Capstone Project #2 Milestone Report

Chris Malec

Statement of the Problem:

Biological research creates enormous quantities of image data. Often times, it requires experts to infer the boundaries of different cellular structures or anatomical features. The more computer vision can be harnessed to identify features in microscopy images, the faster data can be gleaned from scans. Since hand annotation can take much, much longer than the actual data collection, this would speed up data collection drastically, leading to more insights and discovery.

Dataset Collection and Cleaning:

Specifically, this project will look at the identification of mitochondria within neuronal cells from Scanning Electron Microscope (SEM) images. A relatively new technique in this field is known as Focused Ion Beam - SEM, in which a three dimensional tissue sample is sequentially imaged by the SEM, and then a layer is ablated (vaporized) by a beam of focused ions, allowing the layer below to be imaged. This procedure was used to create a 5 micron x 5 micron x 5 micron volume of data representing neuronal cells.

Though there are many features of interest in a biological system the mitochondria are particularly important since they create the energy for the cell. Therefore, a group of researchers took the time to hand annotate 330 images to separate out mitochondria pixels from non-mitochondria pixels (**CVLab SEM dataset**). This dataset is notable in that the data is truly three dimensional, the data is given as an image stack in which each pixel is actually a 5 nanometer x 5 nanometer x 5 nanometer voxel.

The filetype for the image was a series of multipage TIF files. There were four files, two containing training images and the groundtruth images (images whose pixels are annotated as 1 for mitochondria and 0 for non-mitochondria), and two files containing testing images and their respective groundtruth images.

Once the files were downloaded, they could be converted to an ImageSequencer object through the PIL (Python Image Library) package. This object allowed me to easily iterate through the sequence of tif images and create a four dimensional numpy array. I chose the dimensions to be compatible with conventions used by tensorflow, which I plan on using later, therefore, the numpy array had size # examples x # channels x # pixels wide x # pixels high. Since electron microscopy images are in grayscale, there was only one channel, but again, it is kept to keep the array in tensorflow's expected format.

I made a function to do the necessary operations to convert the given file to a numpy array and used it on all four TIF files. Finally, I saved the numpy arrays for use in further notebooks, and took a look at one of the image slices to get an idea of what the data looked like.

Exploratory Data Analysis:

I had several questions to answer about the data. First off, I visualized the 3D volume by looking at several slices along the z-axis of the imaged volume. I also created a visualization of the outside of the rectangular prism of image data. Since the resolution is roughly the same in all three dimensions, the xz and yz planes look very similar to the xy plane, except narrower.

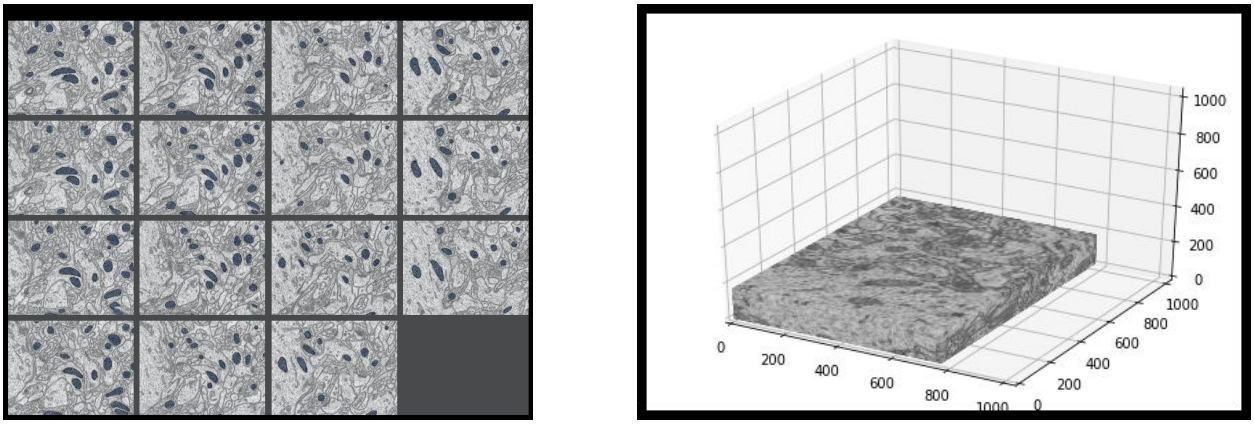

Figure1: Image slices along the z-axis and image volume as a rectangular prism

What is the average mitochondria area, the average volume?

Working with the ground truth images for the training data, I was able to add up the average total area for each image by counting the number of pixels that were labeled as mitochondria for each image. However, I needed to also count the total number of mitochondria in each image.

This would be exceptionally tedious to do manually, so I used a simple algorithm that counted the number of ‘humps’ in each line of an image. Each hump was considered to be a mitochondria, and when the number of humps changed, it was assumed that the raster line had moved beyond some number of mitochondria. This algorithm agreed with my manual count for three random selected images.

