Image Colorization using WGANs

Team Blurons - Course Project CS726

Github Repository

Satwik Murarka (190020101) Mihir Nandawat (190020071)
Deepak Thorat (190020037) Rhythm Shah (190110074)

Problem Statement:

Adding useful color information to monochrome images is known as image colorization. Image colorization has many applications, including image restoration, astronomical photography, CCTV footage, electron microscopy, etc. In this project, we trained a DCGAN model with UNet Generator and further studied the effects of Wasserstein Loss Function along with its variants on the model performance and checked whether they were able to mitigate the shortcomings of baseline DCGAN such as mode collapse and instability during training. The network was trained on a publicly available dataset, namely Places365. The Peak Signal-To-Noise Ratio (PSNR) and Mean Absolute Error (MAE) were used as metrics to evaluate the performance of our models compared to the baseline model.

Literature Survey:

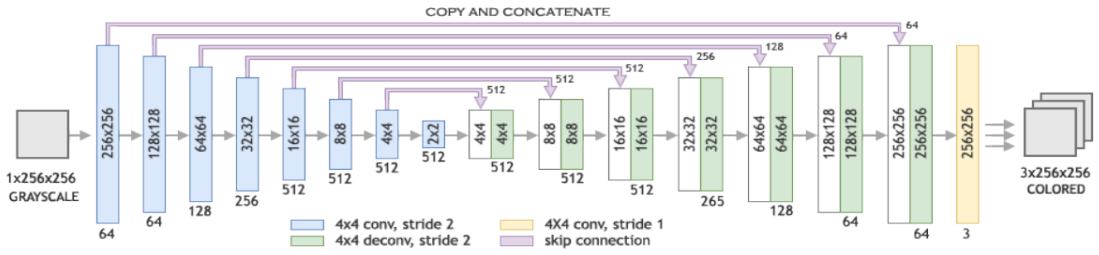
The idea of using DCGAN for image colorization was discussed in this paper

Generative Adversarial Network (GAN):

The generator and discriminator are two smaller networks that make up a GAN. The generator's goal is to produce results that are as close to real data as possible. The discriminator's job is to figure out if a sample came from the generator's model distribution or from the original data distribution. Both subnetworks are trained simultaneously until the generator can consistently produce results that the discriminator can't classify.

Deep Convolutional GANs (DCGAN):

The generator's input in a traditional GAN is randomly generated noise data z. But in DCGAN the input is a grayscale image with zero noise. In this model, fully connected layers are replaced by convolutional layers which include unsampling instead of max-pooling. This helps to train the model without consuming large amounts of memory. However, there is an information bottleneck that prevents the flow of low-level information in the network. To fix this problem, features from the contracting path are concatenated with the unsampled output in the expansive path within the network. This architecture is called U-Net, where skip connections are added between layer i and layer n-i.



U-Net Architecture (1x256x256)

DCGAN's cost function is as follows, it is a min-max optimization problem:

$$\min_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \min_{\theta_G} -\mathbb{E}_z \left[\log(D(G(\mathbf{0}_z|x)))\right] + \lambda \|G(\mathbf{0}_z|x) - y\|_1$$

$$\max_{\theta_D} J^{(D)}(\theta_D, \theta_G) = \max_{\theta_D} \left(\mathbb{E}_y \left[\log(D(y|x))\right] + \mathbb{E}_z \left[\log(1 - D(G(\mathbf{0}_z|x)|x))\right]\right)$$

Binary Cross-Entropy (BSE) loss is used to train traditional DCGAN, which leads to some problems which are as follows:

- **Mode Collapse**:- While training DCGAN under certain circumstances, it is unable to generate outputs belonging to different classes. It happens when rather than learning the underlying distribution of the data, Generator learns to consistently trick the discriminator by generating images under a certain mode defeating the entire purpose.

- **Vanishing Gradient**:- This problem arises when a large input space is mapped to a small one, which causes the gradient to disappear while backpropagating. Here, Generator is prone to it. Disappearing gradients (small gradients) make every step of the descent increasingly small which on a compounding scale can lead to almost zero progress in learning. This results in a strong discriminator, leading to a poor discriminator.

