Cell Segmentation using U-Net

Meet Kathiriya - 160050001
Shreyas Pimpalgaonkar - 160050024

Introduction

In this project we implemented the U-Net architecture, a Fully Convolutional Neural Network using Keras library in python. We trained the model to segment the cells and boundaries in microscopic images of biological cells.

Network Architecture

1. Double convolution Layers with increasing filter count accompanied by max pooling layers in the beginning.
2. Upsampling layers ahead with decreasing filter counts to increase the resolution of the images.
3. In order to localize, high resolution features from the contracting path are combined with the upsampled output.

Data Preprocessing and Augmentation

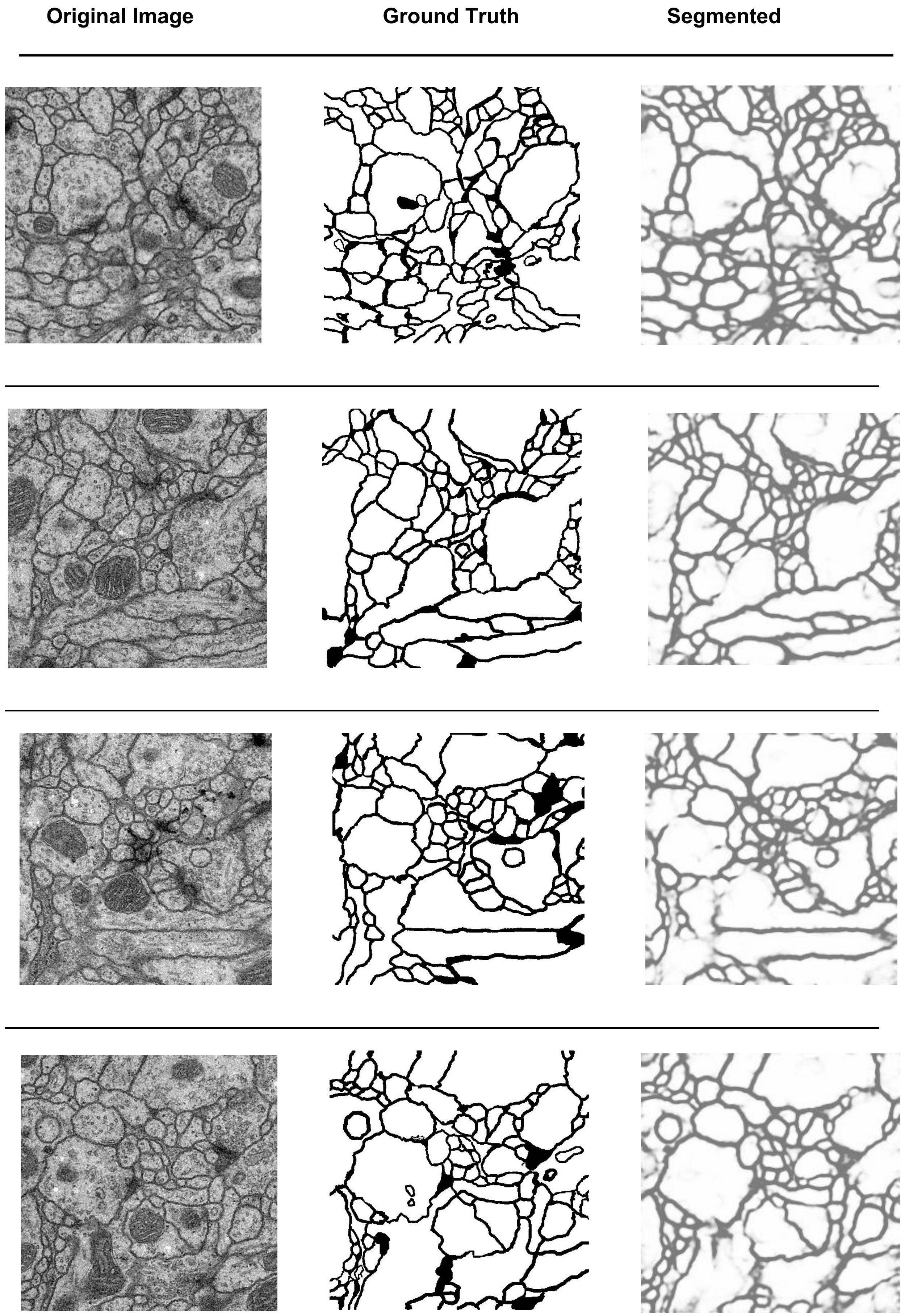
We used the Augmentor library of python to generate 3000 images and labels of size 256x256 from a dataset of 30 images. The following operations were performed in sequential order with a certain probability for each operation.