I applied the same reasoning to find the total number of mitochondria in the volume, by applying my earlier function to each slice, I assumed that when the number of shapes counted changes, then the image slices had moved beyond one of the mitochondria. In this way, I counted the average number of pixels per mitochondria and average number of voxels per mitochondria. I multiplied by 25 nm^2 to get an average area of 70,013.9 nm^2 (a square about 53 pixels on a

side) and by 125 nm^3 to get an average volume of 13,430,224.3 nm^3 (a cube about 48 pixels on a side).

In later steps, these estimates can inform me as to how large an image must be in order to contain at least one mitochondria on average.

Are the cells spherical, elliptical, some other odd shape?

By calculating the edges of the mitochondria from the ground truth images, I could reconstruct a 3D scatterplot that displays the surfaces of the 3D mitochondria. Some mitochondria are definitely spherical, while others are stretches out. Most shapes appear to be convex, which is good, because my counting algorithm wouldn't work well if the mitochondria routinely hooked around or made odd 's' shapes.

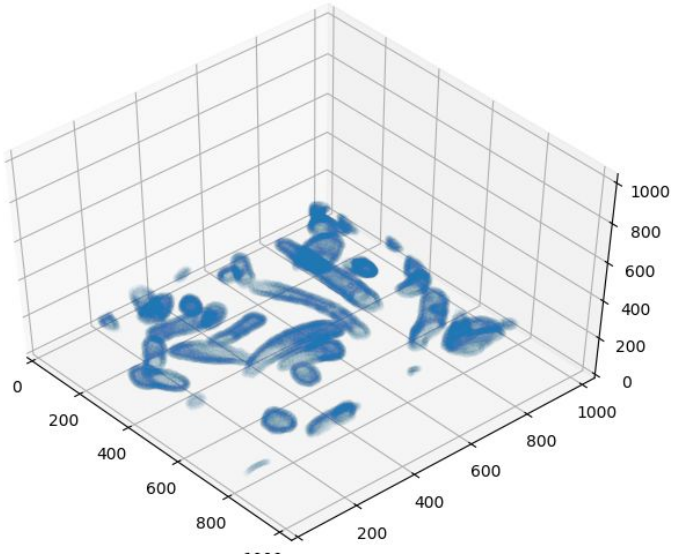

Figure 2: 3D scatterplot of mitochondria surfaces.

How much variation is there in contrast, is there a difference in the two regions?

The microscope image is really a measurement of the intensity of electrons reflected to each pixel from the surface. The microscope operator will typically try to prevent saturation of the pixel intensity to either zero or maximum. Indeed, creating a histogram of pixel intensities, reveals that the pixels do not span the entire possible range, and there is no pixels at saturation value.

When the two regions are plotted separately, it can be seen that the mitochondria are darker than the general image. This can also be seen from casual inspection of the image slices. However, the pixel intensity range in the mitochondria region entirely overlaps the pixel intensity range of the non-mitochondria region. This means that though the mitochondria by in large

reflect less electrons than the non-mitochondria regions, this cannot be used as a good method of identifying the two regions.

It's good to know that the solution to the problem at hand cannot be achieved simply by setting a threshold level and taking pixels with a value lower than the threshold as mitochondria. It will actually be worthwhile to train a neural net to segment the images.

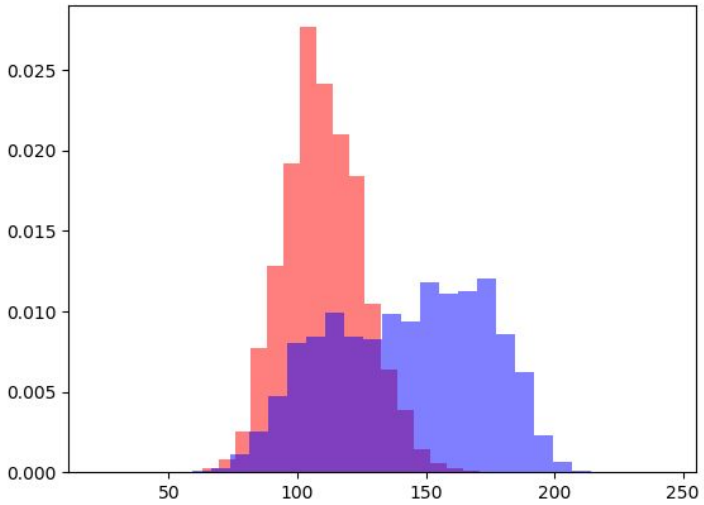

Figure 3: Histograms of mitochondria (red) and non-mitochondria (blue) pixel intensities.

