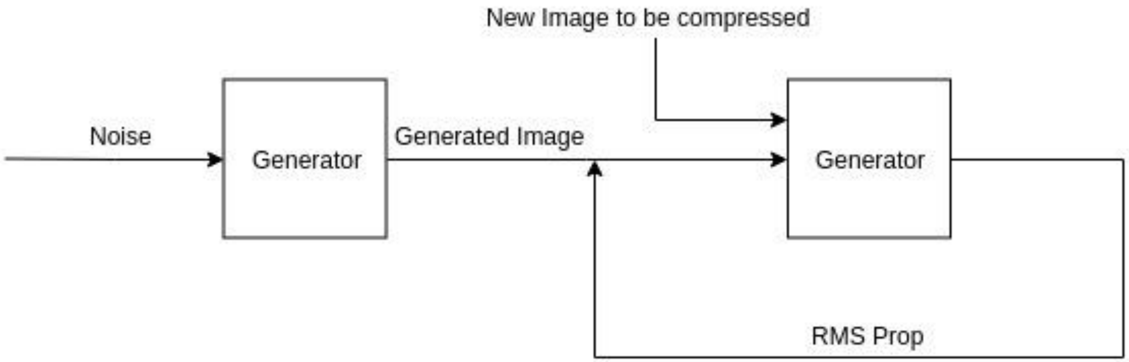
Generative Adversarial Networks (GAN)

GANs - A brief Introduction

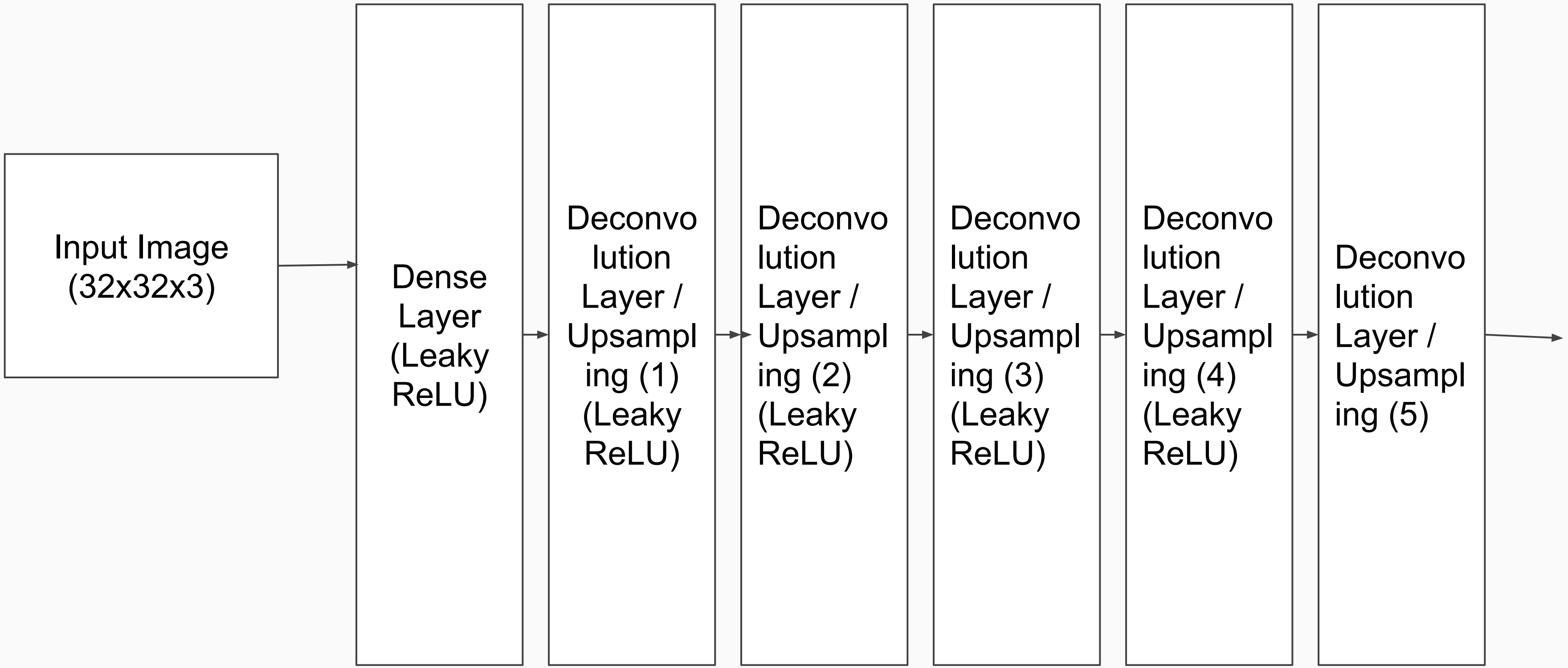
- Generator Adversarial models (or GANs) are learning models which consist of a combination of the generator model and a Discriminator model working in tandem to achieve better learning
- This model makes use of the concepts of Game Theory and the objective (loss) function is generally modelled as a 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.

