TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A Major Project Final Report
On
Carpet Design with Generative Adversarial Networks

Submitted By:

Sailesh Shiwakoti	(076BCT063)
Suraj Niroula	(076BCT089)
Surya Narayan Chaudhary	(076BCT091)

Submitted To:

Department of Electronics and Computer Engineering
Pulchowk Campus
Lalitpur, Nepal

April, 2024

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A Major Project Final Report
On
Carpet Design with Generative Adversarial Networks

Submitted By:

Sailesh Shiwakoti	(076BCT063)
Suraj Niroula	(076BCT089)
Surya Narayan Chaudhary	(076BCT091)

Submitted To:
Department of Electronics and Computer Engineering
Pulchowk Campus
Lalitpur, Nepal

In partial fulfillment for the award of a Bachelor's degree in Computer Engineering

Under the Supervision of
Asst. Prof. Daya Sagar Baral

April, 2024

‘

Letter Of Approval

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled **'Carpet Design using Generative Adversarial Networks'** submitted by **Sailesh Shiwakoti, Suraj Niroula, and Surya Narayan Chaudhary** in partial fulfillment of the requirements for the Bachelor's Degree in Computer Engineering.

Supervisor: **Asst. Prof. Daya Sagar Baral**
Department of Electronics and Computer Engineering
Institute of Engineering, Pulchowk Campus
Director, Center for Information Technology
Pulchowk Campus, Institute of Engineering

External Examiner: **Asst. Prof. Prakash Poudyal**
Department of Computer Science and Engineering, School of Engineering
Kathmandu University
Dhulikhel, Nepal

DATE OF APPROVAL: APRIL, 2024

Copyright

The authors have agreed that the Library, Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus may make this report freely available for inspection. Moreover, the authors have agreed that permission for extensive copying of this project report for scholarly purpose may be granted by the supervisors who supervised the project work recorded herein or in their absence, by the Head of the Department wherein the project report was done. It is understood that the recognition will be given to the authors of this project and to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering in any use of the material of this report. Copying or publication or the other use of this report for financial gain without approval of the Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus and authors' written permission is strictly prohibited.

Request for permission to copy or to make any use of the material in this project in whole or part should be addressed to department of Electronics and Computer Engineering, IOE, Pulchowk Campus.

Acknowledgments

We would like to express our first and foremost gratitude towards the Department of Electronics and Computer Engineering, Pulchowk Campus for providing us the opportunity to undertake this project which has helped us learn and implement our skills and knowledge.

We extend our heartfelt appreciation to Daya Sagar Baral for his exceptional guidance and supervision throughout the course of our group project. His insightful feedback, constant support, and invaluable expertise have played a pivotal role in shaping the trajectory of our project.

We extend our sincere gratitude to Mr. Sriniwas Acharya, our esteemed mentor, for his invaluable contributions during this time and expertise to conduct insightful meetings with us, delving into the intricacies of our project. His guidance was instrumental in identifying necessary changes and proposing additional features.

We would like to express our deepest appreciation to Dr. Jyoti Tandulkar for his unwavering guidance, continuous supervision, and invaluable suggestions throughout the project. His astute consultations have been instrumental in propelling us to achieve remarkable results.

Finally, we would like to express our gratitude to all individuals directly or indirectly associated with this project, whose support and contributions have been instrumental in our progress.

Sincerely,
Sailesh Shiwakoti (PUL076BCT063)
Suraj Niroula (PUL076BCT089)
Surya Narayan Chaudhary (PUL076BCT091)

Abstract

The project centers around the utilization of Generative Adversarial Networks (GANs) to augment the interior design paradigm. Through the submission of carpet designs drawn using symmetry groups by users, the system leverages the generative potential of GANs to iteratively refine and materialize these concepts with heightened efficiency. Then the image acts as input for a neural style transfer model which uses pretrained VGGNet19 for edge detection and feature abstraction. The generated stylized image is provided to the user, who upon confirmation gives the size, which is then colorized appropriately using Pix2Pix, upscaled using seam carving, quantized using octree quantization, and then converted into a grid where each cell represents the color of yarn to use during manufacturing along with a color palette.

Keywords: Generative Adversarial Networks (GANs), Interior Design, Neural Style Transfer, VGGNet19, Pix2Pix, Seam Carving, Octree Quantization.

Contents

Letter Of Approval iii
Copyright iv

Abstract **vi**

Contents **ix**

List of Figures **xi**

List of Abbreviations **xii**

1 Introduction **1**
1.1 Background 1
1.2 Problem Statements 2
1.3 Objectives 2
1.4 Scope 3
1.5 Application 3

2 Literature Review **4**
2.1 Generative Adversarial Networks for Image Generation 4
2.2 Deep Convolutional Neural Networks for Image Classification 4
2.3 Unsupervised Representation Learning with DCGAN 4
2.4 U-Net: Convolutional Networks for Biomedical Image Segmentation 5
2.5 Image-to-Image Translation with Conditional Adversarial Networks 5
2.6 StyleGAN 5
2.7 Neural Style Transfer 5
2.8 Symmetry and Group Theory 6
2.9 Software solutions 6

3 Related Theory **7**
3.1 Generative Adversarial Networks (GANs) 7
3.2 DCGAN: Deep Convolutional Generative Adversarial Network 8
3.2.1 Architecture and Components 8
3.2.2 Loss Function 8

3.3 Convolutional Neural Networks (CNNs) 9
3.4 U-Net 9
3.5 Pix2Pix 10
3.6 VGGNet 11
3.7 Neural Style Transfer 12
3.8 Symmetry Groups 13

4 System Design **16**
4.1 Feasibility Study 16
4.2 Requirement Analysis 16
4.2.1 Functional Requirements 17
4.2.2 Non-Functional Requirements 17
4.3 System Architecture 18

5 Methodology **21**
5.1 Data Collection and Preprocessing 21
5.2 GAN Model Development 22
5.3 Dataset Augmentation 22
5.4 Dataset Preparation for pix2pix Model 22
5.5 Dataset Augmentation 23
5.6 Improving Sketch Realism 25
5.7 Fourier Transformations 25
5.8 Hough Transform 27

6 Experimental Setup **29**
6.1 Data Collection 29
6.2 Dataset Augmentation 30
6.3 Results 30
6.3.1 Hardware Configuration 31
6.3.2 Software Environment 31
6.4 Training Parameters 32
6.4.1 Two Time-Scale Update Rule 32
6.4.2 Vanishing Gradient 33
6.5 Image Quantization 33
6.6 Seam Carving for Image Resizing 34

7 System Design **40**
7.1 DCGAN Architecture 40
7.2 VGG network 43
7.3 System Architecture 44

8 Project Results **47**
8.1 Output from GAN 48
8.2 Output from pix2pix model 48
8.3 Output from VGG19 model 49
8.4 User Interface and User Interaction 52
8.5 Neural Style Transfer 53

9 Future Steps **55**

10 Discussion **57**

11 Conclusion **60**
References 60

List of Figures

3.1 Rotation of a square by 90 degrees. 13
3.2 Patterns with symmetry groups D_7 and R_{12}. 14
3.3 Pattern with horizontal translation symmetry. 14
3.4 Pattern without frieze symmetry. 15
3.5 Examples of patterns with different wallpaper groups. 15

4.1 Sequence Diagram 19

5.1 Color channels varying during dataset augmentation 24
5.2 Making sketches from real rug images 25
5.3 Original Carpet Image and Segmentation with Highlights 26
5.5 Outputs from Hough Transform and Probabilistic Hough Transform to detect edges and subsequently pattern in carpet 28

6.1 Segmentation of images during feature extraction 31
6.2 Separating components of rug during feature engineering 32
6.3 Some rugs' features are not captured in segments 32
6.7 Comparison of different color quantization methods. 34
6.8 Example from vectorizer.ai output showing the difference between vectorized art, contains lesser more representative colors 34
6.9 Resized images to specified dimensions. 36
6.4 K-Means Clustering with K=3 37
6.5 K-Means Clustering with K=20, more clusters can result noisy patterns . . . 38
6.6 Visualization of various patterns, where highlighted portion indicates the feature of the rug. 39

7.1 Generative Adversarial Network Model 40
7.2 U-Net as Generator 41
7.3 Training procedure for the generator 42
7.4 Training procedure for the discriminator 43
7.5 VGG-19 Architecture 44
7.6 Sequence Diagram 45

8.1 Carpet generated when random input is provided to GAN 48
8.2 Carpet colorized by pix2pix model 49
8.3 Carpet Design Steps in system. (e) and (f) and Neural Style Transfer enhanced designs by taking input as the final design from user (d) 50
8.4 User Generated Designs using Symmetry Groups 50
8.5 Enhanced Images using GAN by taking 8.4 as input 51
8.6 Different Symmetry Groups in ReactJS drawn by user 52
8.7 Combined Design 52
8.8 Images returned from Django API 53
8.9 Comparison of original and quantized images, along with the color palette . 53
8.10 High-frequency artifacts as seen from edge detection algorithm(Sobel Edge Detection) 54
8.11 Comparison of the style transfer output before and after introducing variation loss (trained upto 1000 steps) 54

List of Abbreviations

GAN	Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
PGAN	Progressive Growing of GANs
CycleGAN	Cycle-Consistent Generative Adversarial Network
VGG Net	Visual Geometry Group Network

1. Introduction

1.1 Background

The artistry of rug and carpet design stands as a pivotal factor in elevating the aesthetic allure and holistic ambiance of interior spaces. However, conventional methodologies in rug design are occasionally hindered by limitations in customization options and the time-intensive nature of the creative process. The paradigm shift introduced by leveraging the formidable capabilities of Generative Adversarial Networks (GANs) holds the promise of revolutionizing the rug design landscape, ushering in an era of unparalleled personalization and creativity.

Generative Adversarial Networks, or GANs, are cutting-edge artificial intelligence models that engage in a dynamic interplay between a generator and a discriminator. This intricate dance of algorithms results in the creation of entirely novel and contextually relevant designs. By harnessing the power of GANs, rug designers can break free from the constraints of traditional design methods, opening the door to an infinite spectrum of possibilities.

The integration of GANs into the rug design process is poised to democratize creativity, enabling users to actively participate in the conception of unique and personalized rug designs. Through intuitive interfaces, users can seamlessly navigate and manipulate design parameters, tailoring their rugs to align with specific preferences, color schemes, and spatial requirements. This newfound level of customization empowers individuals to curate designs that not only harmonize with their interior aesthetics but also reflect their personal style, fostering a deeper connection between the individual and their living spaces.

To further enrich the user experience, the incorporation of drag-and-drop features becomes a pivotal aspect of this innovative approach. By implementing user-friendly interfaces that facilitate effortless manipulation of design elements, individuals can intuitively craft rugs that resonate with their vision. This interactive and accessible design process transcends the conventional boundaries of artistic expertise, making rug design a collaborative and enjoyable endeavor for individuals of varying design proficiency.

Moreover, the infusion of user-provided images as references for rug suggestions adds another layer of user experience. This feature transcends the traditional confines of design inspiration, allowing users to draw from their own life experiences, memories, or even specific themes. Whether it be a photograph, a piece of artwork, or an image capturing a cherished moment, these visual cues serve as a foundation for GANs to generate rug designs that

embody the essence of the user's unique narrative.

1.2 Problem Statements

1. Limited Customization: Traditional rug design methods offer limited options for customization, making it difficult for individuals to find rug designs that align with their unique preferences and interior design concepts.

2. Time-Consuming Process: Designing a rug from scratch can be time-consuming and may require expert knowledge, posing a challenge for individuals who want to quickly visualize rug options for their spaces.

3. Lack of Visual Guidance: Clients often struggle to envision how a specific rug design will complement their existing interior decor, leading to uncertainty and decision-making challenges.

1.3 Objectives

The primary objectives of this project is:

1. Develop a Generative Adversarial Network (GAN)-based model showcasing the ability to generate a diverse array of visually captivating and personalized rug designs. This model will cater to varying user preferences and interior styles by incorporating an input template design provided by the user.

The secondary objectives of this project are:

1. Implement a sophisticated image-based generation system that empowers users to upload images of their interior spaces. This system will intelligently analyze these images and generate personalized rug design suggestions that seamlessly integrate with the existing decor and layout.

2. Create an intuitive and interactive platform, equipped with user-friendly drag-and-drop features and an extensive palette of color and pattern options, enabling users to seamlessly customize rug designs to match their unique aesthetic preferences.

3. Elevate the interior design experience by providing users with immersive, realistic visualizations of the generated rug designs within their own spaces.

1.4 Scope

The project endeavors to revolutionize the domain of rug and carpet design through the innovative integration of Generative Adversarial Networks (GANs). Our ambitious scope aims to transcend the constraints of traditional rug design methodologies by offering an expansive array of customization options while streamlining the typically time-consuming creative process. By harnessing the transformative power of GANs, our project seeks to democratize creativity within the realm of rug design, empowering users to explore limitless design possibilities and achieve unprecedented levels of personalization.

Furthermore, our project encompasses the development of an interactive carpet design platform empowered by GAN networks. This platform will provide users with a comprehensive selection of design categories, intuitive design tools, and efficient workflows, effectively reducing design complexities and time investment. Additionally, our platform will offer a curated palette of quantized colors, ensuring coherence and harmony in design compositions while facilitating seamless integration into diverse interior settings.

1.5 Application

The proposed solution involves integrating GANs into the rug design process to provide users with a dynamic and interactive platform. This application empowers individuals to actively participate in the conception of unique and personalized rug designs. Through intuitive interfaces, users can navigate and manipulate design parameters, tailoring their rugs to align with specific preferences, color schemes, and spatial requirements. The project also incorporates drag-and-drop features, making rug design accessible to individuals with varying levels of design proficiency. Additionally, the integration of user-provided images as references adds a layer of user experience, allowing GANs to generate rug designs that reflect the essence of the user's unique narrative, drawing inspiration from their life experiences and visual cues.

2. Literature Review

2.1 Generative Adversarial Networks for Image Generation

The work by Goodfellow et al. [1] on "Generative Adversarial Networks" has substantial relevance to our rug generation project. Their pioneering research introduced the concept of GANs, which have found widespread use in image generation tasks. By leveraging the adversarial training process, GANs demonstrate remarkable capabilities in producing realistic and diverse images. Our project draws inspiration from Goodfellow et al. principles and techniques to develop a GAN-based model for generating visually appealing and personalized rug designs.

2.2 Deep Convolutional Neural Networks for Image Classification

The paper by Krizhevsky et al. [2] titled "ImageNet Classification with Deep Convolutional Neural Networks" may not directly relate to rug design, but it holds relevance. This work primarily focuses on image classification using deep convolutional neural networks (CNNs), a fundamental component in various computer vision tasks, including rug design. CNNs facilitate the extraction of meaningful features and patterns from images, which can be instrumental in generating and recommending rug designs.

2.3 Unsupervised Representation Learning with DCGAN

In the realm of rug design, the work by Radford et al. on unsupervised representation learning with Deep Convolutional Generative Adversarial Networks (DCGAN) is particularly relevant [3]. DCGANs have proven successful in generating realistic images by training deep convolutional neural networks in an adversarial framework. Applying DCGANs to rug design can enable the creation of diverse, visually appealing rug designs that cater to various interior styles and preferences.

2.4 U-Net: Convolutional Networks for Biomedical Image Segmentation

The U-Net architecture, proposed by Ronneberger et al.[4] in 2015, revolutionized biomedical image segmentation. Recognizing the challenges in segmenting structures of varying scales, U-Net introduces a U-shaped network with a contracting path to capture context and an expansive path for precise localization. This design, accompanied by skip connections, enables the model to seamlessly integrate low and high-level features. U-Net's application extends beyond biomedical fields, finding use in diverse image segmentation tasks due to its ability to handle limited data and generate accurate pixel-wise predictions efficiently.

2.5 Image-to-Image Translation with Conditional Adversarial Networks

Isola et al.[5] 2018 work on pix2pix addresses the broader challenge of image-to-image translation. Pix2pix employs Conditional Generative Adversarial Networks (cGANs) to learn mappings between input and output images. By introducing adversarial networks, it successfully transforms labels, sketches, or maps into realistic images. Pix2pix's versatility is evident in applications such as photo colorization, style transfer, and generating real-world images from synthetic representations. The model's architecture, involving a U-Net generator and a PatchGAN discriminator, has inspired subsequent advancements in conditional image synthesis.

2.6 StyleGAN

The work by Karras, Laine, and Aila on StyleGAN[6] presents a novel generator architecture for generative adversarial networks (GANs), borrowing from style transfer literature. This architecture enables automatic separation of high-level attributes and stochastic variation in generated images, facilitating intuitive control of synthesis. It improves distribution quality metrics, interpolation properties, and latent factor disentanglement. Their proposed automated methods for quantifying interpolation quality and disentanglement are applicable across various architectures. Additionally, they introduce a diverse dataset of human faces, enhancing research resources in this field.

2.7 Neural Style Transfer

In their seminal work, Gatys, Ecker, and Bethge [7] present a neural algorithm for artistic style transfer. This innovative approach allows for the separation and recombination of content and style in images, shedding light on the intricate processes underlying human

artistic creation. Specifically, a specific block and its corresponding convolutional layers are selected for the content image, while more than one convolutional layer is used for the style image. The content of an image is represented by the values of the intermediate feature maps, facilitating the synthesis of new images that combine the content of one image with the style of another.

2.8 Symmetry and Group Theory

The work by David Eck[8] delves into the exploration of frieze, rosette, and wallpaper symmetry, examining the concepts of translation and rotation operations on objects within these symmetrical systems. Symmetry within these systems is identified by letter combinations, such as "p11", while rosettes are characterized by the count of rotations that divide the 360-degree space to replicate components. Understanding these symmetrical properties is essential for analyzing and creating visually pleasing designs, providing valuable insights into the principles governing symmetry in various contexts.