Statistical Inference

An important question in the Data Science problem is how to present the data to the machine learning model. Should the data be presented as a series of image slices? Or maybe the data should be presented as a series of volumes. The thicker the volumes used, the less training examples, but perhaps they are more useful examples. It is important to note here, that the original paper describing U-Net image segmentation used only 30 training examples to achieve effective image segmentation.

With this issue in mind, I wanted to look at image correlation between different slices along the z-axis. By this I mean, testing how correlated each image slice was with the image at the bottom of the stack. Subsequent image slices are not independent, but pictures at different depths of many of the same cellular structures. Presumably, the bottom image will be correlated perfectly with itself, a little less with the image above it, and a little less with the image above that.

I wrote a function to calculate the Pearson correlation coefficient of two series of pixel intensities, as well as a 95% confidence interval. I could find built in functions to get the correlation coefficient, but the confidence interval functionality will take more looking. I then

performed the statistical test on each layer of the training data in turn to create a graph of correlation coefficient vs distance from the bottom image.

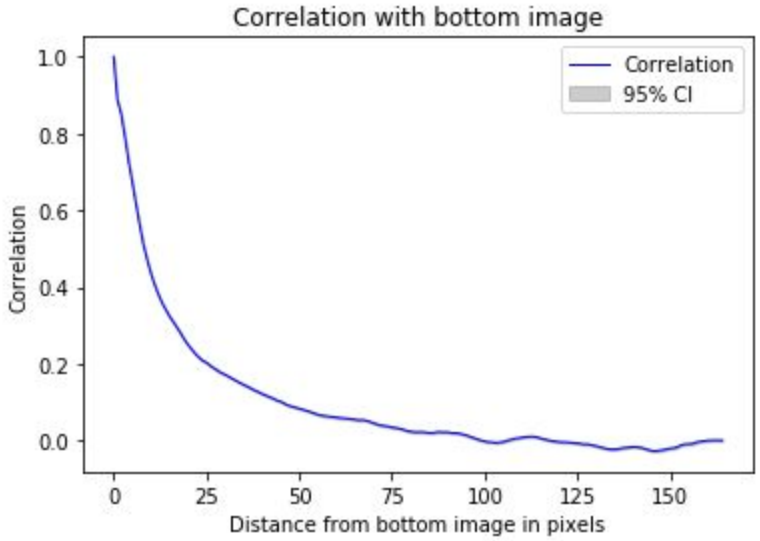

Figure 4: Correlation coefficient vs distance from the bottom image slice.

Another important concept in image segmentation, particularly for microscopy data (which can be scarce) is data augmentation. Data augmentation involves transforming the image by shifting, resizing, flipping, distorting, or otherwise altering the image to increase the volume of training examples as well as to avoid overfitting.

I was interested to know how well, a transformation such as flipping the image from left to right might decorrelate adjacent layers. It turns out, such operations are very effective, so that data augmentation can be used to greatly increase the number of training examples that appear independent of each other.

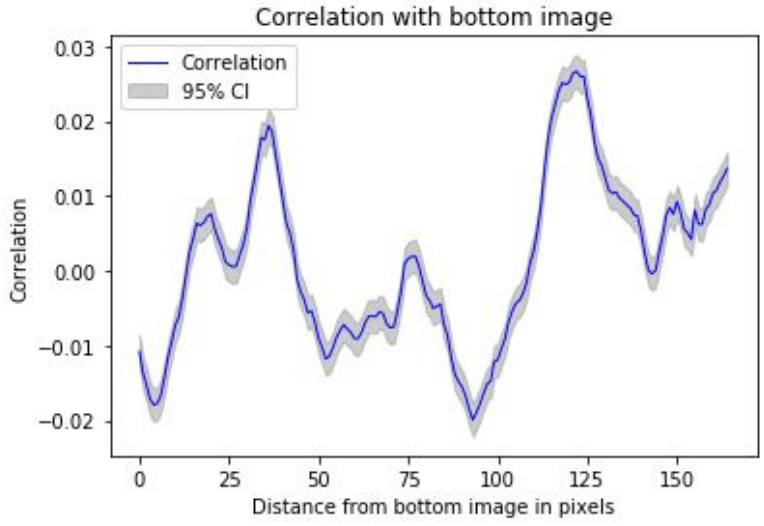

Figure 5: Correlation coefficient vs distance from the bottom image slice after a horizontal flip.

Noise and distortions were other transformations I looked at. Adding random noise, which amounted to adding a random pixel intensity drawn from a normal distribution, affect the correlation of adjacent images as expected. Greater noise lead to images being less correlated.