Architecture





Generator (Architecture)



Input Image (32x32x3)

Dense Layer (Leaky ReLU)

Deconvolution Layer / Upsampling (1) (Leaky ReLU)

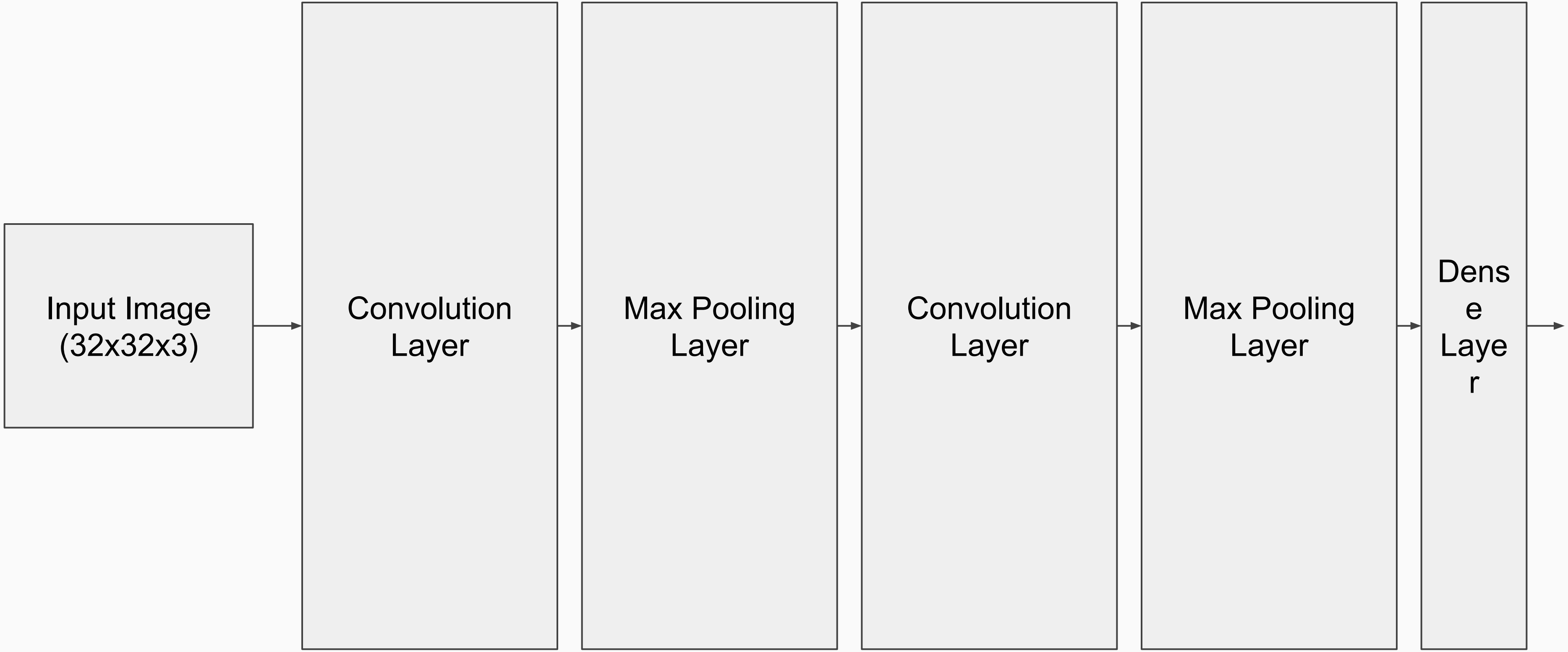
Deconvolution Layer / Upsampling (2) (Leaky ReLU)

Deconvolution Layer / Upsampling (3) (Leaky ReLU)

Deconvolution Layer / Upsampling (4) (Leaky ReLU)

Deconvolution Layer / Upsampling (5)

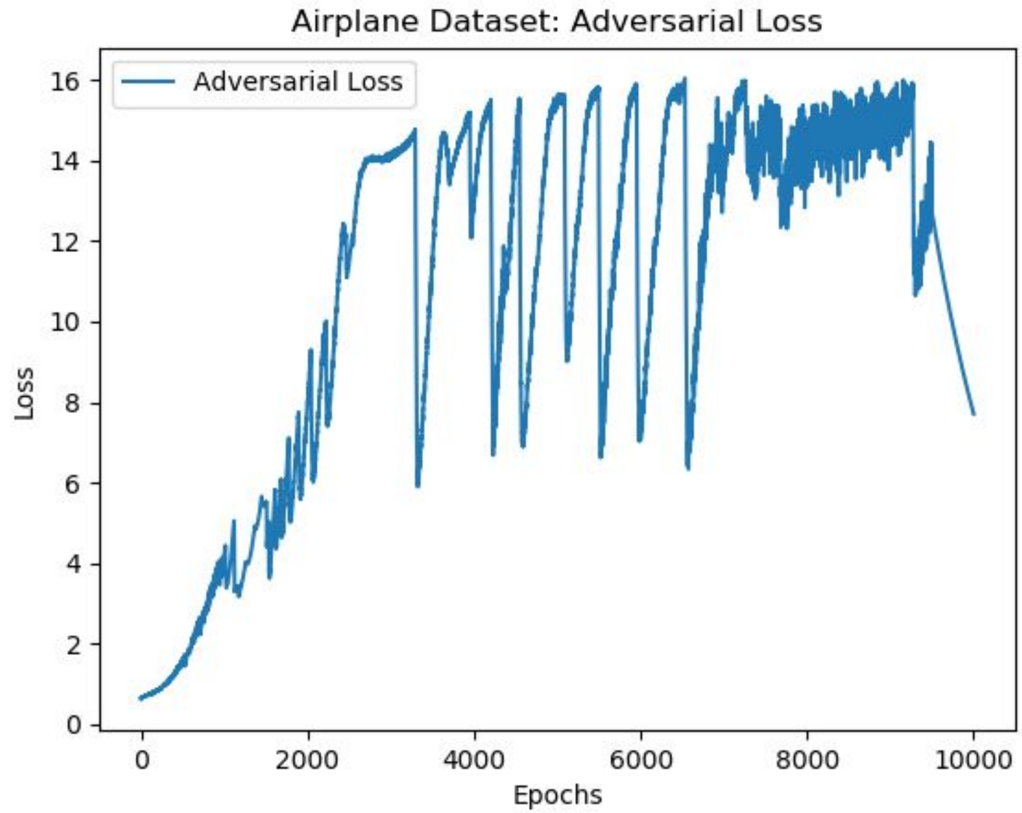
Discriminator (Architecture)



