Project Report - ECE 285

Alexandra Mikhael
ECE
A15763534

Devanshi Panchal
ECE
A59026424

Abstract

The objective of this project is to start by replicating the research presented in Phillip Isola et al.'s paper, "Image-to-Image Translation with Conditional Adversarial Networks" [1]. Subsequently, we conducted experiments using various loss functions and alternative generator-discriminator architectures, and have summarized the outcomes of these experiments.

1 Introduction

In computer vision, numerous challenges boil down to transforming an image from one representation to another, essentially involving image-to-image translations. One of the most used method to achieve this is Conditional Generative Adversarial Networks (cGAN). The cGAN is a model used in deep learning, and enables more precise generation and discrimination of images to train machines and allow them to learn on their own. This concept was first published in 2014 by Mehdi Mirza and Simon Osindero.

cGAN is a neural network that enables the generation of data that aligns with specific properties, which can be class labels, textual descriptions, or other traits, by harnessing the power of conditions. It is a type of GAN model where a condition is put in place to get the output.

Through the implementation of our reference paper [1], we aim to address the task of image-to-image translation utilizing cGANs. Our motivation to choose this topic stems from the increasing demand for automated systems capable of transforming images from one domain to another, a fundamental challenge in computer vision and graphics.

Key aspects of the problem include the need for accurate and reliable translation between diverse image domains, such as converting satellite images to maps or transforming sketches into realistic photographs. These translations are essential for various applications, including image enhancement, style transfer, and domain adaptation.

GANs are designed to learn a loss function that is adaptable to the data and can be utilized across a wide range of tasks. These tasks would traditionally require diverse types of loss functions. In the case of Conditional GANs, they are trained to learn a structured loss that can penalize any structure that deviates between the output and the target.

While there have been previous attempts to apply GANs in a conditional setting, the distinctiveness of this proposal lies in its universal applicability, not being confined to a specific application. Furthermore, this framework distinguishes itself from prior works through its architectural choices for the generator and discriminator. It employs a U-Net- based generator and a convolutional PatchGAN classifier, which only penalizes structure at the scale of image patches, enhancing its uniqueness.

Our proposed approach is to re-implement the paper, modify the architectures of the generator and discriminator, and try implementing different loss functions to see its effect on the output. For the first part of the approach we will implement the Johnson's ResNet[4] architecture, more specifically ResNet6 and ResNet9 for the generator and try an ImageGAN network for the discriminator. For the second part we will implement the U-Net architecture used in the paper with the Least-square and Wasserstein Loss functions.

Looking at the results and the generated images for the different parts of our work, we saw that Encoder-Decoder network with ResNet9 and BCE loss function as well as UNet with Wasserstein loss perform the best out of all the options.

2 Related Work

We have mainly done extensive literature review of the following two papers for the architectures used in this project.
"Image-to-image translation with conditional adversarial networks." Isola, Phillip, et al.

In this paper, the authors explore conditional adversarial networks as a versatile solution for image-to-image translation tasks. These networks not only learn to map input images to output images but also develop a loss function to optimize this mapping. This enables the application of a uniform approach to problems that typically require distinct loss formulations. They demonstrate the effectiveness of this method in synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images, among other tasks.

The framework used in this paper is unique in that it is not tailored to any specific application, making it significantly simpler than most others. Additionally, the method differs from previous works in several architectural choices for both the generator and the discriminator. Unlike past approaches, "U-Net"-based architecture is used for the generator and a convolutional "PatchGAN" classifier for the discriminator, which penalizes structure at the scale of image patches. To explore the generality of conditional GANs, they test the method on a variety of tasks and datasets, including both graphics tasks, like photo generation, and vision tasks, like semantic segmentation.

Perceptual losses for real-time style transfer and super-resolution. Johnson, J., Alahi, A., Fei-Fei, L. (2016).

The authors train feed-forward transformation networks for image transformation tasks, but instead of using per-pixel loss functions that rely solely on low-level pixel information, they use perceptual loss functions based on high-level features from a pretrained loss network. During training, perceptual losses provide a more robust measure of image similarity than per-pixel losses, and at test time, the transformation networks operate in real-time.

