



Indian Institute of Technology, Guwahati
Department of Data Science and Artificial Intelligence

Object Removal using Mask R-CNN

Project Report

Submitted by:

Poojitha Goli

Roll No: 220150003

Course: DA312

Instructor: Dr. Chiranjib Sur

A project report submitted as part of the curriculum
for the 6th semester B.tech
in *Data Science and Artificial Intelligence*

April 18, 2025

Declaration

I, Poojitha Goli, of the Department of Data Science and Artificial Intelligence, University of Reading, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

Poojitha Goli
April 18, 2025

Abstract

Image post-processing is a technique used to enhance the quality of images captured by a camera. One interesting problem under this category is the removal of unwanted objects from an image. The challenge begins with identifying the object of interest, as manually outlining or highlighting it can be time-consuming. To address this, deep neural networks are employed to automatically generate a mask for the object, which helps identify the exact pixels that need to be removed. The second challenge is filling in the missing pixels after object removal while preserving the integrity of the background. The system must infer the background hidden behind the object from the context provided by the image.

This project combines image segmentation and inpainting techniques to enable users to remove objects from images efficiently. Two state-of-the-art models, Mask R-CNN and DeepFillv2, are used in this system, with an Intersection Over Union (IoU) calculation to assist in object selection. We evaluate the system using images from the MS-COCO dataset as well as images taken on an iPhone. The effectiveness and limitations of the models are demonstrated through several examples, and insights into the system's performance are provided.

Keywords: Object Removal, Mask R-CNN, DeepFillv2, Image Segmentation, Image Inpainting

Report's Total Word Count: *[Enter word count here]*

Program Code: The program code for this project is available on Github: [\[https://github.com/Poojitha237/object_remove/tree/main/object_remove-main\]](https://github.com/Poojitha237/object_remove/tree/main/object_remove-main)

Contents

List of Figures	v
List of Tables	vi
1 Introduction	1
1.1 Background and Motivation	1
1.2 Problem Statement	1
1.3 Objectives	1
1.4 Scope of the Project	1
1.5 Tools & Technologies Used	2
1.6 Structure of the Report	2
2 Literature Review	4
2.1 Early Methods	4
2.2 Deep Learning-Based Segmentation	4
2.3 Generative Inpainting	4
2.4 Integrated Approach for Object Removal	5
2.5 Conclusion	5
3 Methodology	6
3.1 Dataset Description	6
3.2 System Overview	6
3.3 System Architecture	6
3.4 Object Segmentation using Mask R-CNN	7
3.4.1 Handling Multiple Masks in a Bounding Box	7
3.4.2 Handling Ambiguities with Identical Proportions	8
3.4.3 Advantages of the IoU-Based Approach	8
3.5 Model Fine Tuning	9
3.6 Inpainting	9
3.6.1 The Problem of Missing Pixels	9
3.6.2 Gated Convolutions	9
3.6.3 Improvement with Gated Convolutions	10
3.6.4 Generative Adversarial Networks (GANs) for Inpainting	10
3.7 DeepFill v2 Architecture	11
3.8 Pipeline Workflow	11
4 Results and Evaluation	12
4.1 Example Results and Analysis	12
4.2 Observations and Limitations	15

5 Conclusion Future works	16
6 User Guide	17
6.1 How to Implement	17
6.1.1 Step 1: Clone the Repository	17
6.1.2 Step 2: Install Dependencies	17
6.1.3 Step 3: Download Pre-trained Weights	17
6.1.4 Step 4: Run the Application	17
6.2 User Controls	18
6.3 Notes	18
References	19

List of Figures

3.1	System pipeline for object removal using Mask R-CNN, IoU-based selection, and DeepFill v2 inpainting.	7
3.2	Left: Two masks that have pixels inside the bounding box. Right: Multiple masks entirely inside the bounding box.	7
3.3	Comparison of Intersection over Union (IoU) for different bounding boxes. . .	8
4.1	Example 1: Outdoor Park Scene — (a) Original image with selection, (b) Masked result, (c) Inpainted result.	13
4.2	Example 2: Mountain Viewpoint Scene — (a) Original image with selection, (b) Masked result, (c) Inpainted result.	13
4.3	Example 3: Group Photo Setting — (a) Original image with selection, (b) Masked result, (c) Inpainted result.	14
6.1	Graphical User Interface for drawing bounding box and removing the object .	18

List of Tables

1.1 Tools and technologies utilized in the project 2

List of Abbreviations

SMPCS School of Mathematical, Physical and Computational Sciences

Chapter 1

Introduction

1.1 Background and Motivation

Removing objects from a picture is a difficult but fascinating process that calls for getting rid of undesired features. Often, when taking pictures, we come across distractions or obstructions we want to get rid of. Although finding the item to be removed is fairly simple, the actual difficulty is in rebuilding the background pixels concealed behind the object. This project uses artificial intelligence especially deep neural networks to both remove objects and smartly infer the missing areas, therefore addressing that difficulty.