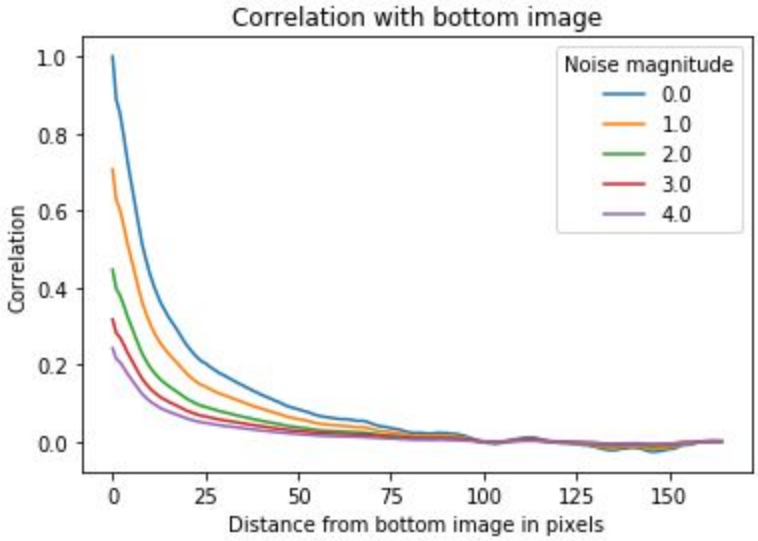

Figure 6: Correlation plots with different magnitudes of random noise.

Finally, distortions were introduced by creating a random vector field that displaced pixel intensities from the original image and then interpolated back to the original grid. This results in the borders of features being altered. Two parameters, alpha and sigma control the distortion. Higher alpha creates larger distortions, meaning that the original pixels are moved farther away on average. Higher sigma smooths out the distortions so that the borders of features remain roughly intact though the exact shape may be changed.

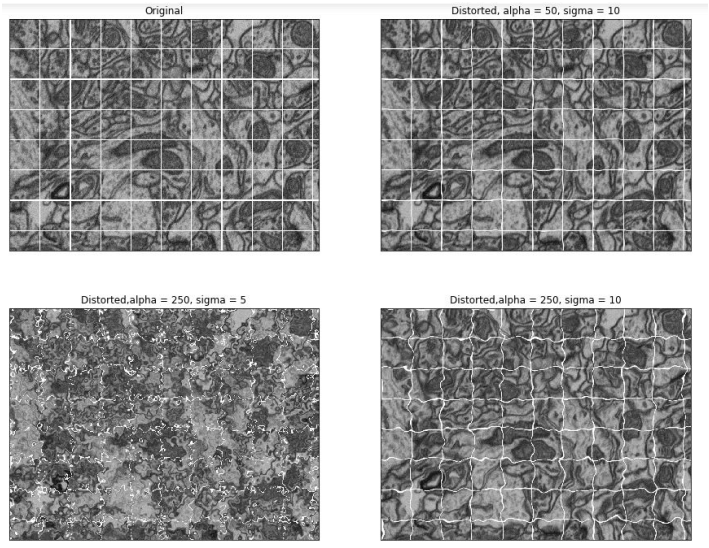

Figure 7: Examples of image distortion via a random vector field and smoothing function.

These distortions caused a decorrelation of adjacent image slices. Even for somewhat large distortions, the effect is less than for a transformation such as a flip. It is important to keep in mind, that decorrelating the images is not the entire goal of the transformations. Training a model that is robust to small changes from the training data and so performs well on unseen data is an equally important goal.

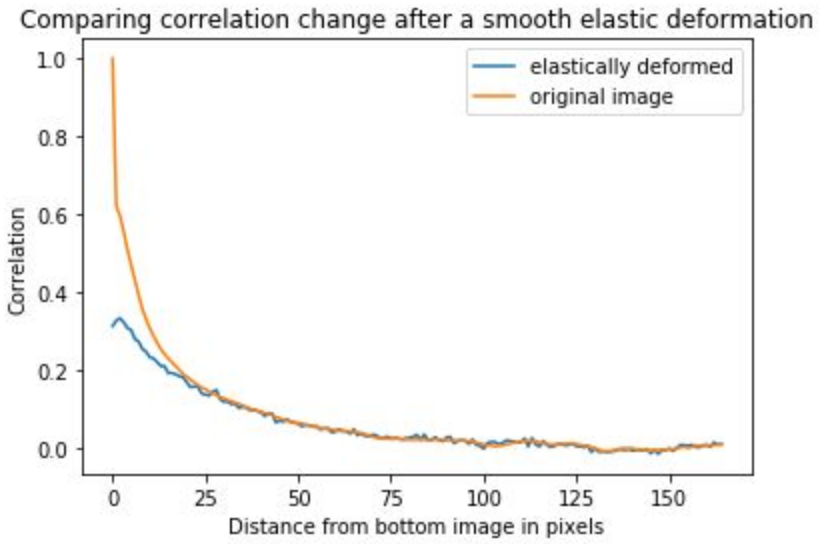

Figure 8: Correlation vs distance from bottom image for no transformation and a distortion described here.