Instance Eraser using Instance Segmentation and Generative Learning

Renu Krishna Gutta
Department of ECE
UCSD
rgutta@ucsd.edu

Rohith Reddy Rachala
Department of ECE
UCSD
rrachala@ucsd.edu

Abstract

We propose an "Instance Eraser" algorithm that removes all instances of a specified object from an image and recovers the background at the removed spot. Our pipeline involves using state-of-the-art image segmentation models to segment a given image and remove all instances of a particular category based on user input. We then use an existing state-of-the-art image-to-image generative learning model to reconstruct the background at the removed spot. Our proposed algorithm has potential applications in image editing, video editing, medical image analysis, and satellite image analysis. We evaluate the performance of our algorithm on a large dataset of images and compare it with existing state-of-the-art algorithms. A pytorch implementation of the project can be found at: Github

1 Introduction

The problem we aim to solve in this project is to develop an algorithm that can remove all instances of a specified object from an image and recover the background at the removed spot. This project is inspired by Google's MagicEraser feature on the Pixel smartphone. The objective is to create a convenient solution for removing specific objects from an image and seamlessly replacing them with background elements that blend naturally with the surrounding environment. Our tool goes beyond manual selection by offering users a list of available object classes to choose from for removal. Once a class is selected, all instances of that class in the image are automatically erased and replaced with surrounding pixels to ensure smooth continuity with the background. To showcase its usability and functionality, we provide a user-friendly interface where users can upload their image and select the object class they want to remove.

The motivation for this problem is that often we may want to remove unwanted objects or distractions from an image while preserving the background. For example, in landscape photography, unwanted objects such as poles, wires, or buildings may obstruct the view and disturb the composition. In medical imaging, artifacts such as noise or motion blur may obscure the underlying image information. In satellite imaging, clouds or other atmospheric interference may obstruct the view of the earth's surface. Therefore, developing an algorithm that can remove these unwanted objects while preserving the background can be highly beneficial in various applications.

The key parts of the problem are instance segmentation, instance removal, and image reconstruction. Image segmentation involves dividing the image into different regions based on their similarity or dissimilarity. Instance removal involves identifying and removing all instances of a specified object category from the image. Image reconstruction involves filling in the removed areas with meaningful background information to preserve the image's visual coherence.

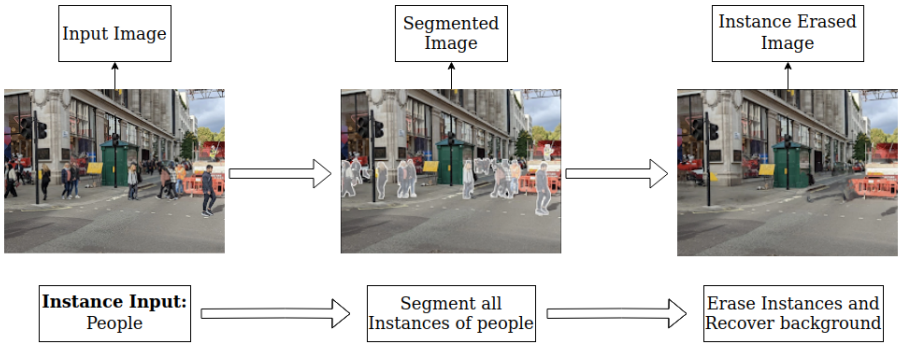

Figure 1: Method overview

2 Method Overview

Our method as referenced in Fig.2 for developing the "Instance Eraser" algorithm involves the following steps:

- **Dataset Generation**: To begin, we use exisiting benchmark dataset for training our image segmentation model. We later modify the same dataset by randomly whitening out background regions (not objects) to create inputs for training the image inpainting models.
- **Instance segmentation**: The image segmentation step involves utilizing techniques such as Mask R-CNN with a U-Net decoder or using a pre-trained Mask R-CNN model. This process divides the input image into different regions based on their similarity or dissimilarity.
- **Instance removal**: Once the image has been segmented, we proceed to the instance removal step. Users will provide input specifying the object class they want to remove. Using the image segmentation results, we identify and remove all instances of the specified object category from the image. The removed object pixels are replaced with white background pixels.
- **Image reconstruction**: The final step is image reconstruction, also known as image inpainting. The instance-removed image obtained from the previous step serves as input to a generative model. We have explored multiple approaches, including Pix2Pix (supervised learning) with a ResNet generator, Pix2Pix with a U-Net generator, and CycleGAN (unsupervised learning). These models are trained to produce a reconstructed image that maintains visual coherence by filling in the removed areas with meaningful background information.