The authors' neural network architecture incorporates 5 residual blocks between the downsampling and upsampling stages, leveraging residual connections to facilitate learning the identity function—a crucial aspect for image transformation tasks where the output image must retain the structure of the input image. Applied to style transfer, their model achieves comparable performance to existing models while significantly enhancing speed. In single image super-resolution, training with a perceptual loss enables the model to reconstruct fine details and edges more effectively. These innovations result in a network that not only trains more easily and efficiently but also delivers superior visual quality in its outputs.

3 Method

The first step was to try reimplementing the Pix2pix paper[1], which uses a U-net based generator and a PatchGAN discriminator. We have then tried to use different loss functions for our cGAN network that was proposed in the paper. The two loss functions we are investigating in this project are : 1) Least-Square GAN loss and 2) Wasserstein loss.

Next, we modified the generator and discriminator architectures. For the generator, we tried including residual blocks between the encoder-decoder network of the generator. Additionally, we also experimented with using an ImageGAN discriminator, instead of a PatchGAN architecture as used in the paper, to observe the changes.

3.1 Pix2Pix Reimplementation

Conditional Generative Adversarial Networks (cGANs) consist of two components: a generator and a discriminator. The generator aims to produce valid samples from a constant noisy input, while the discriminator is trained to distinguish between real and fake inputs. The generator continually attempts to fool the discriminator by classifying its generated samples as real, while the discriminator strives to identify the generated samples as fake and the training set data as real. The discriminator is also trained on real data to learn the difference between authentic and generated samples. After the training process, the cGAN's generator is capable of producing synthetic images that closely resemble real ones.

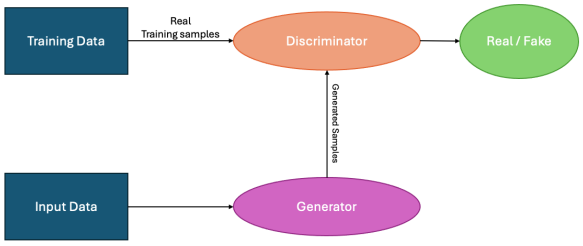

Figure 1: Vanilla GAN architecture

Now for Image-to-image translations, the GAN architecture is more complex.
The training data consists of two images : input image and true image. The input to the generator is the input image. The generator model is now an AutoEncoder model, i.e., it would have both encoder and decoder in it. This Generator architecture makes sense since adding an encoder in the model would enable Generator to extract the essential features from the input image, and the decoder would generate an image close to the true image.
Now for the discriminator, we first need to train it on the ground truth, so we train it on real samples, i.e., the input and true image from the training data. Then we would pass the output image from the generator alongside the input image from the training data and see its performance based on loss functions.

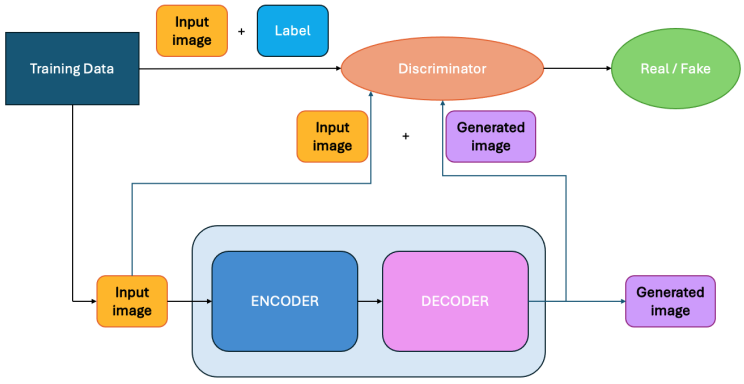

Figure 2: GAN architecture for image-to-image translations

The loss function of cGAN is defined as both adversarial loss and L1 loss as used in the paper. The adversarial loss is expressed as :

$$L_{cGAN}(G, D) = E_{x,y}[logD(x, y)] + E_{x,z}[log(1 - D(x, G(x, z)))] + \lambda L_{L1}(G) \quad (1)$$

where,

$$L_{L1}(G) = E_{x,y,z}[||y - G(x, z)||]$$

Here, x is the input image, y is the output image and z is the random noise vector fed to the generator to learn to producs fakes that are similar to the ground truth y.

