Instance Segmentation of Histology Images using Deep Learning

Rohit Anilkumar
(Data Scientist, Tata Elxsi)

Objective

Develop a deep learning model for identifying cell nuclei from histology images. The model should have the ability to generalize across a variety of lighting conditions,cell types, magnifications etc. The generated mask should have the same size as that of the corresponding raw image.

System Requirements

16 GB RAM, Intel i5 7400 Processor, 1TB Hard Disk, NVIDIA GTX 1050 GPU, Ubuntu OS.

Image Processing and Data Augmentation

The dataset shared has 590 images. Since the dataset is small , the premise was suitable for data augmentation. ImageGenerator function from keras was used for the purpose. Basic transformations like shifting width and height, rotation and zooming were used. Since this was an image segmentation task, the same transformations should be applied to the raw image and its mask. The seed parameter should be same for both the raw image and mask image ImageGenerators so that same transformations can be applied to the raw image and its corresponding mask.

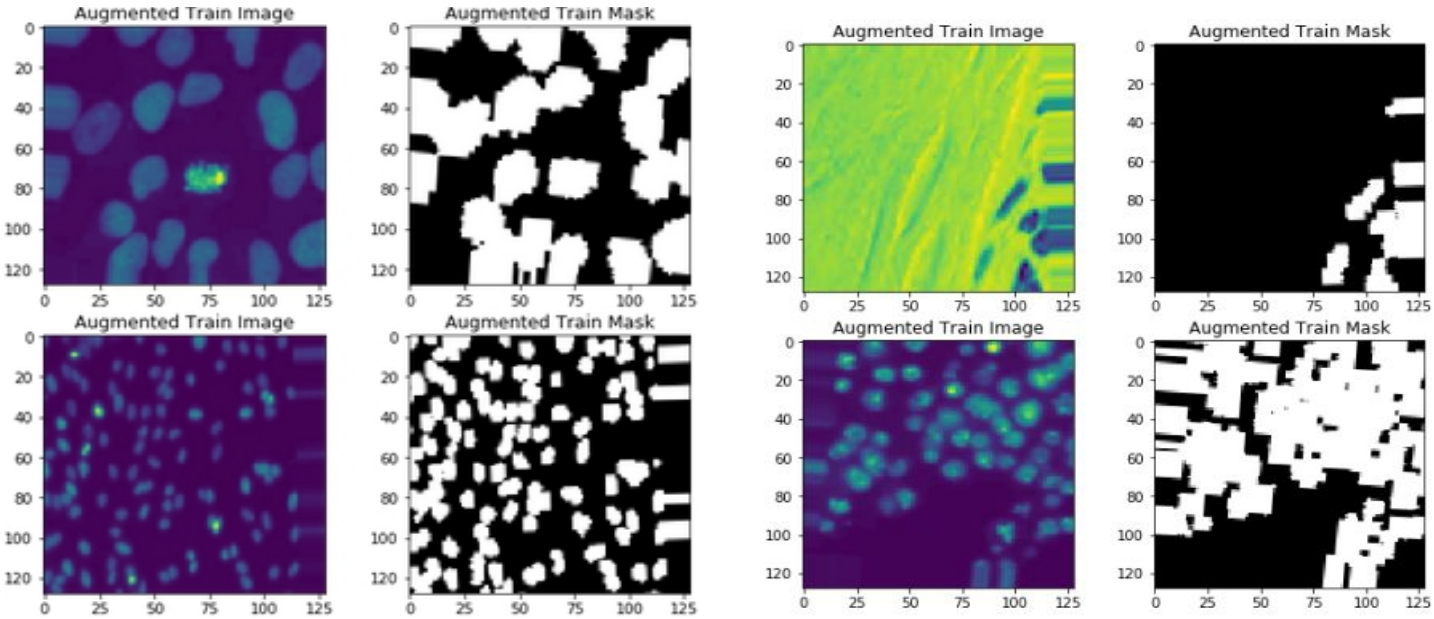


Fig 1: Augmented training image and its corresponding masks. The augmentedImageChecker() method in the ipynb notebook was used for generating the above representations.

The training images are having various resolution so all the images are resized to a uniform size. The model was trained on 256x256, 128x128 pixels images. Best results were obtained for 128x128 image size.

Neural Network

For the purpose of instance segmentation, a UNet architecture is used. The architecture consists of a contracting path and a symmetric expanding path. The former helps to capture context and the latter helps in localization. Along with accuracy, a user defined function which calculates the dice similarity coefficient is also used as the evaluation metric for the neural network.

The dice similarity coefficient (DSC) was used as a statistical validation metric to evaluate the spatial overlap accuracy of automated probabilistic fractional segmentation of images with the ground truth images[1]

The model was trained using the augmented data for 50 epochs with steps per epoch set at 200 and Adam as optimizer.