1.2 Problem Statement

Segmenting the object to be removed and then realistically inpainting the missing area to match the background are the two steps that make up the object removal challenge. A sophisticated, accurate pixel-level prediction is needed for this. Our project uses DeepFillv2 for inpainting and Mask R-CNN for segmentation to address these issues.

1.3 Objectives

The primary objectives of this project are:

- To build an automated pipeline for object removal in images.
- To utilize Mask R-CNN for object detection and segmentation.
- To implement DeepFillv2 for context-aware inpainting.
- To enable user interaction through object selection using Intersection over Union (IoU).
- To evaluate the effectiveness and limitations of the system on both benchmark (MS-COCO) and real-world images.

1.4 Scope of the Project

This project shows how deep learning techniques can be applied effectively to the task of object removal from images, which has broad implications for a variety of fields. The scope consists of:

- **Photo Editing and Enhancement:** Offering a user-friendly tool to both novice and expert editors to eliminate unwanted elements from photos, enhancing their visual quality without the need for manual labor.
- **Surveillance and Security:** Helping to anonymize or de-identify sensitive subjects in security footage while preserving contextual integrity is part of surveillance and security.
- **Digital Restoration:** Digital restoration is the process of helping to rebuild damaged artwork, old photos, or other visually significant images when portions of the image are obscured or missing.
- **Content Moderation:** Automating the removal of inappropriate or unwanted content from images shared on social media or online platforms.

Future advancements like real-time video object removal, integration with web or mobile applications, and the use of more sophisticated generative models for improved visual coherence can all be made possible by this project.

1.5 Tools & Technologies Used

Tool/Library	Purpose
Python 3.x	Programming language used for scripting and backend logic
PyTorch	Framework for implementing deep learning models
Torchvision	Provides access to pre-trained Mask R-CNN models
OpenCV	Used for image processing and GUI interactions
NumPy	Supports efficient numerical and array operations
Matplotlib	For visualizing images and results

Table 1.1: Tools and technologies utilized in the project

1.6 Structure of the Report

The rest of this report is organized as follows:

- **Chapter 2: Literature Review**
Discusses related work in the field of object detection, segmentation, and image inpainting.
- **Chapter 3: Methodology**
Outlines the methods used, including model architecture, dataset preparation, and the pipeline design.
- **Chapter 4: Implementation**
Describes the technical implementation of the system and integration of various components.
- **Chapter 5: Results and Evaluation**
Presents experimental results, evaluation metrics, visual outputs, and performance analysis.

- **Chapter 6: Discussion**
Interprets the results and discusses the effectiveness and limitations of the approach.
- **Chapter 7: Conclusion and Future Work**
Summarizes the report and suggests possible directions for future improvements or extensions.
- **Chapter 8: References**
Lists the sources, research papers, and tools referenced throughout the report.
- **Chapter 9 (i): User Guide**
Provides step-by-step instructions for running the code, dependencies, and GitLab access.
- **Chapter 9 (ii): Additional Results**
Includes any extra figures, tables, or experiments that support the main text.

Chapter 2

Literature Review

In the fields of image processing and computer vision, object removal is a thoroughly researched problem. The objective is to locate and remove particular objects from a picture, then visually complete the missing areas. Classical image inpainting techniques have historically been used for this task, but more intelligent and resilient approaches have surfaced with recent deep learning advancements.

2.1 Early Methods

To fill in the gaps in images, traditional techniques like **PatchMatch**, **texture synthesis**, and diffusion-based inpainting methods were frequently employed. These methods often struggle when dealing with large missing regions or scenes that are semantically complex, but they perform well in small or texture-consistent areas. They usually rely too much on nearby pixels and don't understand the global context, which results in artifacts or strange outputs.