L1 distance is added to the generator loss for low-frequency correctness of the generated image. L1 distance is preferred over L2 distance as it produces images with less blurring [1].

U-Net Generator

In U-Net generator architecture, encoder and decoder are mirror images of each other.

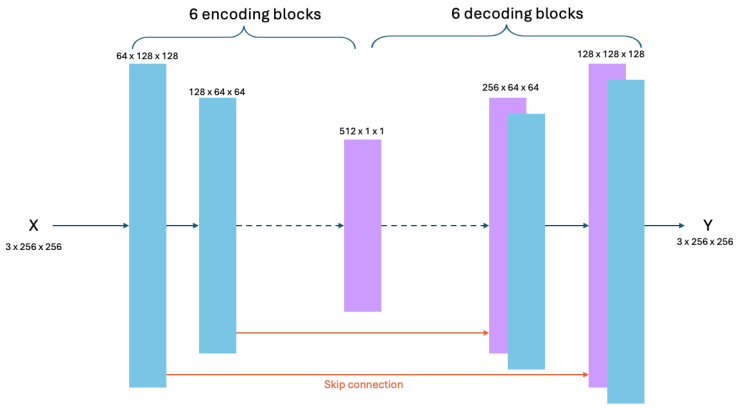

Figure 3: Our U-Net Generator Architecture

A U-Net architecture is proposed for the generator, which adds skip connections to the traditional encoder-decoder network. Here, skip connections are introduced between layers (i) in the encoder and their counterparts (N - i) in the decoder, allowing direct transmission of shared information and eliminating the need to unnecessarily pass all the information through several layers, including the bottleneck of the encoder-decoder network.

In our project, the encoder consists of six convolutional blocks. Each block doubles the number of filters from the previous block, reducing the image dimensions by half. These blocks consist of a convolutional layer followed by a batch normalization layer and a ReLU activation function. Following the encoder, a bottleneck layer serves as a conduit to transfer the encoded features to the decoder. The initial convolutional layer before the 6 encoding layers doesn not have batch normalization.

The decoder, being the mirror image of the encoder, doubles the image size with each block. However, there is a slight modification in the decoder's input: we concatenate the output of the previous decoder layer (or convolutional block) with the corresponding output from the mirrored encoder layer.

All ReLUs in the encoder are leaky, with slope 0.2, while ReLUs in the decoder are not leaky. After the last layer in the decoder, a convolution is applied to map to the number of output channels (3), followed by a Tanh function.

PatchGAN Discriminator

The discriminator, called D, looks at small sections(patches) (70x70) of an image to decide if they look real or fake. Instead of just saying if the whole image is real or fake, it looks at each part separately. This method, known as PatchGAN, helps to judge the realism of different parts of the image.

In the context of the this paper, the discriminator uses four convolution blocks with some specific numbers of filters, i.e., 64, 128, 256, 512 in each convolutional block. Each convolutional block in Discriminator would consist of 1 convolutional layer, 1 batch-normalization layer, and one LeakyReLU layer as an activation function.

3.2 Different Loss Functions

As described in the previous sections, the original paper uses the min-max loss, also known as Binary Cross-entropy Loss (BCE loss) and L1 Loss for training. Here, we have tried to use Least-Square GAN loss, more specifically MSE loss and Wasserstein Loss functions.

Least-Square GAN Loss

Traditional GANs often face vanishing gradient issues with the sigmoid cross-entropy loss function, especially for samples positioned correctly but far from real data. Addressing this, researchers introduced LSGANs in [2], which use the least squares (LS) loss function. The function minimizes the Pearson X^2 divergence between the real and generated data distributions. This helps the generator to produce samples that are more like the real data. In standard GANs, the discriminator provides feedback in the form of a binary real/fake decision. In contrast, the Least Squares Loss function provides a continuous measure of how real or fake the discriminator thinks a sample is. This richer feedback helps the generator to improve.
We specifically tried Mean-Squared Error (MSE) loss which is essentially the least square loss divided by the number of observations n.