1. 90/180/270 degree rotation
2. rotation upto 15 degrees
3. random flipping
4. gaussian distortion with different kernel sizes
5. random cropping

Training and testing

We trained the model for 5 hours on a MacBook Pro with a 2.7 GHz CPU for 4 epochs on 3000 images upto an training accuracy of around 80%. Cross entropy loss function with Adam optimiser at a constant learning rate was used. Output thresholding done to get binary segmented image.

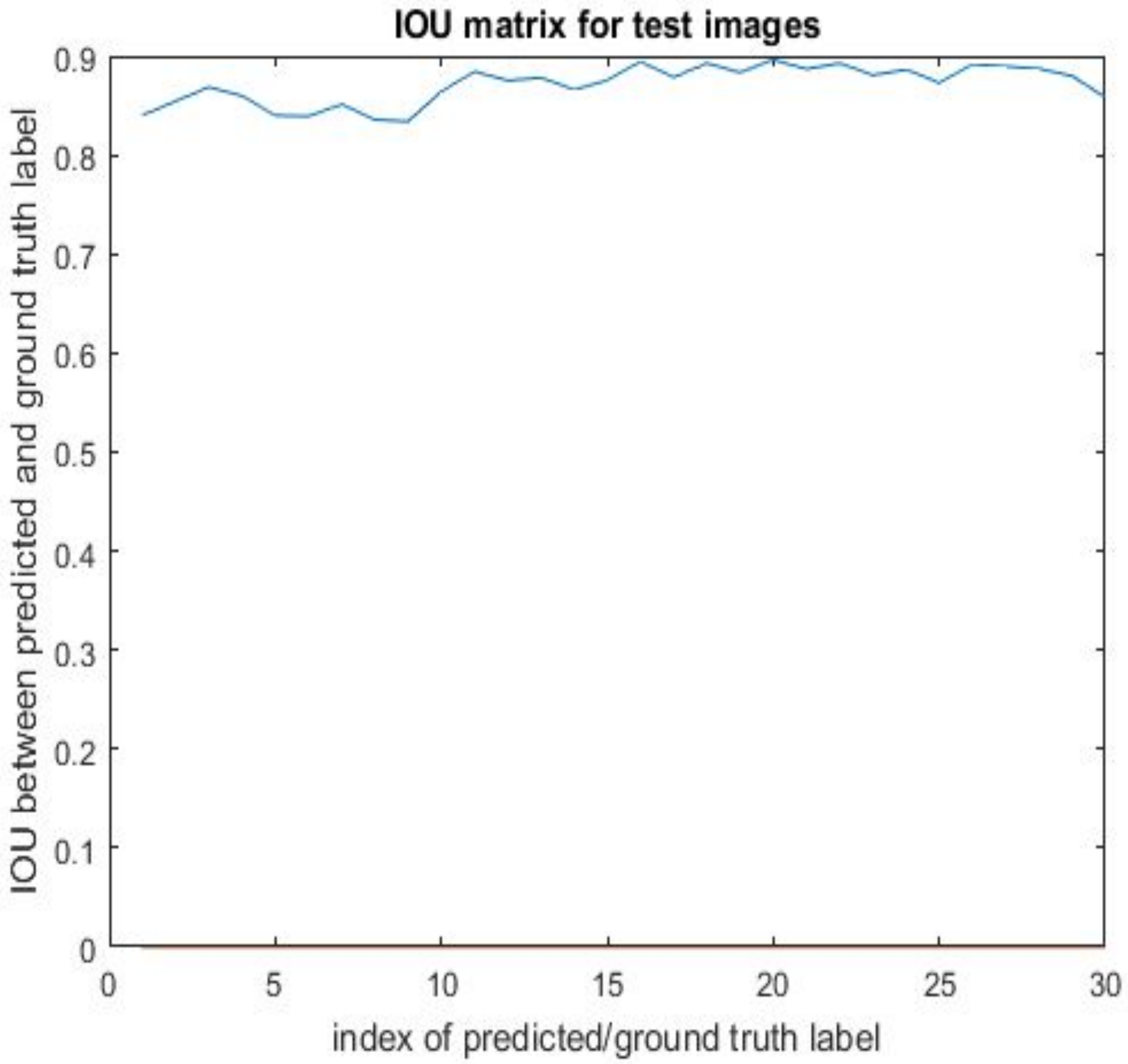
Results



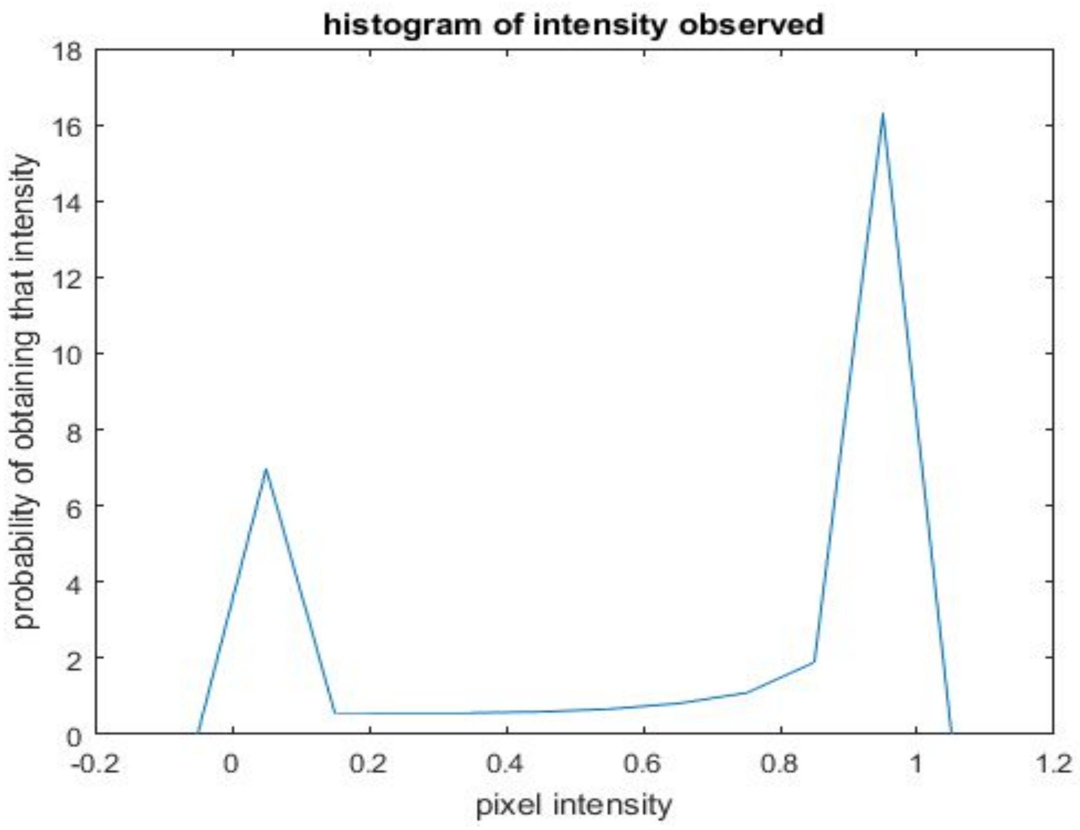
Analysis Results on Predicted segmentation images :

We tested our trained U-Net on 30 images for whom ground truth images were available. We compared results with ground-truth images by first performing binary classification on predicted labels using k-means clustering(with two clusters) followed by calculating IOU between generated binary image and ground truth image.

We are getting satisfactory IOU values between 80 to 90 percent. IOU metric plot ->



We also created a histogram of probability/ intensity generated after segmentation (before clustering), to observe distribution of intensities our model generated.



Training with Dice Loss :

We trained a model with dice loss rather than cross entropy loss function and found increment of 3-5 percent in accuracy of the trained model.