Wasserstein Loss Function:-

When the discriminator's job is simple, the above minimax loss function can cause the GAN to get stuck in the early stages of GAN training. Use of the Wasserstein Loss function circumvents the issue faced by Baseline DCGAN by el. It is an approximation of the Earth Mover distance, which is a measure of the distance between two probability distributions over a region. Wasserstein Loss approximates the Earth Mover's Distance between real and generated distributions. The equation for Wasserstein Loss is as follows:

$$\min_g \max_c \mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$$

Where c represents the critic which is the discriminator. We now use the name critic instead of Discriminator since it is no longer discriminating between two classes but trying to maximize the distance between its evaluation of a fake and real example.
For the Wasserstein GAN to work well we have to take into account an additional condition which is the critic needs to be 1-Lipschitz Continous or 1-L Continous. According to this condition, the norm of the gradient should be at most 1 at every point. This is important to ensure that the loss function is continuous, differentiable, doesn't grow too much, and maintains stability during training.
Now for ensuring the 1-L condition we can apply to methods to the critic neural network:

- **Weight Clipping**: This method ensures L-1 continuity by forcing the weights of the critic to be constrained to a fixed interval. The weighted outside this interval are clipped or set to the maximum or minimum limit. The disadvantage of this method is that it limits the critic's ability to learn.
- **Gradient Penalty**: This ensures continuity by adding a regularization term to the loss function. This penalizes the critic if the gradient norm is greater than one. This is done by sampling points by randomly interpolating between real and fake images.

$$\min_g \max_c \mathbb{E}(c(x)) - \mathbb{E}(c(g(z))) + \lambda \mathbb{E}(||\nabla c(\hat{x})||_2 - 1)^2$$

Expression for training a WGAN-GP

Peak Signal To Noise Ratio:-

PSNR is defined as the ratio between the maximum possible power of a signal and the power of undesirable losses that affect its representation.It is used to quantify the reconstruction quality of images.The expression for PSNR is given as:

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log \end{aligned}$$

Here MAX_I is the maximum possible pixel value of the image

Mean Absolute Error:-

Mean absolute error is the mean of the absolute difference between the individual pixel values of the generated and original image. Its expression is as follows:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

Our Approach

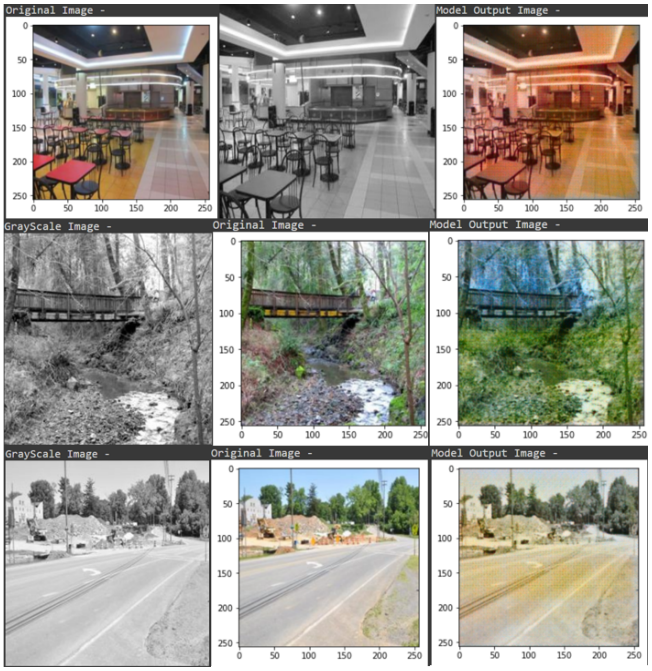
To counter the shortcomings of the DCGAN model, we approached the problem by experimenting with different loss functions. We first changed the loss to a Wasserstein loss, furthermore, we tried a Wasserstein loss with a gradient penalty. We trained the model on 5000 images from the Places365 dataset and tested it on 50 images due to Colab's GPU and RAM limitations. The models were trained for 10 epochs with a batch size of 20. The learning rate for optimizers was chosen to be 5e-5. L*a*b color space was used for the input image's matrix representation which contains a dedicated channel to depict the brightness of the image and the color information encoded in the remaining two channels. This space prevents any sudden variations in both color and brightness which is usually experienced on a small scale with RGB space.
The architectures of the models were as follows:-

