WiDS-2023: Camouflage Object Masking mentored by Gaurav Misra

Ananya Chinmaya, Roll No.210070008

January 29, 2023

Contents

1 Introduction

Camouflage Object Masking deals with the task of identifying and segmenting out objects that are not well-distinguished from their backgrounds. This is a specialized problem within the domain of Computer Vision, which has been seeing and continues to see rapid advances and novel techniques. Camouflage Object Masking has several applications across varied fields including medical image segementation, rare species discovery, the miliary, and so on. Recent work also suggests that research in AI-based segmentation of camouflaged objects could hold key insights into how the hierarchial levels of human visual perception are structured.
The implementation was done using Deep Learning and Convolutional Neural Networks(CNNs). For the code, Python libraries such as PyTorch and segmentations-models-pytorch were used to implement the U-Net architecture to use the given training dataset and use the same for segmentation of images of camouflaged objects.

2 Weekly Updates

Week 1: Introduction to Machine Learning, Deep Learning and Convolutional Neural Networks. The resource provided for this week was the DeepLearning.AI Deep Learning course taken by Andrew Ng on Coursera.

Week 2: Diving Deeper into Convolutional Neural Networks(CNNs) and Introduction to PyTorch. The resources for this week were an MIT-OCW lecture on CNNs and a YouTube playlist to get familiarized with PyTorch and learning how to create a CNN architecture on Python

Week 3: Understanding the Problem Statement of Camouflage Object Masking. The resources provided included papers to be read on U-Net architecture, image segmentation and Camouflage Object Masking. This was to learn the theory behind image segmentation.

Week 4 and 5: These weeks were all about implementing and fine-tuning the architectures studied. There were no resources for this week except for the given training dataset. A working model of the code was presented at the end, along with the results obtained.

3 Theory Learnt

3.1 Basics of Deep Learning

The learning from here has been uploaded on my Github repository for this project in the form of wids-notes.pdf.
But to summarize, we start with Binary classification which is just a method to see whether an object is present or not in a particular image. We use Logistic Regression to determine the "probability" of y(which represents the image) being 1(indicating that the object is present) and 0(indicating that the object is not present) for the image. Our goal is to set the other parameters such that $\hat{y}$ is closest to y.
The Loss or Error function needs to minimized. A lower loss function means a better model. $J(w, b) = \Sigma_{i=1}^{m} L(\hat{y}_i, y)$ is the cost function which is a summation of the losses. The goal is to minimize the cost function which is done by Gradient Descent. We iterate repeatedly till the function has a slope of zero which indicates a minima.

3.2 Convolutional Neural Networks

Neural Networks consist of an input layer and an output layer and hidden layers in between. The hidden layers are a part of the architecture and are not "visible". We decide and fine tune the parameters of all the hidden layers to best suit our architecture.
Forward Propagation is basically using the current parameters to recieve an output, compare the output with the ideal output and record the losses which are **Back propagated**. This serves as a method to fine-tune the results based on a feedback mechanism. We can decide the number examples in the training dataset and the number of times we iterate through the dataset. Each iteration is called an EPOCH.
Convolutional Neural Networks (CNNs) are Neural Networks that have multiple hidden layers. Each layer takes in input from the previous layers(Forward Propagation) and provides feedback during the Backward propagation And the multiple iterations going through the training dataset is how the parameters are fine tuned by setting a **Learning Rate** to increase the accuracy of the Convolutional Neural Network.

Other **Hyper-Parameters** also need to be set such as:

- number of iterations
- choice of activation function
- number of hidden layers
- number of hidden units
- Momentum
- Learning Rate, etc.

This is the summarization of the structure of CNNs

3.3 U-Net Architecture and Image Segmentation

The U-Net architecture consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. The U-Shape comes from the encoding and decoding blocks.
It is a very common architecture for Image segmentation.

4 Code Implementation

Uploading the Dataset: The dataset is loaded onto the computer. There is a Test dataset, a validation dataset and the Train dataset.
The U-Net architecture:We convert images into vectors. We have functions to create an encoder block, a convolutional block, and decoder block.