2.9 Software solutions

Various software tools exist to facilitate rug and carpet design, offering customization, visualization, and design assets to creators.

Galaincha is a robust custom rug designing software tailored for carpet designing, rendering simulations, and preprocessing, featuring a user-friendly interface [9]. Explorug provides customers the ability to personalize rugs, visualize them in their intended room settings, and place orders directly [10]. Ramro Design simplifies the rug design process by generating royalty-free, color-separated, and ready-to-use rug designs [11].

Other design software options include Only1DollarDesign, which aims to democratize design accessibility across various industries [12], and OnlyHundred, which offers a treasure trove of unique designs to inspire creators across different fields [13].

3. Related Theory

3.1 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) consist of a generator network (G) and a discriminator network (D) that are trained in an adversarial manner. The objective of the generator is to produce synthetic data that closely resembles real data, while the discriminator's goal is to distinguish between real and generated data.

The generator takes a random noise vector (z) as input and maps it to a high-dimensional space to generate synthetic samples ($G(z)$). The discriminator, on the other hand, receives both real data (x) and generated data ($G(z)$) and aims to classify them correctly as real or fake. The outputs of the discriminator can be interpreted as the probability of the input being real ($D(x)$) or generated ($D(G(z))$).

The training process of GANs involves a min-max game between the generator and discriminator. The generator aims to generate samples that are indistinguishable from real data, while the discriminator aims to correctly classify real and generated samples. The objective function of the GAN can be defined as:

$$\min_G \max_D V(D, G) = E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

where $p_{\text{data}}(x)$ represents the distribution of real data and $p_z(z)$ represents the distribution of the input noise vector.

In this formulation, the discriminator aims to maximize $V(D, G)$ by correctly classifying real and generated samples, while the generator aims to minimize $V(D, G)$ by generating samples that the discriminator cannot distinguish as fake.

The training of GANs involves iteratively updating the generator and discriminator networks. In each iteration, the discriminator is trained first by minimizing the discriminator's loss function:

$$\mathcal{L}_{\text{D}} = -\left(E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]\right)$$

This loss encourages the discriminator to correctly classify real and generated samples.

Next, the generator is trained by minimizing the generator's loss function:

$$\mathcal{L}_{\text{G}} = -E_{z \sim p_z(z)}[\log(D(G(z)))]$$

This loss encourages the generator to produce samples that are classified as real by the discriminator.

The training process continues by iteratively updating the discriminator and generator using their respective loss functions until convergence.

3.2 DCGAN: Deep Convolutional Generative Adversarial Network

Deep Convolutional Generative Adversarial Networks (DCGANs) are a class of generative models designed for image synthesis tasks. They combine deep convolutional neural networks (CNNs) with the GAN framework to generate high-quality and realistic images.

3.2.1 Architecture and Components

A DCGAN consists of two main components: the generator and the discriminator. The generator's role is to create images from random noise, while the discriminator aims to distinguish between real and generated images.

The generator takes a random noise vector $\mathbf{z}$ as input and produces an image $G(\mathbf{z})$. It typically consists of transposed convolutional layers (also known as deconvolutional layers) to upscale the noise vector into a higher-dimensional space that resembles the image space.

The discriminator, on the other hand, is a CNN that takes an image x as input and produces a probability $D(x)$ that the input image is real. It involves a series of convolutional layers followed by fully connected layers to perform the classification.

3.2.2 Loss Function

The objective of a DCGAN is to train both the generator and discriminator simultaneously through a two-player minimax game. The discriminator's task is to maximize its ability to differentiate between real and generated images, while the generator's task is to minimize the discriminator's ability to do so.

The loss function for the discriminator D can be formulated as the binary cross-entropy between the true labels (1 for real images and 0 for generated images) and the discriminator's predictions:

$$L_D = -E_{x\sim p_{\text{data}}(x)}[\log D(x)] - E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Conversely, the generator's loss function is aimed at minimizing the probability that the discriminator correctly classifies the generated images as fake. This encourages the generator to produce images that are increasingly similar to real images:

$$L_G = -E_{z\sim p_z(z)}[\log D(G(z))]$$

3.3 Convolutional Neural Networks (CNNs)

‘ Convolutional Neural Networks (CNNs) are a class of deep neural networks specifically designed for processing grid-structured data like images. They leverage convolutional layers to automatically learn hierarchical features from the input data.

3.4 U-Net

U-Net is a convolutional neural network architecture designed for semantic segmentation tasks. It consists of an encoding path followed by a decoding path to capture context and achieve precise localization.

The U-Net architecture comprises the following components:

Contracting Path (Encoder)

The contracting path consists of convolutional and max-pooling layers to capture context and reduce spatial dimensions.

$$\begin{aligned}&\text{ConvBlock}(x, filters) = \\ &\quad \text{Conv}(x, filters, 3 \times 3) \rightarrow \text{ReLU} \rightarrow \text{Conv}(x, filters, 3 \times 3) \rightarrow \text{ReLU}\end{aligned} \tag{3.1}$$

$$\begin{aligned}&\text{ContractingBlock}(x, filters) = \\ &\quad \text{ConvBlock}(x, filters) \rightarrow \text{MaxPooling}(2 \times 2)\end{aligned} \tag{3.2}$$

Bottleneck

The bottleneck connects the contracting and expanding paths, capturing essential features.

$$\begin{aligned}&\text{Bottleneck}(x, filters) = \\ &\quad \text{ConvBlock}(x, filters) \rightarrow \text{Dropout}(0.5)\end{aligned} \tag{3.3}$$

Expanding Path (Decoder)

The expanding path consists of up-sampling and concatenation operations to recover spatial information.

$$\begin{aligned}&\text{UpConvBlock}(x, filters) = \\ &\quad \text{Conv}(x, filters, 2 \times 2) \rightarrow \text{ReLU} \rightarrow \text{Conv}(x, filters, 3 \times 3) \rightarrow \text{ReLU}\end{aligned} \tag{3.4}$$

$$\begin{aligned}&\text{ExpandingBlock}(x, filters) = \\ &\quad \text{UpConvBlock}(x, filters) \rightarrow \text{Concatenation} \rightarrow \text{ConvBlock}(x, filters)\end{aligned} \tag{3.5}$$

Output Layer

The output layer produces the final segmentation map.

$$\begin{aligned} &\text{OutputLayer}(x, \text{num_classes}) = \\ &\quad \text{Conv}(x, \text{num_classes}, 1 \times 1) \rightarrow \text{Softmax} \end{aligned} \tag{3.6}$$

Unlike traditional networks, U-Net avoids fully connected layers and relies solely on the valid part of each convolution. To handle large images, an overlap-tile strategy is employed during segmentation, extrapolating missing context for border pixels by mirroring the input image. This strategy is crucial to overcome GPU memory limitations and enables seamless segmentation of arbitrarily large images.

Given limited training data in biomedical tasks, U-Net uses extensive data augmentation, including elastic deformations. This allows the network to learn invariance to deformations without encountering them in the annotated image corpus, a crucial aspect in biomedical segmentation where realistic deformations are common.

3.5 Pix2Pix

Pix2Pix, introduced by Isola et al.[5], is a conditional generative adversarial network (cGAN) designed for image-to-image translation tasks. Operating with a paired dataset, the model learns a mapping from input to output images using adversarial and task-specific losses.

Pix2Pix is a conditional generative adversarial network designed for image-to-image translation tasks, where it learns a mapping from an input image to an output image.

Objective Function

The objective function of Pix2Pix consists of an adversarial loss and a pixel-wise L1 loss.

Adversarial Loss

The adversarial loss encourages the generator G to produce images indistinguishable from real images, and the discriminator D to differentiate between real and generated images.

$$\begin{aligned} \mathcal{L}_{\text{GAN}}(G, D) = E_{x,y}[\log D(x, y)] \\ + E_x[\log(1 - D(x, G(x)))] \end{aligned} \tag{3.7}$$

Pixel-wise L1 Loss

The pixel-wise L1 loss enforces a similarity between the generated image and the ground truth.

$$\mathcal{L}_{\text{L1}}(G) = E_{x,y,z}[||y - G(x, z)||_1] \tag{3.8}$$

Combined Objective

The combined objective is a weighted sum of the adversarial and pixel-wise L1 losses.

$$\mathcal{L}_{\text{Pix2Pix}}(G, D) = \mathcal{L}_{\text{GAN}}(G, D) + \lambda \mathcal{L}_{\text{L1}}(G) \tag{3.9}$$

where λ is a hyperparameter controlling the importance of the pixel-wise L1 loss.

Training involves iterative updates of the generator and discriminator to find a Nash equilibrium. Pix2Pix has shown success in various image-to-image translation tasks, including style transfer and sketch-to-image generation.

3.6 VGGNet

VGGNet, proposed by Simonyan and Zisserman in their work [14], is a convolutional neural network architecture designed for large-scale image recognition tasks. The main focus of their investigation was to explore the impact of the network's depth on its accuracy in the context of image recognition.

The VGGNet architecture consists of the following components:

Convolutional Blocks

The basic building block is a series of convolutional layers with small 3x3 filters and Rectified Linear Unit (ReLU) activation functions. This design choice allows the network to capture complex patterns while keeping the number of parameters relatively low.

$$\begin{aligned} &\text{ConvBlock}(x, filters) = \\ &\quad \text{Conv}(x, filters, 3 \times 3) \rightarrow \text{ReLU} \end{aligned} \tag{3.10}$$

Stacking Blocks

VGGNet stacks multiple ConvBlocks to create deeper representations. By repeating ConvBlocks, the network can learn hierarchical features of increasing complexity.

$$\begin{aligned} &\text{Stack}(x, num_layers, filters) = \\ &\quad \text{ConvBlock}(\ldots \text{ConvBlock}(x, filters) \ldots) \quad (\text{repeated } num_layers \text{ times}) \end{aligned} \tag{3.11}$$

Pooling

Max-pooling layers are used to reduce spatial dimensions and extract the most salient features from the input data.

$$\begin{aligned}&\text{MaxPooling}(x) = \\ &\quad \text{MaxPool}(x, 2 \times 2)\end{aligned} \tag{3.12}$$

Fully Connected Layers

The network ends with fully connected layers for classification. These layers process the extracted features and output the final predictions.

$$\begin{aligned}&\text{FullyConnected}(x, units) = \\ &\quad \text{FC}(x, units) \rightarrow \text{ReLU} \quad \text{(for hidden layers)} \\ &\quad \text{FC}(x, \text{num_classes}) \rightarrow \text{Softmax} \quad \text{(for the output layer)}\end{aligned} \tag{3.13}$$

VGGNet architecture is commonly utilized for transfer learning in various image recognition tasks due to its simplicity and effectiveness.

3.7 Neural Style Transfer

Neural style transfer[7] is an optimization technique used to take two images—a content image and a style reference image—and blend them together so the output image looks like the content image, but "painted" in the style of the style reference image.

Loss Function

NST aims to generate an image G that combines the content of a content image C and the style of a style image S using a loss function:

$$L(G, C, S) = \alpha L_{\text{content}}(G, C) + \beta L_{\text{style}}(G, S) \tag{3.14}$$

Content Loss

Measures the difference in content between G and C:

$$L_{\text{content}}(G, C) = \frac{1}{2} \sum_{i,j} (G_{ij} - C_{ij})^2 \tag{3.15}$$

Style Loss

Captures differences in style using Gram matrices:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \tag{3.16}$$

$$L_{\text{style}}(G, S) = \sum_l \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - S_{ij}^l)^2 \tag{3.17}$$

3.8 Symmetry Groups

As described by David Eck in his work on symmetry[8], a "symmetry" of a pattern is a transformation, or motion, for which the pattern is the same before and after the transformation. As a simple example, if a square is rotated by 90 degrees around its center, the square looks the same before and after the rotation. On the other hand, if it is rotated by 45 degrees, one can see that the square has been tilted.

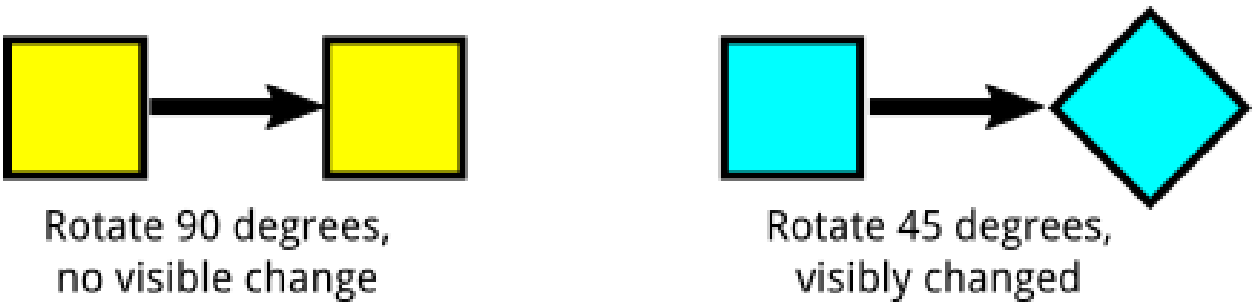


Figure 3.1: Rotation of a square by 90 degrees.

A square has other symmetries. It can be rotated 180 degrees or 270 degrees, and it can be reflected vertically, horizontally, and through either of its diagonals.

The set of all symmetry transformations of a pattern form what is called the "symmetry group" for that pattern. Different patterns have different symmetry groups. The symmetry group of a square is named D_4.

For some patterns in the plane, there is a point at the center of the pattern that is left in place by every symmetry. The possible symmetry operations for such a pattern are rotations about the center of the pattern and reflections about a line that passes through the center. The pattern on the left below has 7 rotation symmetries (including the identity transformation) and 7 reflection symmetries. Its symmetry group is called D_7. The pattern on the right has 12 rotation symmetries (including the identity) and no reflection symmetries. Its symmetry group is called R_{12}.

We can define rotation groups with any number of rotations. For example, the rotation group R_{137} is the symmetry group of a pattern that has 137 rotation symmetries and no reflection symmetries. The groups that we get when we add reflection symmetry are called the dihedral groups. For example, the dihedral group D_{137} is the symmetry group of a pattern that has 137 rotation symmetries and 137 reflection symmetries. The rotation and dihedral groups are sometimes referred to collectively as rosette groups. The web app rosette.html

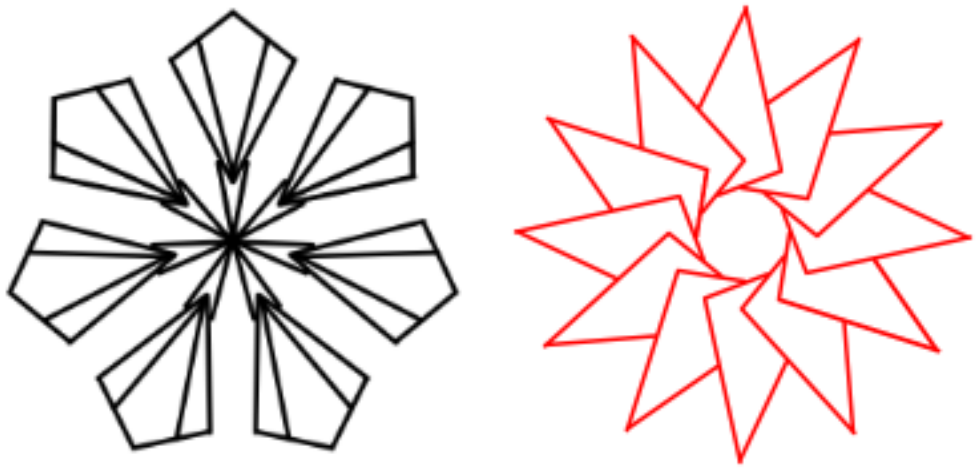


Figure 3.2: Patterns with symmetry groups D_7 and R_{12}.

lets you draw patterns with symmetry group R_n or D_n for n up to 20. There is a set of radio buttons that allows you to select the number of rotation symmetries, and there is a checkbox that adds reflection symmetries. When you draw something in the white area, all the symmetric transformations of the thing that you draw are also shown.

If we drop the requirement that a pattern should have a center point that is left in place by all symmetry operations, then we get new kinds of symmetry. In particular, we get translation symmetries. A translation operation moves everything by a certain distance in a certain direction. If a pattern has one translation symmetry, it automatically has an infinite number of them. For example, if moving everything one inch to the right is a symmetry of a pattern, then moving two inches to the right, or three inches, or four inches are also symmetries. A pattern that has a translation symmetry is necessarily infinite. Here is an example of a pattern that has a horizontal translation symmetry.

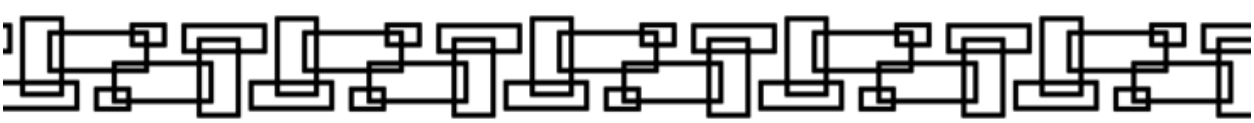


Figure 3.3: Pattern with horizontal translation symmetry.

A frieze group includes translations symmetries in one direction (but not in a second independent direction). Furthermore, the group is "discrete" in the sense that there is a minimum translation distance that is a symmetry. For example, this picture has arbitrarily small horizontal translation symmetries, so its symmetry group is not a frieze group:

With this restriction, there are only seven different frieze groups. They have names like $p1m1$ and $pma2$ (though other names are also used). In the program, there is a set of radio buttons for selecting the frieze group that you want to use.