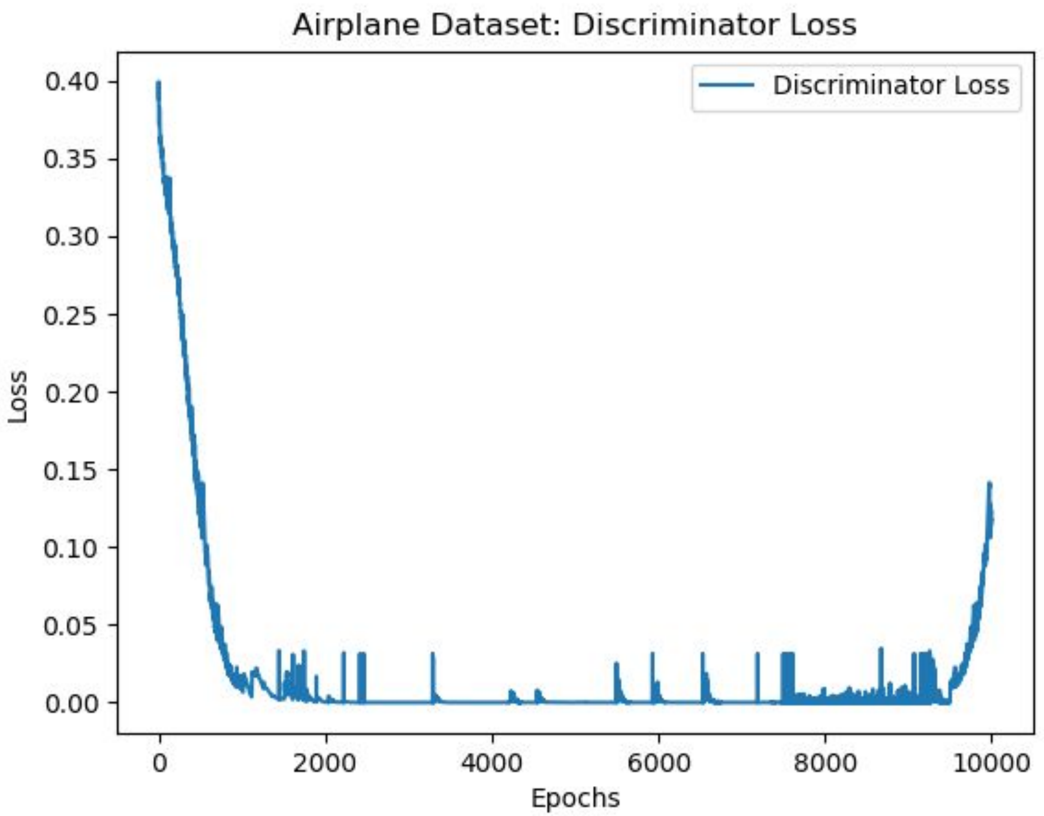
Algorithm

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
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)

