Table of Contents

List of All Figures

Abstract

As a general-purpose approach to image-to-image translation issues, conditional adversarial networks are being looked into. These networks learn a loss function to train this mapping in addition to learning the mapping from the input image to the output image. As a result, it is conceivable to use the same generic technique to solve issues that in the past called for completely distinct loss formulations. We show that this method is efficient for a variety of tasks, including colorizing photographs, reconstructing objects from edge maps, and synthesizing photos from label maps. In fact, other internet users (many of them artists) have submitted their own experiments with our method since the publication of the pix2pix program connected with this paper, further confirming its broad applicability and simplicity of adoption without the need for parameter tinkering.

1. Introduction

In image processing, many problems are based on “translating” an input image into a corresponding output image. The regulated conversion of a given source image to a target image is known as image-to-image translation. An example might be the conversion of Gray photographs to color photographs. We define automatic image-to-image translation, which is related to automatic language translation, as the process of converting one possible representation of a scene into another given enough training data.

Convolutional neural networks (CNNs) have already made tremendous advancements in this field, becoming the go-to solution for a range of picture prediction issues. Despite the fact that CNN's learn to minimize a loss function - an objective that rates the quality of results, a lot of manual work still goes into creating efficient losses. But we must use caution when making wishes, just like King Midas! The CNN will typically provide fuzzier results if we use a naive approach and ask it to minimize the Euclidean distance between the predicted and ground truth pixels.

In this paper, we investigated a generic method for image-to-image translation in which we train our custom model and compare it with other available technology. It is based on the conditional generative adversarial network, where a target image is generated, conditional on a given input image. In this instance, the Our GAN modifies the loss function to ensure that the produced picture is both a plausible translation of the input image and plausible within the content of the target domain. Our additional contribution consists of outlining a straightforward framework that is nevertheless effective and of examining how certain critical architectural decisions affect the final product.

2. Literature Review

2.1 Structured losses for image modeling:

Per-pixel classification or regression problems are frequently used to create image-to-image translation issues. These formulations treat the output space as "unstructured" in the sense that, given the input image, each output pixel is taken to be conditionally independent of

every other output pixel. Instead, conditional GANs learn a structured loss. Structured losses punish the output's combined configuration. This type of loss has been studied extensively in

the literature using a variety of techniques, such as conditional random fields [10], the SSIM metric feature matching, nonparametric losses, the convolutional pseudo-prior, and losses based on matching covariance statistics. The conditional GAN differs from other models in that the loss is learned and, in theory, can punish any structure that differs between the output and the goal.

2.2 Conditional GANs:

The use of GANs in the conditional setting is not new to us. GANs have been trained using discrete labels, text, and even images in earlier and ongoing research. The future frame prediction, product photo production, image generation from sparse annotations, and image prediction from a normal map have all been addressed by the image-conditional models. While additional terms (such as L2 regression) were employed to make the output conditional on the input, some other articles that used GANs for image-to-image mappings just applied the GAN unconditionally. Inpainting, future state prediction, image alteration directed by user constraints, style transfer, and superresolution have all seen outstanding results in these publications.

Each technique was designed with a particular application in mind. Nothing about our framework is application-specific.This makes our setup far more straightforward than others. Additionally, our approach differs from earlier efforts in a number of architectural decisions made for the generator and discriminator.

Contrary to prior work, we developed a "U-Net" like architecture for our generator and a convolutional "PatchGAN" classifier for our discriminator that only penalizes structure at the scale of individual image patches. Our generator uses a different convolution block unit which has residual networks so that model can become deeper, It was previously suggested in to use a PatchGAN architecture in the discriminator to record local style statistics. Here, we demonstrate that this method works on a larger variety of issues and examine the impact of varying the patch size.

3. Proposed Solution

The Our GAN architecture involves the careful specification of a generator model, discriminator model, and model optimization procedure. Both the generator and discriminator models use standard Convolution-BatchNormalization-ReLU blocks of layers as is common for deep convolutional neural networks.