Figure 3.4: Pattern without frieze symmetry.

The simplest frieze group, $p111$, has translation symmetry and no other symmetry. The other groups add additional symmetries. For example, $p112$ adds a 180-degree rotation symmetry. (180 degrees is the only possible rotation symmetry in a frieze group.) $p1m1$ adds a reflection through a vertical line, while $pm11$ adds reflection through a horizontal line, and $pmm2$ adds both. (An "m" stands for "mirror" and indicates a reflection symmetry.) The groups $p1a1$ and $pma2$ add another kind of symmetry called a glide reflection. A glide reflection consists of a translation followed by a reflection over a line that is parallel to the direction of translation.

If we add translation symmetry in a second, independent direction, we get wallpaper groups. It turns out that there are only 17 different wallpaper groups (again, considering only discrete groups). Here are three examples made using the groups $p6m$, pgg, and $p4m$.

Figure 3.5: Examples of patterns with different wallpaper groups.

If a wallpaper group has any rotational symmetries, then the smallest rotational symmetry must be one of 180, 120, 90, or 60 degrees. A wallpaper group can also have reflection symmetries and glide reflection symmetries. An "m" in the group name indicates a reflection symmetry. A "g" indicates a glide reflection symmetry.

4. System Design

4.1 Feasibility Study

- Market Potential: The global rug industry enjoys a significant market, with countries like India, Iran, Pakistan, and Nepal being major exporters. Rugs hold cultural and historical significance, with auctions featuring carpets used by royalty. Additionally, it is common for individuals to have at least one rug or carpet in their homes, highlighting the demand for diverse design options.

- Challenges in Rug Selection: The process of selecting the perfect rug can be cumbersome and time-consuming. Individuals often find it difficult to visualize how a particular rug will fit into their room decor, making decision-making a daunting task. Our project aims to address this challenge by providing a platform that simplifies the design process and allows users to easily explore different rug options.

- Technical Feasibility: The study confirmed the technical feasibility of our project, considering the availability of advanced GAN algorithms and existing interior design datasets. Leveraging these resources, we can generate personalized and visually appealing rug designs that cater to individual preferences.

- Economic Viability: An economic analysis revealed the financial viability of our project within the allocated budget. By streamlining the rug selection process and providing an interactive platform, we anticipate a high user adoption rate. This, in turn, creates opportunities for revenue generation through user subscriptions, partnerships with rug manufacturers, and potential expansion into related interior design services.

- User Convenience: Our system aims to save users' time and effort by offering a user-friendly interface that allows them to visualize different rug options in their rooms. By providing this convenience, we expect a large user base to embrace our platform, making it a practical solution for rug selection and interior design.

4.2 Requirement Analysis

The requirement analysis phase aimed to identify the essential and desirable features of the rug design system. This process involved understanding user needs, studying industry

standards, and considering technical constraints. The identified requirements serve as a roadmap for the development of the system.

4.2.1 Functional Requirements

- User Registration and Authentication: Users should be able to create accounts, log in, and maintain their profile information.
- Carpet Design Generation: The system should generate diverse rug designs based on user preferences, including color, pattern, and style.
- Customization Options: Users should be able to customize rug designs by adjusting colors, patterns, and sizes.
- Design Recommendation: The system should provide personalized rug design recommendations based on user preferences and uploaded images.
- Rug Catalog Management: The system should maintain a catalog of available rug designs, including their specifications, prices, and availability.
- Social Sharing: Users should have the option to share their rug designs and room layouts on social media platforms.
- Administration and Content Management: The system should provide an administrative interface to manage user accounts, rug designs, and other content.

4.2.2 Non-Functional Requirements

- Usability: The system should have an intuitive and user-friendly interface to ensure ease of use for users with varying technical backgrounds.
- Performance: The system should be able to handle multiple user requests simultaneously and provide quick response times for generating rug designs and rendering room layouts.
- Scalability: The system should be designed to accommodate a growing user base and handle increasing data storage and processing requirements.
- Security: User data, including personal information and uploaded images, should be securely stored and protected from unauthorized access.
- Compatibility: The system should be compatible with different web browsers and device platforms to ensure a seamless user experience across various devices.

- Reliability: The system should be reliable and available for users at all times, minimizing downtime and disruptions.
- Maintainability: The system should be designed and implemented in a modular and extensible manner, allowing for easy maintenance and future enhancements.

4.3 System Architecture

The system is an interactive application that enables users to create custom carpet designs. The system consists of a frontend developed using ReactJS, a backend implemented in Django, and utilizes machine learning models such as StyleGAN and Pix2Pix for image generation and colorization.

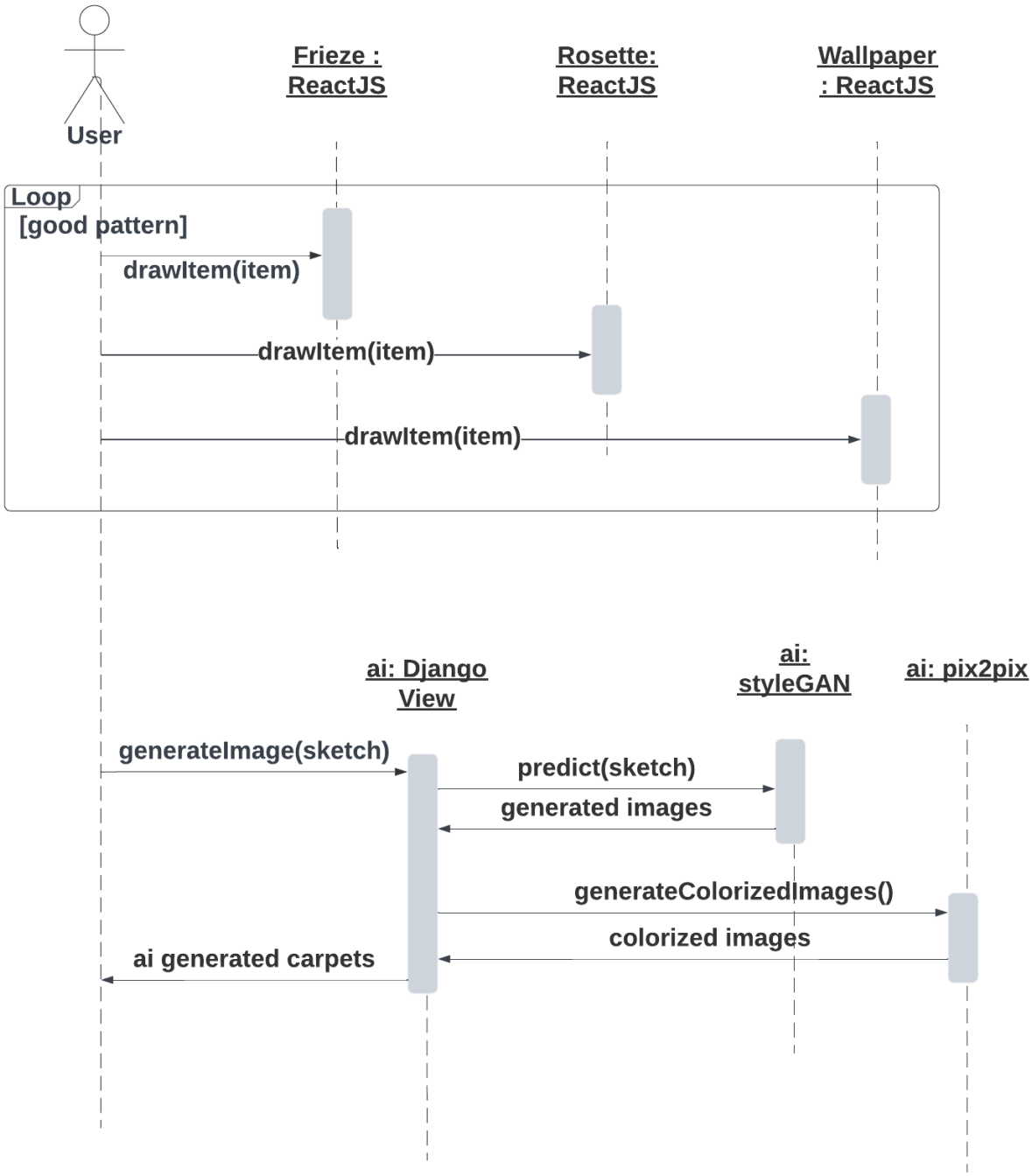


Figure 4.1: Sequence Diagram

1. **Drawing Carpet Image:** The *User* interacts with the *ReactJS Frontend* to draw a carpet image using a set of drawing tools. During this process, the user has the option to choose from different patterns, including frieze, rosette, or wallpaper patterns.

2. **Pattern Parameter Selection:** Once the user completes the drawing, they can select various pattern parameters such as symmetry, color, line width, and motifs (e.g., circle,

rectangle, line, hand-drawn elements) using the frontend interface.

3. **Backend Processing:** The *Frontend* sends the user's input, including the drawn image and selected parameters, to the *Django Backend*.

4. **Image Generation using Neural Style Network:** The *Backend* instantiates the *StyleGAN* model, a generative adversarial network (GAN), to predict a high-resolution carpet image based on the user's drawing and selected parameters. The architecture of StyleGAN is complex, incorporating multiple layers for feature extraction and generation.

5. **Pix2Pix Colorization:** The predicted image may have undesirable color characteristics. To enhance the visual appeal, the *Backend* passes the generated image to *Pix2Pix*, a conditional GAN designed for image-to-image translation. Pix2Pix colorizes the image to some extent, providing a more refined output.

6. **Result Presentation:** The *Backend* returns the processed images, including the StyleGAN-generated and Pix2Pix-colorized versions, along with color palette, to the *Frontend*. The *Frontend* then displays these images to the *User* for evaluation and selection.

5. Methodology

The methodology employed in this project focuses on harnessing the power of Generative Adversarial Networks (GANs) to revolutionize the rug design paradigm. GANs offer a promising avenue for generating personalized and visually captivating rug designs based on user preferences. The methodology encompasses data collection, preprocessing, GAN model development, and dataset augmentation to enhance the interior design experience.

5.1 Data Collection and Preprocessing

The first step involved gathering a diverse collection of rug design images from various sources. Pinterest, renowned for its rich visual content, was tapped using a web crawling approach. The Selenium web driver was employed to visit Pinterest pages, simulating user interaction and enabling the scraping of a plethora of rug images. This automated process captured images beyond the initial page load, ensuring a comprehensive dataset.

Additionally, the George Washington University Museum website and the Dundee University Museum Textiles Collection website served as valuable resources. Employing the BeautifulSoup library, rug images from these platforms were extracted, enriching the dataset with historical and cultural designs. Moreover, web scraping from e-commerce platforms such as Amazon and eBay augmented the collection.

The extraction of images from Joseph V. McMullan's book "Islamic Carpets" presented a unique challenge. Since the book was in PDF format, images were intermingled with textual content. Computer Vision techniques were employed, using image processing to identify and extract the rug images while omitting pages with textual content. This approach ensured a curated collection of rug designs.

Also, the bing-image-downloader Python package allowed us to easily retrieve images from the Bing search engine. Leveraging this tool, we acquired a diverse set of images relevant to problem. This process streamlined our data collection efforts, enabling us to gather a substantial dataset efficiently. In total, we downloaded a significant number of images(1000), ensuring a comprehensive coverage of various aspects by containing diverse types of carpets.

Moreover, our data collection process utilized eBay and Etsy.com as another sources, yielding a substantial dataset for analysis. From eBay, we gathered approximately 300 images, predominantly showcasing carpets of diverse shapes, with circular designs being particularly prevalent despite instances of repetition. Meanwhile, our extraction from Etsy.com

resulted in around 2900 carpet images. Although not as vibrant as those sourced from Pinterest, these images exhibited a diverse range of rug types, encompassing Persian, Oriental, and a selection of European designs.

5.2 GAN Model Development

The heart of the methodology involves the development of the GAN model. The GAN architecture, comprising a generator network (G) and a discriminator network (D), was chosen due to its remarkable ability to generate realistic and diverse samples. The generator takes random noise vectors as input and generates synthetic rug designs, while the discriminator distinguishes between real and generated images.

For the rug design task, a customized generator architecture was formulated. The generator network takes latent vectors as input and generates rug designs. The discriminator network assesses the authenticity of the generated designs. The generator and discriminator networks are trained in iterations in an adversarial manner, where the generator aims to produce images that deceive the discriminator while the discriminator aims to accurately differentiate real from generated images.

5.3 Dataset Augmentation

To create a robust and diverse dataset for GAN training, dataset augmentation techniques were employed. K-means clustering was used to categorize rug images based on dominant color patterns. This clustering process grouped images with similar color characteristics, contributing to the dataset's diversity. Images were segmented to isolate essential parts such as central patterns, edges, and the area between them. K-means clustering was applied to each segment, further enriching the dataset with various color patterns.

The augmented dataset, consisting of numerous color-pattern combinations, provided a comprehensive training foundation for the GAN model. Images were resized to a uniform resolution of 64x64 pixels and converted to floating-point representations for compatibility with the GAN architecture.

5.4 Dataset Preparation for pix2pix Model

In the initial stages, a GAN model was employed to generate rug patterns. Subsequently, the focus shifted to generating real images from sketch images. To achieve this, the available dataset was converted into sketches.

To enhance the training dataset, variations were introduced in the color channels to facilitate the generation of differently colored rugs.

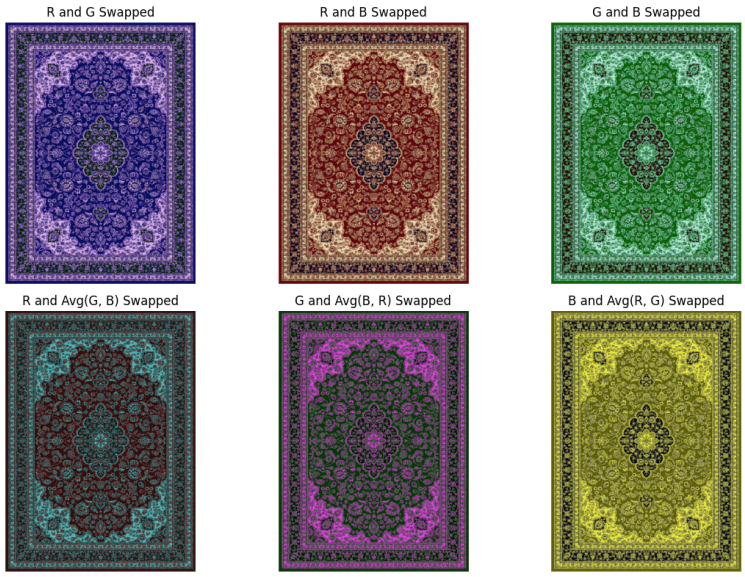
The cGAN development involves the generator with U-Net architecture and the discrim-

inator. For black-and-white or sketched images, an edge detection algorithm , bitwise not operator, and Gaussian Blur was applied to improve the overall quality.

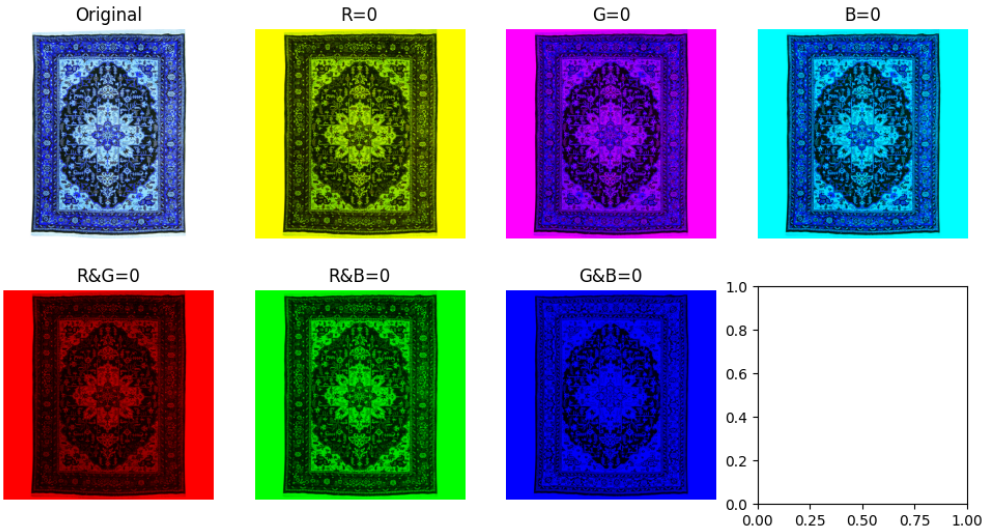
5.5 Dataset Augmentation

Given the critical issue of a small proportion of good images and unattractive sketches, a dataset augmentation strategy was devised. This involved rotating images clockwise by 90 degrees and varying R, G, and B pixels.