2.2 Deep Learning-Based Segmentation

Convolutional neural networks (CNNs) have significantly improved object detection and segmentation. **Mask R-CNN** is one of the most well-known models; it was first presented by He et al. in 2017. It adds a branch for pixel-level segmentation mask prediction to the Faster R-CNN framework. Mask R-CNN is frequently used for example segmentation tasks and is trained on datasets such as **MS-COCO**, which comprises more than 80 object categories. Because of its capacity to produce precise object masks, it is especially helpful in applications where exact segmentation is essential, such as object removal.

2.3 Generative Inpainting

Recent advances in deep generative models have greatly improved image inpainting quality. Yu et al.'s **DeepFillv2** is one such model. Through the use of gated convolutions and a two-stage (coarse-to-fine) network architecture, the model is able to comprehend both the local textures and the global structure of images. Furthermore, it has a contextual attention mechanism that makes it easier to replicate pertinent textures from nearby areas, resulting in more realistic inpainting outcomes.

2.4 Integrated Approach for Object Removal

Segmentation and generative inpainting together provide a potent object removal pipeline. In this project, the user-selected object is precisely identified and masked using Mask R-CNN. After that, the masked image is sent to DeepFillv2, which adds realistic background content to the area that was removed. Even in complex scenes, this integrated approach ensures high-quality results while minimizing user input.

2.5 Conclusion

According to recent research, deep learning models have surpassed conventional techniques in both segmentation and inpainting tasks. This project demonstrates the efficacy of contemporary AI techniques in practical visual applications by utilizing cutting-edge models such as Mask R-CNN and DeepFillv2 to achieve accurate object removal with few artifacts.

Chapter 3

Methodology

Here we will describe the methods and techniques that are used to achieve object removal from images. In this process we will work on object segmentation and image inpainting to provide realistic processed outputs.

3.1 Dataset Description

We used the MS-COCO dataset for training and evaluation. It contains a wide variety of objects in natural scenes, with pixel-level annotations suitable for segmentation tasks. Additionally, we tested the pipeline on images captured using a smartphone to assess real-world performance.

3.2 System Overview

The system follows a two-stage pipeline. First, the object of interest is identified and segmented using Mask R-CNN. Next, the segmented region is passed to an inpainting model (DeepFill v2) to fill in the removed area with contextually relevant pixels.

3.3 System Architecture

The system follows a modular pipeline composed of three main components: object segmentation, object selection, and image inpainting. These modules work sequentially to achieve the goal of automatic object removal and background restoration.

The first step in the pipeline is **Object Segmentation**. In this phase, we use Mask R-CNN, a state-of-the-art deep learning model, to segment the object that needs to be removed from the image. The model is trained to detect various objects and create a mask, which marks the region of the object to be removed.

Following segmentation, the next step is **Object Selection**. In this phase, the user selects the object to be removed using an Intersection over Union (IoU) calculation. The IoU score ensures that the selected object is precisely defined and corresponds accurately to the segmented object.

Finally, in the **Image Inpainting** stage, the system reconstructs the background behind the removed object. This is done using DeepFill v2, an advanced inpainting model. DeepFill v2 leverages contextual information from the surrounding image to fill in the missing pixels, effectively restoring the image to a natural look without the removed object.

Figure 3.1 illustrates the overall architecture of the system and shows how these components interact to perform object removal and inpainting.

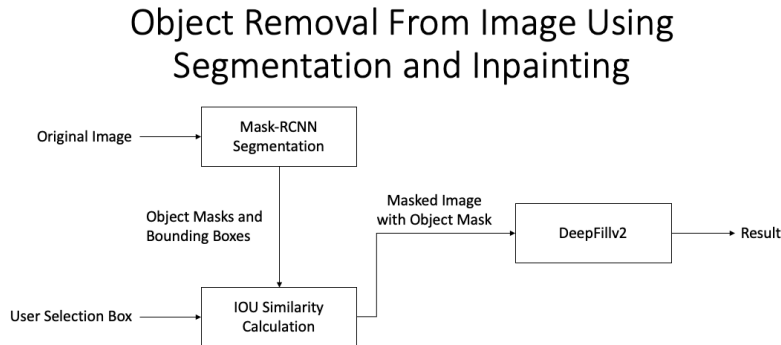


Figure 3.1: System pipeline for object removal using Mask R-CNN, IoU-based selection, and DeepFill v2 inpainting.

3.4 Object Segmentation using Mask R-CNN

The first method for object segmentation involves using Mask R-CNN to identify and create a mask of the object to be removed. This technique works well when there is a single mask within the user-defined bounding box, as shown in Figure 3.2. However, challenges arise when multiple masks are present within the bounding box.