The reason for choosing this method is that it combines state-of-the-art techniques from computer vision and generative modeling to develop an effective and efficient algorithm. Instance segmentation allows us to identify and segment different regions of an image, while instance removal enables us to remove specific instances of a specified object category. Finally, image reconstruction enables us to fill in the removed areas with meaningful background information.

3 Method Description

3.1 Dataset Generation

For instance segmentation tasks, we utilized the COCO-2017 dataset. But for image inpainting problem, during the training phase, our objective was to teach the model to fill in white patches in an image with background information, seamlessly blending the filled-in areas with the rest of the picture. However, we needed to prepare a dataset that focused on generating background information rather than object information.

To create this dataset, we utilized the COCO dataset, which provides rich annotations for object segmentation. By leveraging these annotations, we generated masked polyfill images, where object segments were colored black and the background was represented by white pixels. Next, we employed

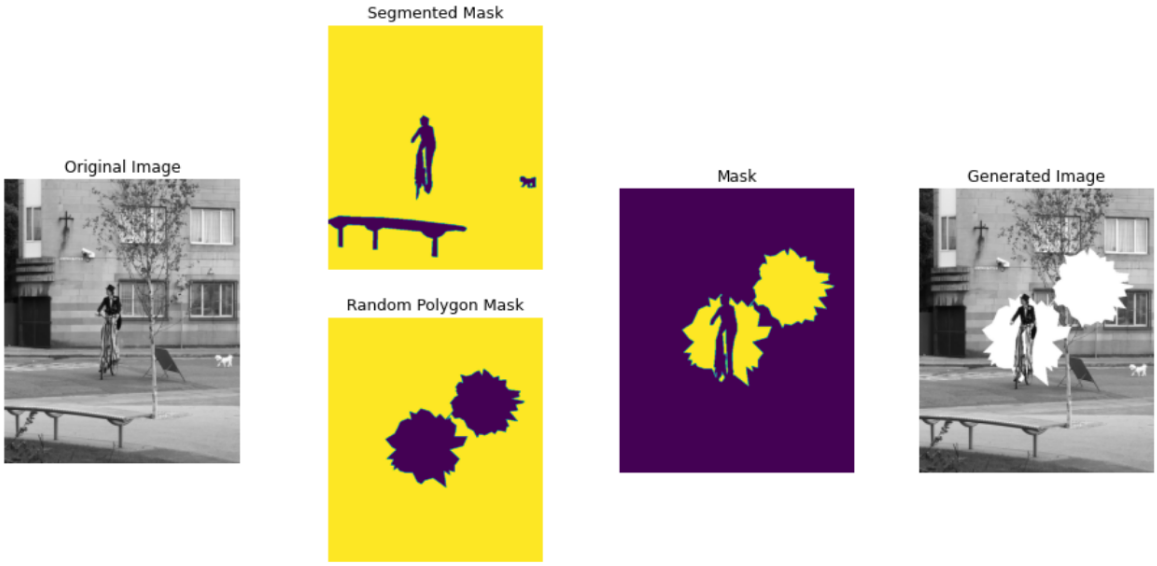

Figure 2: Flowcart of Generating Datasets

a novel approach to generate random polygons. These polygons were overlaid onto a black image of the same dimensions as the input image.

Steps followed in dataset generation process, an example flowchart is showed in Fig.2:

1. Using COCO dataset and segmentation annotations. A masked segmented image is generated, where objects are depicted with black pixels and the background with white pixels.
2. Create random polygons with irregularity and spikes, ranging from 40 to 50 vertices.
3. Overlay the polygons onto a black image, maintaining the same size as the input image.
4. Fill the interior of the polygons with white pixels to represent the background.
5. Perform a logical AND operation b/w the segmented image and the random polygon image to obtain the final mask.

Dataset Significance: The creation of the Instance Eraser dataset offers several advantages for training models to recover background from images. By randomly erasing the background without affecting the foreground objects, the dataset enables models to learn the intricate task of reconstructing the background accurately. This approach ensures that the models focus specifically on background recovery, enhancing their ability to restore images to their original state.