(a) Varying color channels



(b) Swapping strategies for color channels

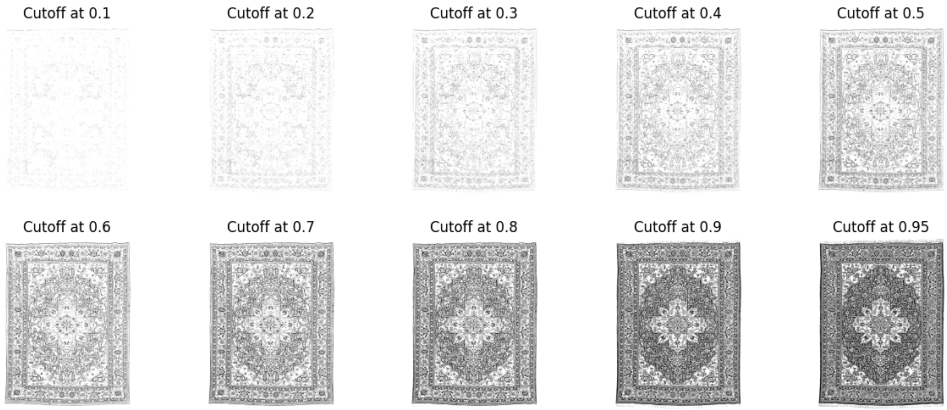


(c) Setting some channels to zero

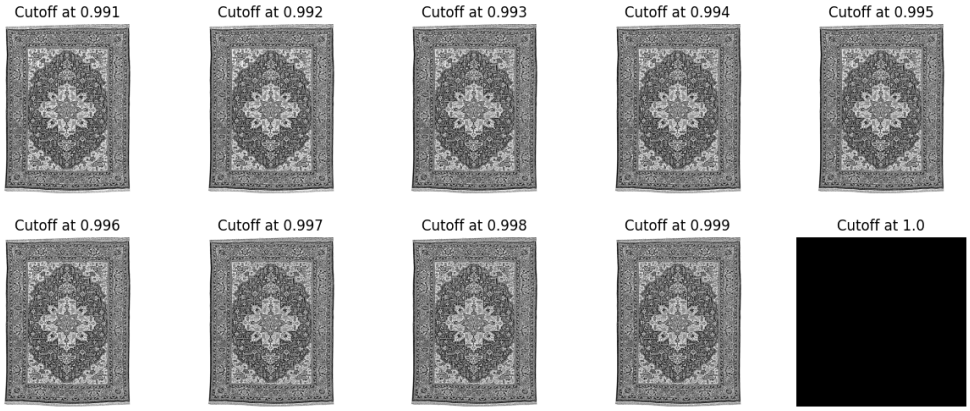
Figure 5.1: Color channels varying during dataset augmentation

5.6 Improving Sketch Realism

To enhance the realism of sketches, we adjusted pixel intensities based on their tendencies towards white or dark shades.



(a) Sketches at various cutoff limits



(b) Sharp edges are obtained at cutoff near 1.0

Figure 5.2: Making sketches from real rug images

5.7 Fourier Transformations

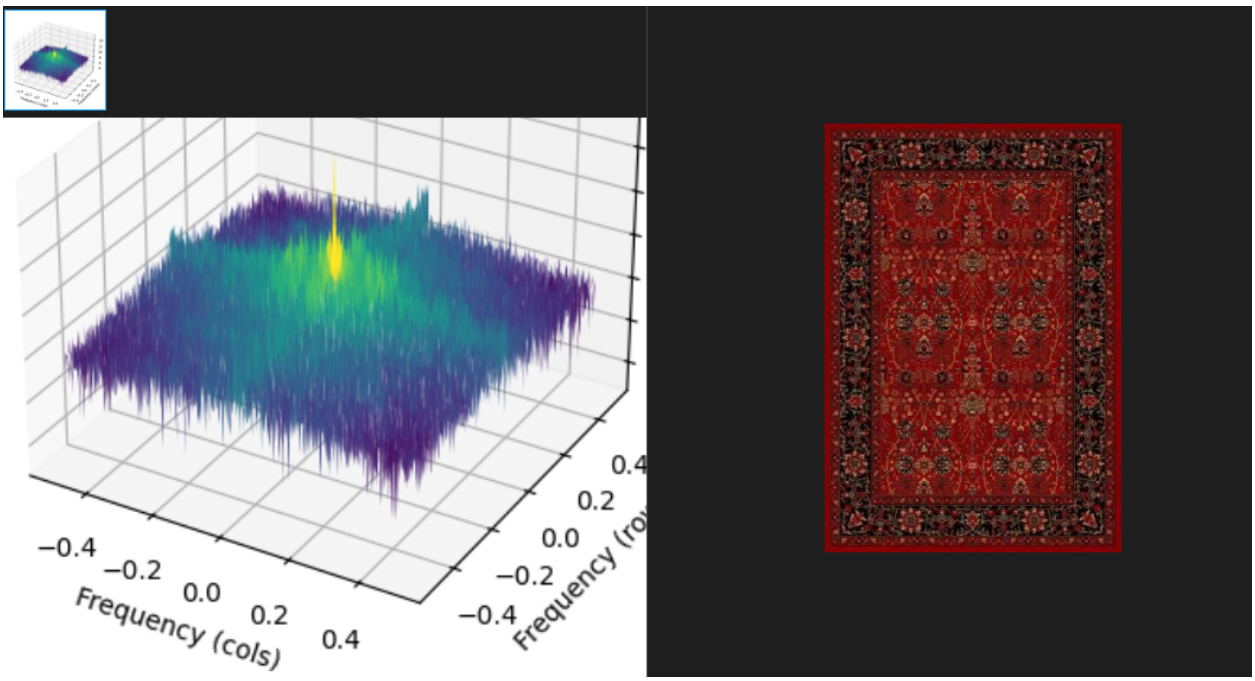
The intricacies of carpet patterns were meticulously explored through Fourier and Wiener transformations utilizing MATLAB tools. Applying Fourier transformation involved converting each carpet image from the spatial domain to the frequency domain, providing a clear spectrum representation highlighting dominant patterns and frequencies. Subsequently, Wiener transformation was employed to refine and enhance the segmented patterns obtained from Fourier transformation. The Wiener filter effectively reduced noise and artifacts, resulting in a more refined representation of the underlying carpet patterns. Figure 5.3 illustrates an original carpet image alongside the segmentation with highlights achieved through Fourier and Wiener transformations.

The applications of Fourier and Wein transformations offer segmentation capabilities and can be instrumental in converting unlabeled datasets into labeled datasets. However, classifying each segmented region into specific motifs or patterns and further categorizing them as Persian or Oriental poses a significant challenge. This task often necessitates domain knowledge and may require manual inspection for accurate classification.

(a) Original Carpet Image

(b) Segmentation with Highlights

Figure 5.3: Original Carpet Image and Segmentation with Highlights



(a) Fourier transform in 3D space and associated carpet, peaks in response corresponds to dense pixel pattern

5.8 Hough Transform

In boundary detection, the problem lies in determining which edges correspond to the boundary. This becomes advantageous for us as we can use it to identify boundaries to represent motifs and patterns.

It is therefore possible to associate with each line of the image a pair (r, θ). The (r, θ) plane is sometimes referred to as Hough space for the set of straight lines in two dimensions. This representation makes the Hough transform conceptually very close to the two-dimensional Radon transform. In fact, the Hough transform is mathematically equivalent to the Radon transform, but the two transformations have different computational interpretations traditionally associated with them [**?**].

The output was obtained as follows, but it did not produce satisfactory results, although a few sharp lines were detected.

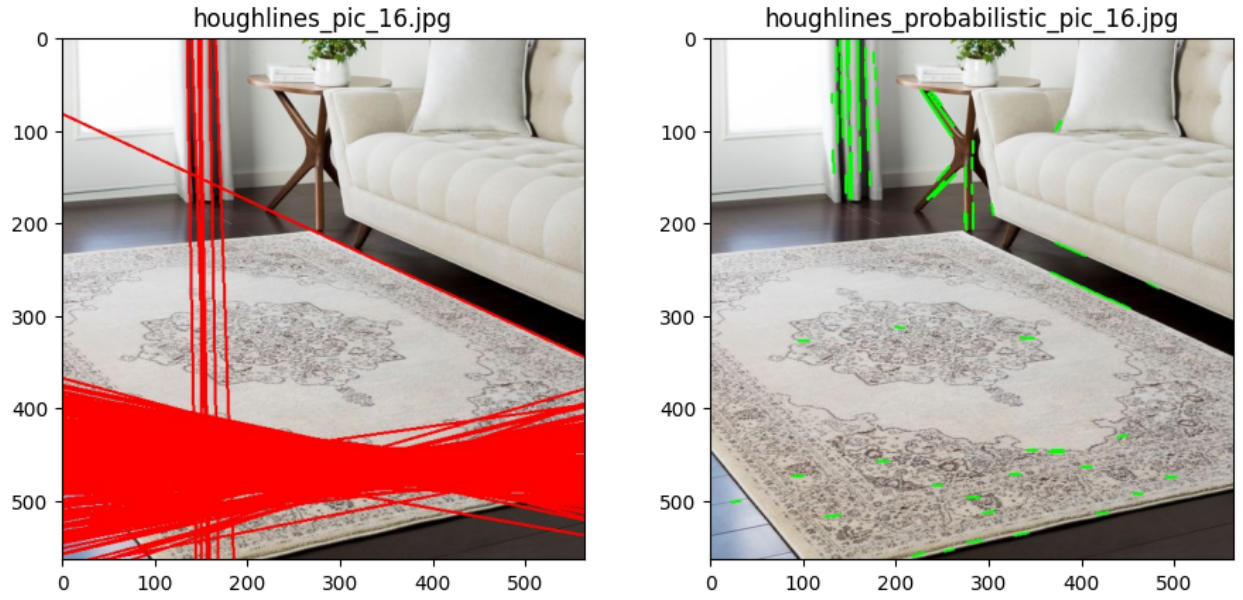


Figure 5.5: Outputs from Hough Transform and Probabilistic Hough Transform to detect edges and subsequently pattern in carpet

6. Experimental Setup

The experimental setup of the rug design project encompasses a comprehensive approach to data collection, dataset augmentation, and GAN model training.

6.1 Data Collection

The foundation of the experimental setup lies in collecting a diverse and comprehensive dataset of rug design images. Multiple sources were tapped to ensure a wide spectrum of design options, ranging from traditional to contemporary patterns. The primary sources for data collection included Pinterest, the George Washington University Museum website, the Dundee University Museum Textiles Collection website, and e-commerce platforms such as Amazon and eBay.

The data collection process began with web scraping techniques. For Pinterest, a Selenium web crawler was employed due to its dynamic loading of images as users scrolled down the page. The web crawler emulated user interaction, scrolling through pages and extracting image links. A key challenge here was managing the repetition of links, which could lead to an infinite loop. To address this, a list of downloaded images was maintained to avoid redundant downloads.

For the George Washington University Museum website and the Dundee University Museum Textiles Collection website, BeautifulSoup was employed to parse the HTML content and extract image links. The websites featured a collection of historical and cultural rug designs, contributing to the dataset's diversity.

Additionally, web scraping was employed to extract rug images from Amazon and eBay. Customized search queries like "Persian rugs" and "Oriental rugs" were used to retrieve images from these e-commerce platforms. The images were then downloaded and stored for further processing.

One unique challenge arose with the extraction of images from Joseph V. McMullan's book "Islamic Carpets," which was in PDF format. To overcome this, Computer Vision techniques were employed to differentiate between pages with images and pages with textual content. OpenCV was utilized to identify text regions, allowing the extraction of images that were not interspersed with text.

6.2 Dataset Augmentation

The choice and preparation of the dataset play a pivotal role in the training of any machine learning model. For the rug generation project, a diverse dataset containing images of rugs in various styles, colors, and patterns was curated. The dataset was pre-processed to convert it into a suitable format for training the pix2pix model. Images were resized to a consistent resolution, and any irrelevant metadata was removed to ensure a clean and uniform dataset.

With the collected images forming the initial dataset, dataset augmentation techniques were applied to diversify and enrich the dataset. The goal was to enhance the GAN model's ability to generate a wide range of rug designs that cater to varying user preferences.

The dataset augmentation process involved two main steps: color-pattern clustering and image segmentation. Color-pattern clustering aimed to categorize rug images based on dominant color patterns. K-means clustering was chosen for its simplicity and efficiency. Different values of k were tested to determine the optimal number of clusters. It was observed that $k = 6$ produced visually distinct color patterns without introducing excessive noise.

Subsequently, images were segmented to isolate key components: the central pattern, edges, and the area between them. The central pattern was often a densely connected circular or square motif, while the edges featured distinct patterns. Image segmentation allowed for a finer analysis of different rug sections, contributing to the diversity of the dataset.

For each segment, K-means clustering was applied separately to generate multiple clusters based on color patterns. This approach led to the creation of a dataset containing numerous images, each representing a different color-pattern combination. As a result, the dataset was enriched with a wide range of designs that catered to various user preferences.

6.3 Results

One of the pivotal accomplishments is the meticulous collection of a diverse and comprehensive dataset. This dataset incorporates rug images from various sources, including online platforms, museum websites, books, and e-commerce platforms. The dataset's diversity is crucial in ensuring that the generated rug designs encompass a wide array of styles, patterns, and color schemes, catering to the varied preferences of users seeking personalized interior design solutions.

Furthermore, the dataset augmentation techniques applied have proven effective in enhancing the quality and diversity of the dataset. By employing segmentation and clustering methods, the augmentation process has generated multiple variations of each rug image, allowing the GAN model to capture intricate design details and variations. This augmenta-

tion process contributes to the robustness and versatility of the GAN model, enabling it to generate visually appealing and realistic rug designs.

The development of the customized GAN architecture tailored for rug design is a significant achievement. This GAN model serves as the creative engine responsible for generating rug designs based on user inputs and preferences. The model's architecture has been carefully designed to ensure compatibility with rug design features and patterns, enhancing its ability to produce coherent and aesthetically pleasing designs. Initial training of the GAN model using a subset of the augmented dataset has been initiated, laying the groundwork for subsequent refinements and improvements.

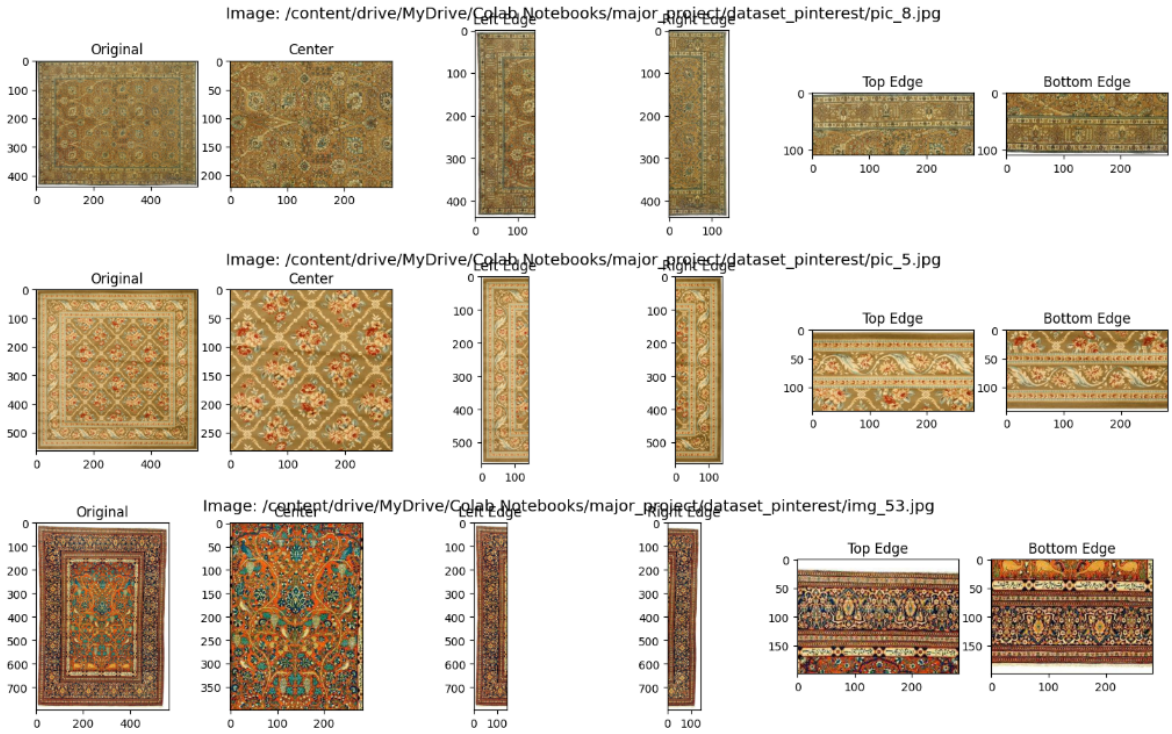


Figure 6.1: Segmentation of images during feature extraction

6.3.1 Hardware Configuration

The experimental setup began with a robust hardware configuration to ensure efficient model training. The computations, particularly for training deep neural networks, demand substantial computational power. In this project, a high-performance GPU (Graphics Processing Unit) was utilized to accelerate the training process provided by Kaggle.

