Object Removal From Image Using Segmentation and Inpainting

Robert Jiang
UCSD

Abstract

Image post processing is a technique used to enhance the original image taken by a camera. One interesting problem that falls under this category is removing unwanted objects found in an image. The first step is to identify the object of interest but manually outlining or highlighting can be a tedious task. As such, we employ deep neural networks to automatically create a mask of the object to identify the exact pixels that need to be removed. The second challenge with this is being able to automatically fill in the missing pixels after removing an object that keep the background intact. The system needs to be able to infer the background that is hidden behind the object from the context that the image provides. This project combines image segmentation and inpainting techniques to allow for users to select objects to remove from an image. Two state of the art models, Mask R-CNN and DeepFillv2, along with an IOU calculation for user object selection are used. We show some examples of our system on images from MS-COCO and images taken on an iPhone and evaluate the effectiveness and limitation of the models.

1 Introduction

Object removal is a difficult but interesting task of removing an object from an image. Oftentimes, when we take pictures, we would like to remove unwanted objects that get in the way of our shot. It is easy to identify the object to remove, but filling in the pixels of the background once the object is removed is much harder. This project tackles this issue by using artificial intelligence, specifically deep neural networks, to remove an object and infer the missing pixels.

One way to detect the object is to have the user manually erase the unwanted object by coloring it in. However, this is inconvenient for the user as objects can take many different shapes and manually erasing them can be tedious. One solution to this is to have the user draw bounding box or circle around the object and remove all pixels inside the boundary. This is much easier for the user, but by removing all pixels inside the bounding box, we might be erasing background pixels that could be useful for inferring missing pixels once the object is removed.

Instead of relying solely on user input to detect the object, we can use deep learning segmentation models to help, specifically Mask R-CNN. These models are able to identify and segment out different objects in an image. By using the mask generated by these models along with user input bounding boxes, we are able to identify the objects more precisely and efficiently. Once the object has been masked out of the image, it can then be passed to a generative inpatining network, Deepfillv2 to get the final output image.

The result is an integrated system when given a bounding box by the user, the system is able to identify and remove 80 different types of objects defined in the MS-COCO dataset.

2 Related Work

One version of object detection is to create a bounding box of the approximate location of the object as done in the YOLO detection algorithm[7]. Then we can use this box and remove all pixels inside the box. However, the bounding box includes pixels that are not actually part of the object of interest. Thus, by removing all pixels in the box, we might be getting rid of information that is helpful for reconstructing the missing parts of the image.

Image inpatining has also recieved lots of attention in recent years. Many of these approaches including using predefined edges as inputs [5] involve an autoencoder method using generative adversarial networks with convolutions to gain a latent representation of the image. Then reconstruct the image with the missing pixels filled in using up-sampling techniques.

One proposed method is patch based inpatining using GANs[1]. This method uses a combination of different generative adversarial networks to generate missing pixels in a given image. However, this approach only focuses on holes in the image that are regular shapes like squares or rectangles.Thus, it is not robust to many different objects. But everyday items come in many different shapes so we need to find a different method that can handle irregularly shaped objects to remove from the image.

3 Method

This project uses a combination of image segmentation and masks selection with image inpainting. Two pretrained models Mask-RCNN and DeepFillv2 are used consecutively. Mask-RCNN model and weights come from Pytorch implementation from TorchVision Models and Deepfillv2 pretrained model comes from [6].