3.2 Instance Segmentation

3.2.1 MaskRCNN

Mask R-CNN (Mask Region-based Convolutional Neural Network) is a popular deep learning architecture for object detection, instance segmentation, and image segmentation tasks. It is an extension of the Faster R-CNN framework, incorporating an additional branch for pixel-level segmentation masks.

The architecture of Mask R-CNN as shown in Fig. 3 consists of three main components: a backbone network, a region proposal network (RPN), and a mask prediction network.

1. Backbone network: The input image is passed through the backbone network - a ResNet50 architecture, which consists of multiple residual blocks. Each residual block performs convolutional operations to extract features from the input image. The network gradually reduces the spatial dimensions of the feature maps while increasing their depth, capturing both low-level and high-level features. This network is pre-trained on the COCO dataset,

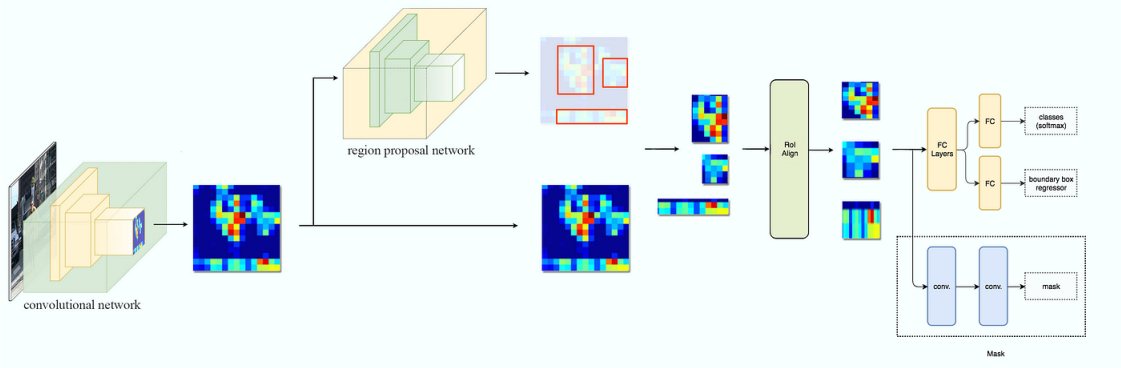

Figure 3: Mask RCNN Architecture

enabling it to learn generic visual representations. The backbone network is responsible for extracting high-level features from the input image. Typically, a convolutional neural network (CNN) such as ResNet or VGGNet is used as the backbone. These networks are pre-trained on large-scale image classification datasets like ImageNet, enabling them to learn generic visual representations.

2. Region Proposal Network (RPN): The RPN generates region proposals, which are potential bounding boxes containing objects of interest. It operates on the feature maps generated by the backbone network and uses anchor boxes of various scales and aspect ratios to propose object locations. The RPN scores each anchor box based on its likelihood of containing an object and refines the box coordinates to tightly fit the objects.
3. Mask Prediction Network: The mask prediction network takes the proposed regions from the RPN and performs two tasks: object classification and instance segmentation. For object classification, it predicts the class label for each proposed region. For instance segmentation, it generates a binary mask for each region, assigning a value of 1 to the pixels belonging to the object and 0 to the background.

During training, Mask R-CNN requires annotated data with bounding box labels and pixel-level segmentation masks. It uses a multi-task loss function that combines the losses for bounding box regression, object classification, and mask prediction. This enables end-to-end training of the network.

3.2.2 Pretrained MaskRCNN

Pretrained models of Mask R-CNN on the COCO (Common Objects in Context) dataset are widely available. COCO is a large-scale dataset with over 80 object categories and diverse images. Pretraining on COCO allows the models to learn general object representations and provides a good starting point for various computer vision tasks. These pretrained models can be fine-tuned on specific datasets or used as feature extractors for transfer learning.

By leveraging the power of Mask R-CNN, we can perform accurate object detection, instance segmentation, and image segmentation tasks. The pretrained models on COCO provide a valuable resource for researchers and practitioners, enabling them to build upon the learned representations and achieve high performance in their specific applications.

3.3 Image Inpainting

3.3.1 Pix2pix

Pix2Pix is a popular generative adversarial network (GAN) architecture that focuses on image-to-image translation tasks. It is designed to learn a mapping between an input image and an output image, typically with a conditional setting where the output image depends on the input image. Pix2Pix has been successfully applied to tasks such as image colorization, image style transfer, and image-to-image translation.