6.3.2 Software Environment

For the software environment, Google Colab was chosen as the primary platform for developing and executing the machine learning pipeline. Google Colab provides free access to

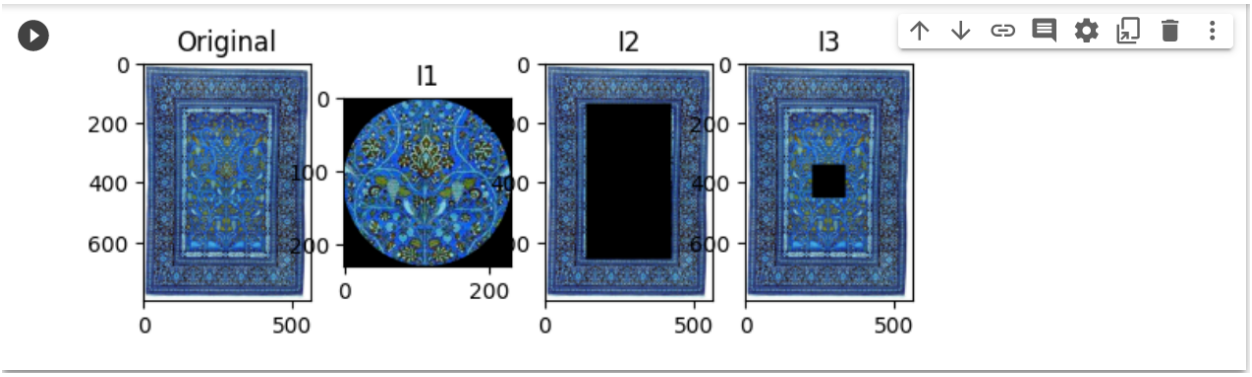


Figure 6.2: Separating components of rug during feature engineering

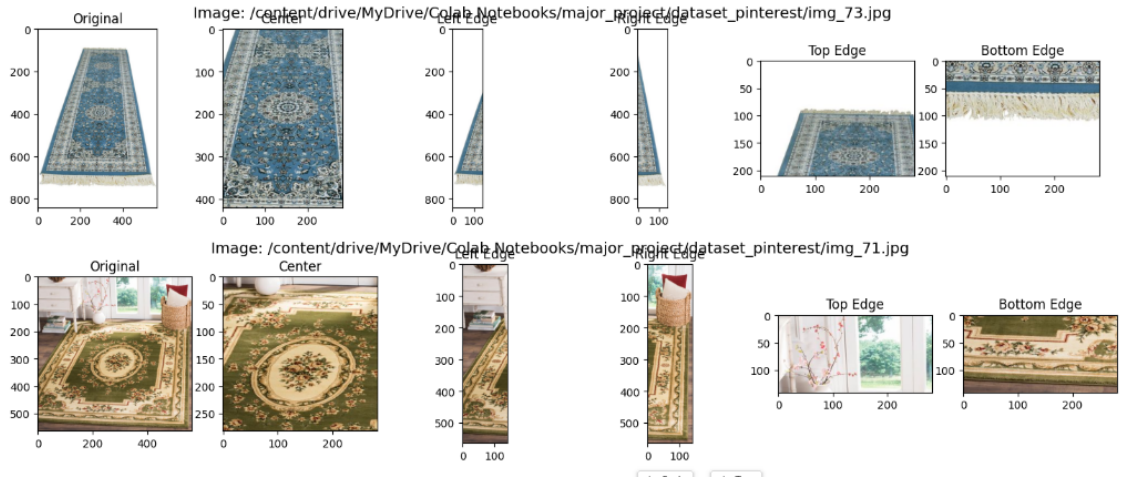


Figure 6.3: Some rugs' features are not captured in segments

GPU resources, making it an ideal choice for experimenting with deep learning models. The project leveraged the TensorFlow framework, an open-source machine learning library, to implement and train the pix2pix cGAN. TensorFlow's compatibility with Google Colab and extensive community support made it a preferred choice for this project.

6.4 Training Parameters

Configuring the hyperparameters for training is a critical aspect of achieving optimal model performance. The pix2pix model's hyperparameters, including learning rate, batch size, and the number of epochs, were carefully tuned to strike a balance between model convergence and avoiding overfitting. The learning rate was set at 0.001, and the batch size of 64 was determined based on the available GPU memory to ensure efficient training.

6.4.1 Two Time-Scale Update Rule

The two time-scale update rule (TTUR) is proposed to stabilize GAN convergence by setting the learning rate of the generator lower than that of the discriminator. This ensures that the

generator moves slower than the discriminator, preventing it from driving the discriminator into new regions without capturing its gathered information. According to the literature review of [6], the authors proved that a general class of games, including the GAN game, converges under mild assumptions to a stationary local Nash equilibrium when trained under TTUR. They also recommended using the Adam stochastic optimization to avoid mode collapse and the Fréchet inception distance for evaluating GAN performances. Thus, we used the Adam optimizer.

6.4.2 Vanishing Gradient

Conversely, if the discriminator learns too fast compared to the generator, it could almost perfectly distinguish between real and generated samples. In such cases, the generator could get stuck with a very high loss, and the gradient would be close to zero, resulting in the vanishing gradient problem. This occurs when the discriminator is too good at distinguishing samples, preventing the generator from learning. To address these issues, we adjusted the learning rates during training: the learning rate of the generator was set to 1×10^{-4}, while that of the discriminator was set to 2×10^{-4}.

6.5 Image Quantization

In the manufacturing of carpets, it is essential to quantize the colors in the carpet image. This process involves mapping each color obtained after quantization to the corresponding wool color. Various methods have been utilized for this purpose, including:

- **Octree Algorithm For Color Quantization**: This method involves constructing a tree-based data structure to partition the color space, enabling efficient color quantization. It strikes a balance between accuracy and speed, making it a popular choice.
- **Median Cut Algorithm For Color Quantization**: In this approach, the color space is recursively partitioned based on the median values of color components, resulting in distinct color regions. It is a fast and effective method that provides a good trade-off between speed and the number of colors in the quantized palette.
- **Maximum Coverage Algorithm For Color Quantization**: This method focuses on maximizing the coverage of the input color space with the selected color palette, potentially providing better color fidelity.

The output of the comparison among different quantization methods is shown in Figure 6.7.

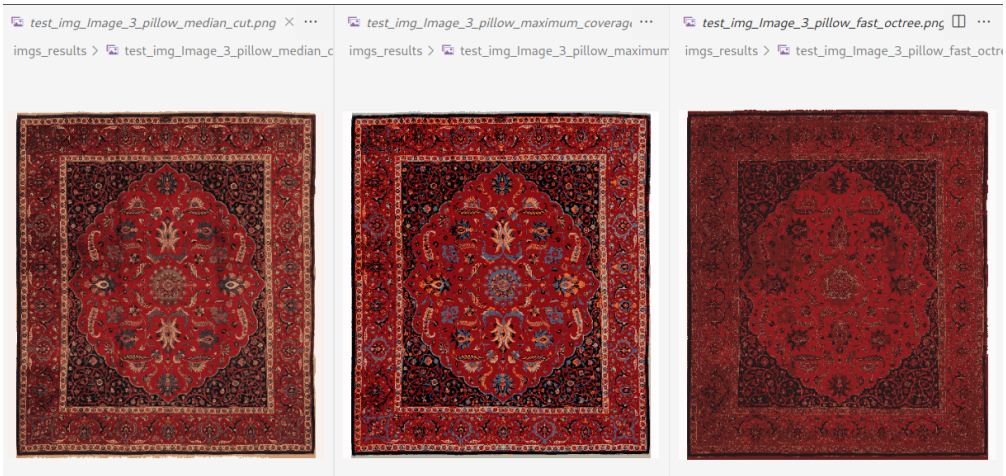


Figure 6.7: Comparison of different color quantization methods.

Currently, we are using the median cut method, which offers a balance between speed and the number of colors in the palette. This trade-off allows us to achieve an efficient quantization process while maintaining an acceptable level of detail in the carpet design.

Although websites such as vectorizer.ai produce impressive results, their free versions often include a watermark on the output images.

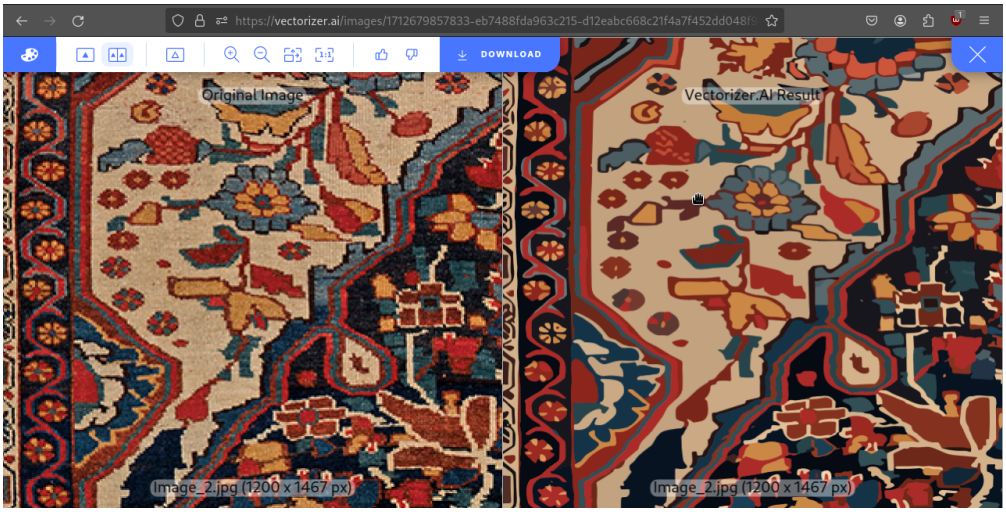


Figure 6.8: Example from vectorizer.ai output showing the difference between vectorized art, contains lesser more representative colors

6.6 Seam Carving for Image Resizing

Seam carving, also known as liquid re-scaling, enables content-aware resizing of images by identifying and manipulating seams, or paths of least importance, within the image. Seam carving can reduce or extend the size of an image by removing or inserting seams, respectively.

Additionally, it allows for the manual selection of areas that should remain unmodified, and it can remove entire objects from photographs.

The primary goal of seam carving is image re-targeting, which addresses the challenge of displaying images without distortion across various media sizes, such as cell phones or projection screens.

Seams in the algorithm can be vertical (a path of connected pixels from top to bottom with one pixel per row) or horizontal (a path of connected pixels from left to right with one pixel per column). The importance or energy of each pixel is measured based on its contrast with neighboring pixels, using various methods such as gradient magnitude, entropy, visual saliency, or eye-gaze movement.

The seam carving process involves the following steps:

1. Begin with an image.
2. Calculate the weight, density, or energy of each pixel using the chosen method.
3. Create a list of seams ranked by energy, with lower energy seams being less important to the image's content.
4. Remove or insert seams as necessary, depending on whether the image needs to be reduced or expanded.
5. Produce the final, resized image.

We employ seam carving to resize the carpet images according to custom input dimensions, such as 3 meters by 3 meters, while maintaining the desired aspect ratio and content integrity. Given that one centimeter corresponds to four pixels, it is crucial to adjust the images precisely to meet these specific measurements. Seam carving allows us to achieve this resizing in a content-aware manner, enabling us to tailor the carpet designs to the exact specifications required by our application. This technique ensures that the resized images align accurately with the provided input dimensions.

(a) Resized to 0.1m * 0.1m (b) Resized to 0.1m * 0.5m (c) Resized to 0.5m * 0.1m

Figure 6.9: Resized images to specified dimensions.

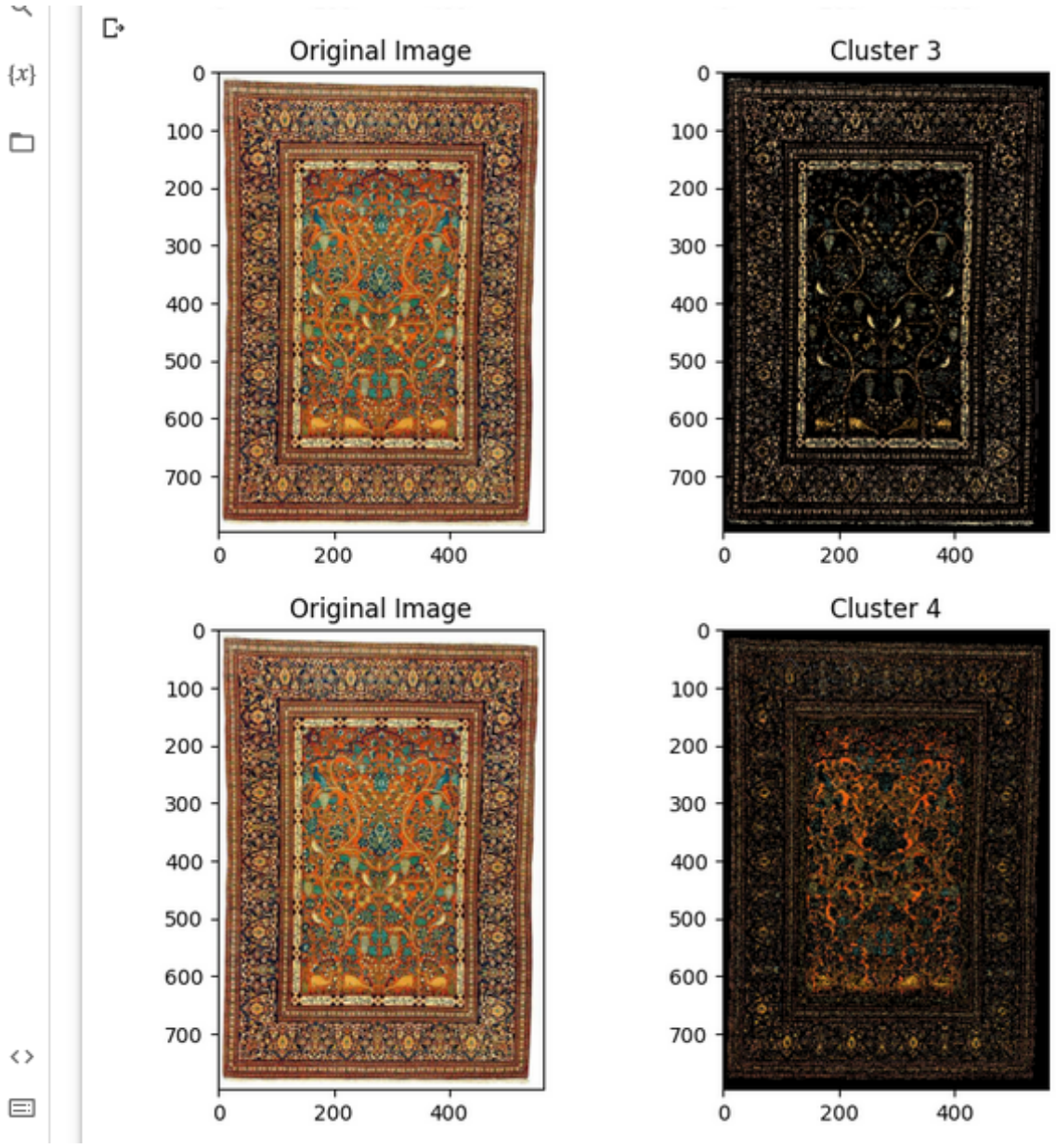


Figure 6.4: K-Means Clustering with K=3

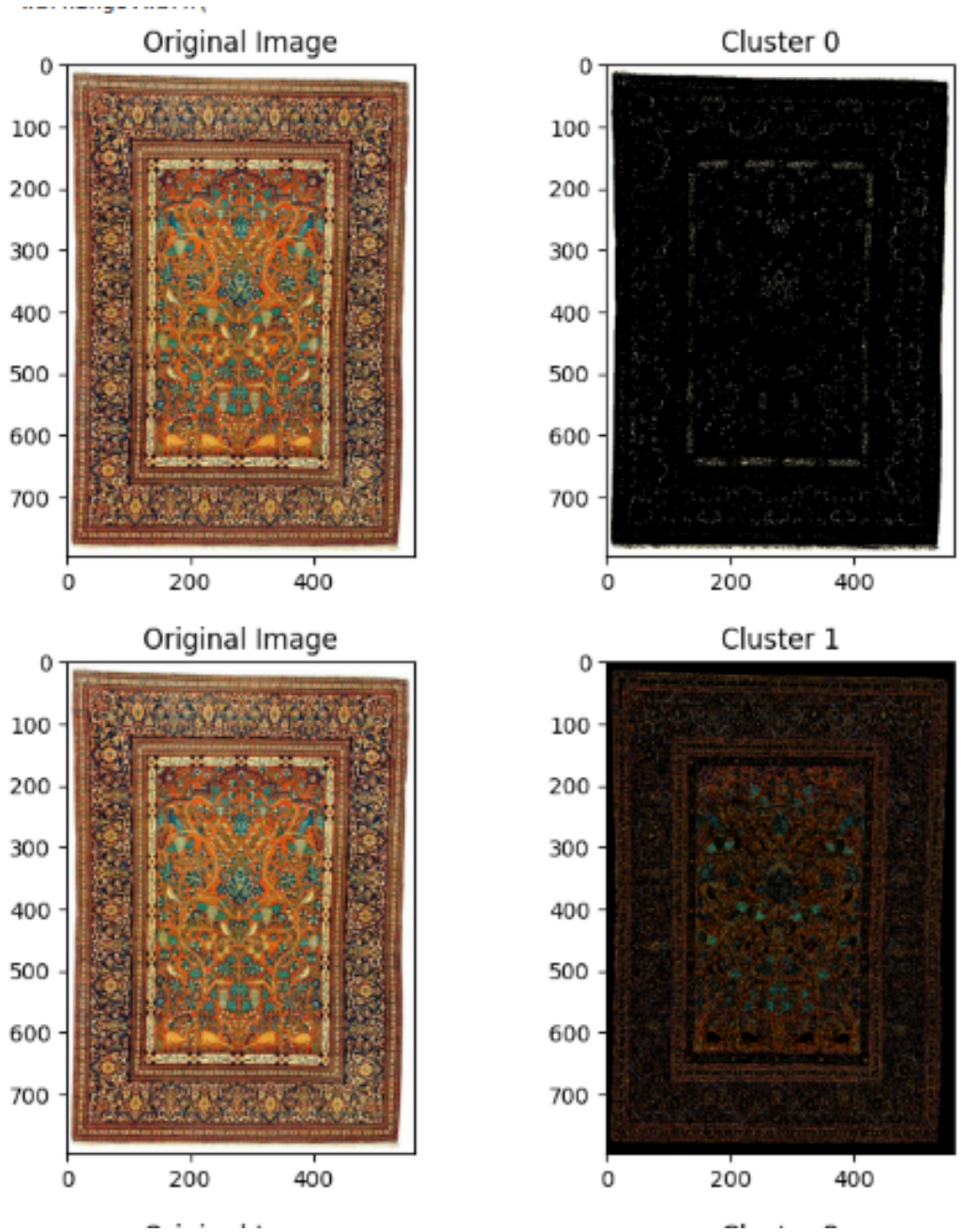


Figure 6.5: K-Means Clustering with K=20, more clusters can result noisy patterns