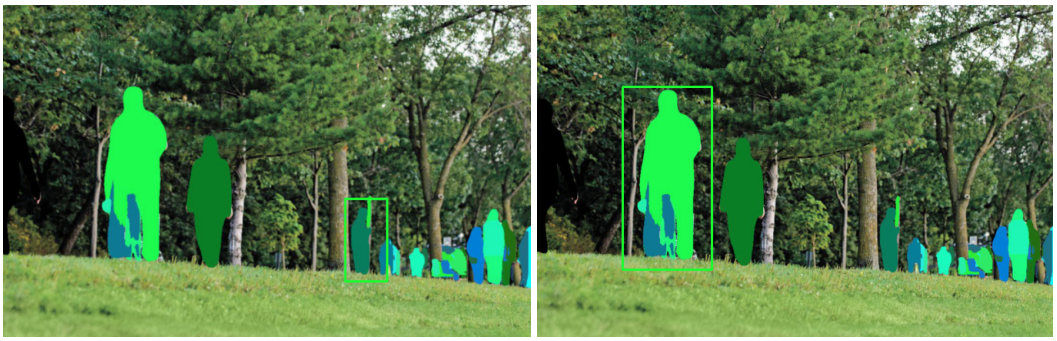
The input image will also be resized such that the longer side of the image will not exceed 680 pixels will preserving aspect ratio to maintain best results using these models. In this section, we will go over the main features of each model along with some different methods of mask selection and reasons for using one versus the others.

3.1 Object Detection and Selection

Object detection is handled using a state of the art segmentation model Mask-RCNN[2].This is a instance segmentation model built on top of Fast-RCNN, a region proposal convolutional network used for object detection and bounding box estimation[8]. In addition to the bounding box and class label branches, Mask RCNN introduces a third branch to estimate object masks. Just like with Faster R-CNN, Mask-RCNN uses a Region Proposal Network to identify regions of interest. Then in parallel bounding box and class predictions, ROI align layers are used to predict the object mask. A pretrainted Mask-RCNN model using a Resnet50 backbone was used as the segmentation model.

Two different methods of selecting the object were considered. Both involve the user drawing a bounding box around the object they would like to have removed.

The first method would be to find a mask produced by Mask-RCNN that resides inside that box. However, this would only work if there is only 1 mask in the bounding box drawn by the user. So a slight modification was made to account for the situation where multiple masks are present in the box shown in figure 1a Rather than taking any mask that has pixels in the box, we take the mask with the highest percentage of pixels inside the box. Proportion is used rather than absolute pixel to make this invariant to object size. The calculation is done by counting the number of pixels inside the box divided by the total number of pixels of the mask. This way, the mask with the highest percentage of pixels in the box is selected.



(a) (b)

Figure 1: Left image shows two masks that have pixels inside bounding box.Right image shows multiple masks entirely inside bounding box.

One limitation to the above method is if there are multiple masks with the same proportion of pixels in the bounding box. A simple example of this case is if there are 2 masks entirely inside the bounding box shown in figure 1b. So method 2 involves looking at the bounding boxes of each object and comparing that to the user selected bounding box. Intersection over union is used to compare the similarity of the user selected bounding box and the object detection bounding box. The object with the highest bounding box IOU score compared to user input bounding box will be chosen as the object.

$$IOU(\text{Box A, Box B}) = \frac{\text{Intersection(Box A, Box B)}}{\text{Union(Box A, Box B)} + \epsilon} \tag{1}$$

This method solves both problems listed above with multiple masks being inside the bounding box. Additionally, the computation is less expensive than counting individual pixels in the masks. Furthermore, we can easily parallelize the computation of all IOU scores by stacking all the bounding boxes together and using Numpy.

3.2 Inpainting

Once we have the object mask, this problem reduces down to filling in the missing pixels of the image known as image inpainting. The model that handles this portion is DeepFillv2, a generative gated convolutional neural network. One major difference between regular computer vision tasks and image inpainting is that in the later, 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.