The architecture of Pix2Pix as shown in Fig. **??** consists of two main components: a generator network and a discriminator network.

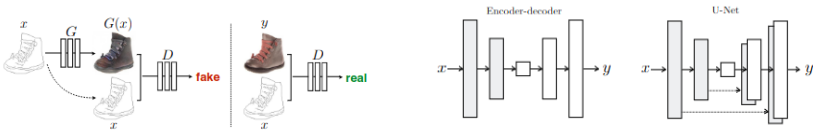

Figure 4: **Left**: Conditional GAN **Right**: Two choices for the architecture of the generator. The "U-Net" is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

1. Generator Network: The generator takes an input image and aims to generate a corresponding output image that matches a desired target. It consists of an encoder-decoder structure, often based on a convolutional neural network (CNN). The encoder part of the network gradually reduces the spatial dimensions of the input image while capturing its high-level features. The decoder part upsamples the encoded features to generate the output image. The encoder and decoder are connected through skip connections, allowing information from different scales to be combined effectively. By training the generator to minimize the difference between the generated image and the target image, it learns to produce realistic and visually coherent outputs.
2. Discriminator Network: The discriminator network is responsible for distinguishing between real images and the images generated by the generator. It takes both real and generated image pairs as input and outputs a probability score indicating the likelihood of the image being real. The discriminator is typically a convolutional neural network that processes the image pairs and performs binary classification.

During training, Pix2Pix utilizes an adversarial training scheme. The generator and discriminator are trained in a competitive manner, with the generator aiming to produce more realistic images to fool the discriminator, while the discriminator aims to correctly classify real and generated images. This adversarial process helps the generator learn to generate high-quality images that resemble the target distribution.

To compute the loss during training, Pix2Pix combines an adversarial loss, which encourages the generator to produce realistic outputs, with a task-specific loss, which measures the difference between the generated output and the target image based on the specific image-to-image translation task.

The strength of Pix2Pix lies in its ability to generate highly detailed and visually plausible output images. The conditional setting allows it to learn complex mappings between different visual domains. The skip connections in the generator network facilitate the preservation of fine-grained details from the input to the output. By training the generator and discriminator together, Pix2Pix achieves a more stable and effective training process compared to earlier GAN architectures.

3.3.2 CycleGAN

CycleGAN is a powerful deep learning architecture for unpaired image-to-image translation. Unlike Pix2Pix, which requires paired training data with corresponding input-output image pairs, CycleGAN can learn the mapping between two image domains using only unpaired images from each domain. This makes it particularly useful when paired data is difficult or expensive to obtain.

The architecture of CycleGAN as shown in Fig. **??** consists of two generator networks and two discriminator networks:

1. Generator Networks: CycleGAN employs two generator networks, G_A and G_B. G_A aims to translate images from domain A to domain B, while G_B performs the reverse translation from domain B to domain A. Each generator follows an encoder-decoder structure similar to Pix2Pix, utilizing convolutional neural networks (CNNs) to learn the mapping between the input and output images.

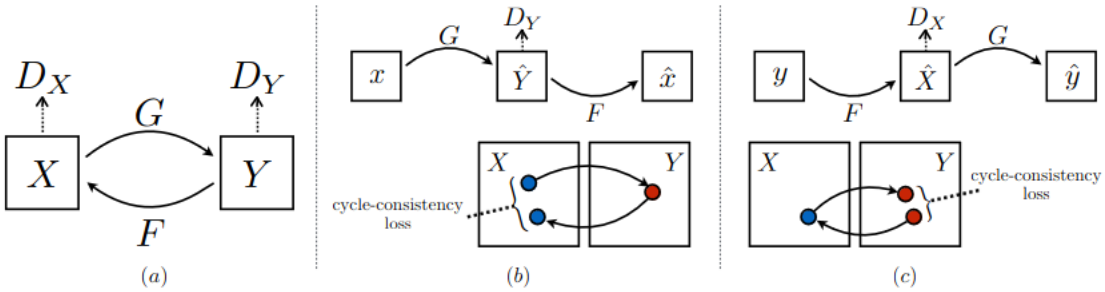

Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

2. Discriminator Networks: CycleGAN also includes two discriminator networks, D_A and D_B. D_A is trained to differentiate between real images from domain A and translated (fake) images produced by G_A. Similarly, D_B distinguishes between real images from domain B and translated images from G_B. The discriminators are CNN-based classifiers that assess the realism of the input images.

