

Generative Neural Network based Image Compression

SMAI Team 57 - Project Overview





Introduction and Motivation

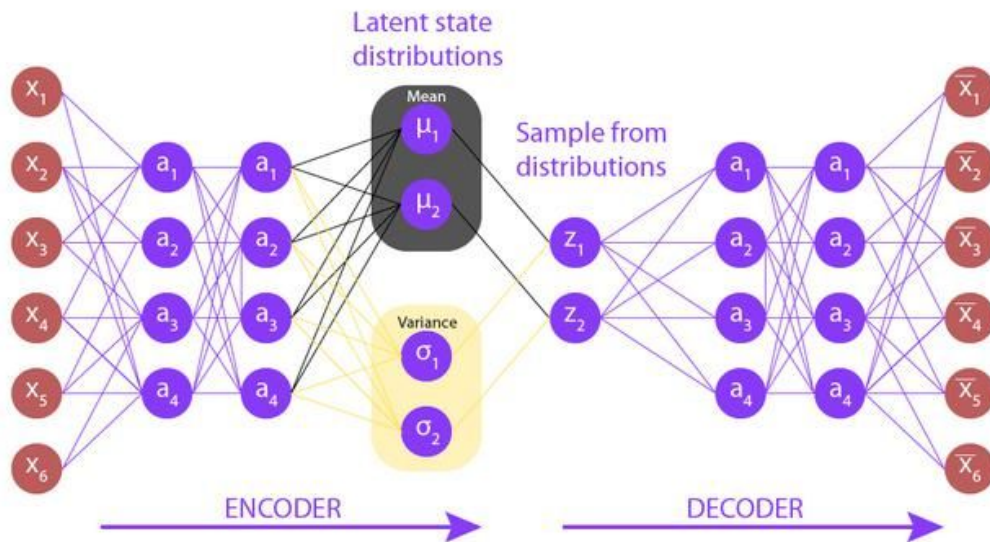
Traditional lossy image compression techniques such as JPEG and WebP are not data-specific, i.e. they are not designed specifically to handle individual datasets, in which the images are semantically related to each other. Hence these techniques do not make use of the semantic relations among the images in a specific dataset, and do not achieve the best possible compression rates. This has led to a growth in the research towards deep neural network based compression architectures. These models tend to achieve orders-of-magnitude better compression rates while still maintaining higher accuracy and fidelity in their reconstructions.



Previous Work(Literature Review)

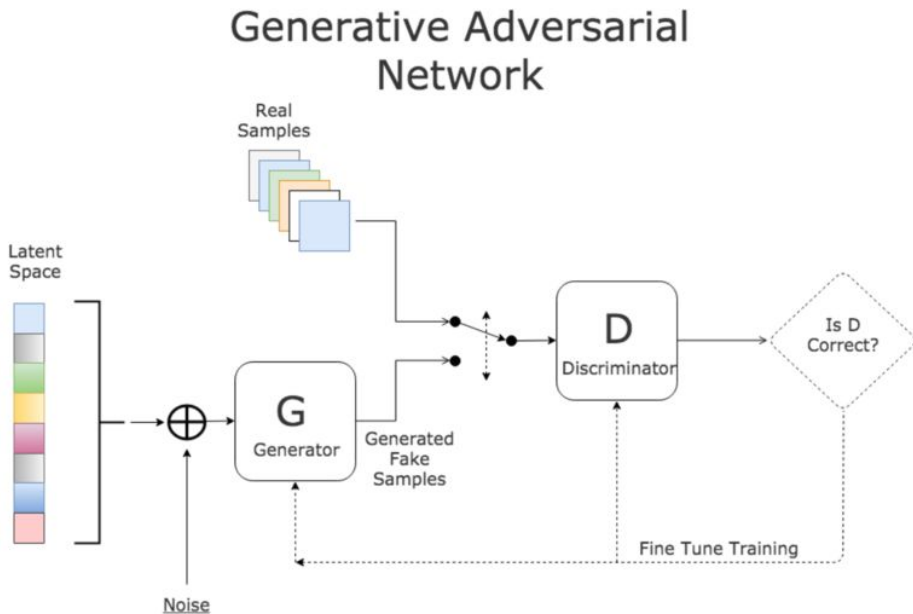


Variational Autoencoders



Given an input dataset \mathbf{x} , characterized by an unknown probability distribution $P(\mathbf{x})$, the objective is to model or approximate the data's true distribution P using a parametrized distribution p_{θ} having parameters θ . Let \mathbf{z} be a random vector jointly-distributed with \mathbf{x} . Conceptually \mathbf{z} will represent a latent encoding of \mathbf{x} and hence the model trains to minimize the error between \mathbf{x} and $D(\mathbf{z})$, where $D(\mathbf{z})$ is the image decoded from \mathbf{z} .