Gated convolution is how this model deals with invalid masked out pixels throughout the network[9]. Gated convolution is an extension of partial convolutions[4]. Partial convolutions avoids invalid pixels by keeping track of the mask throughout the layers of the network and updating the mask to adapt to the changing sizes of activations.

$$Out\_Partial_{i,j} = \begin{cases} W(X \odot M)\frac{\text{sum(1)}}{\text{sum(M)}}, + b & \text{sum(M)} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

W is the convolutional weights. X is the input region corresponding to the current convolution window and M is the corresponding mask region. The mask update rule is below,

$$m'_{i,j} = \begin{cases} 1, & \text{if sum(M)} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

One problem with partial convolutions is that the mask updates are rigid updates that do not take into account how many valid pixels are in the input. Additionally, as we go deeper in the network the mask values are always either 0 or 1 but each pixel contains information about a larger region of the input that might contain valid and invalid pixels. This is because the receptive field of each pixel location gets larger and larger as we go deeper into the network. Gated convolutions improve on this

by adding learn-able gating weights that are dynamically set during model training. This introduces another set of weights for each convolutional layer that is responsible for handling valid and invalid pixels, giving us the following output,

$$Out\_Gated_{i,j} = \phi(Feature_{i,j}) \odot \sigma(Gating_{i,j}) \tag{4}$$

where $Feature_{i,j}$ is the regular convolutional filter output with activation function $\phi()$ and $\sigma(Gating_{i,j})$ is the gating convolutional filter output with sigmoid activation. Sigmoid activation will constrain the output between 0 and 1, giving the mask soft values.

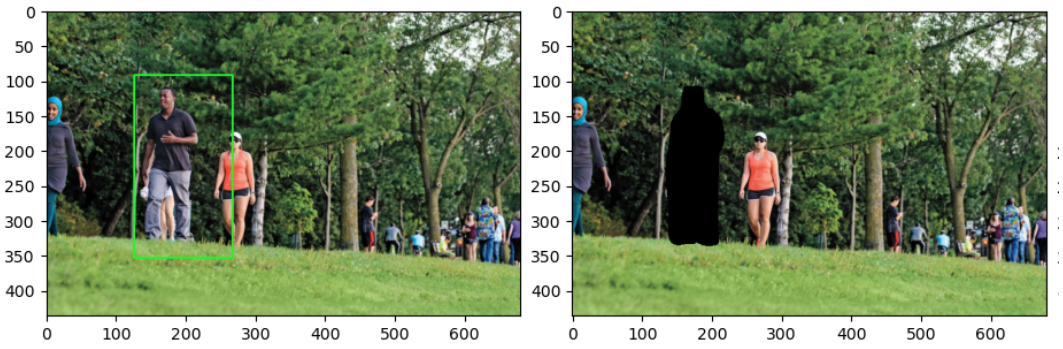
The overall structure of the model is a generative adversarial network(GAN). GANs work by training two models, a generator and a discriminator. The goal is to train a generator model that is able to generate artificial images that fools the discriminator model into predicting the artificial output as a real image. This way we can train a model to generate seemingly real images. This works well with the task of inpatining because we are artificially generating a new image and filling in the missing pixels.

DeepFillv2 uses a two stage generator network, a coarse and refinement network, with contextual attention modeled from [10]. Both of these networks are encoder decoder networks that use gated convolutions in place of all convolution layers. More details on the network can be found here[9]

4 Experiments

Since there is no ground truth labels for our inpainted results, there is no quantitative way to measure the performance of our methods. Instead we will take a look at the following example test images from MS-COCO dataset [3] and images taken using an iPhone. MS-COCO dataset is a collection of over 300,000 everyday images with labelled objects. Mask-RCNN was trained on this dataset to be able to identify 80 different classes of objects. Thus, our system is able to identify and remove those 80 different objects. In the following sections, we will go over each example, highlighting aspects of each example and limitations of the system. Then we will explore possible future improvements to the system.