The key concept in CycleGAN is the cycle-consistency loss, which enables the network to learn mappings between two domains by enforcing consistency when translating an image to the other domain and back. This loss ensures that if an image from domain A is translated to domain B and then translated back to domain A, it should resemble the original image. The same principle applies to images from domain B.

During training, the generators aim to minimize the adversarial loss by fooling the discriminators into classifying the translated images as real. Additionally, they minimize the cycle-consistency loss by reconstructing the original input image after the double translation. The discriminators, on the other hand, try to correctly classify real and fake images, providing feedback to improve the generators' translation quality.

The overall objective of CycleGAN is to learn two mappings, G_A and G_B, that can successfully translate images between the two domains, while maintaining the content and style of the original images. The cycle-consistency loss ensures that the translations are meaningful and consistent.

4 Experiments

4.1 Instance Segmentation

4.2 Image Inpainting

In our pursuit of developing an effective solution for recovering background in images, we experimented with various generative models. Specifically, we trained Pix2pix models with ResNet and UNet as generators, as well as a CycleGAN model. Our objective was to assess the performance of these models and determine the most successful approach in terms of achieving desirable results.

1. Pix2pix with ResNet Generator: We employed the Pix2pix architecture with a ResNet generator to train our first model. This generator is known for its ability to capture long-range dependencies and generate high-quality images. We carefully optimized the hyperparameters to ensure optimal performance. This code was implemented by us
2. Pix2pix with UNet Generator: In our second approach, we utilized the Pix2pix framework with a UNet generator. The UNet architecture has proven to be effective in various image-to-image translation tasks, including segmentation and reconstruction. We fine-tuned the hyperparameters and architectural aspects to maximize the performance of this model.

3. CycleGAN: As an additional experiment, we trained a CycleGAN model, which enables image translation between two domains without requiring paired training data. This model has been successful in tasks involving style transfer and domain adaptation. We adapted the CycleGAN architecture to suit our object removal objective.The details and outcomes of these experiments are discussed further in the results section.

Training and Evaluation: During the training phase, we utilized the Instance Eraser dataset, which we created by combining the COCO dataset with our specialized masking technique. We divided the dataset into training and validation sets to monitor the models' progress and prevent overfitting.

To evaluate the performance of the models, we conducted extensive qualitative and quantitative analyses. We visually inspected the output images generated by each model to assess the quality of the background recovery and the completeness of object removal. Additionally, we employed standard evaluation metrics such as Peak Signal-to-Noise Ratio (PSNR) to quantify the similarity between the generated images and the ground truth.

Inference: During the inference phase, our focus was on identifying and removing pixels that contained object information while leaving the background intact. To achieve this, we employed pretrained instance segmentation models. Among the models we tested i.e, Pretrained MaskRCNN demonstrated good perforamce and was light weight compared to the MaskRCNN we implemented. For inference, we initially processed the input image using MaskRCNN to detect objects and provide pixel masks corresponding to the identified objects. Users were then able to select the objects they wished to remove. The chosen pixels were cleared out, and the resulting image was passed through the generative model to fill in the cleared patches.

In the results section, we present several examples showcasing the performance of our model in removing objects from images and filling the resulting voids with realistic background information.

5 Analysis and Results

The Table.1 presents the Average PSNR values obtained for different generative models used in our experiments. PSNR serves as a metric to measure the quality of the generated images, with higher values indicating better image quality. From the table, it is evident that the Pix2pix model with ResNet as the generator, trained from scratch for 30 epochs, achieved the highest Average PSNR of 20.87. This suggests that this model performed the best in terms of preserving image details and overall image quality.

Experiment Name	**Average PSNR**
Pix2pix with UNet (Pretrained, 20 epochs)	12.72
Pix2pix with UNet (Trained from scratch, 20 epochs)	18.64
Pix2pix with Resnet (Trained from scratch, 30 epochs)	20.87
CycleGAN (Trained from scratch, 15 epochs)	17.28

Table 1: Average PSNR of Different Generative Models

Below are the results of a few successful or distinctive hand picked examples to better understand the performance of our models.