Generative Adversarial Networks

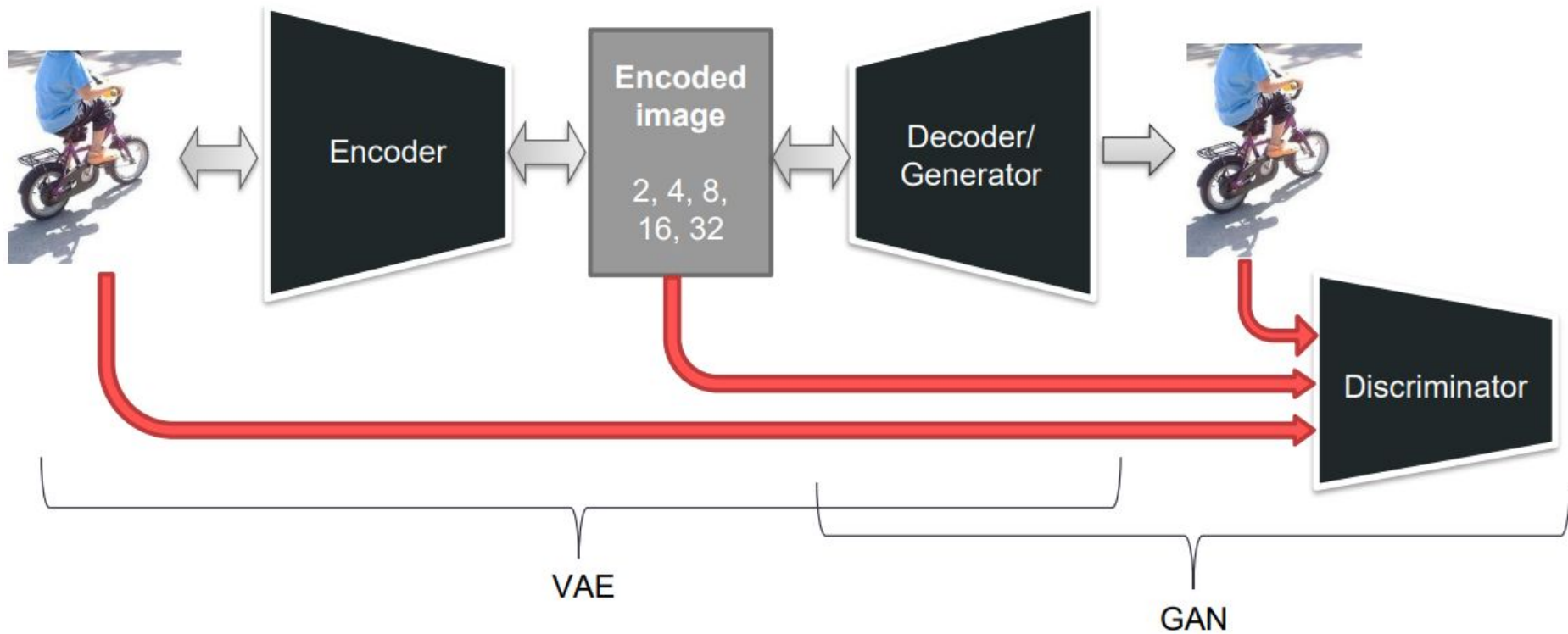


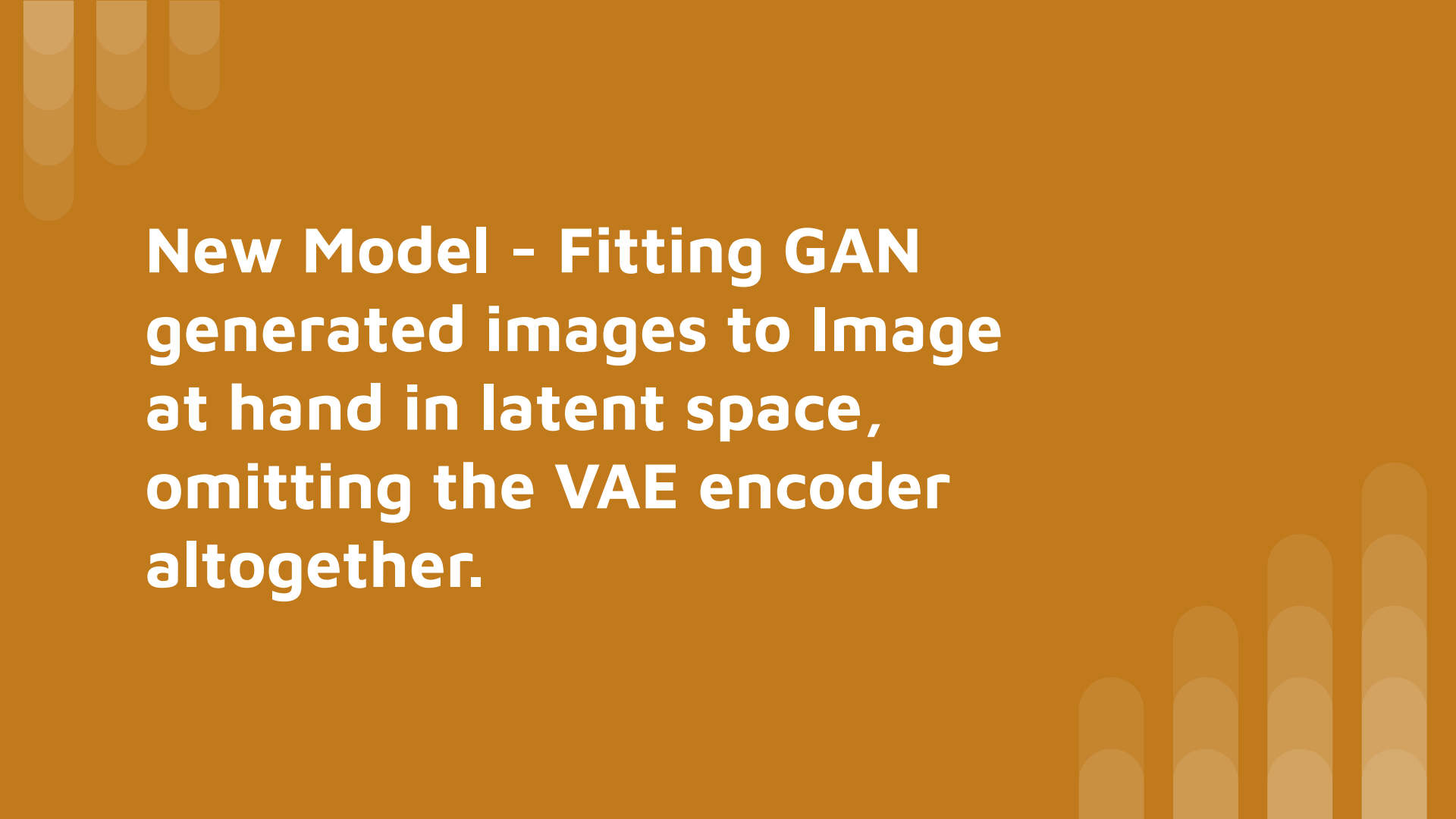
Generative Adversarial models (GANs) are learning models which consist of a combination of the generator model and a Discriminator model working in tandem to achieve better learning. The generator model takes in random noise as input as attempts to faithfully recreate the image it is training again, 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.



VAE Encoder - GAN Decoder

Santurkar et al. [2017] combine Generative Adversarial Networks (GANs) with Variational Autoencoders (VAEs) in their compression scheme, where the decoder of the compressor is the generator part of the trained GAN, which is later combined with the encoder of the VAE. By constructing the compressor with an encoder obtained from a VAE and a decoder from a GAN, authors aim to make use of the strengths of both models. Specifically, **GANs are known to produce high quality, perceptual outputs that are different from the training data**, whereas **VAEs output images that have high fidelity to their originals, though their reconstructions are not necessarily as visually appealing to humans due to the pixel-wise loss they use for training**. In a compression scheme, we need our reconstructions to be both high-quality perceptual images and to be true to their originals. This is why the literature tries to combine these two models.



The background is a solid orange color. In the top-left corner, there are three vertical bars of varying heights, each composed of several overlapping circles. In the bottom-right corner, there are four vertical bars of varying heights, each also composed of several overlapping circles.