Wasserstein GAN Loss

Proposed in [3], this approach entails transforming the discriminator into a critic, which receives more frequent updates compared to the generator model. Instead of predicting probabilities, the critic assigns real-value scores to images and necessitates keeping model weights small. The scoring mechanism aims to maximize the separation between scores assigned to real and fake images. Wasserstein's loss offers the advantage of providing a valuable gradient across most points, facilitating ongoing training of the models.

3.3 Different Architectures Implementation

In this project, we investigated the Johnsons Resnet architectures, more specifically, Resnet6 and Resnet9 for the generator and an ImageGAN Discriminator.

Johnson's Resnet Generator

To improve the model performance, we developed a residual-based encoder-decoder network based on the ResNet model in [4]. Our network is composed of 2 encoding blocks, 6 or 9 residual blocks, and 2 decoding blocks. Each encoding and decoding block follows two-stride convolution/deconvolution-BatchNorm-ReLU structure with reflection padding. Each residual block follows the convolution-BatchNorm-ReLU-convolution-BatchNorm residual connection structure. The initial convolutional layer before the encoders has a larger kernel size of 7, a stride of 1 and padding of 3. While the final convoutional layer after the decoders uses the same settings as the initial block, it is additionally followed by a tanh function.

We chose this architecture as incorporating ResNet blocks into a UNet architecture for GANs combines the strengths of both designs, leading to improved gradient flow, enhanced feature learning, better convergence, higher quality outputs, and greater flexibility. This combination leverages the

advantages of deep residual learning and the encoder-decoder structure of UNet, making it a powerful approach for image-to-image translation tasks.

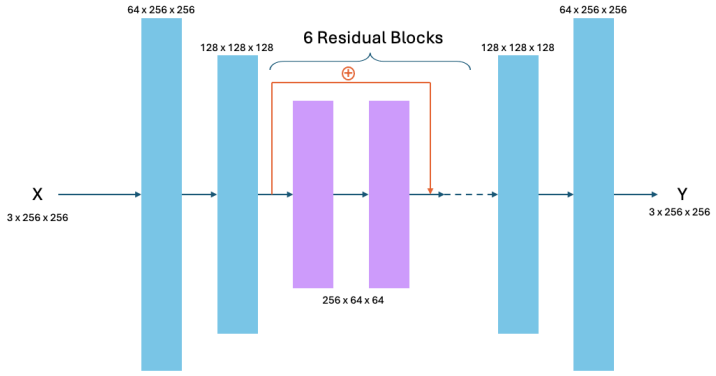

Figure 4: ResNet6 Architecture

ImageGAN Discriminator

ImageGAN discriminator considers the entire input image as a whole for discrimination instead of taking in patches as the input and outputs a single scalar value representing the probability that the input image is real. The model begins with an initial convolutional block followed by a series of convolutional blocks that progressively downsample the input and extract features. Each convoutional layer includes convolution, batch normalization, and Leaky ReLU activation. The network then flattens the feature maps and uses fully connected layers to produce a single output, indicating the authenticity of the image pair.

We chose to experiment with this model as we were curious to visualise the difference in the generated images between the two models. With further literature review we also found out that this type of model for the discriminator is more suitable for tasks where global coherence and overall image quality are crucial, such as image to image translation. This design ensures effective feature extraction and a stable training.

3.4 Training

For the training of our Pix2Pix model, we use Adam optimizers for both the generator and the discriminator with a learning rate = 0.0002, beta1 = 0.5 and beta2 = 0.999. The training dataloader is prepared with a batch size of 16. All of these settings are retained from the original paper. The number of epochs were 400. For the loss functions, we use L1 loss with either BCE loss (for baseline Unet, Johnson's Resnet and ImageGAN architectures) , MSE loss or Wasserstein Loss (experiments with loss function).

The training function iterates over the training data loader, which provides batches of paired images, where each pair consists of an image from one domain (x) and its corresponding image from another domain (y).