3.4.1 Handling Multiple Masks in a Bounding Box

In scenarios where multiple masks are present, a slight modification to the basic Mask R-CNN approach is required. Instead of simply selecting any mask that intersects with the user-defined bounding box, we select the mask with the highest percentage of pixels inside the box. This approach uses proportion rather than absolute pixel count, making it invariant to object size.

The proportion of pixels inside the box is calculated by dividing the number of pixels inside the bounding box by the total number of pixels in the mask. The mask with the highest proportion is selected for removal. This method ensures that the mask most relevant to the user selection is used.

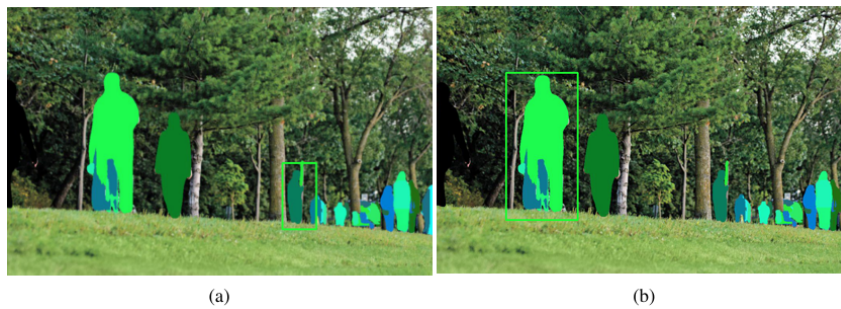


Figure 3.2: Left: Two masks that have pixels inside the bounding box. Right: Multiple masks entirely inside the bounding box.

3.4.2 Handling Ambiguities with Identical Proportions

One limitation of the above method occurs when multiple masks share the same proportion of pixels within the bounding box. In such cases, it becomes unclear which mask to select. For example, in Figure 3.2 (right), there are two masks entirely inside the bounding box, and they may have equal proportions of pixels within the box.

To address this, the second method involves comparing the bounding boxes of each object with the user-selected bounding box. Intersection over Union (IoU) is used to measure the overlap between the two bounding boxes. The object with the highest IoU score is chosen as the selected object for removal. The formula for calculating IoU is as follows:

$$\text{IOU}(\text{BoxA}, \text{BoxB}) = \frac{\text{Intersection}(\text{BoxA}, \text{BoxB})}{\text{Union}(\text{BoxA}, \text{BoxB}) + \epsilon}$$

Where ϵ is a small constant added to avoid division by zero.

This method solves both the problem of handling multiple masks and provides a more efficient calculation than counting individual pixels inside the masks. Additionally, the IoU calculation is computationally inexpensive and can be easily parallelized by stacking the bounding boxes and using Numpy for batch calculations.

3.4.3 Advantages of the IoU-Based Approach

Using IoU for mask selection has several advantages. It not only simplifies the computational complexity but also makes the process robust when there are multiple objects in the user-defined bounding box. The IoU-based method is faster, as it avoids the need to check every individual pixel in the masks. Moreover, this method allows for efficient parallel processing, which makes it scalable for large datasets.

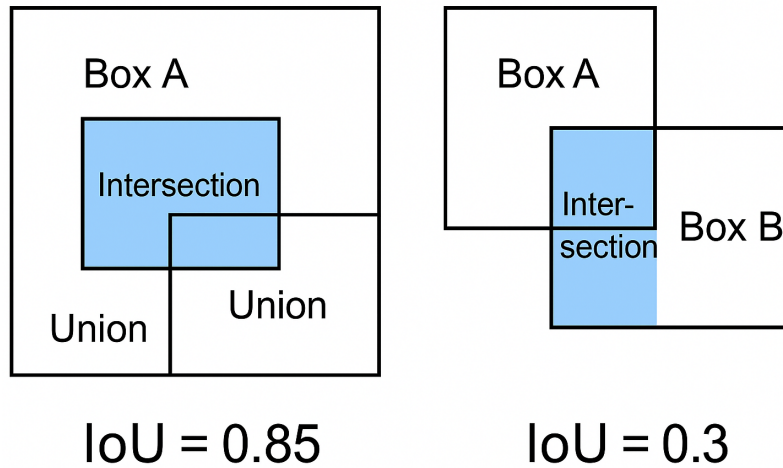


Figure 3.3: Comparison of Intersection over Union (IoU) for different bounding boxes.

By employing these two methods, the system ensures an accurate and efficient selection of the object to be removed, even in cases with multiple overlapping masks.

