4360/6360 Final Project Update 2
Albert You, Jackson Cown, Alireza Vaezi
Dr. Shannon Quinn
11/18/2021

Sartorius - Cell Instance Segmentation (Kaggle Competition)

Review:

We are continuing our work to understand and perform cell segmentation on the SH-SY5Y Neuronal cell line within the constraints of the Sartorius Cell Instance Segmentation challenge on Kaggle. This is an important Machine Learning task because increases in efficiency of neuronal imaging and analysis directly mitigate the bottleneck in pharmaceutical research of drugs related to treating issues caused by these cells. Previously, we discussed generating a coarse segmentation to separate out cell instances, then passing these instances as inputs to a more complex segmentation model. It is worth noting that it may be more practical to perform semantic segmentation on all cells at once, then separate out the specific instances, instead of the inverse.

Updates:

Although training, testing, and annotation data are provided by the competition, they were not in a form where they could efficiently be passed through a training loop in batches for learning in PyTorch. The annotation data is stored in a CSV file, using run length encoding indexed at 1. After defining functions to decode this data and build out annotation masks, we are able to construct a dataset using a predefined Dataset class native to PyTorch. We can use PyTorch's native Dataset and DataLoader classes together to allow for syntactically simple batch iteration during training. We have also been doing more explicit experimentation with building out the U-net architecture in both TensorFlow and PyTorch. At this point, we are unsure which framework will be more practical during the training process, so we are exploring both options to understand the limitations of each.

Next:

The next and final stages of our project include finishing construction of our model based on the U-net architecture, training our model, and fine-tuning our model. It is also likely that we will need to focus on creating augmented data for training in order to reach the desired invariance. This should be a relatively easy process; the PyTorch Dataset class includes functionality for composable transformations on image data using the torchvision library. Another issue we are currently exploring is the size of the images we are segmenting. The segmentation architecture works fine with images of 512x512 and 128x128 dimensions, however, the images we have at hand are of size 704x520 which result in a shape mismatch when it comes to concatenating the layers. As opposed to problems like image classification, we can't take a random crop of the correct size from our image and pass it through. Historically, convolutional architectures have used methods such as cropping the correct size image from each corner and the center of the image, then combining the outputs from each cropped image into one final image. If the image

sizing is an issue, we may have to explore the practicality of a similar strategy. Otherwise we need to modify the dimensions of the layers to fit our dataset.