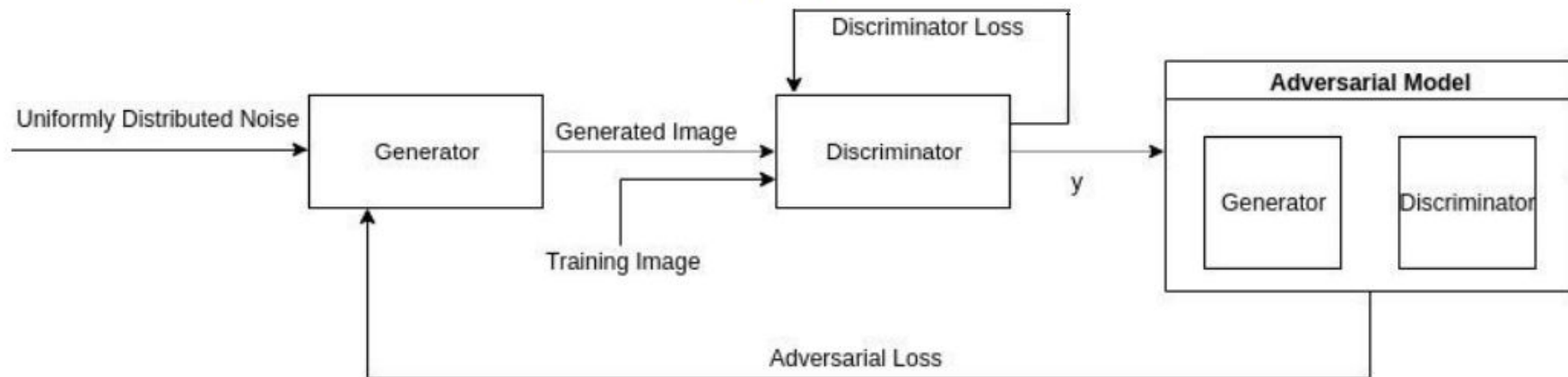
**New Model - Fitting GAN
generated images to Image
at hand in latent space,
omitting the VAE encoder
altogether.**



Training Phase

1. The dataset is split into the training and validation sub-datasets. 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.
2. Discriminator is constructed just for the training purposes and is discarded after training.