3.1 Dataset Creation

Initially 1000 colored scenery images were taken. These images were converted to black and white images(256x256) and then joined with corresponding colored images(256x256) beside each other(as shown in fig.1). Thus the target size of the image becomes 256x512 pixels. Here the left side of the image is the input image i.e black and white image and the right side of the image is the output image i.e colored image.

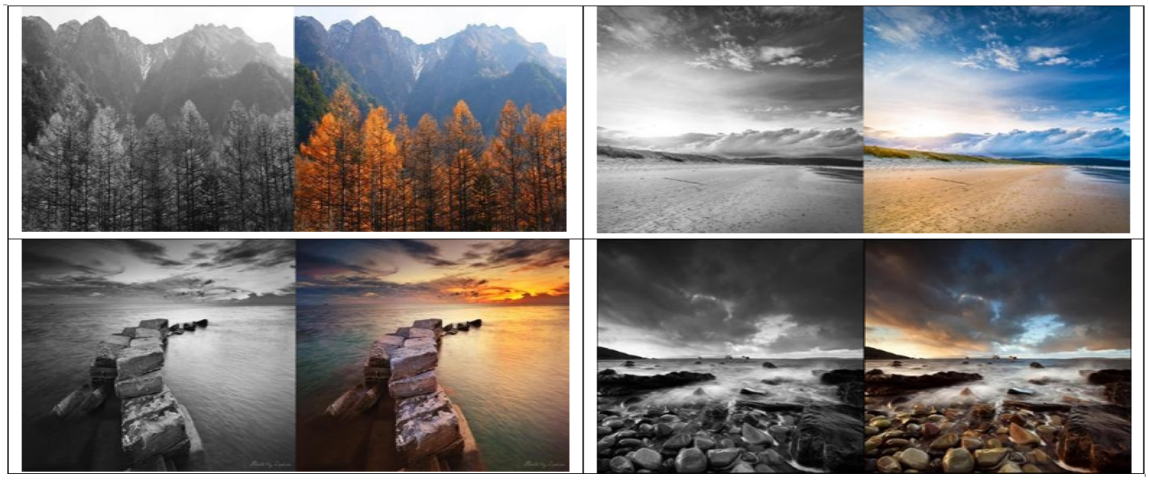

Fig.1:- Dataset having Gray and RGB images

3.2 Model Building and Training

Generator models

3.2.1 U-Net

In the U-net generator model if we feed the picture to an encoder that continuously shrinks the spatial size of the feature block, the network will generalize to store just the crucial features and toss out less relevant information after a sufficient amount of training. The desired output mask will then be produced by the encoder's output and a decoder. The issue was that the decoder layers were not receiving enough context from the encoder output to produce the segmentation mask.

The generator is modified by U-Net. Simply described, a U-Net is a net where the input is downsampled in the first phase and upsampled in the second. In that it entails downsampling to a bottleneck and upsampling once again to produce an output image, the U-Net model architecture is quite similar. However, links or skip connections are created between layers of the same size in the encoder and the decoder, allowing the bottleneck to be avoided.

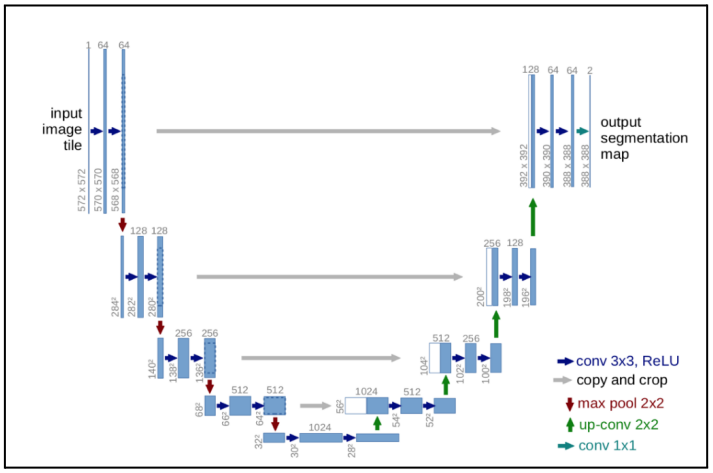

Fig.2:- U-Net model architecture

The architecture is symmetrical and is divided into two main sections: the expansive path on the right is made up of transposed 2D convolutional layers, while the contracting path on the left is made up of the general convolutional process.