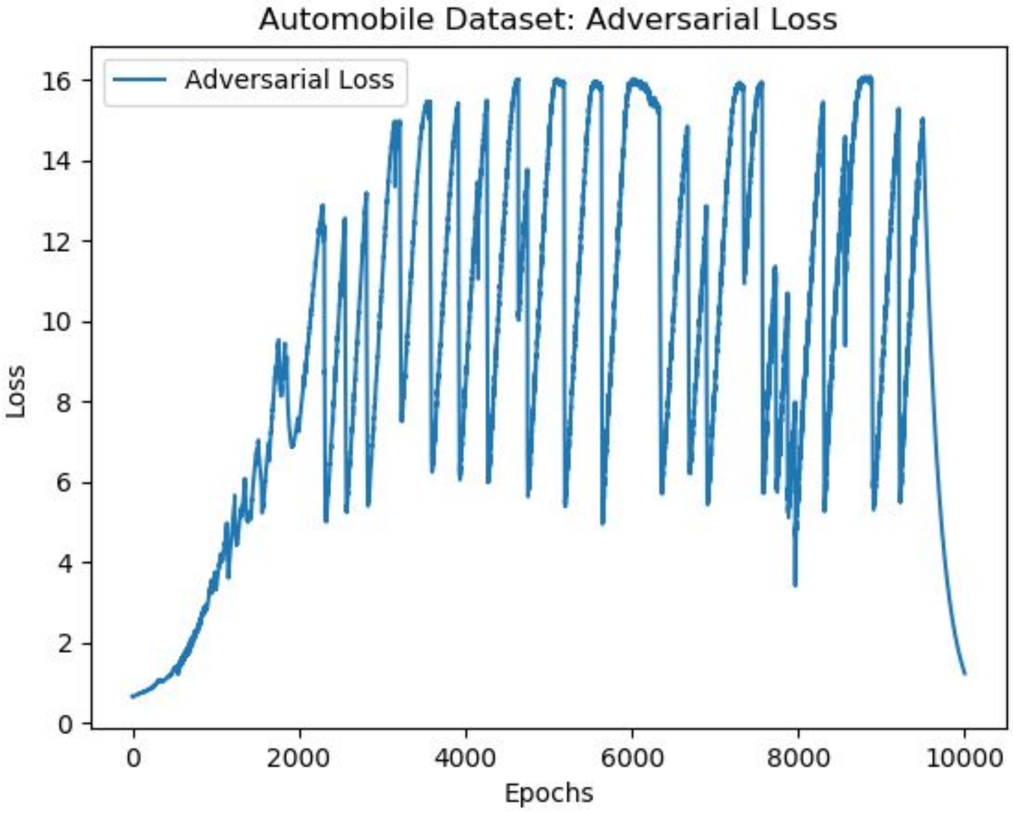
Train Statistics - GAN



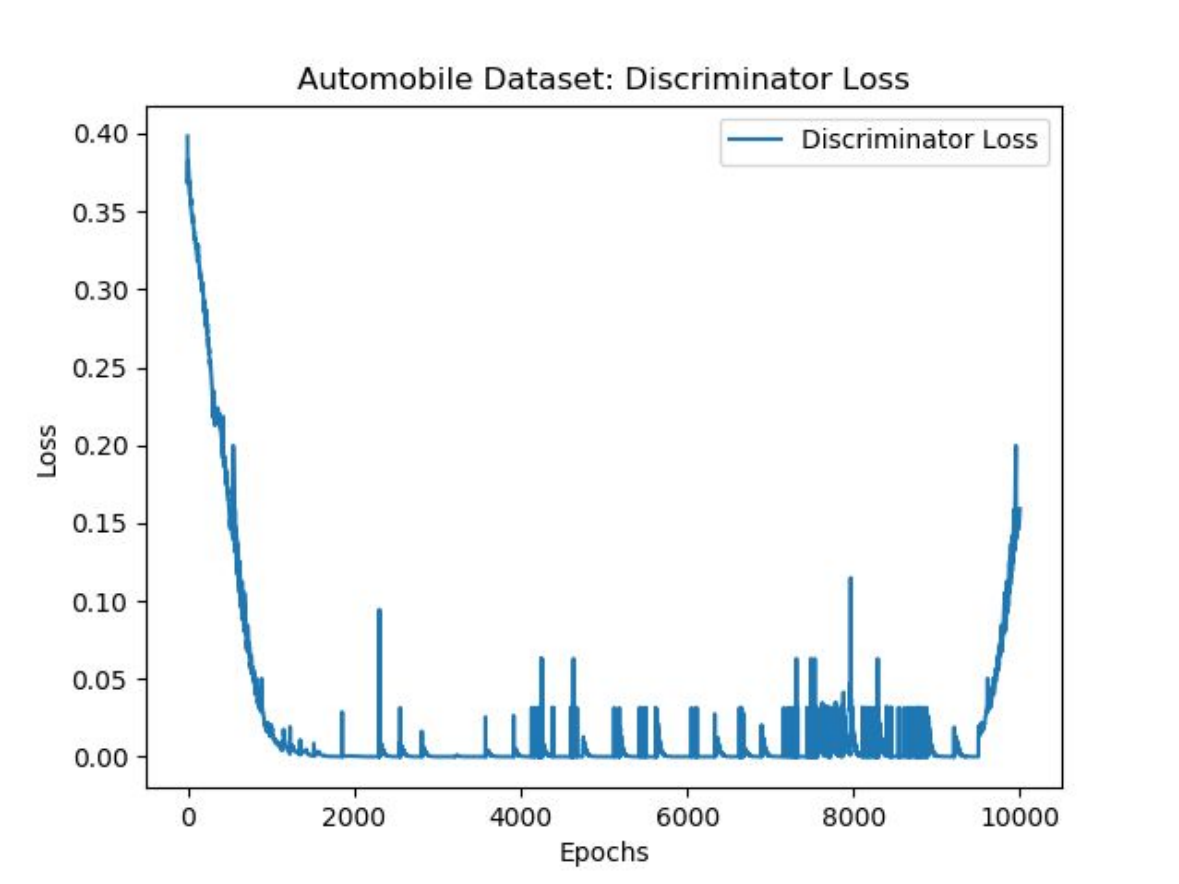
Train Statistics - GAN (contd...)