3.5 Model Fine Tuning

Instead of training a Mask R-CNN model from scratch, we decide to use a pre-trained Mask R-CNN model and perform fine tuning. Mask R-CNN model is trained from scratch on the Data since we lacked both labeled data and computational resources, which was a constraint in this project. Instead, we employed a transfer learning method, which lets us take advantage of knowledge already learnt by the model when trained on a large publicly available dataset such as COCO. For our solution, we applied a pre-trained Mask R-CNN with a ResNet-50 (Feature Pyramid Network) backbone that had been trained on the COCO dataset. It is compatible for transfer learning, hundreds of thousand labeled images in thousands of object classes in "COCO" large-scale object detection, segmentation, and captioning dataset. We froze the lower layers of the ResNet-50 backbone to make the model more appropriate for our custom use case. In lower layers, the model extracts basic visual features including edges and textures which work for most images and need not be retrained.

We change the way of fine-tuning from the feature extractor to the heads of the model, we choose to fine-tune only the box and mask prediction heads. More specifically, the higher layers were trained on our dataset so that the model is now tuned to the exact features within our object images.

This approach for fine-tuning drastically decreased the amount of time used to train the model compared to conducting the training from ground-up, and it also enabled the model to generalize more efficiently to our dataset with sporadic data. This transfer learning approach not only facilitated faster convergence but also enhanced the performance of the model, given that it could leverage the rich features learned from the COCO dataset.

3.6 Inpainting

The next step is to fill in the gaps once we have the object mask generated using Mask R-CNN. This problem reduces to filling in the missing pixels in the image, a technique called image inpainting. The model that handles this portion is DeepFill v2, a gated convolution-based generative network. Unlike typical computer vision tasks, image inpainting is a demanding problem where not all the pixels in the input image provide useful information. In fact, the masked-out pixels in the image can actually give false information if we used a traditional convolutional or feedforward network.

3.6.1 The Problem of Missing Pixels

A central difficulty in image inpainting is that not all image pixels are beneficial, especially the masked-out ones. If we didn't fix the hidden parts, we might add wrong details. The aim of the inpainting model is to forecast absent pixels in a manner that blends smoothly with the rest of the picture, maintaining context and architecture.

3.6.2 Gated Convolutions

The **DeepFill v2** model employs a novel approach called **gated convolution**, which is specifically designed to handle invalid, masked-out pixels throughout the network. Gated convolution is an extension of **partial convolutions**, a technique that avoids using invalid pixels by keeping track of the mask and adapting to the changing sizes of activations during the network's forward pass.

The partial convolution formula used in the original paper is as follows:

$$\text{Out_Partial}_{i,j} = \frac{W(X \odot M)}{\sum(M)} \quad \text{if } \sum(M) > 0, \text{ otherwise } 0$$

Where:

- W is the convolutional weights.
- X is the input region corresponding to the current convolution window.
- M is the corresponding mask region.

The mask update rule for partial convolutions is:

$$m'_{i,j} = \begin{cases} 1 & \text{if } \sum(M) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Although partial convolutions help to address the issue of invalid pixels, they have certain limitations. As the network deepens, the mask values are rigid (either 0 or 1), and each pixel might contain information about a larger region of the input that could include both valid and invalid pixels due to the increasing receptive field of each pixel location.

3.6.3 Improvement with Gated Convolutions

To address the limitations of partial convolutions, **gated convolutions** introduce a set of learnable gating weights that dynamically adjust during training. This allows the model to handle invalid and valid pixels more effectively.

The formula for gated convolutions is:

$$\text{Out_Gated}_{i,j} = \varphi(\text{Feature}_{i,j}) \odot \sigma(\text{Gating}_{i,j})$$

Where:

- $\varphi(\text{Feature}_{i,j})$ is the regular convolutional filter output with an activation function.
- $\sigma(\text{Gating}_{i,j})$ is the gating convolutional filter output with sigmoid activation, which constrains the output values between 0 and 1, providing soft mask values.

3.6.4 Generative Adversarial Networks (GANs) for Inpainting

The **DeepFill v2** model employs a **Generative Adversarial Network (GAN)** architecture, where two models are trained simultaneously: a **generator** and a **discriminator**. The generator's goal is to generate artificial images that look real, while the discriminator attempts to distinguish between real and fake images. This adversarial process helps the generator learn to produce images that are visually convincing.