These 3 blocks are what formed the basis of the U-Net. The U-Net in the code consists of 3 encoder blocks, a convolutional block called the "bottleneck" and 3 decoder blocks. We set the number of epochs and run through the training set.
Displaying the Outputs: The outputs on the test dataset were displayed in a pre defined location on the computer.

5 Results Obtained

The Average Dice Score was around 0.47 These are some of the images passed and the predicted masks.

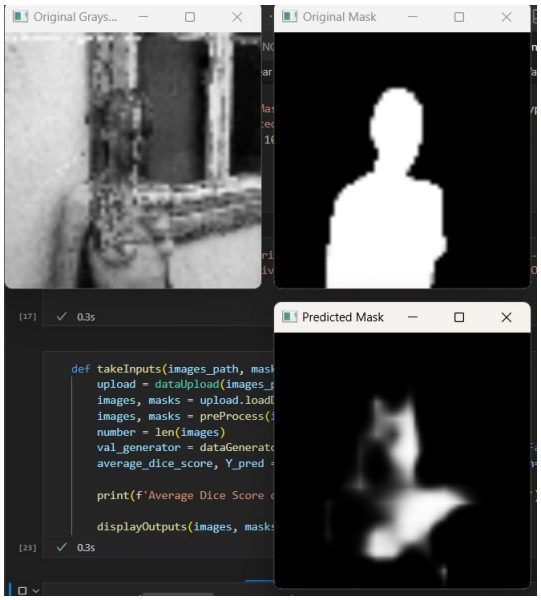

Figure 1: Soldier wearing a Camo Outfit

The above images are from the test dataset.These are the predicted masks(that the code generates) as a comparison to the original masks (the ideal case)

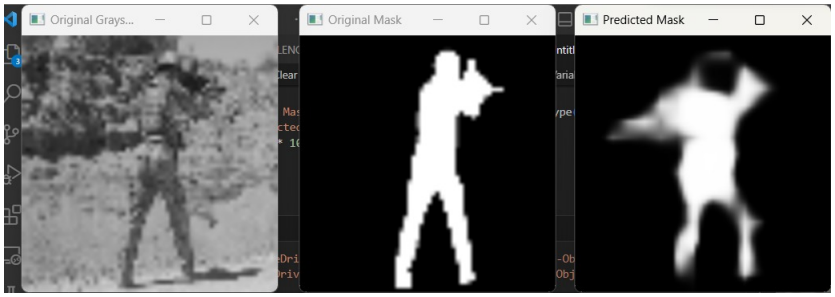

Figure 2: Painted Human Being

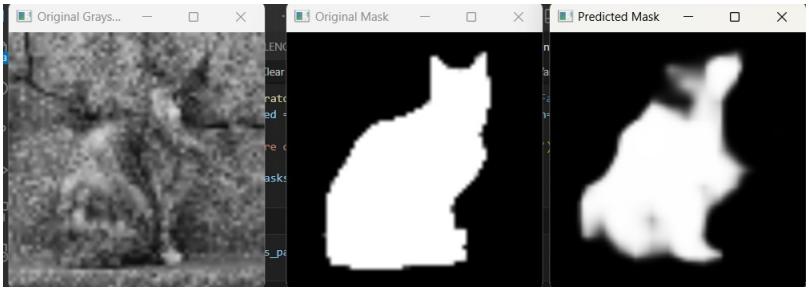

Figure 3: Camouflaged Cat

6 Conclusion

For the above images:
For the soldier wearing the camouflage suit, the CNN is able to broadly make out the shape of the gun, the legs, the body and the head. While obviously not perfect, it gives out the idea of there being a man camouflaged in the picture.
For the painted woman, the performance is not as good as the woman is painted in colours almost exactly as the surroundings. There still seems to be a little detection as the outline of the face.
For the camouflaged cat, the performance of the CNN is pretty good. The ears, the legs and the body are well detected by the CNN built by the code.
Thus, we can conclude that a relatively simple implementation of the U-Net architecture gives recognizably segmented images. More advanced applications would yield much better results.
Link to GitHub Repository: https://github.com/ananyachinmaya/WiDS-2022-Camouflage-Object-Masking/tree/main/Learning