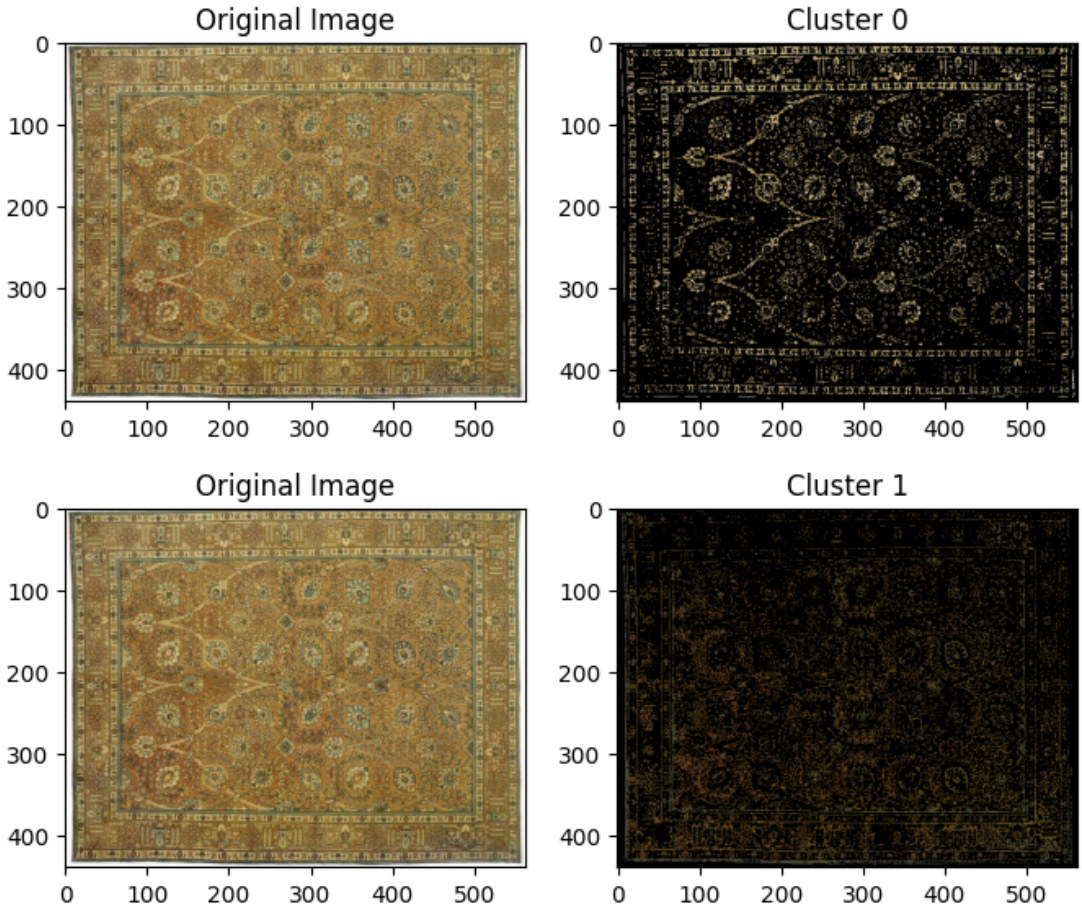


Figure 6.6: Visualization of various patterns, where highlighted portion indicates the feature of the rug.

7. System Design

The system design of the project encompasses the architecture of the generator, discriminator, and DCGAN, along with the training workflow. The following sections provide a comprehensive overview of each component.

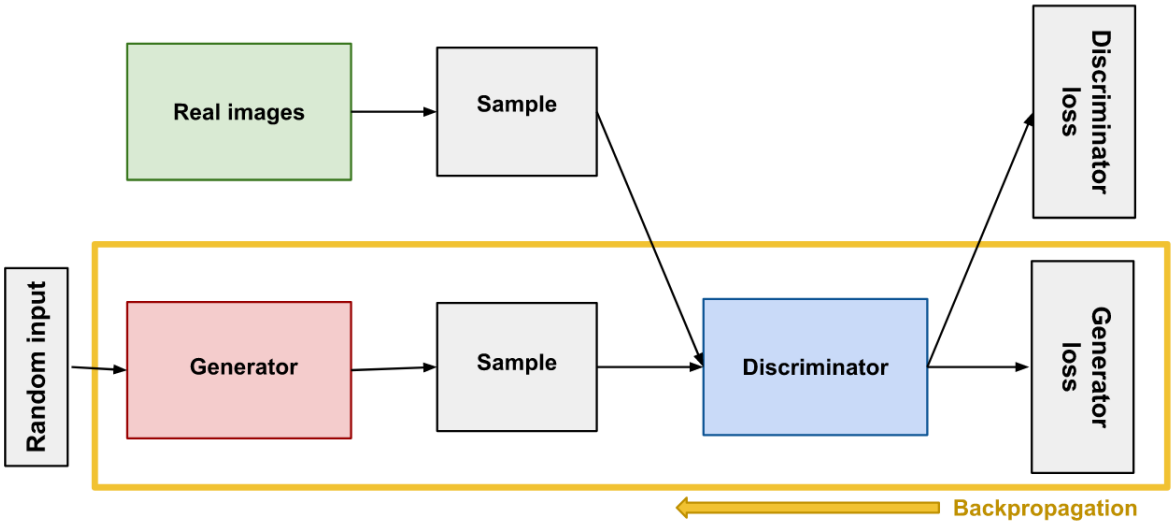


Figure 7.1: Generative Adversarial Network Model

The generator is responsible for generating synthetic rug designs from random noise vectors. It comprises multiple layers of dense, convolutional, and transposed convolutional units.

The discriminator evaluates whether an input image is real or generated by the generator. It consists of convolutional and fully connected layers.

7.1 DCGAN Architecture

The DCGAN combines the generator and discriminator to create a generative adversarial network.

The training workflow involves training the discriminator and the DCGAN iteratively. The discriminator is trained to distinguish real and generated images, while the DCGAN trains the generator to produce realistic images that can fool the discriminator. The training process includes optimization using loss functions and backpropagation.

The system design of the rug generation project is a critical aspect that involves defining the architecture of the conditional generative adversarial network (cGAN) and establishing

the loss functions necessary for training. The system comprises a generator, responsible for creating realistic rug images from input sketches, and a discriminator, designed to distinguish between generated and real images.

Generator Architecture

The generator architecture is based on a modified U-Net, a deep neural network architecture renowned for its success in image-to-image translation tasks. The generator takes a sketch image as input and transforms it into a detailed and vibrant rug image. The architecture consists of downsample and upsample blocks, each contributing to the encoding and decoding processes.

The downsample block reduces the spatial dimensions of the input, capturing hierarchical features. It includes convolutional layers, batch normalization, and Leaky ReLU activation functions. The upsample block, on the other hand, increases the spatial dimensions, allowing the generator to produce high-resolution output. Additionally, dropout layers are applied in some upsample blocks to enhance generalization and prevent overfitting.

Skip connections are employed between corresponding layers in the encoder and decoder, facilitating the flow of low-level details during the upsampling process. The final layer uses a transposed convolution to produce the output image with the same dimensions as the input sketch.

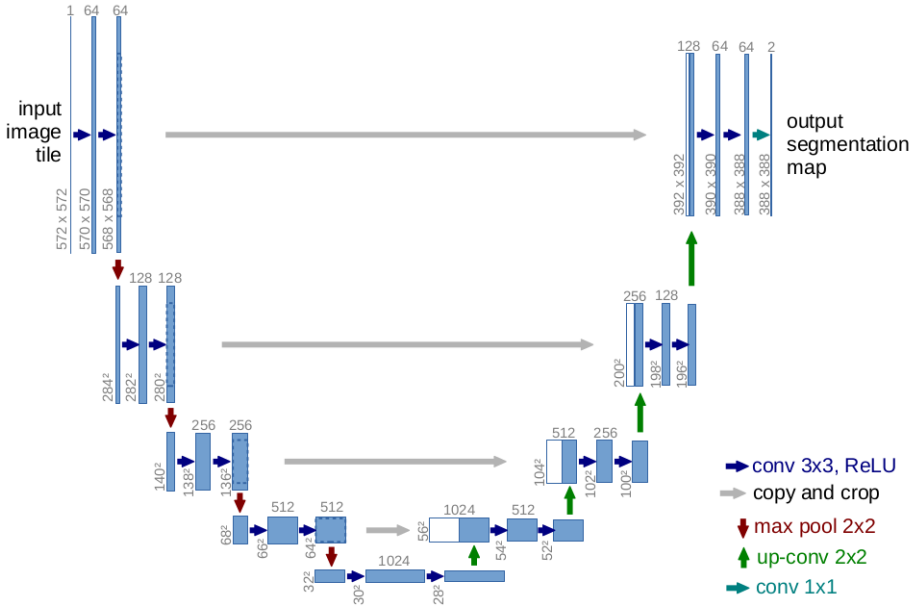


Figure 7.2: U-Net as Generator

Discriminator Architecture

The discriminator is designed as a PatchGAN classifier, aiming to classify whether each patch of an image is real or generated. This architecture promotes high-frequency detail discrimination, contributing to the overall realism of the generated images. The discriminator takes both the input sketch and the generated image as inputs and outputs a classification score.

The architecture of the discriminator involves downsample blocks with convolutional layers, batch normalization, and Leaky ReLU activation functions. A concatenation layer combines the input sketch and generated image before further processing. The discriminator aims to distinguish real images from generated ones by learning relevant features.

Loss Functions

The training of the pix2pix cGAN involves defining appropriate loss functions to guide both the generator and discriminator. The generator is trained using a combination of adversarial loss and mean absolute error (L1 loss). The adversarial loss encourages the generator to produce images indistinguishable from real ones, while the L1 loss enforces pixel-wise similarity between the generated and target images.

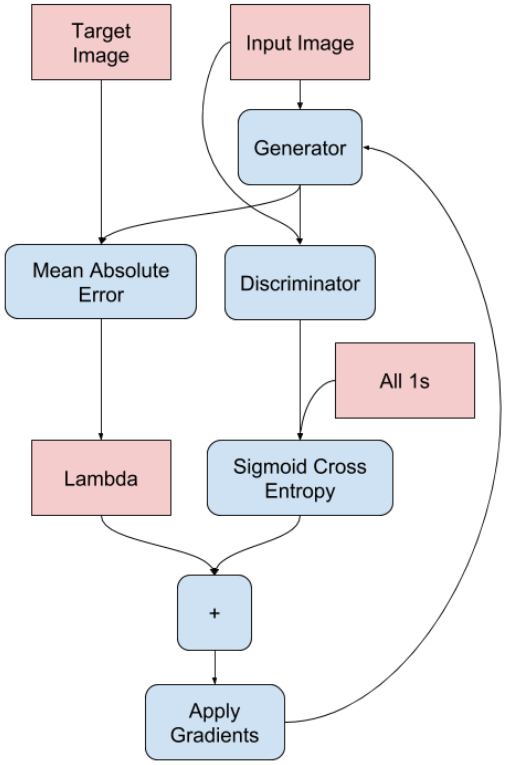


Figure 7.3: Training procedure for the generator

Conversely, the discriminator is trained to distinguish between real and generated images. The discriminator loss comprises a real loss and a generated loss. The real loss ensures accurate classification of real images, while the generated loss penalizes the discriminator for mis-classifying generated images.

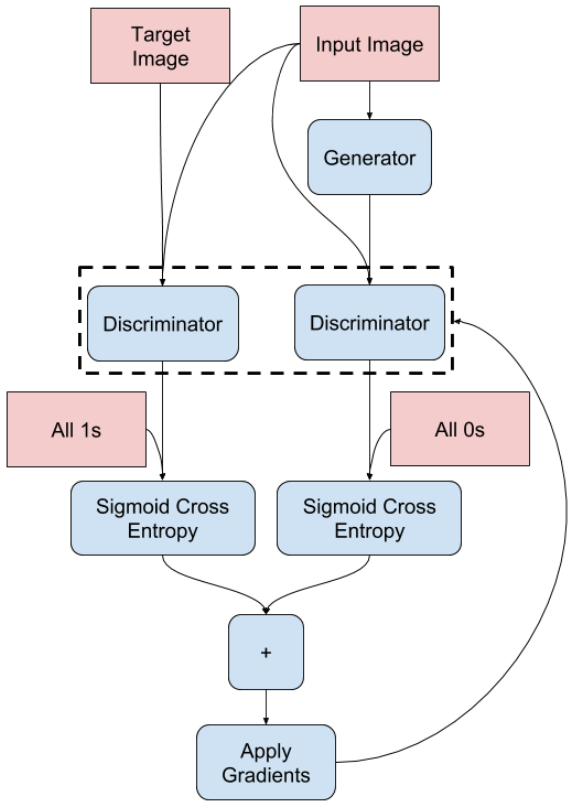


Figure 7.4: Training procedure for the discriminator

Optimizers

The optimization process is crucial for training stability and convergence. The generator and discriminator are optimized using the Adam optimizer with a learning rate of 2×10^{-4} and a beta1 value of 0.5. This choice of optimizer facilitates efficient weight updates during the training process, contributing to the overall effectiveness of the pix2pix cGAN.

7.2 VGG network

The VGG network, a convolutional neural network architecture, plays a pivotal role in the neural style transfer learning. This design leverages the VGG-19 architecture, as illustrated in Figure 7.5[15]

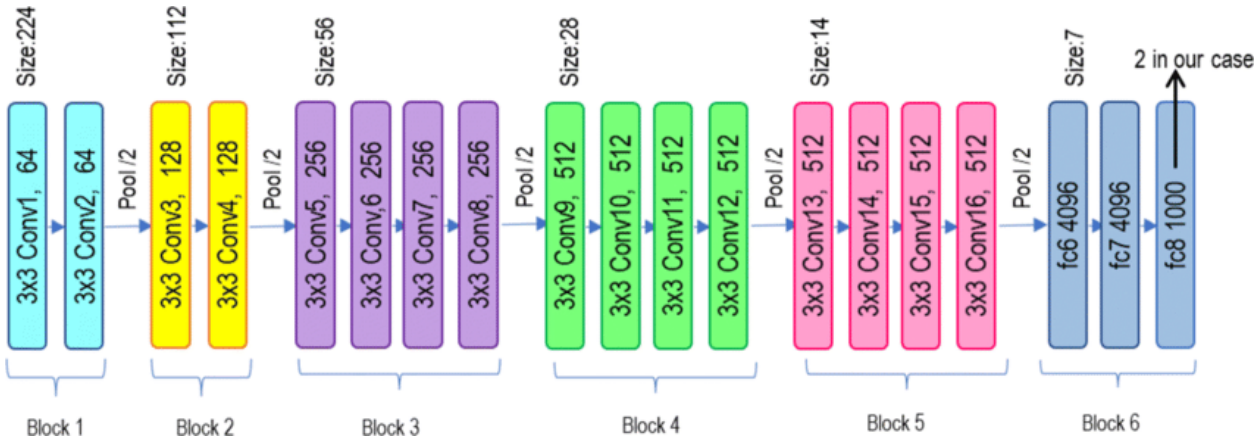


Figure 7.5: VGG-19 Architecture

The VGG-19 architecture consists of multiple convolutional and fully connected layers, providing a deep hierarchical feature representation of input images.

In style transfer, the intermediate layer outputs of the VGG network become instrumental. Specifically, the architecture utilizes the features extracted from VGG-16 to enhance the generation process.

To achieve the generation of a new design from two images, style transfer combines the intermediate layer outputs obtained from the VGG-16 network. This merging process involves a strategic fusion of feature representations, resulting in a novel and synthesized design.

7.3 System Architecture

The system is an interactive application that enables users to create custom carpet designs. The system consists of a frontend developed using ReactJS, a backend implemented in Django, and utilizes machine learning models such as style transfer and Pix2Pix for image generation and colorization.