Real images are passed through the discriminator, and the loss is calculated, aiming to classify real images as 1. Similarly, fake images are fed into the discriminator (with the gradients from the generator detached to avoid backpropagation through the generator), and the loss is computed, aiming to classify fake images as 0. The discriminator loss is the average of the losses from real and fake images.

For the generator, fake images generated by the generator are passed through the discriminator, and the loss is calculated, aiming for the discriminator to classify them as real (1). Additionally, an L1 loss term is incorporated, measuring the pixel-wise difference between the generated image and its corresponding real image, scaled by a hyperparameter lambda = 100. The generator loss is the sum

of the loss and the L1 loss term.

By alternating between training the discriminator and the generator, this training function drives the adversarial learning process, continuing until convergence, resulting in a well-trained GAN model capable of generating high-quality images.

4 Experiments

4.1 Datasets

We used 2 datasets for our project. The Face2comics dataset was used for the training and evaluation of the baseline Unet model and all the experiments and modifications stated above. We used a modified version CelebA dataset to experiment if our model works well with image-inpainting tasks. Both the datasets were divided into train and val folders, each of which had sub-folders of the input and targetted ground-truth images.

4.1.1 Face2Comics

The Face2Comics dataset is designed for the purpose of converting real human faces into comic-style images and vice-versa, facilitating research and development in the field of image-to-image translation. The dataset contains pairs of images where each pair consists of a real human face photograph and its corresponding comic-style representation. The images are in .jpg format. We have used this dataset for the training and testing (different set of pictures used for evaluation) for original U-net model and all the subsequent experiments (different loss functions and modified architectures). Here the input was a comic image and the ground truth was a human face.

Figure 5: Example of paired images in the Face2Comics dataset

4.1.2 CelebA Dataset for Image Inpainting Experiment

CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. The images are in .jpg format. This dataset was modified by us for the purpose of this project. For training we, gave the input as a randomly masked image of a celeb and the ground truth was a fully constructed image (as shown in the left in Figure 6). For validation of the model, the image had a mask at the center and the ground truth was a full image (as shown in the right in Figure 6).

