Segmenting camouflaged objects in the wild
IE643 Challenge Assignment - 2022

Jimut Bahan Pal (22D1594)

October 30, 2022

1 Notes for evaluating the unseen test datasets

Please open the file named **IE643_22D1594_CHALLENGE_TESTING.ipynb**. There, instructions have been provided to make folders for testing purpose. Please make sure that the folders follow this structure during the evaluation.

```
Challenge_Dataset
 | Images
 |    |- Test
 |    |- Train
 |    |- Validation
 |- Masks
 |    |- Test
 |    |- Train
 |    |- Validation
```

The model is automatically downloaded and it is your duty to make the test files, i.e., folders containing images and mask in a structure for proper evaluation. The pipeline also allows us to see the evaluation scores obtained on the final test dataset in red color. That is, the mean Loss, mean Accuracy, mean Dice and mean Jaccard across the full test samples is calculated.

The folder **IE643_22D1594_CHALLENGE** also comes with the report present as **IE643_22D1594_CHALLENGE_REPORT.pdf**, model saved in .pt format and named as **IE643_22D1594_CHALLENGE_MODEL.pt**, and the whole source code with other variants of U-Net models, such as vanilla U-Net, Attention U-Net, Residual UNet++, Attention R2-UNet, R2-UNet etc.,inside the file **IE643_22D1594_CHALLENGE_CODE.ipynb**.

2 ResU-Net architecture

ResU-Net[1] architecture was originally used for segmenting roads from satellite imagery. Later it was used for several different purposes. We have considered this architecture for our purpose since this will be able to detect camoflauged objects which cannot be easily detected by other U-Net [2] variants which are not pre-trained. The architecture uses Resnet-18 pre-trained backbone and we have used sigmoid as activation function in the last layer of the network, since the pixel values should be predicted between 0 and 1. The architecture is shown in Figure 1.

We have used the Pytorch implementation of ResU-Net found in this link https://github.com/monini13/NucleiSegmentationAI/blob/master/model.py. The pipeline is created from scratch for the segmentation purpose which is shared using a Notebook.

3 Data preprocessing

From the samples of the dataset collected as shown in Figure 3, we can see that the dataset can be quite hard even for humans to distinguish between some of the region of interests and unnecessary clutter. The objects are well hidden by giving the same textures as the background of the images. This motivated us to use architecture which supports pre-trained ImageNet weights, since learning can be easy and will be superior to those models which are trained from scratch.

We have used a standard pre-processing technique, i.e., we have resized the images to 256x256 and kept the 3 channels intact, we have also converted the single channel mask to the same dimension. For the images and masks, we have divided the intensity values with 255.0 so that the intensities are between 0 and 1. For the purpose of image augmentation, we have used random vertical and horizontal flips with certain probability (0.5 here), we have also used affine transformations on both the image and masks pairs. For the training robustness we have considered 15 random boxes of size 32x32 pixels placed at any location of the image and mask pairs, which will help increase the robustness of the training procedure. All the data augmentations considered for the training is shown in Figure 2.

4 Training procedure

The code is done in Python3 using the Pytorch framework. The ResUNet model is trained for 500 epochs using Adam optimizer with a learning rate of 1e-04 which uses a weight decay of 5e-04. Training was done using the 1000 images and validated using the 200 images. We have used a batch size of 64 during training. This is possible due to the high compute resources provided, and the results might not be reproducible in google colab. We have used binary cross-entropy as the loss function which can be written as:

$$L_{y'}(y) := -\frac{1}{N}\sum_{i=1}^{N}(y'_i \log(y_i) + (1 - y'_i)\log(1 - y_i)) \tag{1}$$