3.2.2 U-Net++

This network went much further with the skip connection concept. Instead of using the same spatial dimension for the context information between the encoder and decoder, they scaled the context to fit each level of the decoder by taking it from each encoder level and feeding it at the appropriate spatial size.

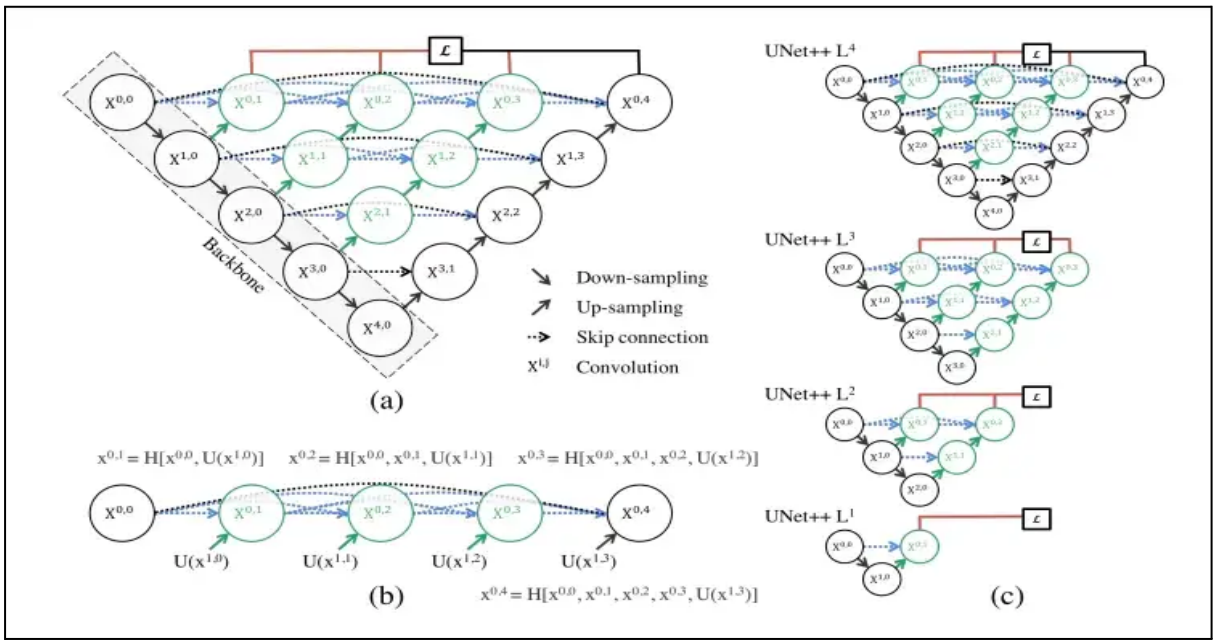

Fig.3:- U-Net++ model architecture

3.2.3 U-Net3+

This has fewer arguments than UNet++ but is comparable. Similar to Attention U-Net but with even less settings, this works really well. Another innovative concept in this work is the Classification Guided Module, which enhances segmentation by using classification findings.