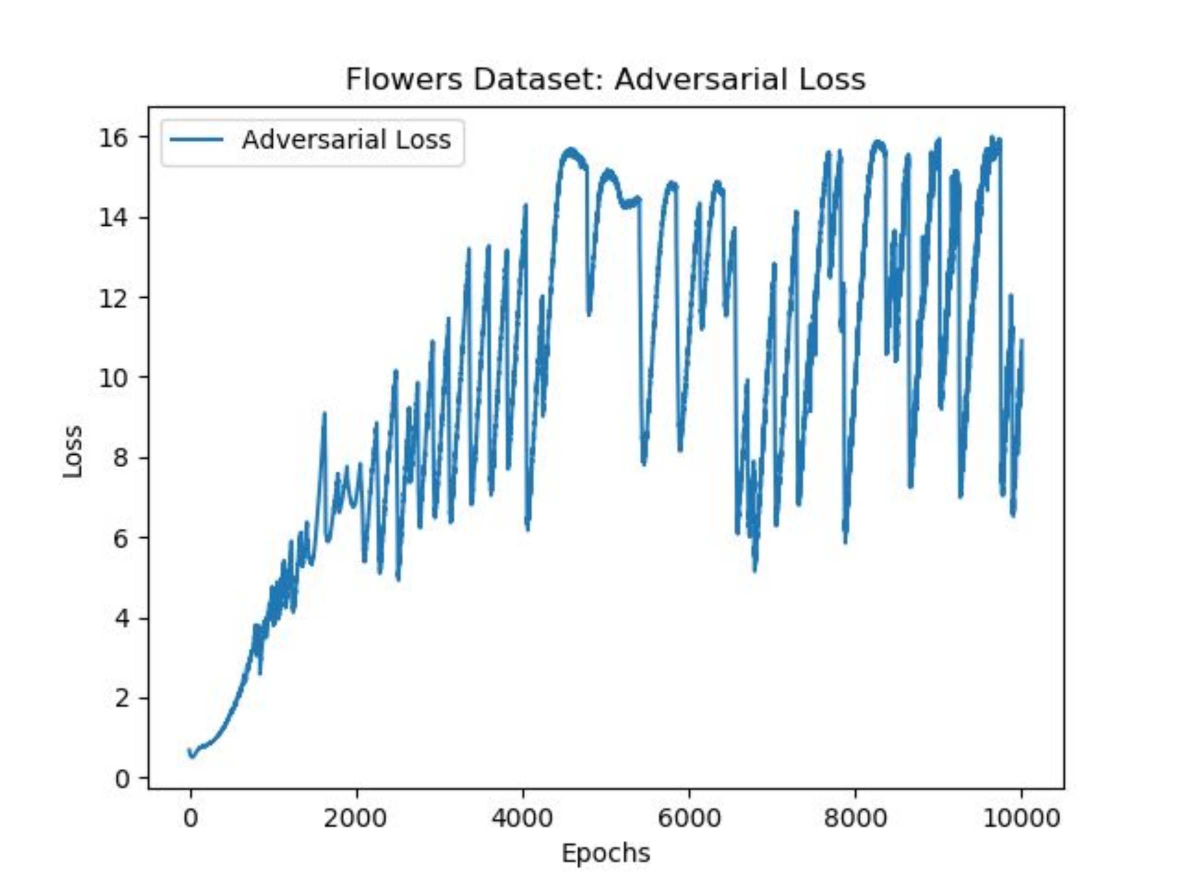
Train Statistics - GAN (contd...)



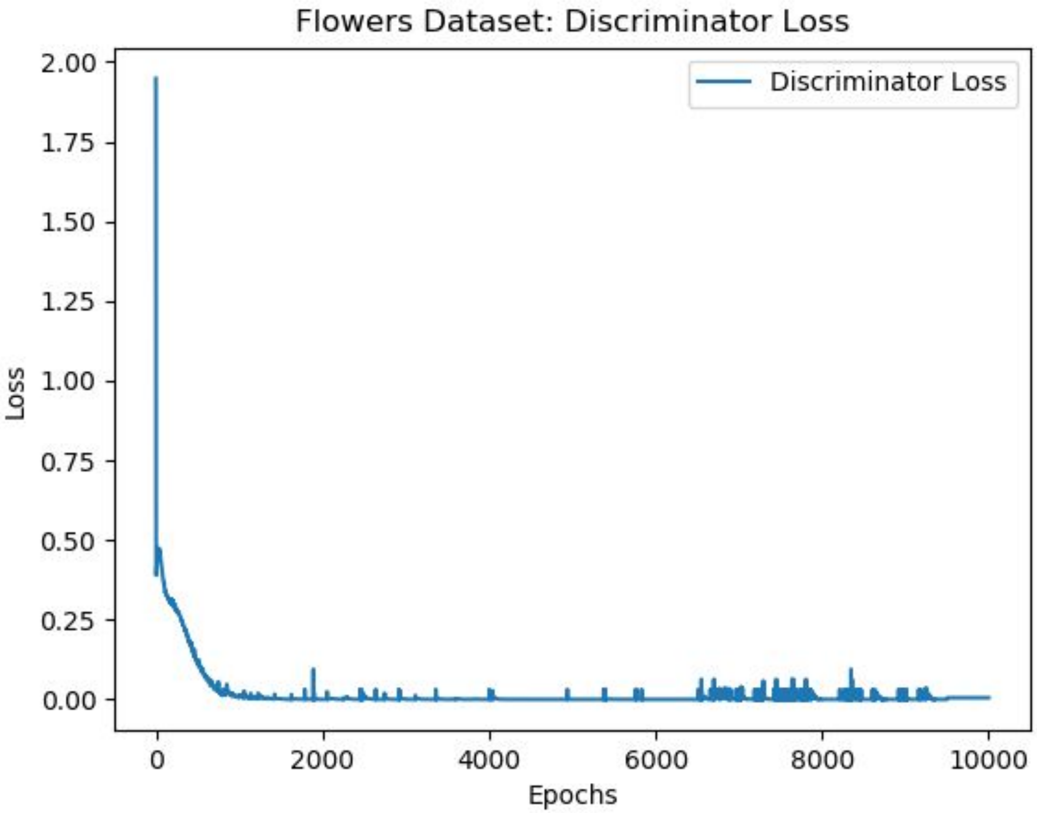
Train Statistics - GAN (contd...)