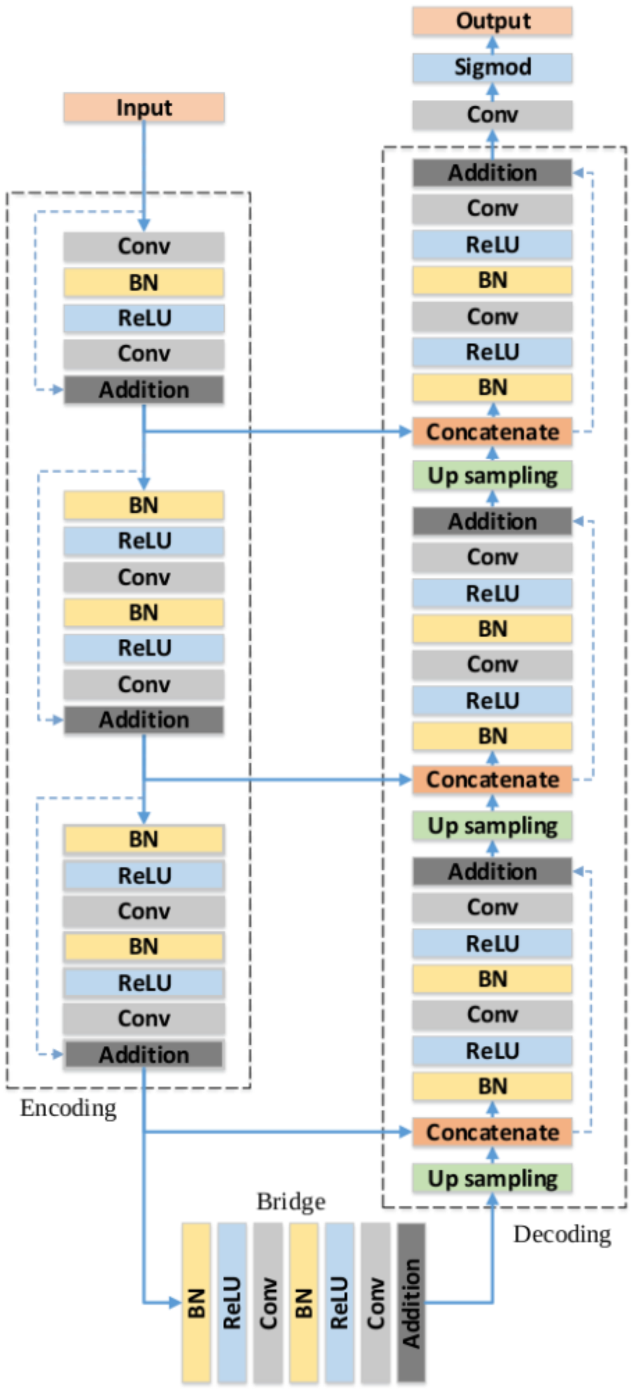
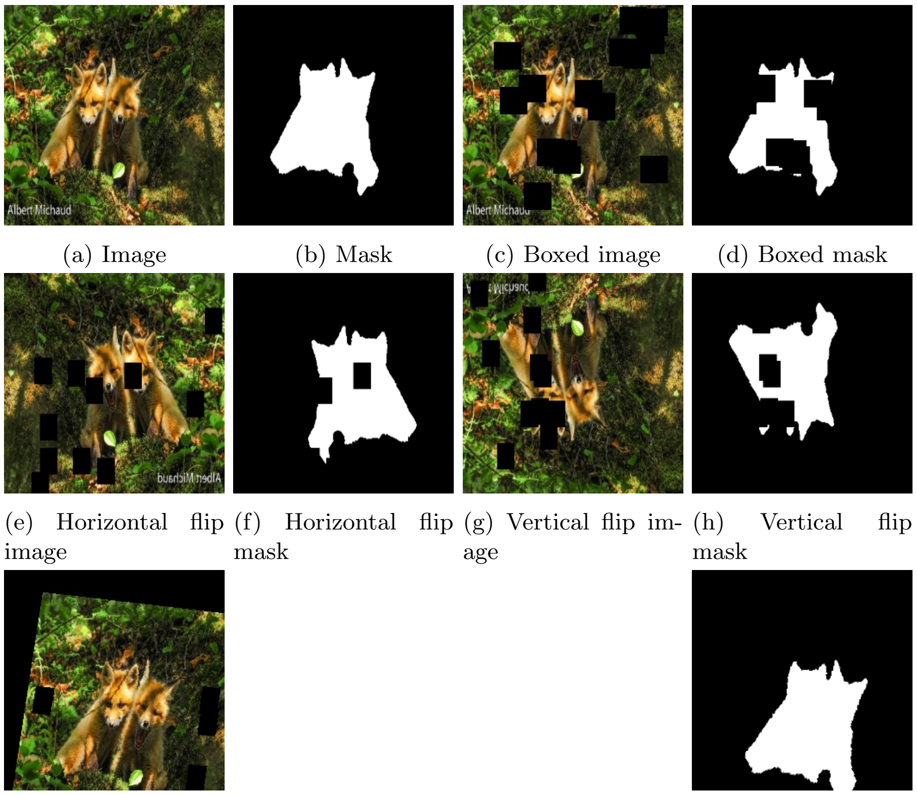