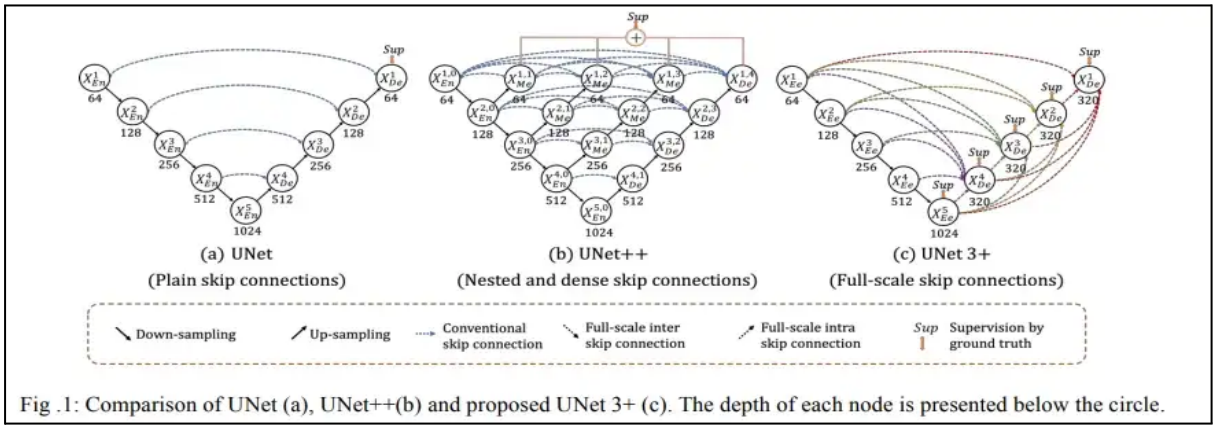

Fig .1: Comparison of UNet (a), UNet++(b) and proposed UNet 3+ (c). The depth of each node is presented below the circle.

Fig.4:- U-Net 3+ model architecture

3.2.4 Linknet

If we replace the operation of concatenation with the addition in U-Net, we get the network called **LinkNet.** LinkNet performs similarly to the U-Net (in some cases even beating the U-Net).

3.2.5 R2U-Net

The recurrent neural network concept is used in this model to give the network a temporal dynamic characteristic. But, it was unable to converge on a unique dataset at all.

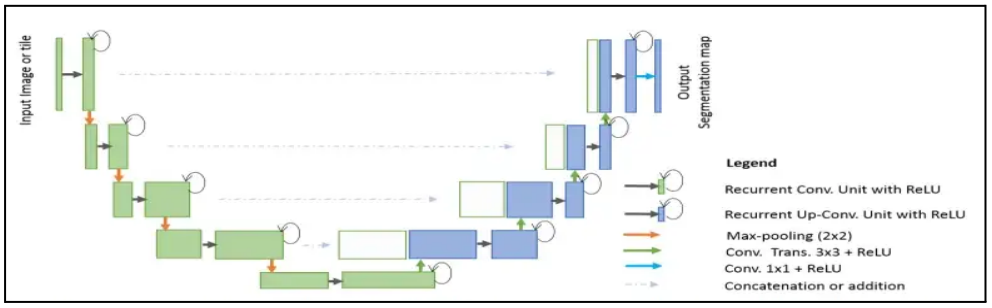

Fig.5:- R2U-Net model architecture

3.2.6 Our Proposed Model

Our Model uses a special convolution unit block that is arranged in a U-like fashion. It has three components: Encoder, Decoder, and skip connections. We mainly enhanced the basic convolution unit block. Our convolution unit block is inspired by two different neural networks. First in residual neural network and recurrent network. The residual network allows output to have the input from previous layers so that a deeper neural network can be formed. A Residual Network can improve the result since input information is added at each layer as a guide to output for better results.

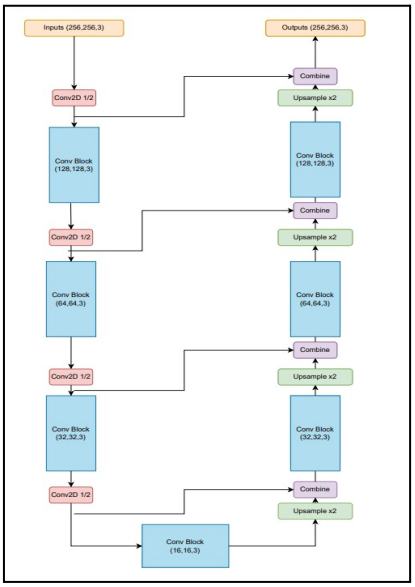

Fig.6:- Network architecture of our proposed model

Our convolution block consists of 6 convolutions each of them followed by Relu and batch normalization. It is a three-stream flow of layers where the first layer is directly added, a single convolution is added and the third is passed through multiple residual and recurrent layers are added together. The Middle layer has 5 convolution layers that are connected to the previous layer by the residual network.