This GAN-based structure is particularly suited for inpainting tasks because the generator is trained to synthesize realistic content for the missing pixels, effectively completing the image in a contextually plausible manner.

3.7 DeepFill v2 Architecture

The **DeepFill v2** architecture utilizes a **two-stage generator network**, comprising a **coarse network** and a **refinement network**. Both networks are encoder-decoder models, with gated convolutions replacing standard convolution layers. The coarse network generates an initial inpainted image, while the refinement network improves the inpainting quality by focusing on finer details and context.

The model uses **contextual attention** to further refine the inpainting, ensuring that the generated pixels are not only realistic but also contextually appropriate with respect to the surrounding pixels.

3.8 Pipeline Workflow

1. Input image is passed to the Mask R-CNN model.
2. Masks for all detected objects are extracted.
3. User or simulated input is used to select the desired object via IoU.
4. The object region is removed from the image.
5. The masked image is processed by DeepFill v2 to inpaint the removed area.
6. Final output is generated.

Chapter 4

Results and Evaluation

Since there are no ground truth labels for our inpainted results, we are unable to quantitatively measure the performance of our method. Instead, we visually evaluate the performance of the system using two example test images: one from the MS-COCO dataset and one taken using an iPhone. The MS-COCO dataset is a large-scale dataset containing over 300,000 everyday images with labeled objects. Mask R-CNN was trained on this dataset to identify 80 different classes of objects. As a result, our system is capable of identifying and removing those 80 object classes.

In the following sections, we present the results for each example, highlighting key aspects of each result and discussing the limitations of the system. Additionally, we will explore possible future improvements to the system.

4.1 Example Results and Analysis

Example 1: Outdoor Park Scene

In this example from the MS-COCO dataset, we see a scene with people on a grassy area surrounded by trees. The user has selected a person in the original image, as indicated by the bounding box. The system has successfully identified and masked this person in the middle image. The final inpainted result shows a clean removal of the selected individual, with the grass and background context preserved naturally. This demonstrates the system's effectiveness in handling outdoor environments with natural elements like grass and trees, which typically provide good texture patterns for inpainting algorithms.

Example 2: Mountain Viewpoint Scene

This image showcases a scenic mountain landscape with people on a rock outcrop. The user has selected a person sitting on the rock (highlighted with a green bounding box in the original image). The system accurately masks this individual in the middle panel. The inpainted result successfully removes the selected person while maintaining the natural appearance of the rock surface and preserving the distant mountain vista and sky. This example demonstrates the system's ability to handle complex natural textures like rock surfaces and distant landscape elements, which are particularly challenging for inpainting algorithms.

Example 3: Group Photo Setting

The example shows a group of children in what appears to be a sports facility or tennis court. Similar to earlier analysis, this demonstrates one of the system's limitations. When the user selects an area containing multiple objects or people (as seen in the top left panel), the system must determine which object to mask based on IoU (Intersection Over Union) calculations. In the masked image (top right), we can see that only one child has been selected for removal despite the bounding box potentially covering multiple individuals or

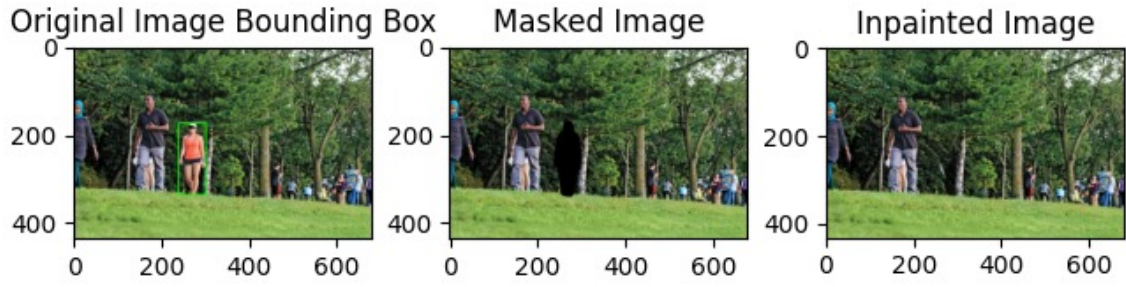


Figure 4.1: Example 1: Outdoor Park Scene — (a) Original image with selection, (b) Masked result, (c) Inpainted result.