Figure 1: ResU-Net architecture. The image is retrieved from https://idiotdeveloper.com/what-is-resunet/



(a) Image (b) Mask (c) Boxed image (d) Boxed mask

(e) Horizontal flip image (f) Horizontal flip mask (g) Vertical flip image (h) Vertical flip mask

(i) Random affine image (j) Random affine mask

Figure 2: Simple data augmentations were considered for the training of the deep neural network architecture. (a) and (b) shows the original image and mask pairs. Same augmentations were applied to the image and mask pair with boxed mask where we cropped a box out of the image and mask pairs as shown in (c) and (d). Horizontal flip of the images and boxed mask is shown in (e) and (f). Vertical flip of image and mask shown in (g) and (h). Affine transformation of image and mask is shown in (i) and (j).

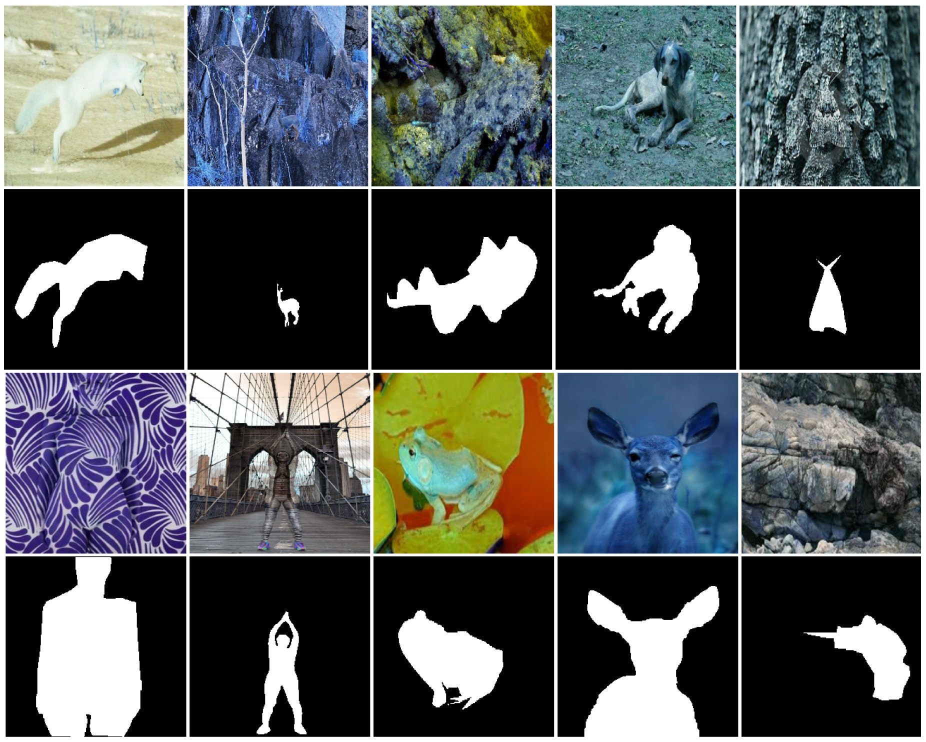


Figure 3: Samples of validation dataset showing that it is even difficult for humans to distinguish and segment objects carefully. There are a lot of natural texture information present in the dataset and using a pre-trained model trained on ImageNet data will certainly do good.

Where, y_i is the predicted class per pixel, y'_i is the original pixel value of segmentation mask. All the pixels in the segmentation mask are averaged (over N) to get the overall loss.

5 Performance metrics

For quantitative analysis the following performance metrics were used, including Dice Coefficient (**DC**), Precision (**PC**) and Jaccard Similarity (**JS**) or Intersection over Union (**IoU**). For calculating these we have to use the following variables, True Positive (**TP**), True Negative (**TN**), False Positive (**FP**), False Negative (**FN**), Ground truth (**GT**), and segmented result (**SR**). Dice Coefficient, Jaccard Index (IoU) and Precision are calculated using the following equations,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$DC = 2 \times \frac{GT \cap SR}{GT + SR} \tag{3}$$

$$IoU = \frac{GT \cap SR}{GT \cup SR} \tag{4}$$

$$PC = \frac{TP}{TP + FP} \tag{5}$$

Generally, accuracy does not work for imbalanced data, hence we use two special metrics which are dice coefficient and jaccard index. These two metrics shows the relative performances of models using the predicted segmentation masks and the original ground truth of the images. The values lie between 0 and 1. The values close to 0 gives bad performance and the values close to 1 gives good performance.