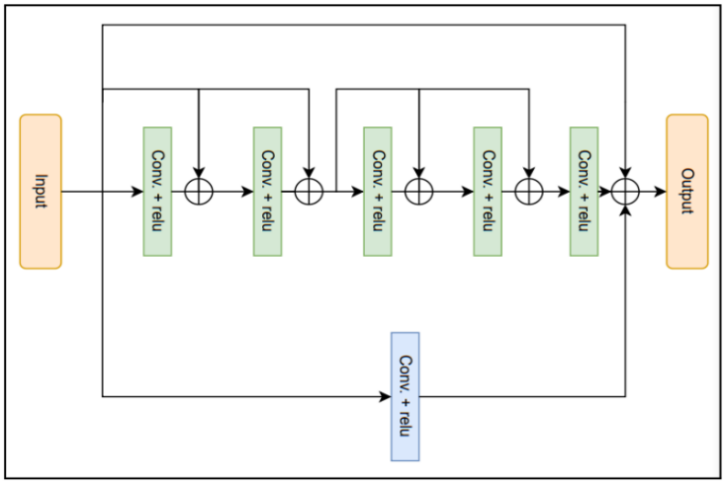

Fig.7:- Architecture of the Conventional unit block

PatchGAN Discriminator Model

The discriminator model takes an image from the source domain and an image from the target domain and predicts the likelihood of whether the image from the target domain is a real or generated version of the source image.

Unlike the traditional GAN model that uses a deep convolutional neural network to classify images, the Pix2Pix model uses a PatchGAN. This is a deep convolutional neural network designed to classify patches of an input image as real or fake, rather than the entire image.