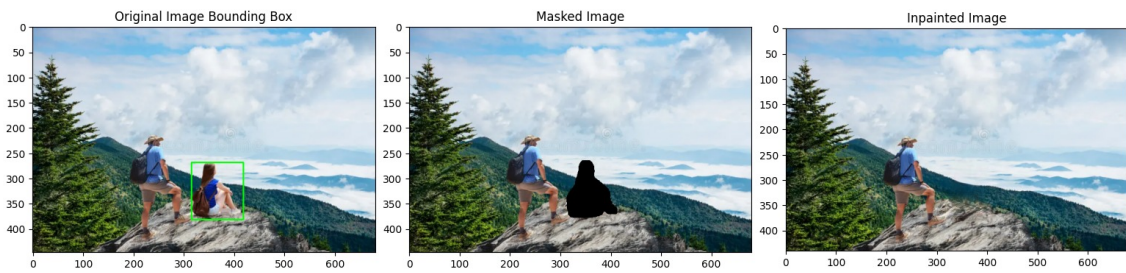
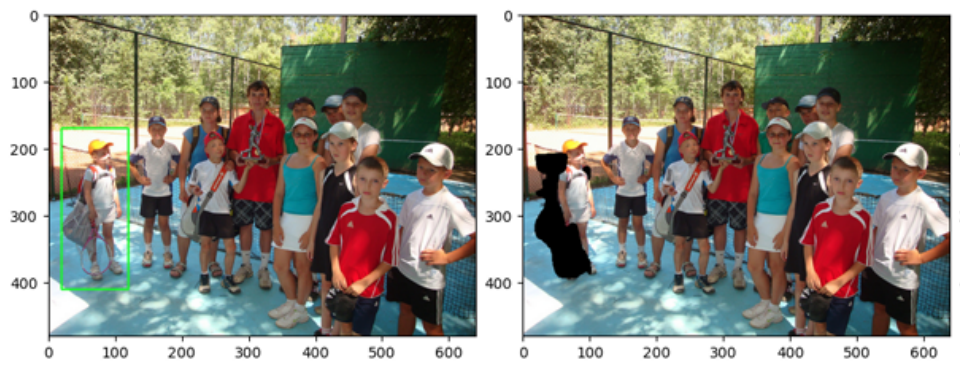


Figure 4.2: Example 2: Mountain Viewpoint Scene — (a) Original image with selection, (b) Masked result, (c) Inpainted result.



(a) User Selection Box and Masked Image



(b) Result

Figure 4: Image from MS-COCO

Figure 4.3: Example 3: Group Photo Setting — (a) Original image with selection, (b) Masked result, (c) Inpainted result.

equipment. This limitation prevents the user from easily removing a person along with items they are carrying or multiple adjacent objects with a single selection.

4.2 Observations and Limitations

Through these examples, we can observe that:

- The system effectively removes selected objects from various environments.
- Natural backgrounds like grass, trees, and rock surfaces are generally handled well.
- The current implementation has difficulty with compound object selection when multiple items need to be removed together.
- The quality of inpainting depends on the complexity of the background and surroundings.

Future improvements might include developing mechanisms for multiple object selection, enhancing the masking algorithm to identify connected objects, and implementing more sophisticated inpainting techniques for complex textures and lighting conditions.

Chapter 5

Conclusion Future works

5.1 Conclusion

In this project, we developed an integrated system for object removal from images by combining deep learning-based segmentation and inpainting techniques. The system utilizes Mask R-CNN for precise instance segmentation, enabling accurate identification and masking of user-selected objects through an IOU-based mask selection strategy. This approach reduces manual effort and increases the precision of object localization compared to traditional bounding box or manual annotation methods. Once the object is masked, DeepFillv2—a generative model with gated convolutions—is employed to inpaint the missing regions, producing visually coherent and realistic results. Experimental results on both MS-COCO and real-world images demonstrate that the system effectively removes a variety of object types while preserving the integrity of the background. Overall, the combination of automated mask generation and advanced inpainting provides a robust and user-friendly solution for object removal tasks.

5.2 Future Work

Although the current system demonstrates promising performance in object removal tasks, there remains significant scope for enhancement. One important direction is to support multi-object removal within a user-defined region. This would enable the system to handle more complex scenes where multiple overlapping or adjacent objects need to be removed simultaneously. In parallel, the mask selection mechanism could be improved by incorporating richer contextual cues or advanced similarity metrics to better resolve ambiguities when objects overlap or share similar Intersection over Union (IoU) scores.