4.1 Example Results

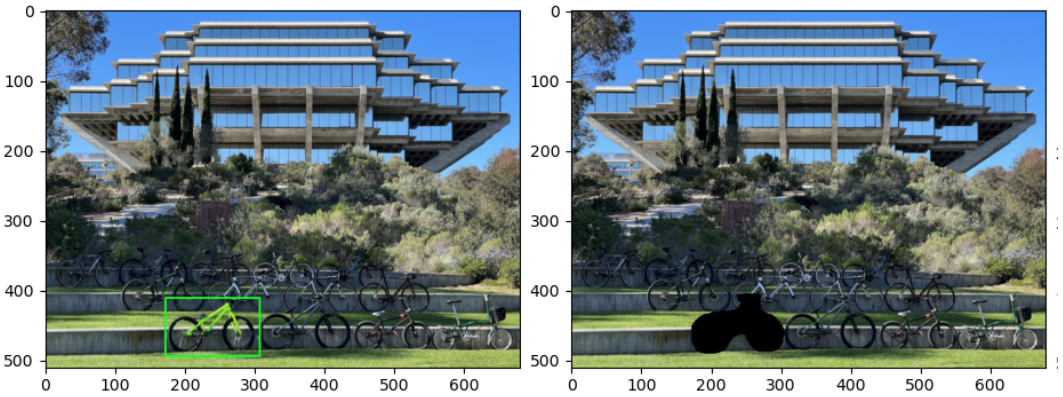


(a) User Selection Box and Masked Image

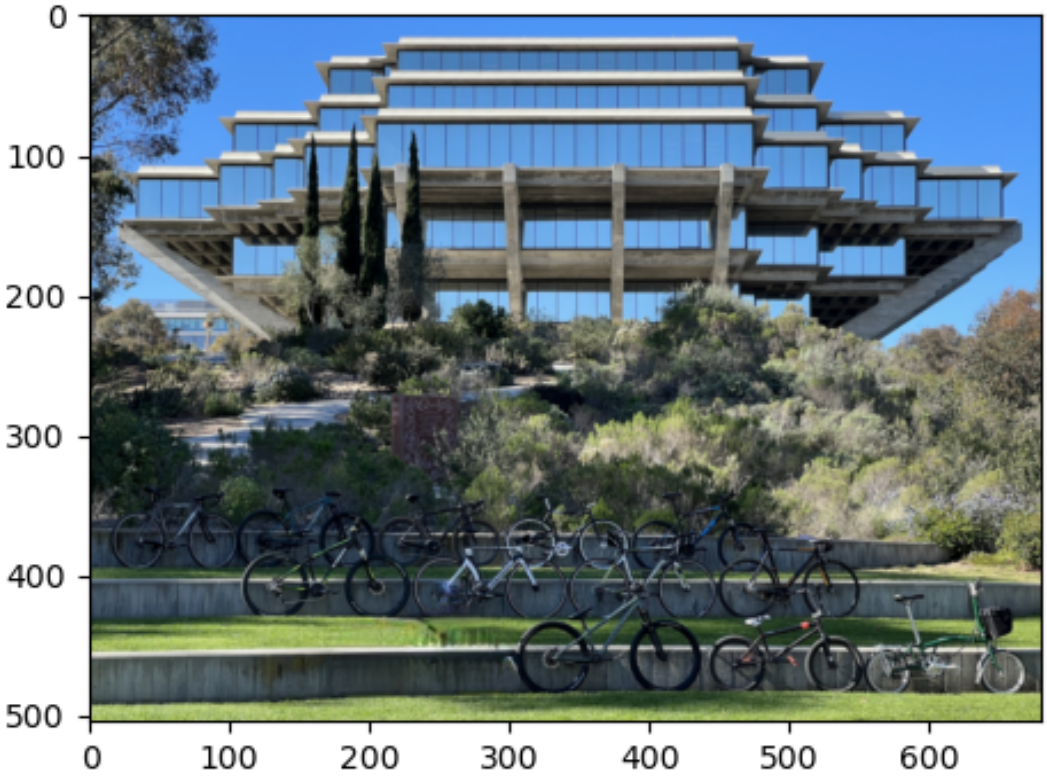
(b) Result

Figure 2: Image from MS-COCO

Looking at 2, we see the user has selected the man wearing a black shirt from the bounding box. We see that the bounding box also covers part of the woman wearing the orange shirt. Even though this is the case, the system chooses the mask of the man because there is a higher change that the user intended to select the man based on the bounding box based on the calculation of the IOUs. And the man is successfully removed from the image.