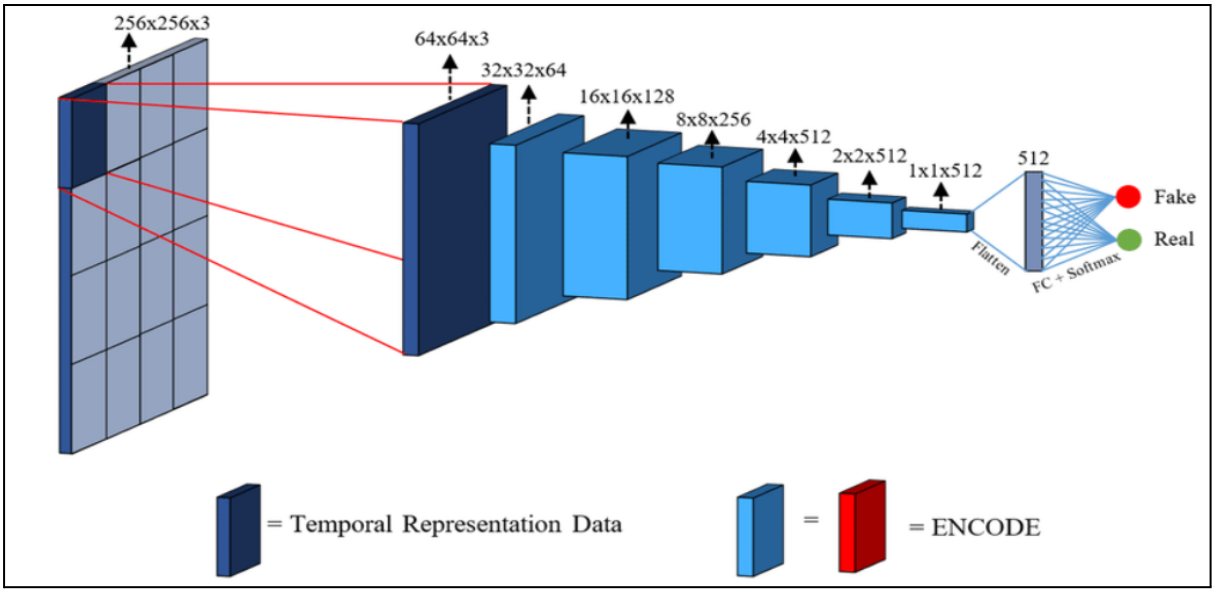

Fig.8:- Model Architecture of PatchGan

Total Loss

The generator model must go through the discriminator model in order to be updated, but the discriminator model can be updated directly. To do this, create a new composite model in Keras and connect the generator model's output to the discriminator model's input. If a created image is authentic or fraudulent, the discriminator model can then make that prediction. If we change the composite model's weights so that the created image is labeled "genuine" rather than "fake," the generator weights will be changed to produce a better false image. To prevent the misleading update, we may additionally mark the discriminator weights as not trainable in this situation. To better match the planned translation of the input image, the generator also has to be modified. In order to compare the created image to the target image, the composite model must output the generated image directly as well. Therefore, the inputs and outputs of this composite model can be summed up as follows:

- Inputs: Source image
- Outputs: Classification of real/fake, generated target image.

The weights of the generator will be updated via both adversarial losses via the discriminator output and L1 loss via the direct image output. The loss scores are added together, where the L1 loss is treated as a regularizing term and weighted via a hyperparameter called lambda (λ), set to 100.

$$\text{loss} = \text{adversarial loss} + \lambda \times \text{L1 loss}$$

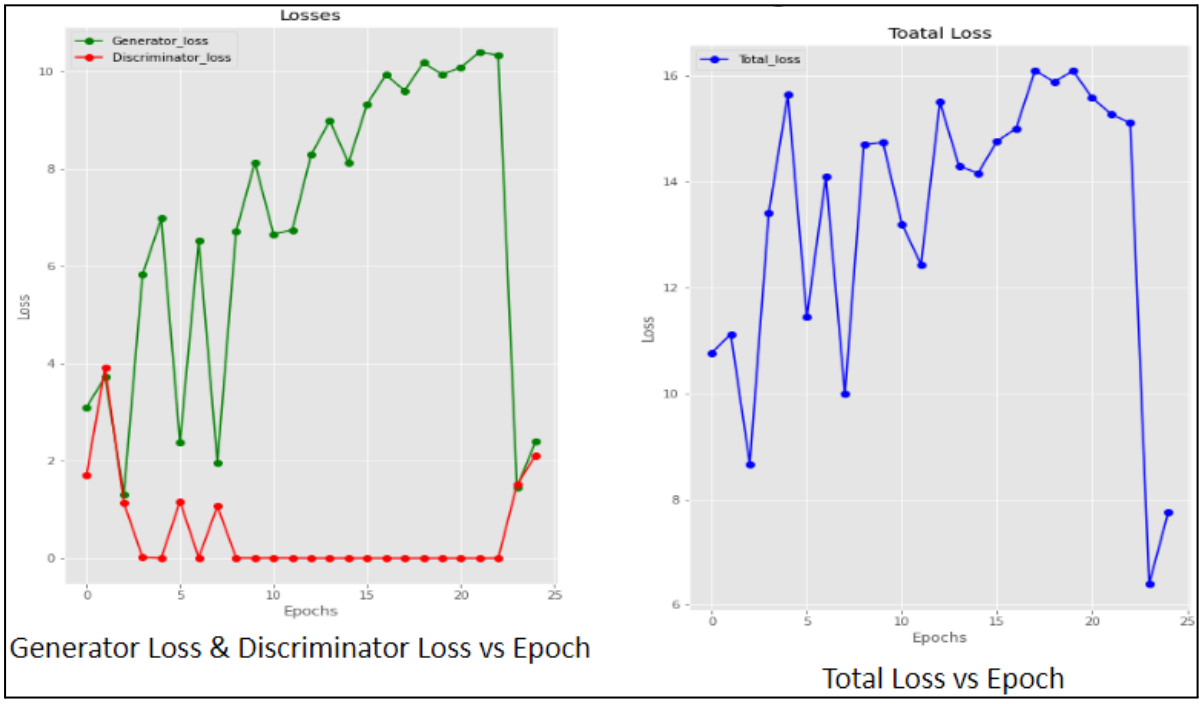

Fig.9:- Losses vs Epoch

4. Results

Every 10 epochs, models are stored in a file and assigned a training iteration number. Additionally, images are produced every 10 epochs and contrasted with the goal images that are desired.