6 Observations and Conclusion

Based on the aforementioned results, it is evident that the model performed satisfactorily in scenarios where the object of interest was set against a plain background or a consistent pattern throughout the image. However, challenges were encountered when dealing with large objects or backgrounds that exhibited high variation or overcrowding. Notably, even pretrained generative models, including those trained for different tasks, struggled to handle even simple cases effectively. The most successful approach involved training the model from scratch using a carefully curated dataset.

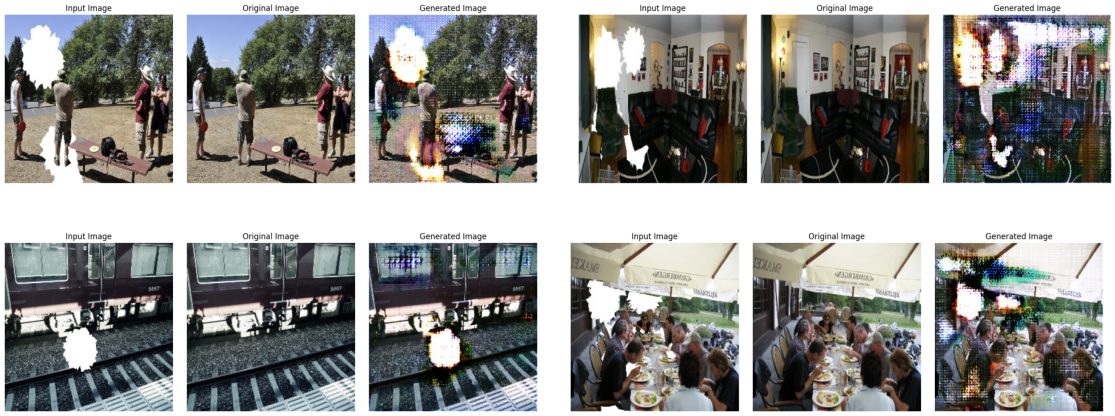

Figure 5: Pix2Pix Architecture with Resnet as a Generator using pretrained weights of a different task. Upon finetuning for 20 epochs on our curated dataset, the model fails to provide any tangible results. The patches and the rest of the images were filled with noise.

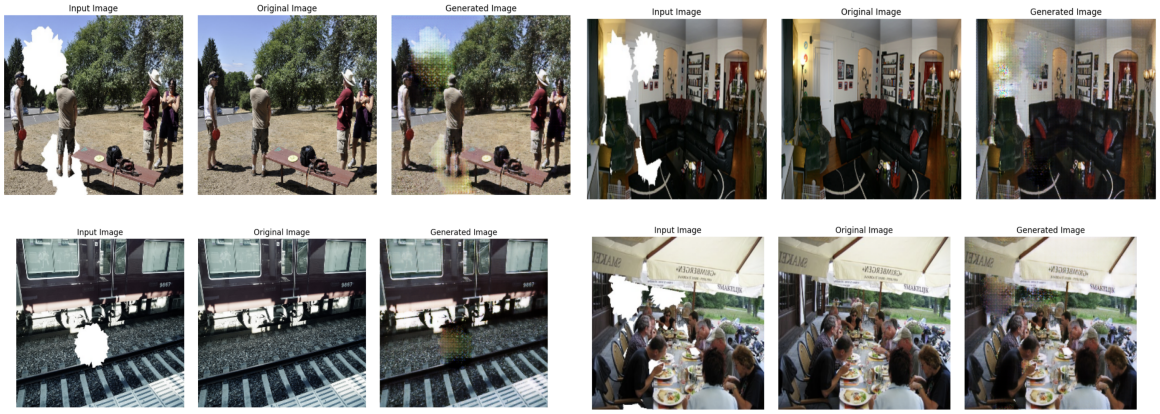

Figure 6: The same Pix2Pix Architecture with Resnet but trained from scratch using the curated dataset for 20 epochs. The model performs much better than the previous model by filling the background with relative contextual information rather than noise.

The observed limitations of pretrained models highlight the difficulty posed by the dataset created for this project, particularly in providing sufficient training information for generative models. Further exploration through extensive studies on the performance of successful models would be valuable in understanding the viability of similar approaches to dataset curation. Notably, successful approaches in our research involved creating image patches that encompassed the object information, allowing models to learn object removal during the training phase.