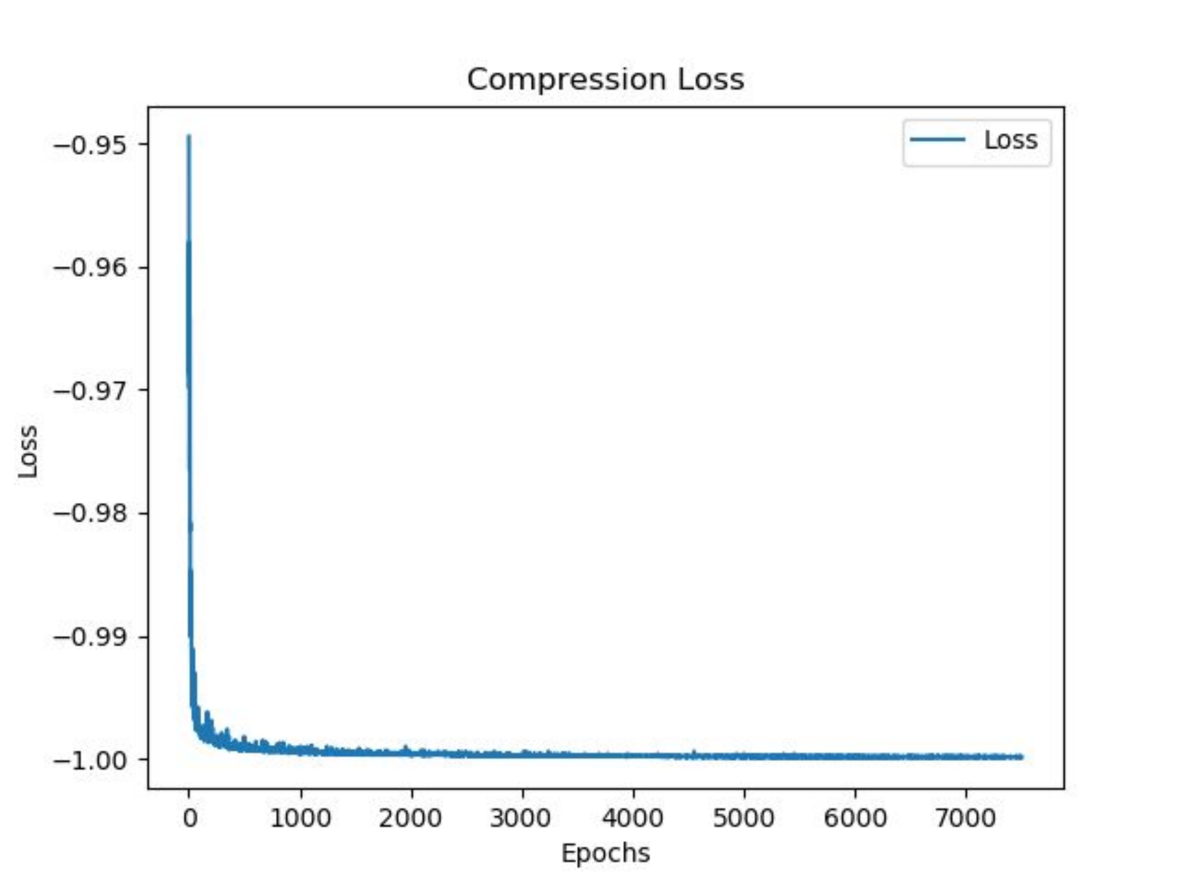
Train Statistics - GAN (contd...)



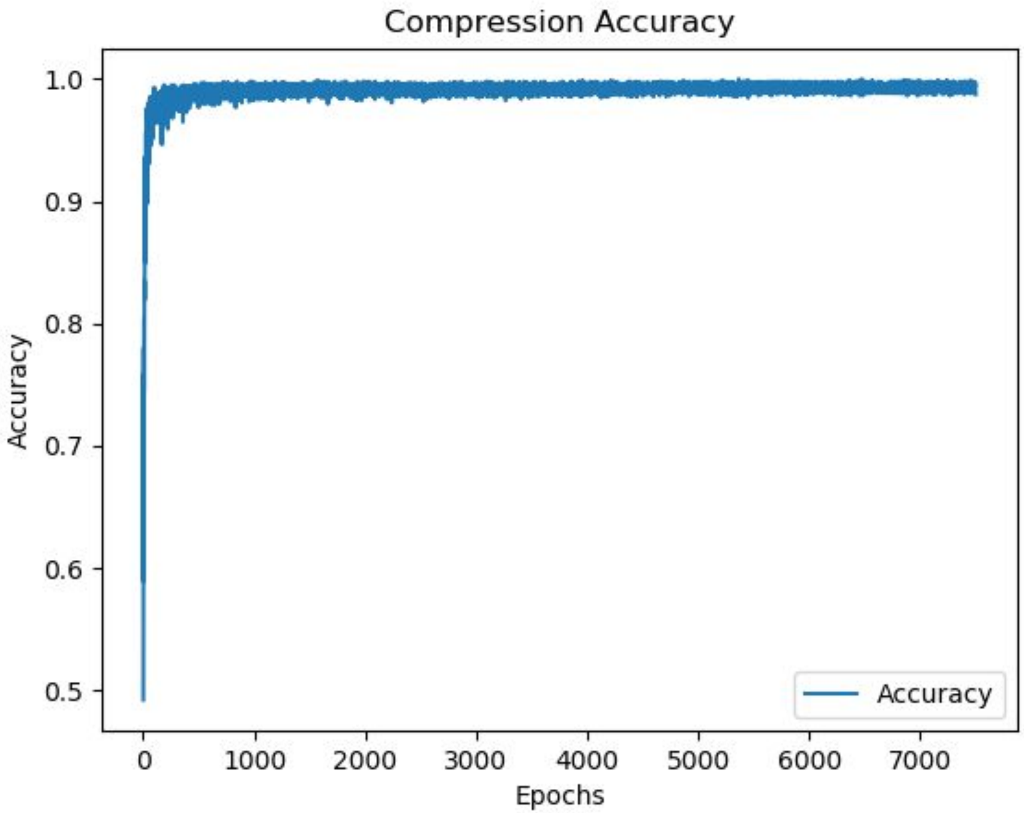
Train Statistics - GAN (contd...)