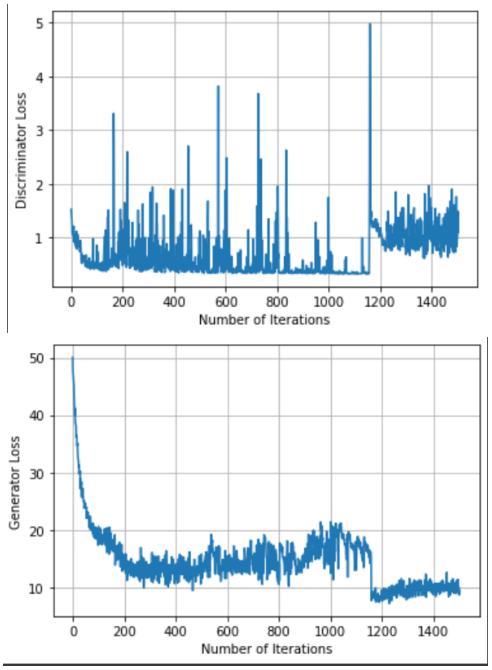
- **DCGAN** - The generator and discriminator both employ convolutional networks in their architectures. The generator follows a special architecture known as U-Net. There are skip connections between the ith and (n-i)th layer. The model is symmetric with n encoding and decoding layers. We downsample using 4*4 convolutional layers with stride 2 which is followed by downsampling followed by batch normalization. A Leaky-ReLU activation function with a slope of 0.2 is used. For upsampling, we also use 4*4 convolutional layers with a stride of 2 followed by batch normalization. For the last layer, we use 1*1 convolution and use a tanh activation function. The discriminator follows a similar architecture as the Generator. In the last layer, convolution is used to map to a 1-D output which is then followed by a sigmoid function to return the probability of the input being real or fake. The input to the discriminator and generator are real colored and black-white images respectively. Binary cross-entropy loss was used along with Adam optimizer with beta1 = 0.5 & beta2 = 0.999.

- **WGAN** - The WGAN follows the same architecture as the above DCGAN. The change comes in the loss function which now uses a Wasserstein loss. We also use a different optimizer namely RMSProp because WGAN performs worse on momentum based optimizers since critic loss is highly non stationary and clip the parameters of the discriminator between [-0.01,0.01]. The Critic is trained multiple times each step while the Generator is trained once per step.
- **WGAN with Gradient Penalty** - For this case, we add a gradient penalty to the loss as a regularization of the discriminator gradient to enforce 1-L continuity. For this case, we also remove the batch normalization in the discriminator.