Figure 6: Example of images in the CelebA dataset

4.2 Results

Original Paper Architecture Results

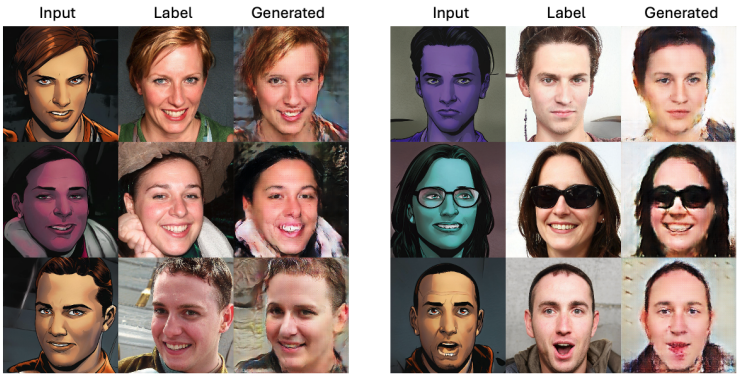

Figure 7: Results for U-Net architecture as implemented in the reference paper

Least-Square GAN Loss

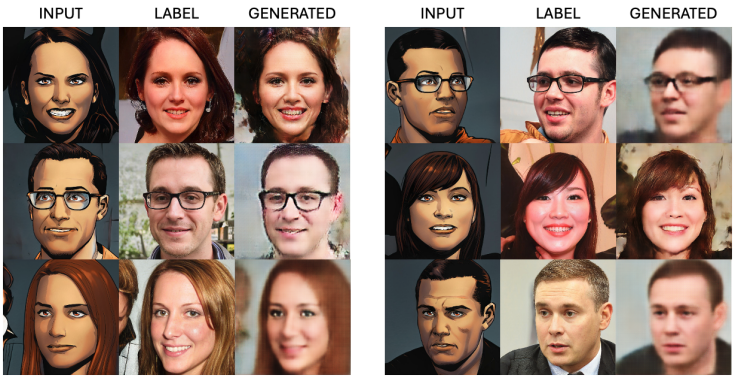

Figure 8: Results for U-Net architecture with Least-Square Loss function

Wasserstein GAN Loss

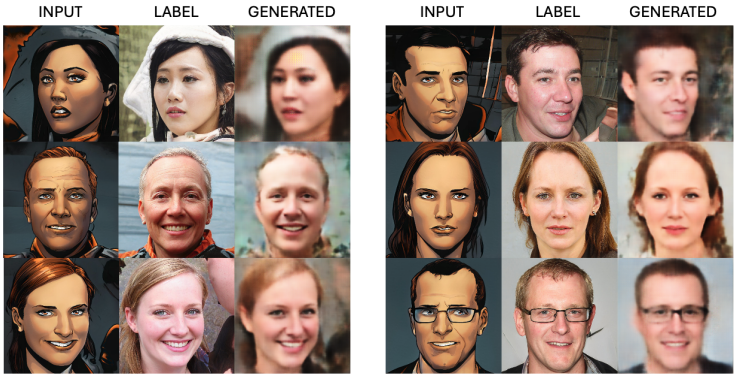

Figure 9: Results for U-Net architecture with Wasserstein Loss function

ResNet6 Architecture

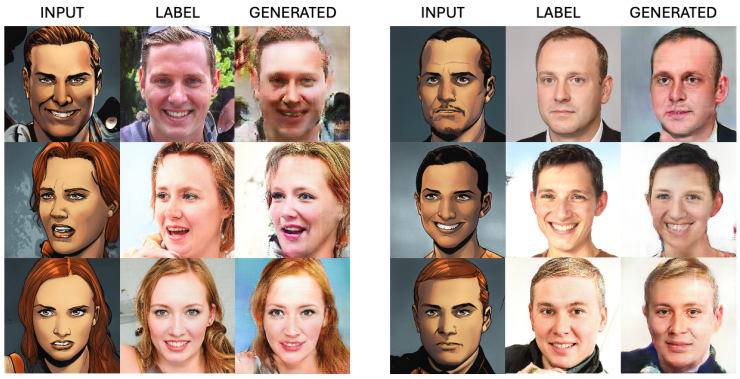

Figure 10: Results for Resnet6 architecture with BCE Loss function

ResNet9 Architecture

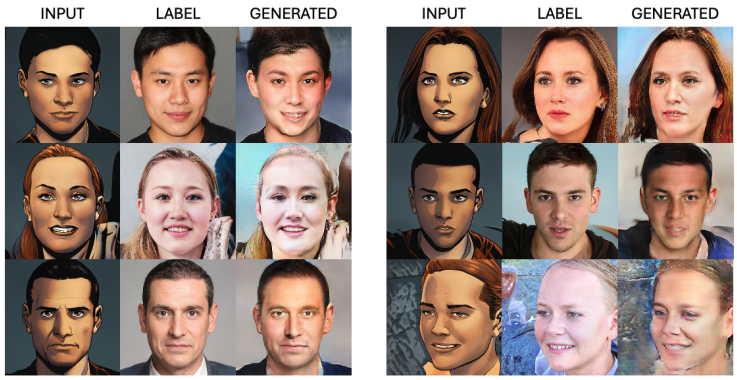

Figure 11: Results for Resnet9 architecture with BCE Loss function

ImageGAN Architecture

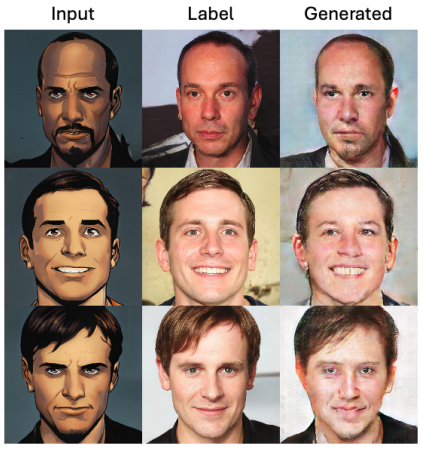

Figure 12: Results for ImageGAN architecture

UNet and PatchGAN Architecture with missing pixel dataset

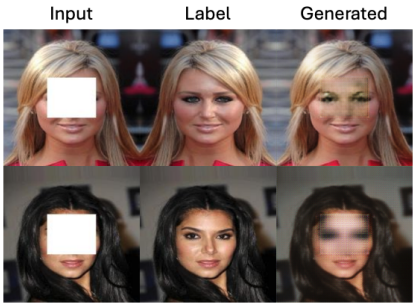

Figure 13: Results for missing pixel images

Discussion

In order to assess the performance of our architectures and loss functions and compare them, we first generated images for the same input image across all (ResNet6, ResNet9, Unet and Least-Square GAN loss, Unet and Wasserstein GAN loss) to better compare them.

As you can see in figure 11, UNet architecture with Wasserstein loss works best when it comes to facial features (eyes, nose, mouth) but is bad when it comes to the hair and background. ResNet9 is best at background and hair but bad with facial features. These results are also reflected in table 1 where between the different loss functions implements, Wasserstein loss function had the lowest losses, and between the different architecture, Unet and ResNet9 had the lowest two losses.

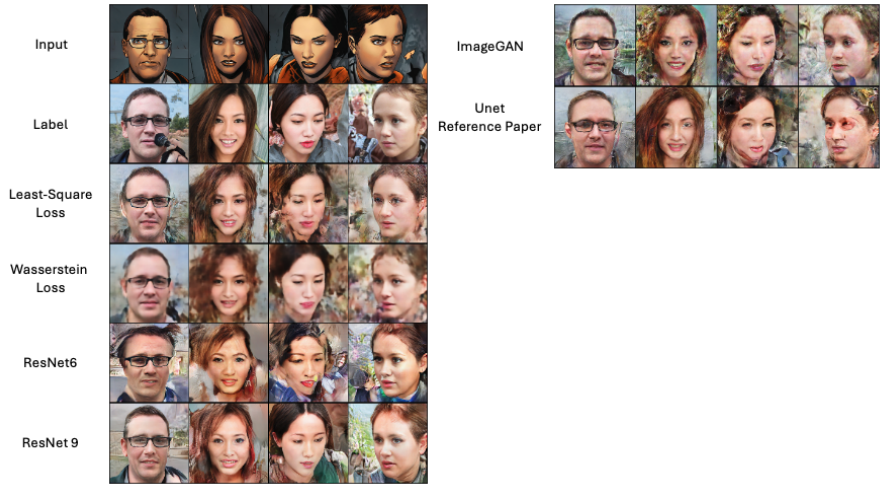

Figure 14: Comparison Result Figure

	G Final Loss	G Median Loss	G Min Loss
UNet	20.8461	19.9804	13.3362
MSE Loss	13.5060	15.2984	10.4875