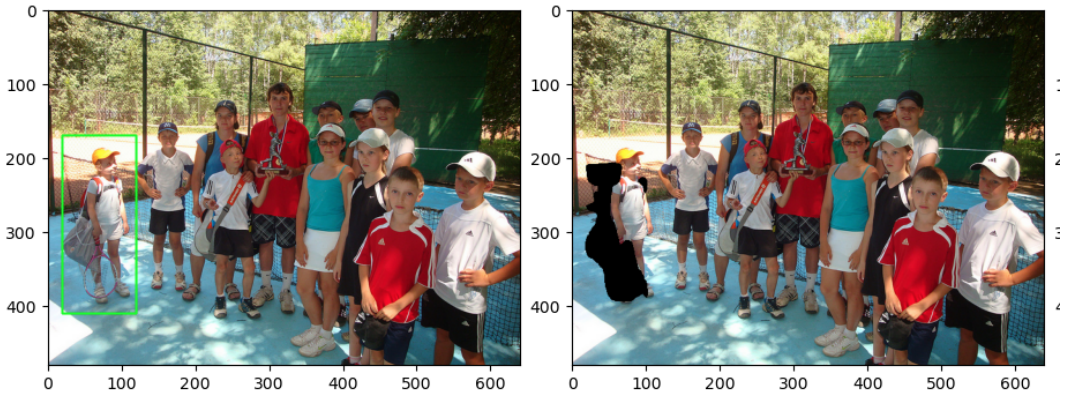
(a) User Selection Box and Masked Image



(b) Result

Figure 3: Image taken with iPhone.

The example in 3 was taken with an iPhone camera. The original resolution was 4032x3024 and has been resized down according to the resize method mentioned above to maintain best performance. The objects are not limited to people as bikes are another class that Mask-RCNN has been trained to identify. Again the box includes pixels of other bikes in the background but the masked object is only the main neon green bike that has been selected to be removed from the image.



(a) User Selection Box and Masked Image

(b) Result

Figure 4: Image from MS-COCO

4 shows an example of a limitation of the current system. The user would like to remove the child along with the bag and tennis racket that he is carrying as indicated by the bounding box. In this case, there are multiple objects within the bounding box and the system only gives us a mask of the object that is most similar to the bounding box based on the IOU calculation. This is not ideal as we would like to remove the entirety of the child along with the things he is carrying.

4.1.1 Future Directions

One improvement that could be made is to allow for multiple masks to be combined to remove multiple objects that are within the bounding box. This would solve the problem we see in 4.

Another area of improvement is to modify the IOU calculation in object selection. If we had two boxes of the same size inside the bounding box provided by the user, then there would be a conflict of which box to choose. Similarly, if there are no boxes inside the user selected bounding box, then all objects would have the same IOU score of 0. We can improve upon this by using different similarity scores that take these cases into consideration.

A further area to look into is interaction with the environment. Objects cast shadows when there is light. Objects can also displace other objects in the case that someone is sitting on the couch. So when we remove an object, we also need to remove all of the interactions and effects that the object has on its environment in order to create a convincing image.

Since this system works for static images, we also could extend the functionality to videos because videos are just long sequences of static images. However the main difference between static images and videos is motion. Thus, one thing we would need to implement is tracking of the same object throughout the entire video and dealing with movement that can cause blurry pictures.

References

[1] Ugur Demir and Gözde B. Ünal. Patch-based image inpainting with generative adversarial networks. *CoRR*, abs/1803.07422, 2018.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[4] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. *CoRR*, abs/1804.07723, 2018.

[5] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z. Qureshi, and Mehran Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. *CoRR*, abs/1901.00212, 2019.

[6] nipponjo. deepfillv20pytorch. `https://github.com/nipponjo/deepfillv2-pytorch`, 2022.

[7] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[8] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[9] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas Huang. Free-form image inpainting with gated convolution, 2018.

[10] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative image inpainting with contextual attention. *CoRR*, abs/1801.07892, 2018.