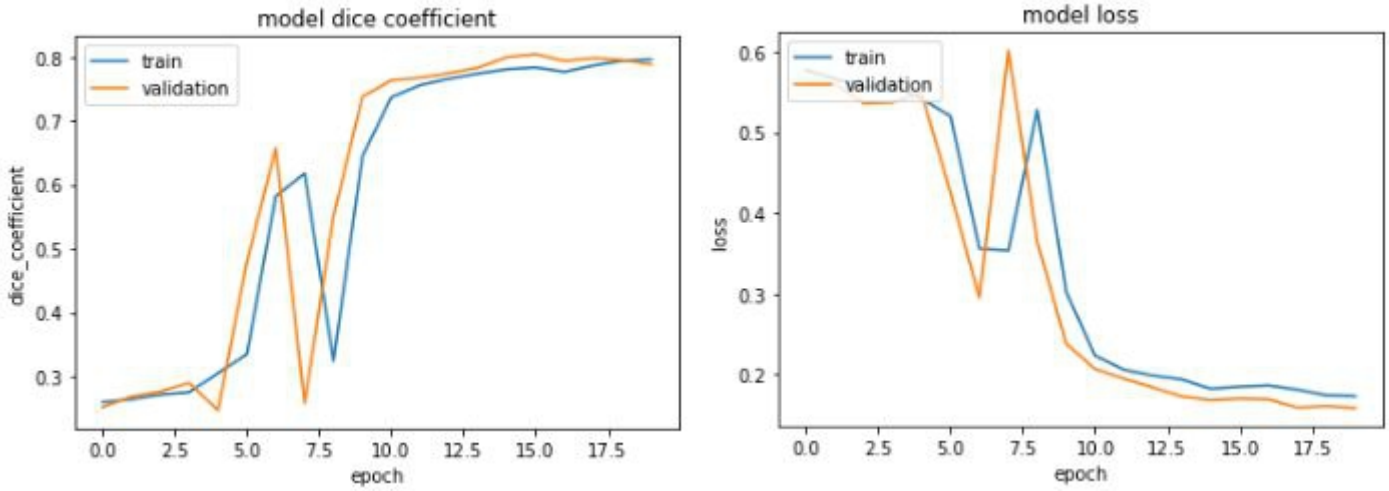


Fig 2: The dice coefficient and loss of the model during training and validation across various epochs

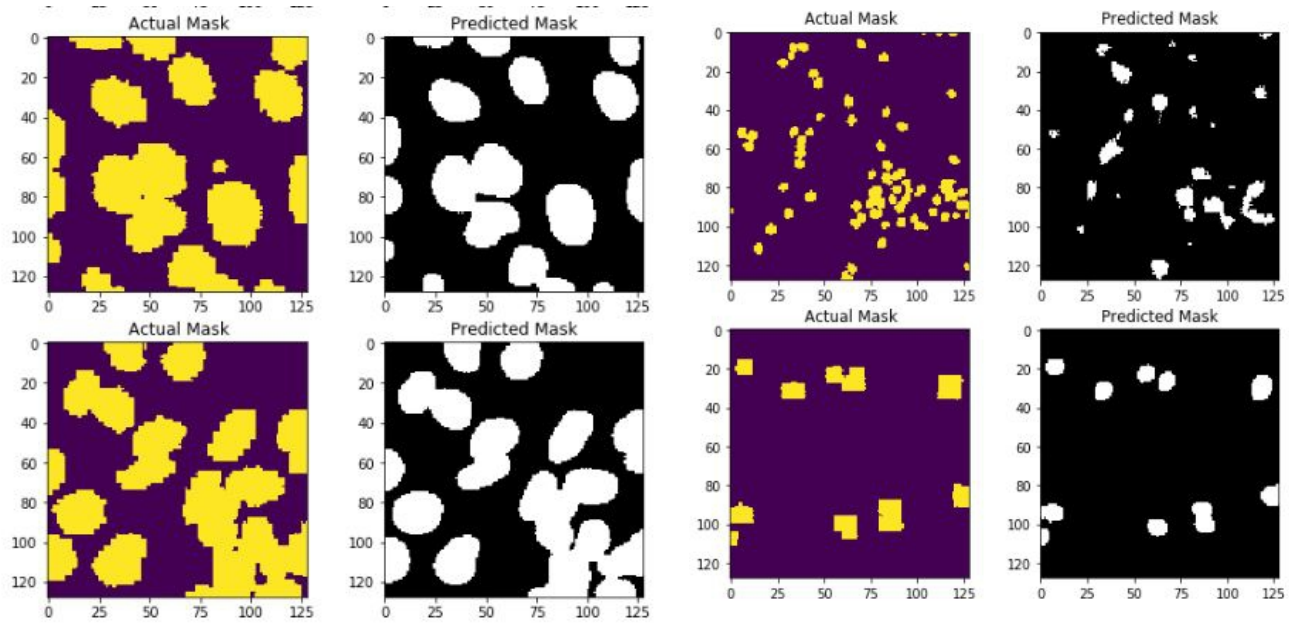


Fig 3: Comparison of mask predicted by the model to the actual mask. The validation datset was used for this.The validationImageChecker() method in the ipynb notebook was used for generating the above representations.