6 Plots and Results

The variation of mean Accuracy, mean Loss, mean Dice Coefficient and mean Jaccard is shown for 500 epochs in Figure 5. Table 1 shows the metrics obtained for the unseen validation dataset after training on the training images and masks pairs for different number of epochs. We have compared a simple baseline of U-Net which gives relatively poor performance. We see that a pre-trained ResU-Net model on ImageNet dataset gives relatively good performance when compared to the vanilla U-Net model. The results got from the ResU-Net model when trained for 500 epochs on the test set is shown in Figure 4.

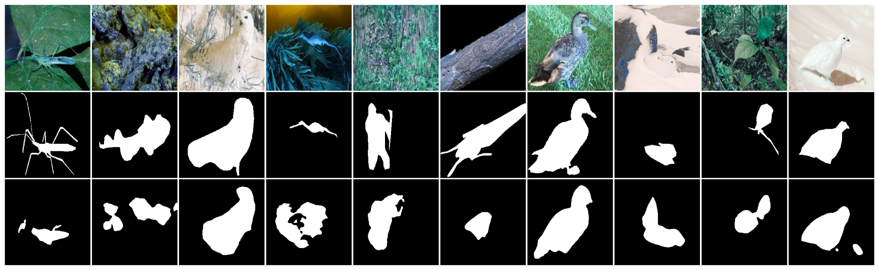
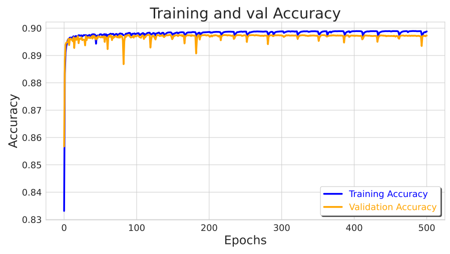


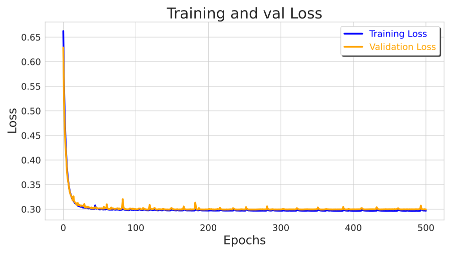
Figure 4: The results of the ResU-Net architecture on test images.

Model	Epochs	Loss	Dice	Jaccard	Accuracy
U-Net [2]	200	0.8595	0.3685	0.2486	0.8214
ResU-Net [1]	500	**0.2898**	**0.6595**	**0.5286**	**0.8952**

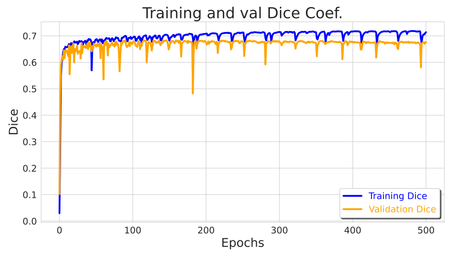
Table 1: Different evaluation metrics for the validation dataset got after training different models.



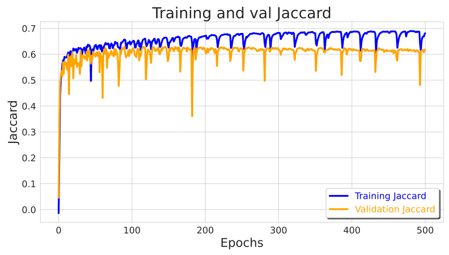
(a) Accuracy.



(b) Loss



(c) Dice Coef.



(d) Jaccard

Figure 5: Training and validation metrics recorded for the ResU-Net model during 500 epochs. We can see minor fluctuations during the training, but that is due to the difficulty in segmenting camouflaged objects.

References

[1] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geosci. Remote. Sens. Lett.*, 15(5):749–753, 2018.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.