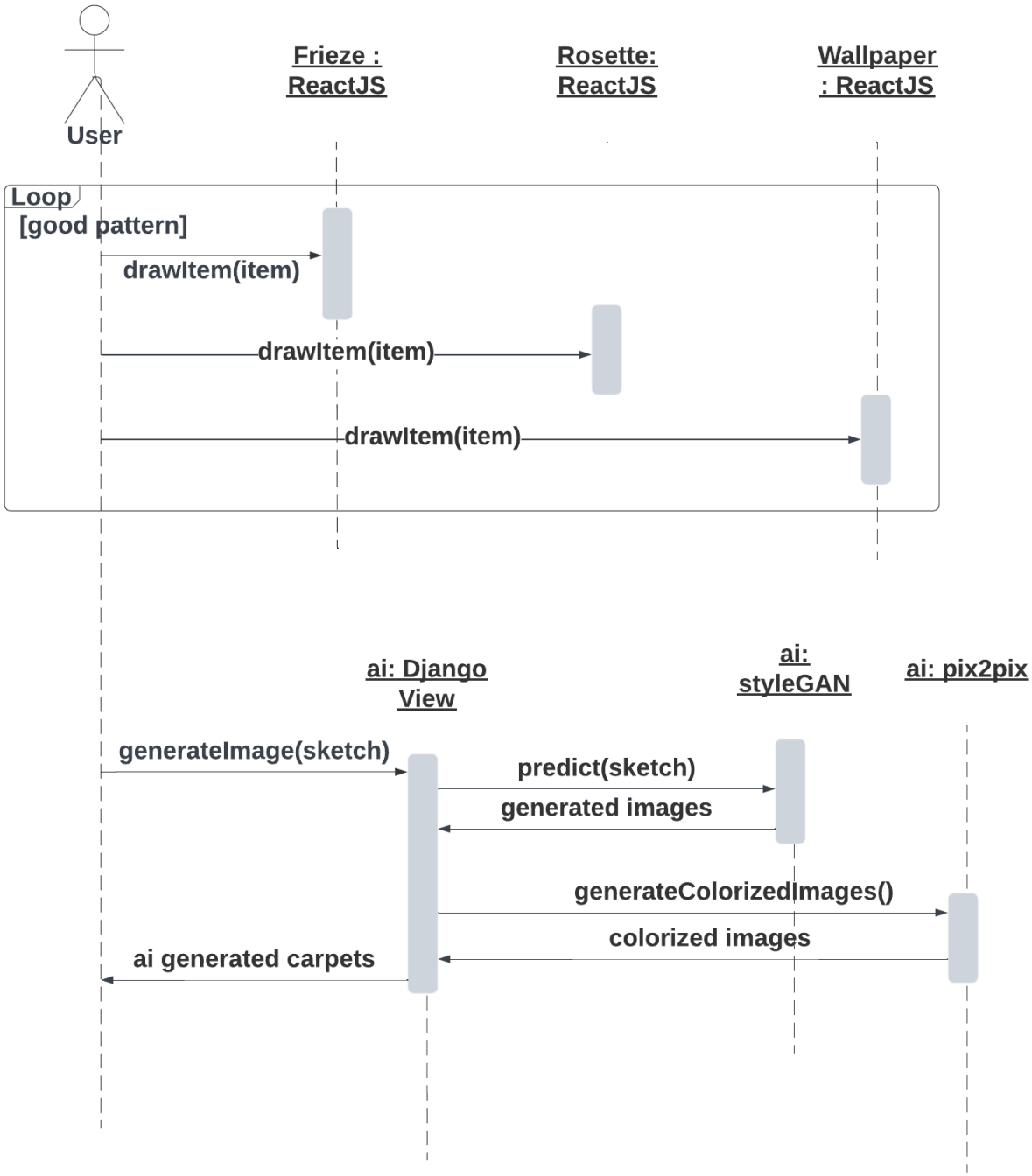


Figure 7.6: Sequence Diagram

1. **Drawing Carpet Image:** The *User* interacts with the *ReactJS Frontend* to draw a carpet image using a set of drawing tools. During this process, the user has the option to choose from different patterns, including frieze, rosette, or wallpaper patterns.

2. **Pattern Parameter Selection:** Once the user completes the drawing, they can select various pattern parameters such as symmetry, color, line width, and motifs (e.g., circle,

rectangle, line, hand-drawn elements) using the frontend interface.

3. **Backend Processing:** The *Frontend* sends the user's input, including the drawn image and selected parameters, to the *Django Backend*.

4. **Neural Style Transfer Image Generation:** The *Backend* instantiates the *style transfer* model, a generative adversarial network (GAN), to predict a high-resolution carpet image based on the user's drawing and selected parameters. The architecture of style transfer is complex, incorporating multiple layers for feature extraction and generation.

5. **Pix2Pix Colorization:** The predicted image may have undesirable color characteristics. To enhance the visual appeal, the *Backend* passes the generated image to *Pix2Pix*, a conditional GAN designed for image-to-image translation. Pix2Pix colorizes the image to some extent, providing a more refined output.

6. **Result Presentation:** The *Backend* returns the processed images, including the style transfer-generated and Pix2Pix-colorized versions, to the *Frontend*. The *Frontend* then displays these images to the *User* for evaluation and selection.

8. Project Results

At the present stage of the project, significant progress has been achieved in laying the foundation for a novel and innovative approach to rug design. The experimental setup phase has been successfully executed, comprising various data collection methods, dataset augmentation techniques, and the development of a customized Generative Adversarial Network (GAN) architecture tailored specifically for rug design. This initial phase of the project has paved the way for the implementation and refinement of the rug design generation process.

8.1 Output from GAN

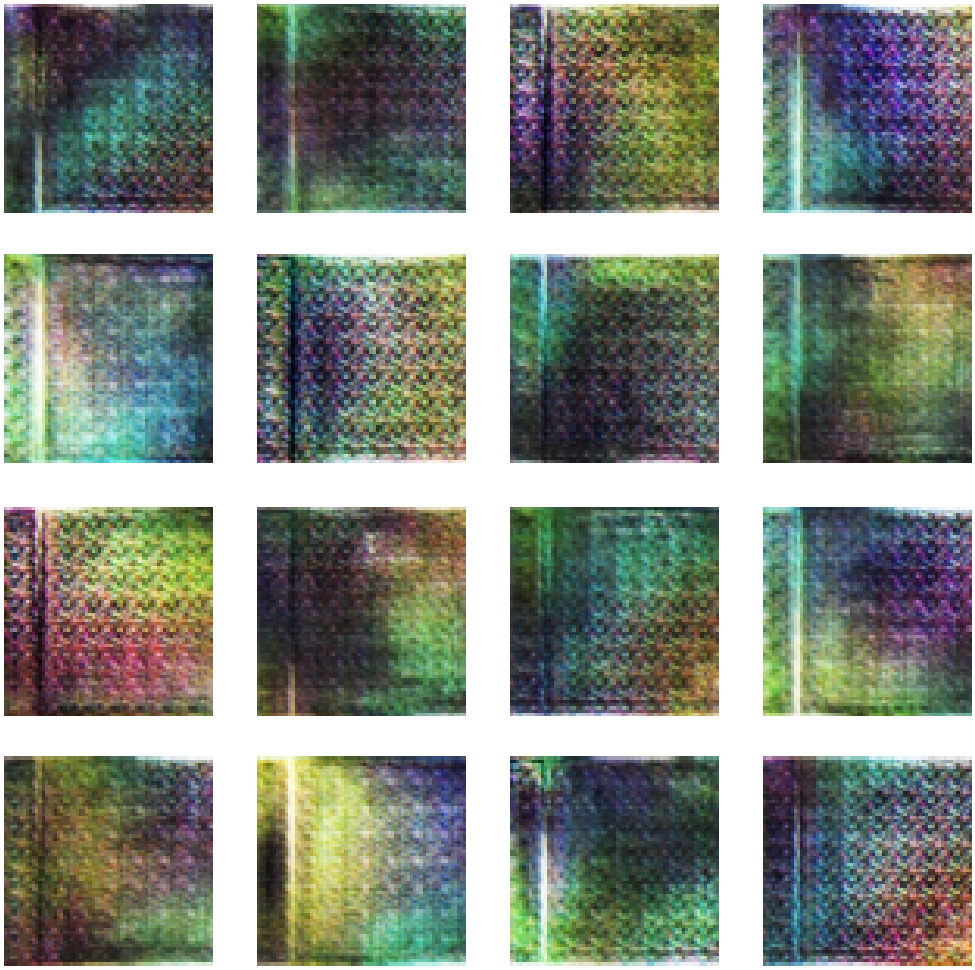


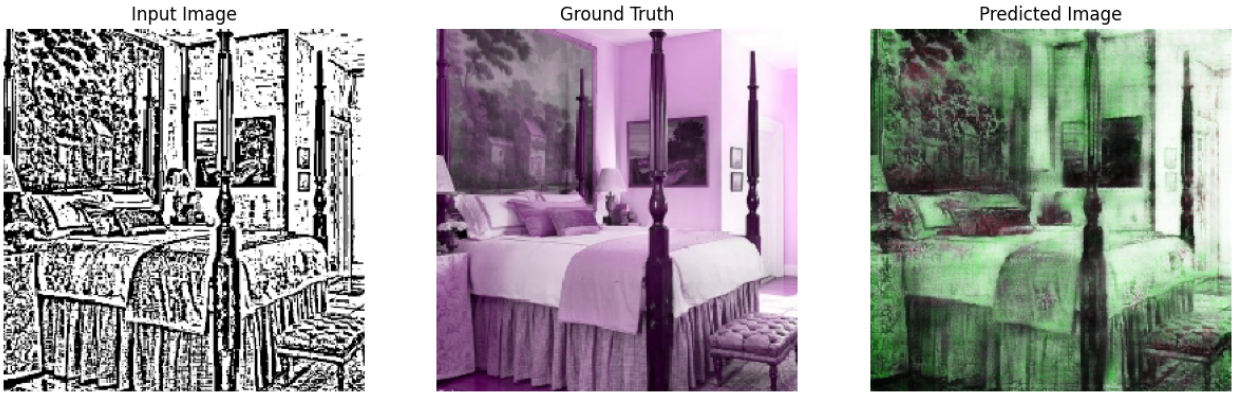
Figure 8.1: Carpet generated when random input is provided to GAN

8.2 Output from pix2pix model

The generated images from the pix2pix models are presented below:

(a)

(b)

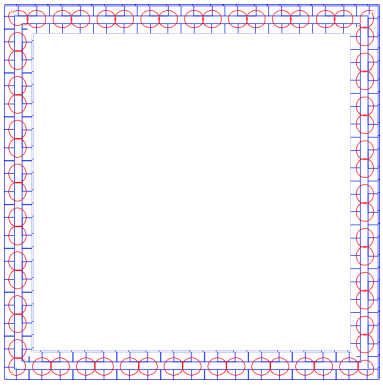


(c)

Figure 8.2: Carpet colorized by pix2pix model

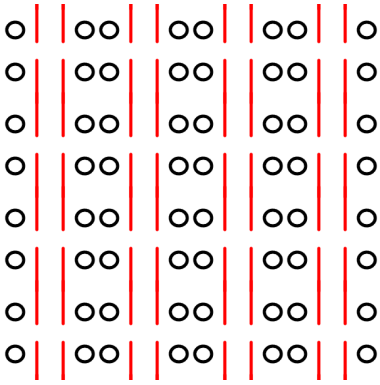
8.3 Output from VGG19 model

The user generated design and GAN enhanced designs are presented below, the frieze symmetry is designed to ensure continuity at the corners, enhancing the visual appeal and seamless of the images.

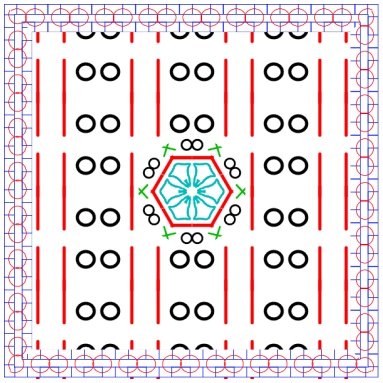


(a) Frieze Symmetry

(b) Rosette Symmetry



(c) Wallpaper Symmetry

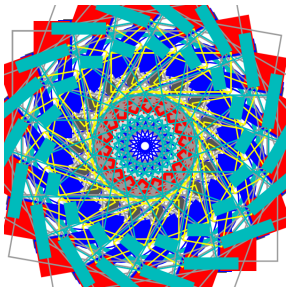


(d) Combined Design

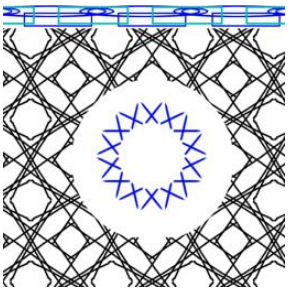
(e) GAN Enhanced Design

(f) GAN Enhanced Design

Figure 8.3: Carpet Design Steps in system. (e) and (f) and Neural Style Transfer enhanced designs by taking input as the final design from user (d)



(a)



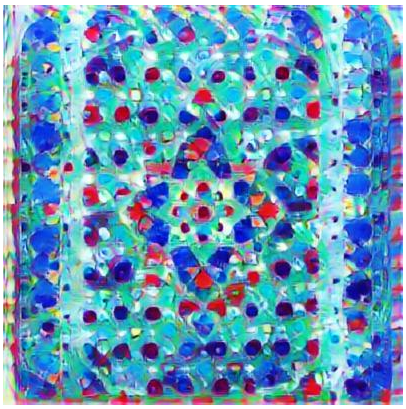
(b)

Figure 8.4: User Generated Designs using Symmetry Groups

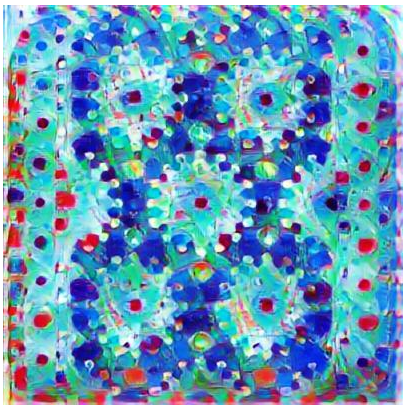
(a)

(b)

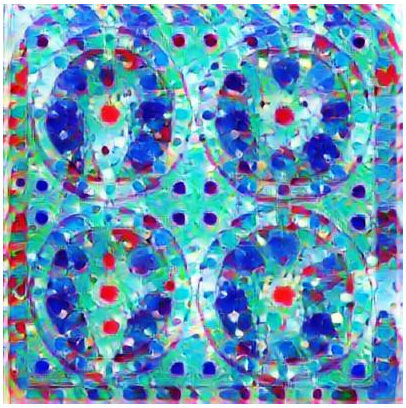
(c)



(d)



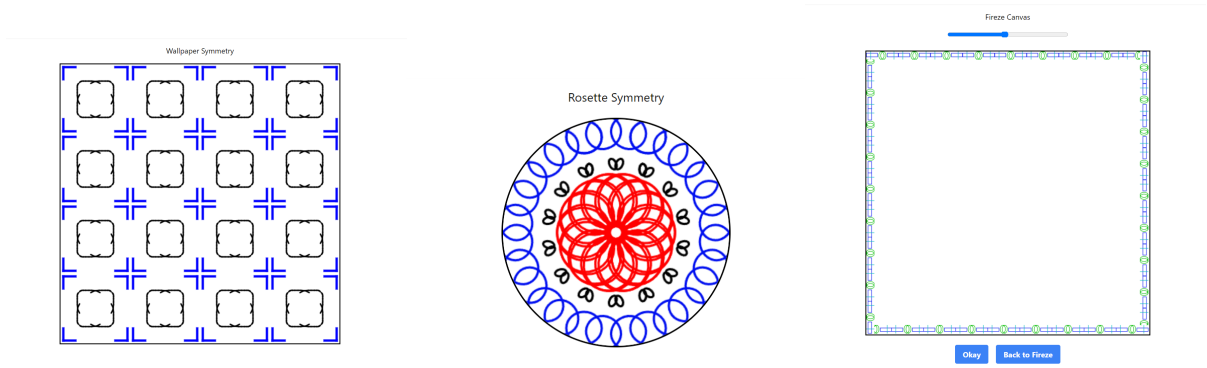
(e)



(f)