Wasserstein Loss	10.3886	12.6698	8.6019
ResNet6	24.3850	27.4108	19.8016
ResNet9	21.1343	22.5969	17.8258
ImageGAN	33.2918	34.8159	24.9081

Table 1: Summary of generator performance losses

Implementation

For the coding part, all the different models of generator and discriminator have been written by us from scratch. Moreover, the different training functions for different losses have also been implemented by us after the literature review. The parts for which we referred existing codes is the data pre-processing, saving of examples and checkpoints and loading checkpoints [10].

Contribution

Both of us have equally contributed to the project. Devanshi wrote the codes for the Unet generator and PatchGAN discriminator model. Alexandra wrote the codes for the ResNet Generators and the ImageGAN discriminator model. For the training function, the code used for our baseline U-net training was written by Devanshi and the training function for Wasserstein Loss was modified by Alexandra. The report and analysis of the final results was done by both us together.

References

[1] Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[2] arXiv:1611.04076

[3] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein genera- tive adversarial networks." International conference on machine learning. PMLR, 2017.

[4] Johnson, J., Alahi, A., Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part II 14 (pp. 694-711). Springer International Publishing.

[5] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. arXiv preprint arXiv:1606.03498, 2016

[6] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. ECCV, 2016.

[7] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. ECCV, 2016

[8] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman. Visually indicated sounds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2405–2413, 2016

[9] Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576 (2015)

[10] https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/pix2pix/pix2pix.py