Training Step

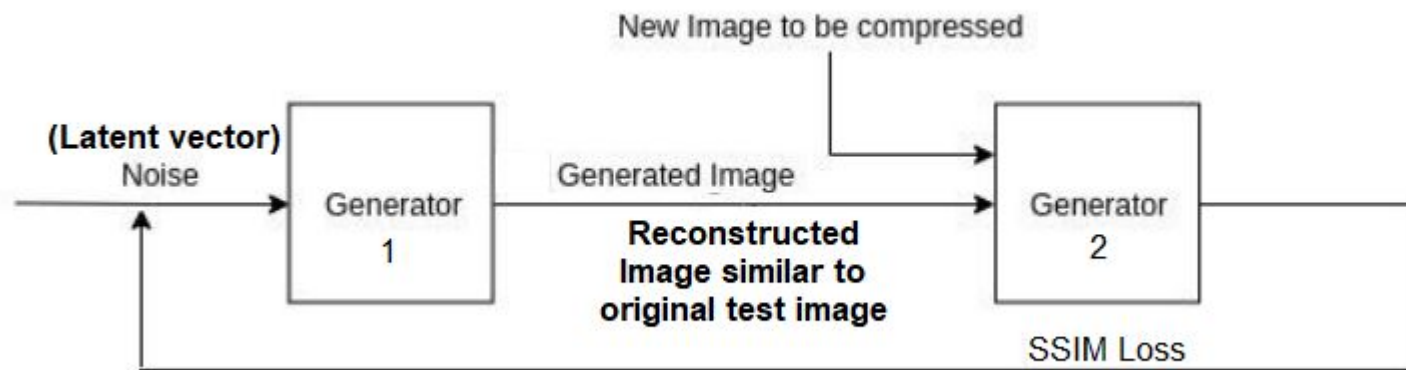


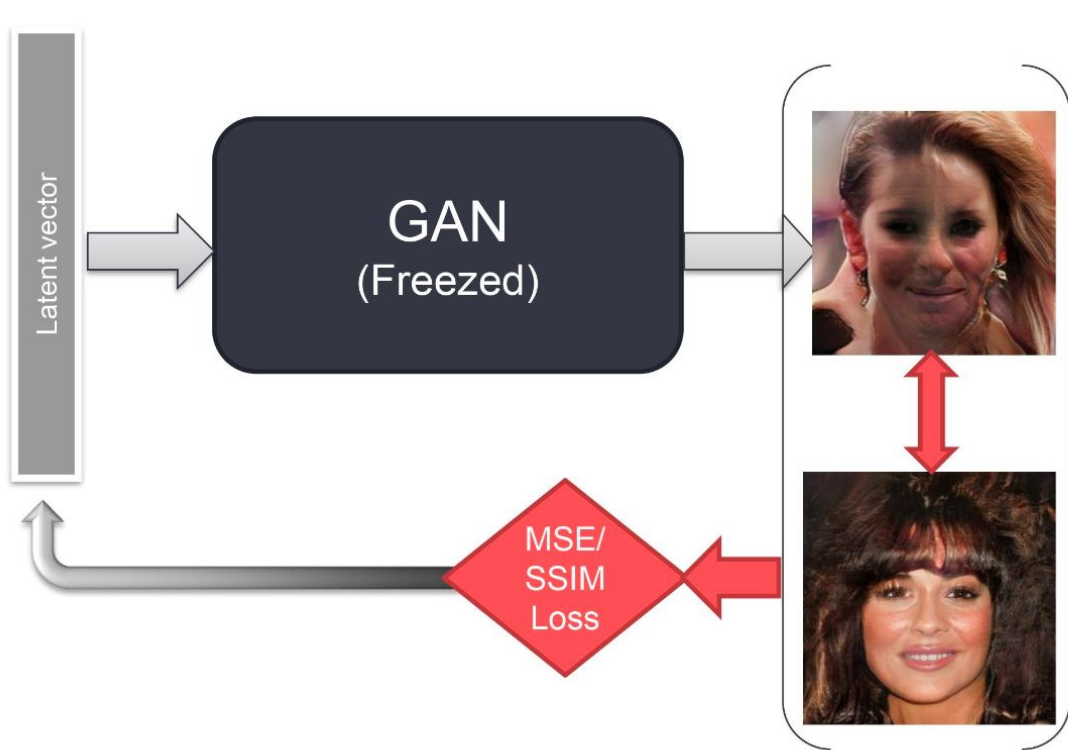


Compression Phase

1. 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
2. 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
3. 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.
4. The output image is then the compressed version of the provided input image with low loss (difference between two images)
5. Perceptual similarity metrics (Structural Similarity Index (SSIM)) used for tuning.

Compression Step





After 100 iterations



After 1000 iterations



Image Quality Metrics

Image quality metrics are useful for us to measure how well the architecture has performed and to define the loss.

MSE(l2) calculates the addition of squared differences between pixel values of two images. While l1 is the addition of differences between pixel values.

$$\ell_2(z') = \|f(z) - f(z')\|_2^2 \quad \ell_1(z') = \|f(z) - f(z')\|_1$$

These metrics are not always a good metric to access image compression quality since they do not take into account the range of variations in pixel values in an image and high, low-frequency components in an image. These factors, however, affect human perception towards quality.



SSIM: Structural Similarity Index Measure

SSIM calculates quality degradation in an image due to image processing tasks such as compression. SSIM is considered a better metric to assess degradation of images because it takes into account visible structures of an image. SSIM is calculated using a combination of variance and covariance terms between two images.

So the perceptual similarity metric, 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: $\frac{1}{N} \sum_{x,y} 1 - SSIM(x; y)$



Results





GAN reversal techniques outperforms traditional methods in the SSIM metric.

<i>Scheme</i>	<i>BPP</i>	<i>CR</i>	<i>PSNR</i>	<i>MSE</i>	<i>SSIM</i>
PNG (Uncompressed)	13.63	1	∞	0	1
K-means ($K = 4$)	1.486	9.172	23.012	342.46	0.7145
JPEG (10%)	0.4848	28.11	26.161	161.87	0.7711
JPEG (1%)	0.2176	62.64	20.465	589.15	0.5280
GAN Reversal (Our approach)	0.2152	63.34	21.192	540.06	0.7073

Summary of the metrics obtained from the baselines and our approach "GAN Reversal"

The table summarizes the performance of the baselines as well as our approach "GAN Reversal" on the proposed metrics. The results show that our approach has a compression ratio (CR) comparable to that of the extreme JPEG (1%) scheme (and even delivers a smaller size in most cases), while clearly outperforming it in the SSIM metric.



SSIM loss for the unseen images, which correlates much better with human perception

For seen images, we can achieve sizes approximately as small as 250 bytes. This number is slightly bigger for unseen images, with an approximate size of 450 bytes after extreme quantization. Note that it's harder to achieve higher fidelity in the reconstructions for unseen images, compared to seen ones, where the simple MSE loss almost always guarantees perfect recovery of the latent vector. They had to use a more sophisticated SSIM loss for the unseen images, which correlates much better with human perception.

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

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

Thankyou