Resizing Generated Masks

One of the expectations was to generate the mask as the same size of the test image. Since the model was trained on a reduced image size, it was necessary to develop an alternate mechanism to resize the generated mask to the size of its corresponding test image. Further analysis revealed that all the images are having variable sizes. Hence a key-value pair mechanism was adopted to store the size of the test images. The filename was set as the key and height and width were used as the values.

```
output_msk_dim.setdefault(file,[])
output_msk_dim[file].append(org_shpe_array[0])
output_msk_dim[file].append(org_shpe_array[1])
```

Fig 4: output_msk_dim dictionary was used to save the dimensions of each image. The image filename was set as key. The above code snippet implements a single key - multi value pair.

Once the masks are generated for all the test images and the dictionary is created, the postProcessing() method will resize the image to its original dimensions by reading the original height and width from the dictionary and resizing the predicted masks. The resized masks are then saved to the destination as grayscale images by setting cmap parameter as gray in the imsave() method.

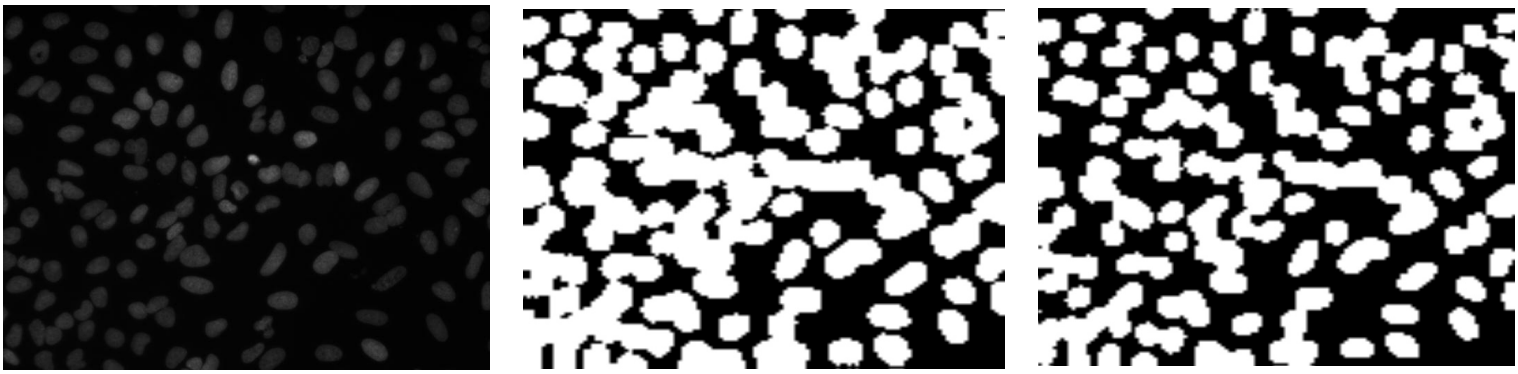
Analysis Of Various Implemented Models

	Model	Model Metrics	Remarks
1	Baseline-256	loss: 0.4187 acc: 0.8305 val_loss: 0.4349 val_acc: 0.8105	Resized images to 256 x 256. Greater misclassification on test images. Long training time
2	Baseline-128	loss: 0.5578 acc: 0.7423 val_loss: 0.5529 val_acc: 0.7481	Resized images to 128 x 128 Misclassification on test data reduced. Faster training timedue to reduced features.
3	Baseline-128 + Data Augmentation + Dice	loss: 0.1736 dice_coef: 0.7958 acc: 0.8856 val_loss: 0.1587 val_dice_coef: 0.7884 val_acc: 0.9362	Employed data augmentation and set Dice Coefficient as evaluation metric.

Table 1

Threshold Analysis on 3(in green)

A threshold value has been set arbitrarily in the testing () method to convert the predicted pixel values to an image. Various values like 0.55, 0.75 etc have been used.



Actual Test Image | Predicted Mask at threshold =0.55 | Predicted Mask at threshold = 0.75

Fig 5: Images at various threshold 0.55, 0.75.

From the above images, its clear that at threshold set at 0.75, the output mask has the ability to distinguish between closeby nuclei.

Actual Test Image | Predicted Mask at threshold =0.55 | Predicted Mask at threshold = 0.75

Fig 6: Images at various threshold 0.55, 0.75.

But at the same time, it fails to identify certain nuclei which is rightly identified when threshold is set at 0.55.

Further analysis helped to arrive at a conclusion that finding an optimal threshold value is paramount in generating a proper mask.

Scope for Improvement

The above results were based on small improvements done on the baseline UNet model and data augmentation. Better results can be obtained by

- Using state of the art neural architectures such as Mask RCNN.
- Better image processing methods can be employed to improve the performance of the model. Based on the analysis, it was found that certain images are having variable lighting which makes nuclei boundary detection tedious. Image processing methods can be used to make the data cleaner.
- Use elastic deformations for data augmentation.

- The analysis and observations were done on RGB images. Other color spaces like LAB can be used. The Lightness parameter in LAB may be used to handle the variable brightness in the images.
- An arbitrary threshold has been set to convert predicted array to the corresponding mask in the testing() method. More analysis is required to estimate the optimal threshold value.
- Test Time Augmentation can be implemented[2].