Figure 8.5: Enhanced Images using GAN by taking 8.4 as input

8.4 User Interface and User Interaction



(a) Wallpaper Symmetry (b) Rosette Symmetry (c) Frieze Symmetry

Figure 8.6: Different Symmetry Groups in ReactJS drawn by user

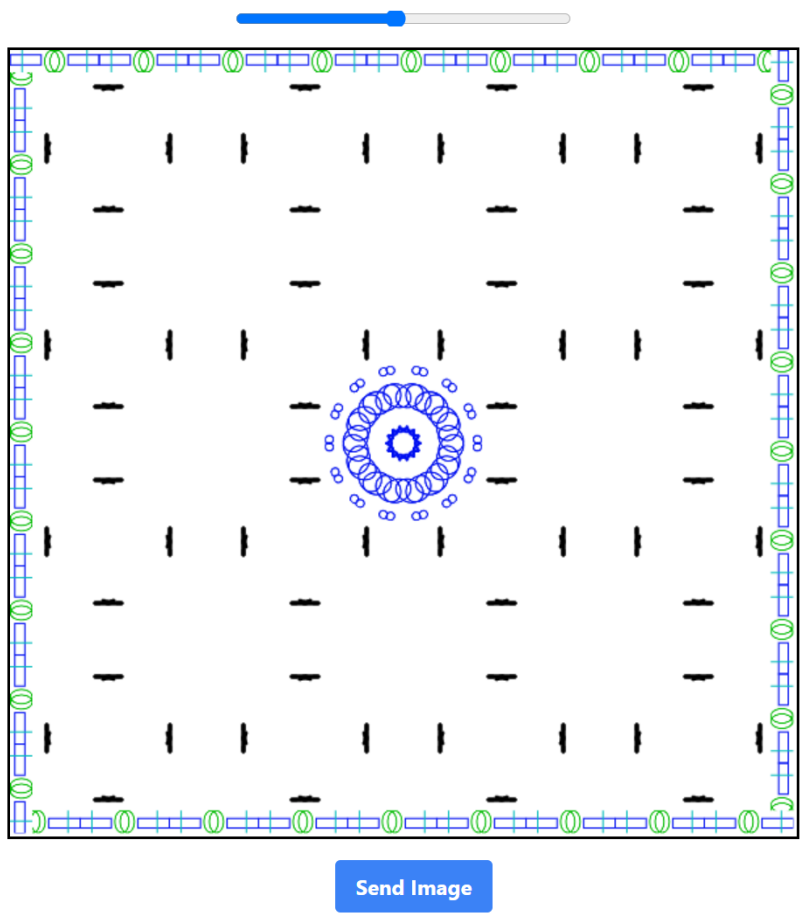
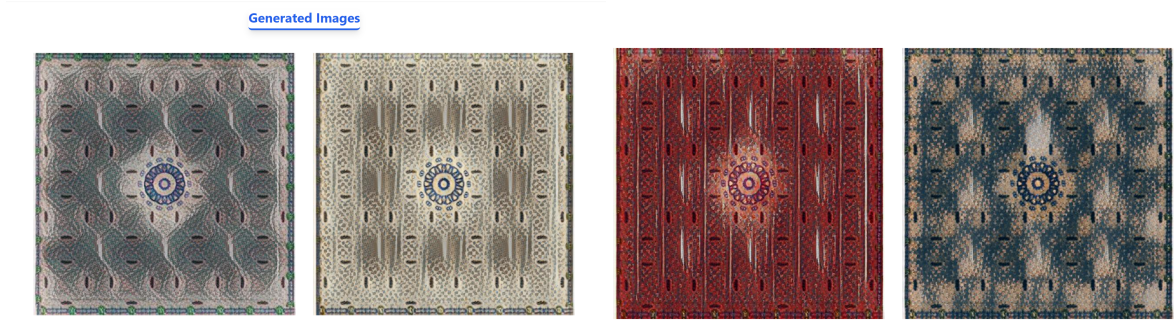
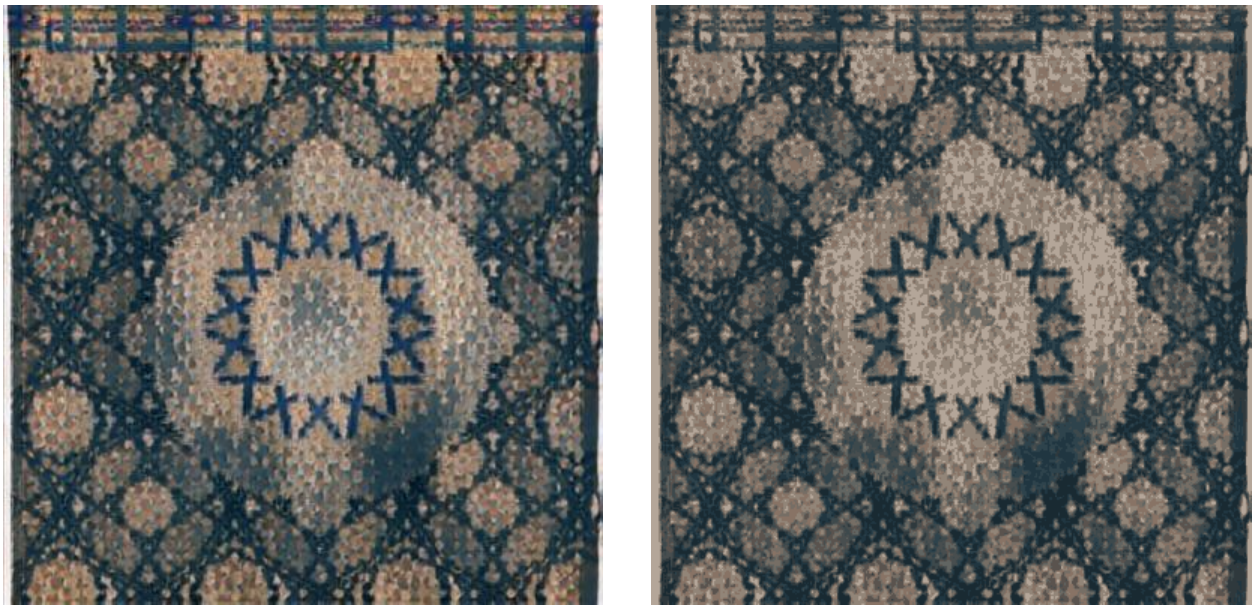


Figure 8.7: Combined Design



(a) (b)

Figure 8.8: Images returned from Django API



(a) Original (b) Quantized

(c) Color Palette

Figure 8.9: Comparison of original and quantized images, along with the color palette

8.5 Neural Style Transfer

One downside to the basic implementation of neural style transfer is that it produces a significant amount of high-frequency artifacts in the output image. To address this issue, a regularization term can be added to penalize high-frequency components in the image. This regularization is known as total variation loss, and its purpose is to minimize abrupt changes in the image, thus smoothing the overall appearance.

The following images illustrate the effects of high-frequency artifacts and the improvements that can be achieved by incorporating total variation loss into the training process:

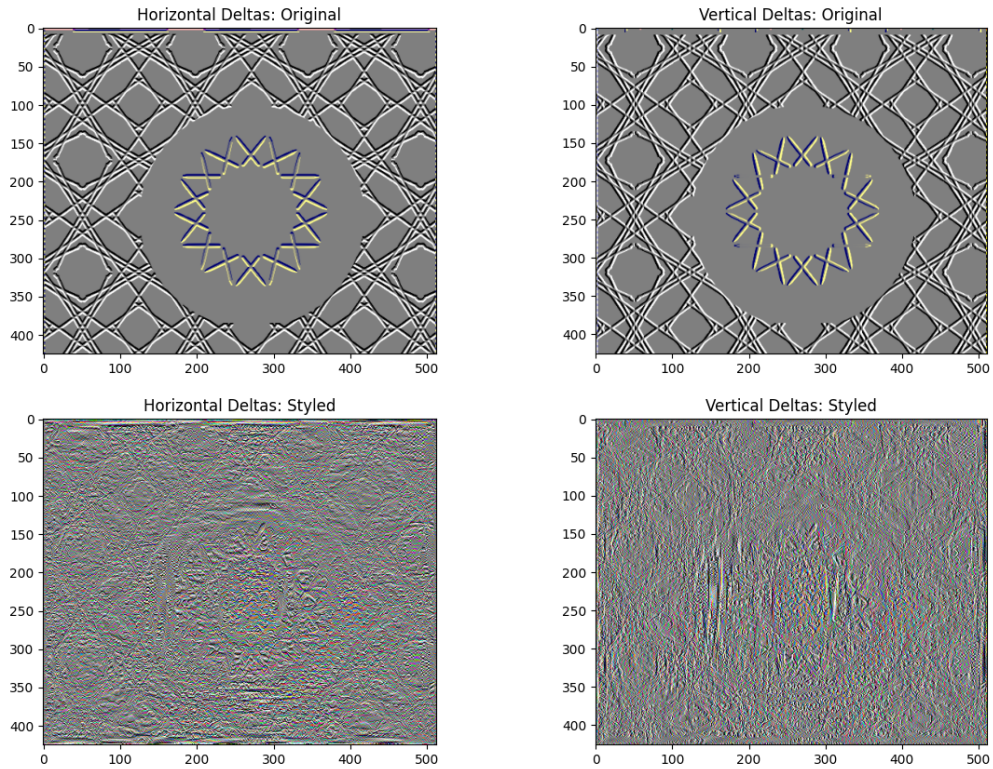
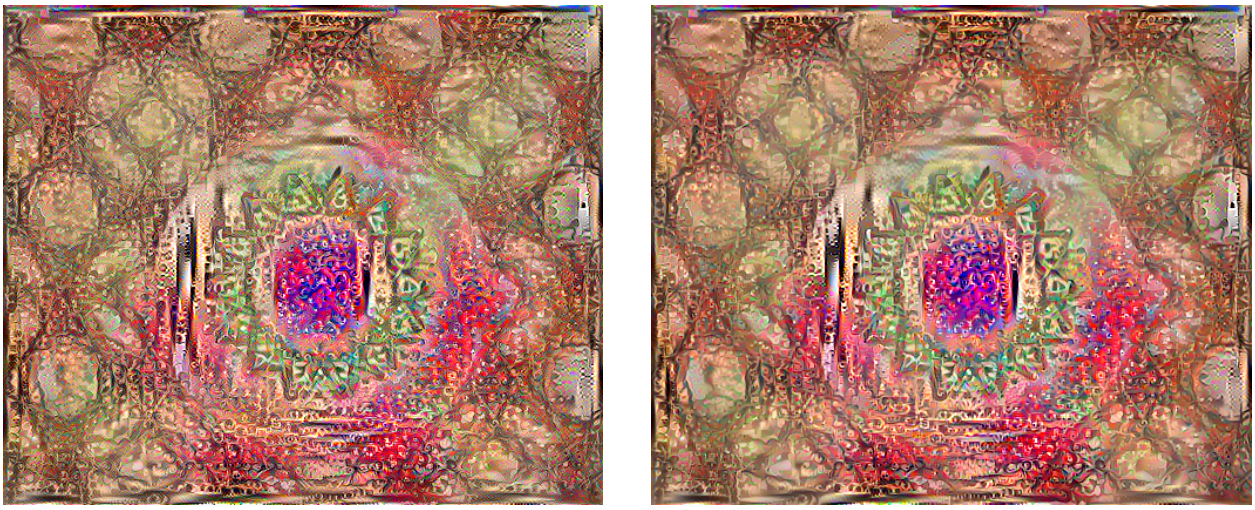


Figure 8.10: High-frequency artifacts as seen from edge detection algorithm(Sobel Edge Detection)



(a) Before introducing variation loss

(b) After introducing variation loss weight.

Figure 8.11: Comparison of the style transfer output before and after introducing variation loss (trained upto 1000 steps)

9. Future Steps

The journey of improving carpet design generation is an exciting exploration of advanced technologies, primarily leveraging Generative Adversarial Networks (GAN), Pix2Pix, and Style GAN. As we reflect on past accomplishments, it is imperative to outline future steps that will drive innovation and enhance the capabilities of our system.

Advancements in GAN, Pix2Pix, and Developing Style GAN Models

Our primary objective centers on the refinement and evolution of existing GAN, Pix2Pix, and style neural network models, leveraging advancements and pre-trained weights. Our ongoing efforts have included continuous training for approximately 100 epochs in Pix2Pix, yielding acceptable results. However, the DCGAN architecture has proven less effective, possibly due to inherent weaknesses in its design. Despite extensive training spanning two months and utilizing significant computational resources, the outcomes remain suboptimal. As we move forward, we face the decision of either enhancing the architecture and initiating training from scratch or leveraging pre-trained models and upgrading them to meet our specific requirements.

Moreover, StyleGAN stands out as a complex and highly effective model capable of generating a wide variety of human faces with remarkable realism. Its optimization towards human face generation presents an intriguing avenue for adaptation to our domain. One approach involves modifying the existing architecture to tailor it towards carpet image generation, while another entails the utilization of pre-trained StyleGAN models from NVIDIA and fine-tuning them with our rug images. This forward-thinking step poses both challenges and excitement as we embark on this innovative journey towards advancing the state-of-the-art in rug and carpet design.

Frontend Enhancements for User Interaction

In conjunction with the ongoing improvements to our models, significant enhancements are planned for the frontend of our system to deliver a more user-friendly and interactive experience. Presently, the fixed input size for carpets imposes constraints on creative flexibility. Our forthcoming initiatives include the implementation of interactive features enabling users to dynamically adjust the height and width of the carpet in real-time.

Furthermore, the integration of a **Random Input** button is scheduled to expedite the

carpet design process by generating input randomly. This feature aims to enhance the design experience by providing diverse and spontaneous suggestions during the creative phase.

In summary, our future endeavors encompass a comprehensive approach to advancing both the AI models and the user interface. By refining the GAN, Pix2Pix, and style transfer models and integrating new AI models, we aim to elevate the quality of generated carpet designs. Concurrently, frontend enhancements will empower users with more interactive tools, fostering a creative and intuitive design process.

Additionally, while our system currently consists of separate components, there is room for improvement in terms of navigation. Consolidating these components into a unified navigation system akin to MSPaint's drawing tools could enhance user experience and streamline usability. Moreover, our testing of the actual grid for manufacturing has revealed that each cell contains the palette color index, which must be matched with the real carpet scale. To address this, users will provide the shape of the carpet along with the associated grid, where each centimeter corresponds to four pixels.

Another limitation of our current system is the lack of mapping between the generated image and its constituent parts. This poses challenges when users attempt to provide input again. To mitigate this issue, future iterations will involve filling the canvas with the generated image and enabling users to interact with it using drawing tools, allowing new drawings to overlay the previously generated image.

10. Discussion

The journey of developing a state-of-the-art carpet generation and manipulation system was marked by various challenges and iterative improvements. The initial approach of converting unsupervised data to supervised using clustering techniques like K-means and DBSCAN faced limitations in terms of computational efficiency, particularly with larger values of k for K-means and a higher number of neighbor points for DBSCAN. Subsequent attempts at rough image portion cutting showed promise but suffered from inaccuracies, prompting the shift towards supervised labeling of dataset components - "center," "body," and "edges."

The utilization of Generative Adversarial Networks (GANs), specifically DCGAN, offered a breakthrough in learning from the unlabeled dataset. However, resource-intensive training posed a constraint, leading to the exploration of Pix2Pix models with U-Net as a generator. While this approach yielded visually appealing results, it lacked diversity and acted primarily as a colorizer for various designs.

To introduce user feedback and enhance user involvement in the design process, ReactJS codes were implemented for Frieze/Rosette/Wallpaper patterns. The UI/UX integration with the Django backend encountered challenges with Pix2Pix's limited capabilities in providing satisfactory results for user-submitted designs.

Further advancements were achieved by integrating style transfer with VGGNet, a CNN-based computer vision model. This fusion resulted in more robust textures on the generated carpets, expanding the design possibilities. The inclusion of user-provided images and preferences allowed for a more personalized and diverse set of outcomes.

The development of a robust carpet design generation system presented several challenges, each met with innovative solutions that have significantly enhanced the overall functionality and user experience.

Managing Input Size and RAM Overload

Effectively managing input sizes to prevent RAM overload emerged as a primary challenge, with the potential consequence of program crashes. Deliberate observation and strategic planning played pivotal roles in achieving a suitable balance. By imposing size constraints and optimizing resource allocation, we successfully mitigated the risk of RAM overload. To expedite setup time, we adopted the memmap function to read bytes corresponding to 1000 images, eschewing the conventional storage of images in formats like .jpg. Instead, we preprocessed images into numpy arrays directly, enhancing efficiency. Notably, we refrained

from saving tensorboard logs to conserve system resources, ensuring a sustainable operation within the allocated 12-hour timeframe on Kaggle. The decision to implement checkpoints was made judiciously, with a checkpoint inserted at the 13th epoch before the session's termination.

Furthermore, the time required to train neural style transfer for each user request was notably high, approximately 100 seconds. However, leveraging precompiled versions from TensorFlow Hub significantly expedited the process compared to the approaches employed in DCGAN and Pix2Pix methods. In particular, these methods utilized pickle to save and load weights into the class, resulting in prolonged processing times. Conversely, while neural style transfer optimization up to 1000 epochs notably decelerated performance, the integration of TensorFlow Hub effectively addressed these inefficiencies, delivering expedited results.

User Interface Design and React Integration

Designing an intuitive and user-friendly interface posed another significant challenge. React components were employed to address this challenge, providing a structured and modular approach to interface development. Incorporating various design options into the model necessitated the use of React's **useRef** and **forwardRef** features. These components played a crucial role in ensuring efficient rendering, allowing for dynamic user interactions with the canvas.

The utilization of mouse movements, keyboard inputs (such as arrow keys), and other user actions became pivotal in optimizing the interface. By leveraging React's capabilities, we synchronized these inputs with the **canvas**, providing users with a seamless and responsive design experience.

Integration with Django and API Development

One of the obstacles was when we used neural style trasfer. given content image and style image, we needed to compute gram matrix first, then trin the model. it suffreed from high variation loss, giving more edges and to minimize it we included total variation loss thus giving more better result.

Integrating the system with Django and developing an API for carpet design generation presented its own set of challenges. A dedicated API was created to facilitate communication between the frontend and the GAN models. This API, integrated into the Django framework, loaded pre-trained model weights stored on Kaggle, allowing for efficient and scalable carpet design generation.

In conclusion, the evolution of the project from unsupervised clustering to GANs and eventually style transfer with VGGNet demonstrates the iterative nature of research and development. The incorporation of user feedback not only enhanced the creative aspect but also improved the practical usability of the system. Challenges were met with strategic shifts, leading to a more refined and effective solution for state-of-the-art carpet design.

11. Conclusion

The project embarked on the ambitious goal of creating a cutting-edge carpet generation and manipulation system, navigating through various methodologies and challenges. From the early attempts with clustering to the resource-demanding GAN models and the subsequent integration of neural style network with VGGNet, each phase contributed to the evolution of the system. The user-centric approach, introduced through ReactJS for pattern selection, marked a significant milestone in engaging users in the design process.

Despite the initial monotony and limitations faced with Pix2Pix models, the project's resilience in adapting and refining methodologies led to the development of a robust system capable of producing diverse, visually appealing, and user-influenced carpet designs. The integration of AI and user feedback showcases the potential for creating a more personalized and inclusive platform for carpet design enthusiasts.

In the future, further enhancements could involve exploring additional AI models, incorporating more advanced user interfaces, and expanding the dataset to foster even greater diversity in design outcomes. The project's journey exemplifies the continuous pursuit of innovation and improvement in the field of AI-driven creative applications.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[3] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2015.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018.

[6] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[7] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

[8] David Eck. Symmetry information. https://math.hws.edu/eck/js/symmetry/symmetry-info.html, 2023. Accessed on: 11 05 2023.

[9] Galaincha. https://galaincha.com.np/, 2023. Accessed: 2023-01-01.

[10] Explorug. https://explorug.com/, 2023. Accessed: 2023-01-01.

[11] Ramro design. https://ramrodesigns.com/, 2023. Accessed: 2023-01-01.

[12] Only1dollardesign. https://only1dollardesign.com/, 2023. Accessed: 2023-01-01.

[13] Onlyhundred. https://onlyhundred.com/, 2023. Accessed: 2023-01-01.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[15] Unspecified. Generalization of convolutional network to domain adaptation network for classification of disaster images on twitter. https://www.researchgate.net/figure/VGG-19-Architecture-39-VGG-19-has-16-convolution-layers-grouped-into-5-blocks-After$_fig5_359771670. Accessed : 12April2024.$