To address the challenges in complex cases, it was found that increasing the number of examples in the dataset and extending the training duration yielded improvements. While CycleGAN demonstrated potential in a few instances, its average performance in generating consistent backgrounds fell short compared to other models. Due to limitations in time and resources, experimentation with other generative models was not feasible, leading to the selection of Pix2Pix as the preferred model for this project.

In conclusion, this project has provided valuable insights and learning experiences for all participants, contributing to the broader understanding of generative models and image inpainting. The findings can be applied to future projects in this field, paving the way for advancements in generative modeling techniques.

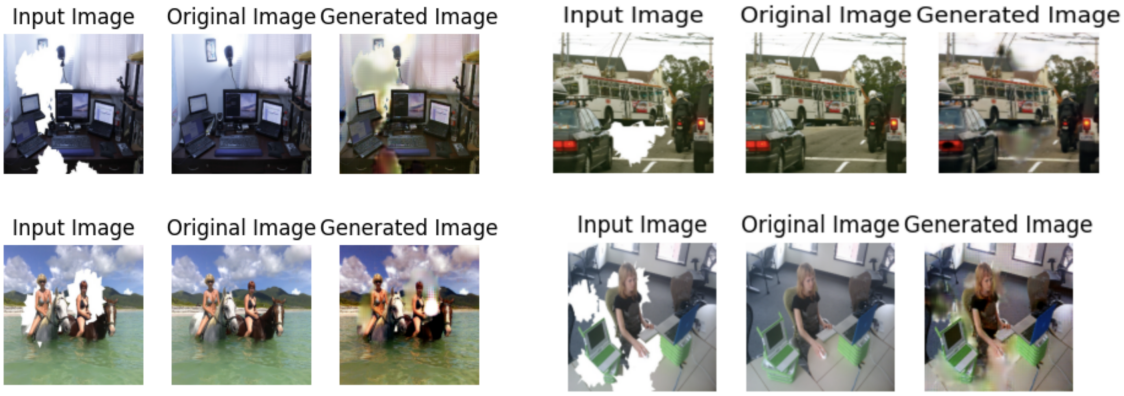

Figure 7: Our most successful model – Pix2Pix Architecture with UNet as the generator, trained from scratch for 30 epochs. Performs well in complex situations with varying background, in which the other models performed very poorly.

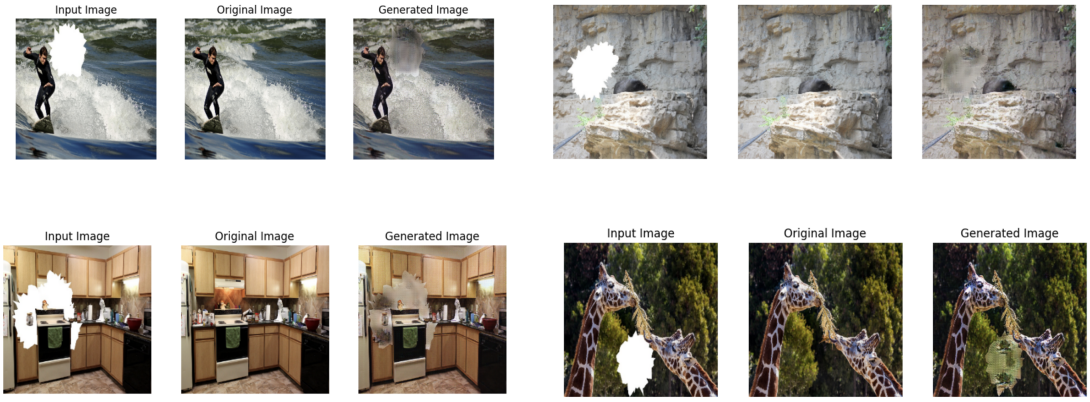

Figure 8: Alternative generative model using CycleGAN. Performs well in a few cases but on an average subpar image generation compared to Pix2Pix.

7 Acknowledgement

The code for Pix2pix with ResNet as generator and MaskRCNN with decoder is implememted by us whereas the Pretrained MaskRCNN, pix2pix with UNet and CycleGan is directly used from github.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[4] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

[5] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016.

[6] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6517–6525, 2017.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer, 2015.

[9] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2242–2251, 2017.

[10] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.