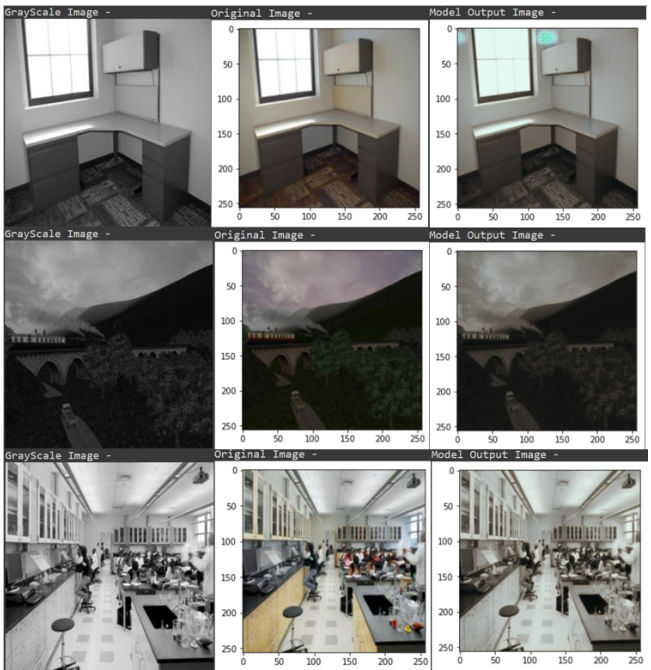
Results

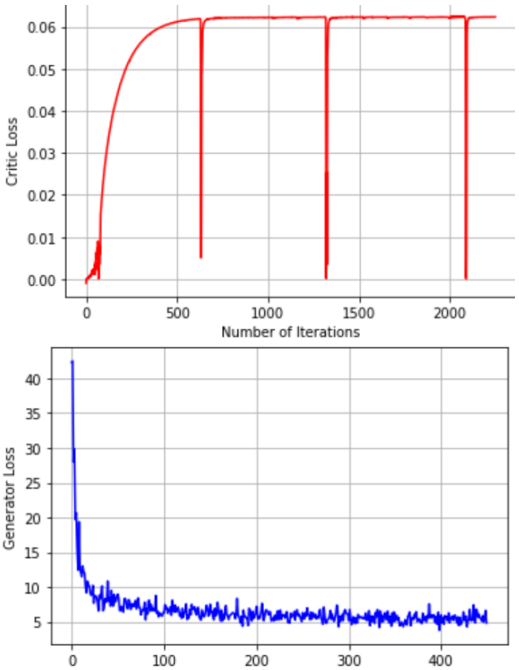
DCGAN Test Set Results:



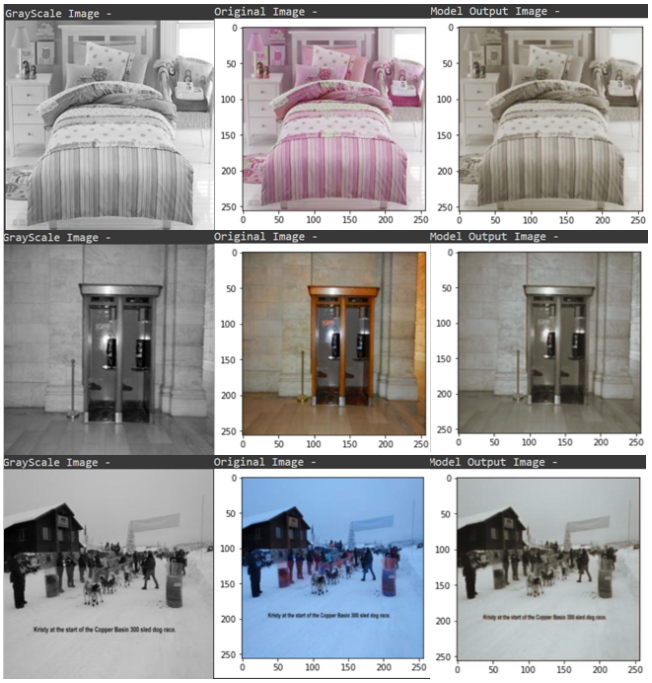


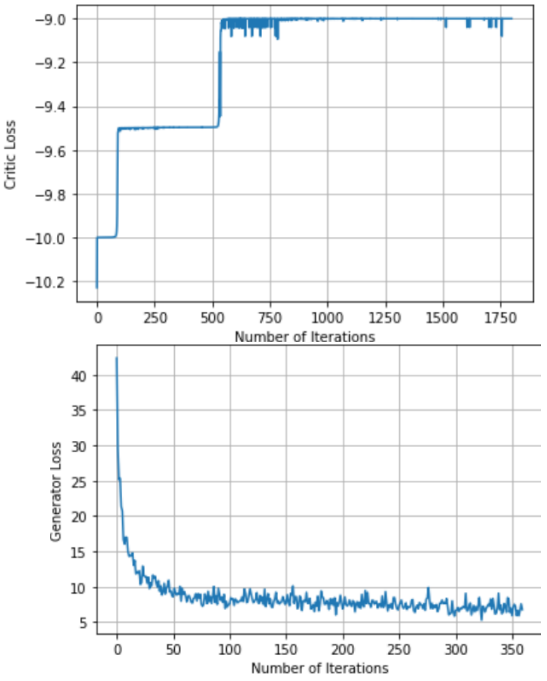
WGAN Test Set Results:





WGAN + GP Test Set Results:





Accuracy Metric:

A testing set of 50 images was taken on which PSNR and MAE metric was calculated, reported average:

Model	PSNR	MAE
Vanilla DCGAN	6.25	113.34
WGAN	6.27	111.46
WGAN with Gradient Penalty	6.29	110.69

Conclusion

We observed the following:-

- PSNR Metric increases in the following order DCGAN → WGAN → WGAN + GP which indicates that the quality of the output images from the generator has improved
- MAE Metric decrease in the following order DCGAN → WGAN → WGAN + GP which indicates that the generated image is closer to the original colored image
- Good quality images were not observed because of insufficient training data and epoch count due to GPU limitations.