These plots can be evaluated at the conclusion of the run and used to choose a final generator model depending on the quality of the images that were produced. After about 25 training epochs, generated images start to look pretty realistic, and the quality seems to hold for the rest of the training period. In comparison to the actual Colored Images image, the first generated image sample below contains more helpful detail.

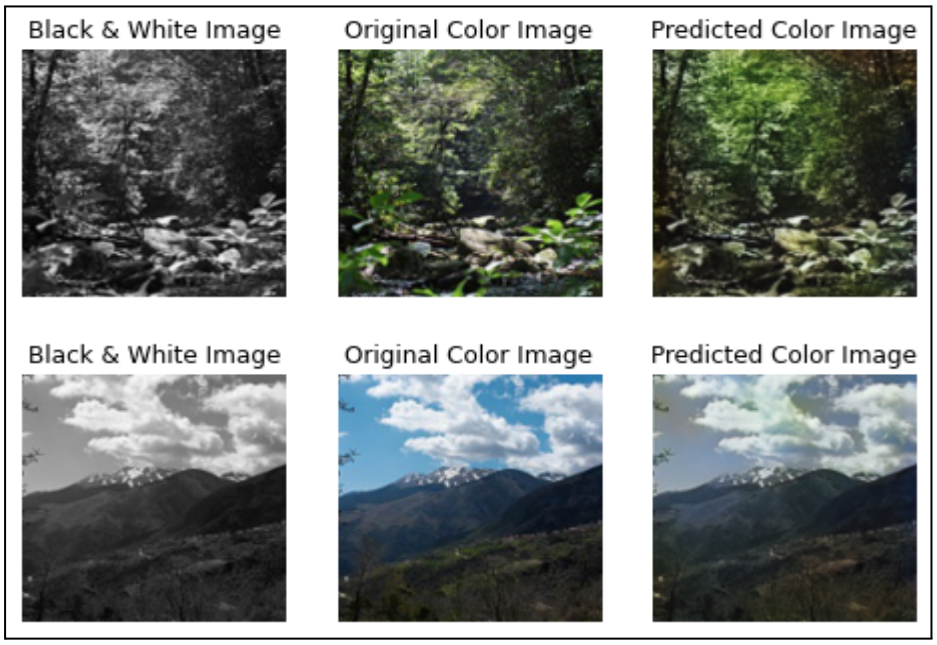

Fig.10:- Result after 25 epochs

We have also made a comparative analysis between our model and some pre-existing models.

	Models	Generator Loss	L1 Loss	Total Loss	Discriminator Loss
1.	Unet	1.2061	**0.0395**	6.7701	0.0006
2.	Unet++	3.0047	0.0746	10.4722	0.0002
3.	Unet3+	1.0446	0.0602	7.7463	0.0002
4.	Linknet	**0.8138**	0.0448	**5.5697**	0.0007
5.	R2Unet	1.5841	0.0504	5.6704	0.0005
6.	Our Model	1.3111	0.0477	6.4000	**0.0001**

Fig.11:- Evaluation Matrix

5. Conclusion

The results in this paper suggest that our model which is also a type of conditional adversarial network has promise for image colorization, particularly those requiring highly organized graphical outputs. These networks learn a loss that is tailored to the job and available data, making them useful in a broad range of contexts. The highly advanced generator model with the property of residual networks and recurrent neural networks surpassed the outcomes performed by previous existing techniques.

6. Acknowledgements

We would like to thank our guide, **Dr. Abhishek Sharma**, for their guidance and feedback during the course of the project. We would also like to thank our department for giving us the resources and the freedom to pursue this project.

7.References

1. https://arxiv.org/abs/1406.2661
2. https://arxiv.org/abs/1611.07004
3. https://phillipi.github.io/pix2pix/
4. https://www.tensorflow.org/tutorials/generative/pix2pix
5. https://en.wikipedia.org/wiki/Generative_adversarial_network
6. https://github.com/phillipi/pix2pix