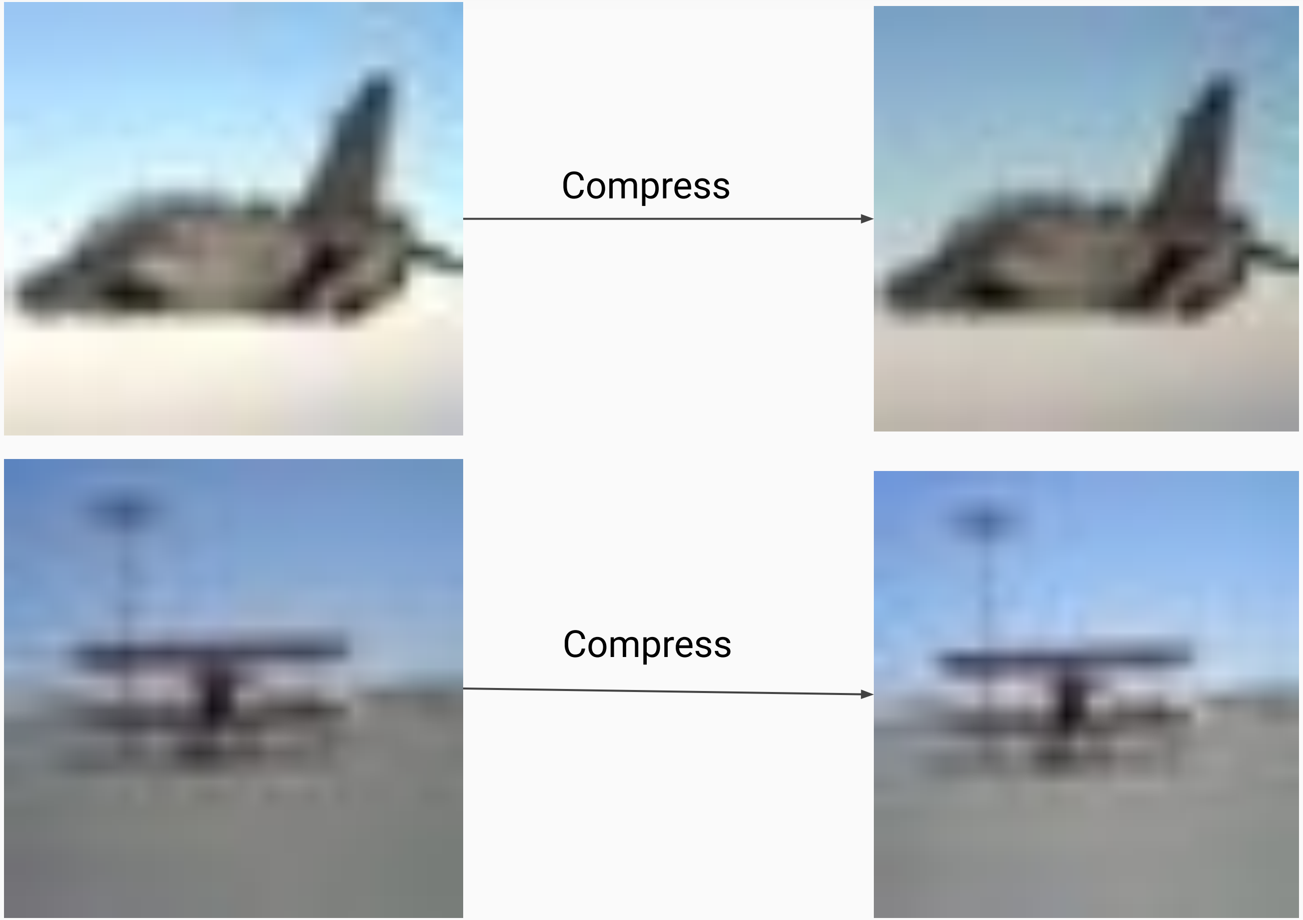
Train Statistics - Compression



Train Statistics - Compression



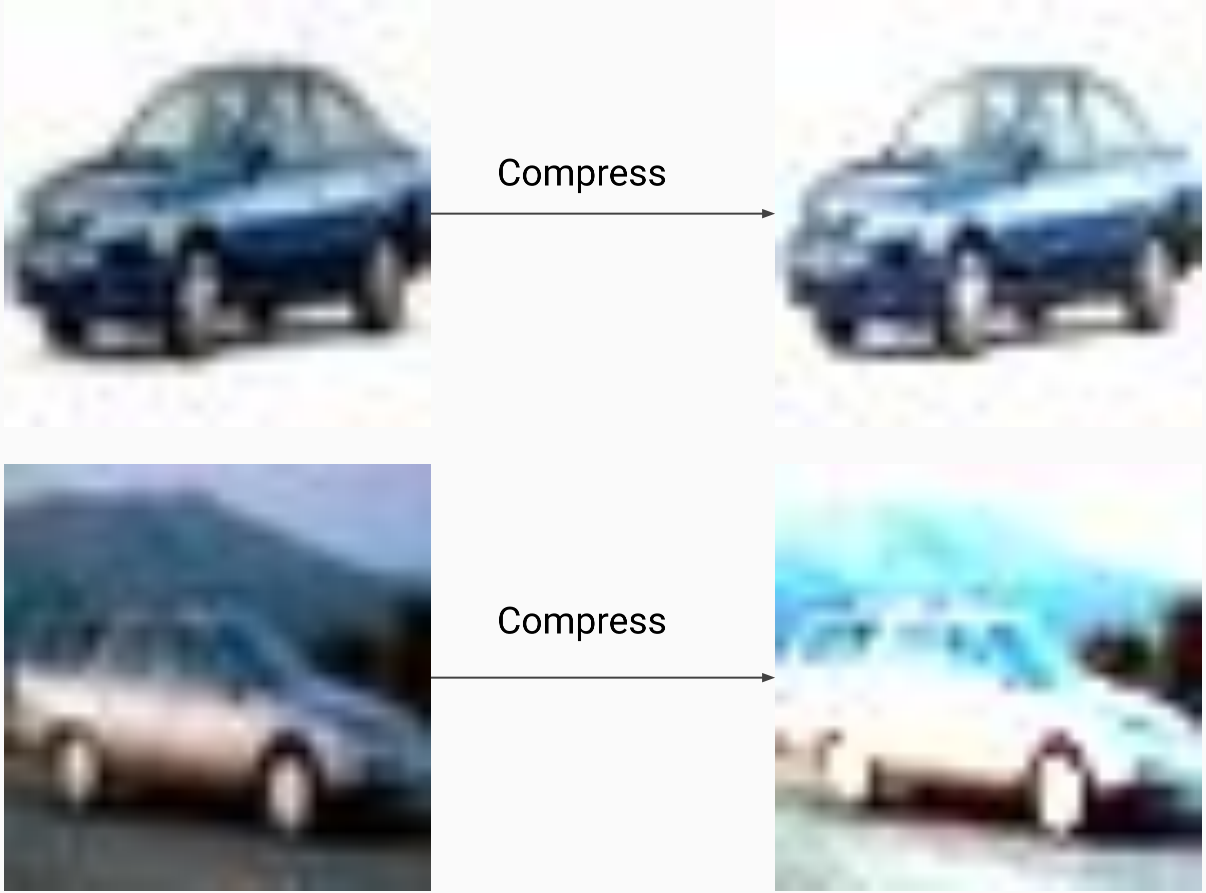
Compression Output (Airplane Dataset)



Compress

Compress

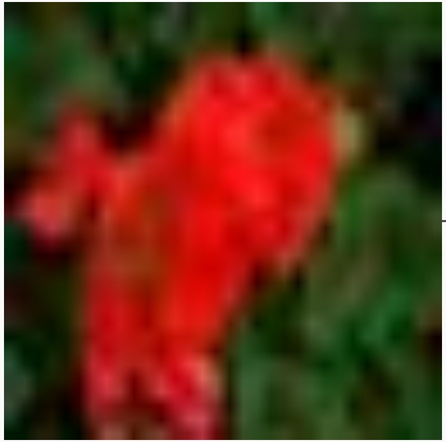
Compression Output (Automobile Dataset)



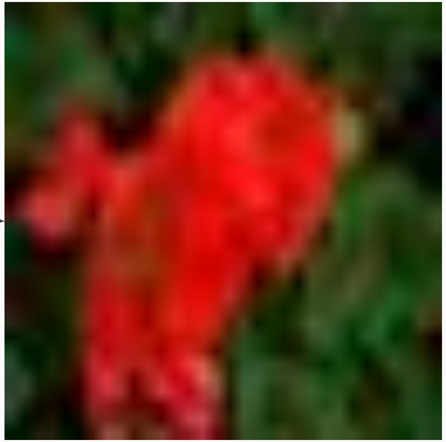
Compress

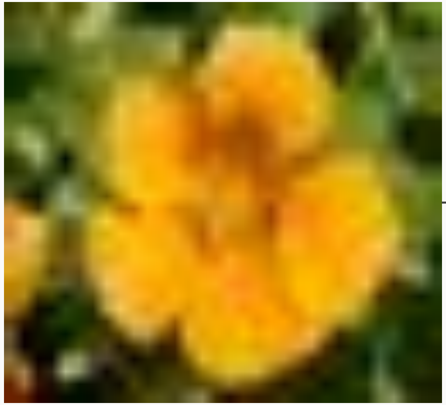
Compress

Compression Output (Flowers dataset)

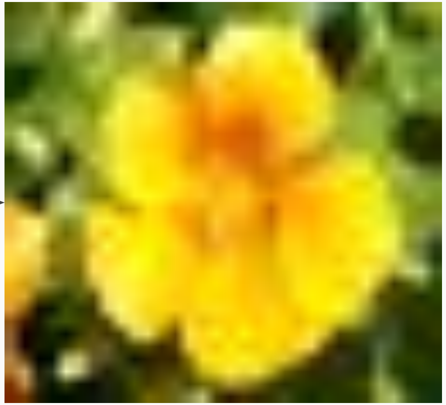


Compress





Compress



Compression Loss Performance

Dataset	MSE	PSNR	MS-SSIM	L1 Error
Airplane	4189.84	16.87	0.93	84741.75
Automobile	20089.50	6.99	0.66	206785.75
Flower	6389.83	13.06	0.79	104047.25

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.

Thank You