Another avenue lies in refining the system's ability to account for environmental effects. Presently, while the primary object is removed, associated artifacts such as shadows or reflections may persist. Future iterations could integrate specialized models capable of identifying and erasing these subtle yet important visual traces (?) to enhance realism. Moreover, expanding the system's capabilities from static images to video sequences presents both a challenge and an opportunity. This would involve integrating object tracking, ensuring temporal consistency, and addressing motion-related artifacts like blur and persistence across frames.

Improvements to the inpainting module itself are also worth exploring. Incorporating diffusion-based approaches or attention-guided mechanisms could significantly boost performance, especially in scenes with complex textures or large missing regions.

Chapter 6

User Guide

This chapter provides detailed steps for setting up and using the Object Removal system on a local machine.

6.1 How to Implement

To set up and run the object removal system locally, follow these steps:

6.1.1 Step 1: Clone the Repository

Clone the GitHub repository using the following commands:

```
git clone https://github.com/treeboor/object-remove.git
cd object-remove
```

6.1.2 Step 2: Install Dependencies

Make sure Python 3 is installed on your system. Then, install the required packages:

```
pip install torch torchvision opencv-python matplotlib numpy
```

6.1.3 Step 3: Download Pre-trained Weights

Download the pre-trained weights for the DeepFillv2 model from the source mentioned in the GitHub repository.

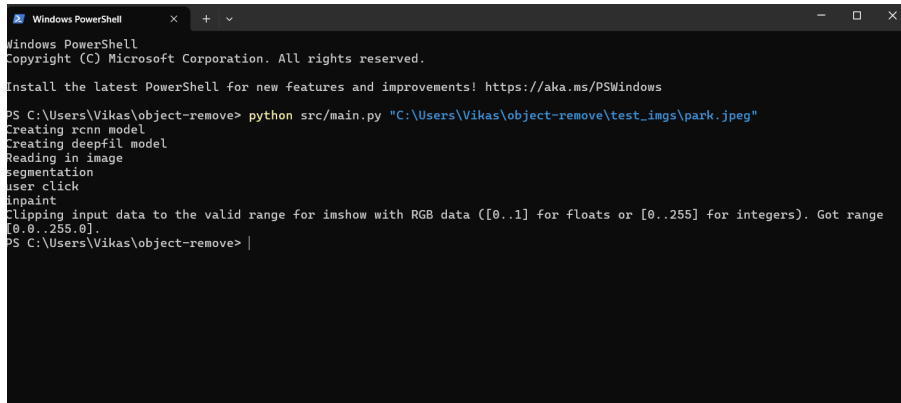
Place the downloaded .pth file into the following directory:

```
src/models/
```

6.1.4 Step 4: Run the Application

Run the application with the path to your target image:

```
python src/main.py path_to_your_image.jpg
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Vikas\object-remove> python src/main.py "C:\Users\Vikas\object-remove\test_imgs\park.jpeg"
Creating rcnn model
Creating deepfill model
Reading in image
Segmentation
User click
Inpaint
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range
[0.0..255.0].
PS C:\Users\Vikas\object-remove> |
```

Figure 6.1: Graphical User Interface for drawing bounding box and removing the object

A new window will open displaying the image you selected.

6.2 User Controls

- **Draw Bounding Box:** Click and drag on the image to draw a bounding box around the object you wish to remove.
- **Reset:** Press `r` to reset the image and clear the bounding box.
- **Continue:** Press `c` to confirm the selected area and proceed with object removal.

6.3 Notes

- **Performance:** Drawing bounding boxes and processing might be slower on systems without a GPU.
- **Model Weights:** Ensure that the pre-trained weights are correctly placed in the `src/models/` directory. If not, the inpainting process will fail.
- **Output:** The system displays three outputs:
 - The original image
 - The masked image (with the object removed)
 - The final inpainted result

References

- [1] Lamport, L. (1994). *LATEX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley.
- [2] Kottwitz, S. (2021). *LaTeX Beginner's Guide: Create Visually Appealing Texts, Articles, and Books for Business and Science Using LaTeX*. Packt Publishing. ISBN: 9781801072588.
- [3] University of Reading. (2023). *Different Styles & Systems of Referencing: Guidance on Citing References for Students at the University of Reading*. Available at: <https://libguides.reading.ac.uk/citing-references/referencingstyles> (accessed June 6, 2023).
- [4] University of Reading. (2023). *Avoiding Unintentional Plagiarism: Guidance on Citing References for Students at the University of Reading*. Available at: <https://libguides.reading.ac.uk/citing-references/avoidingplagiarism> (accessed June 6, 2023).