**GROUP MEMBERS:** Chad Conley, Lucas Switzer, Ryan Walden

**USAGE:**

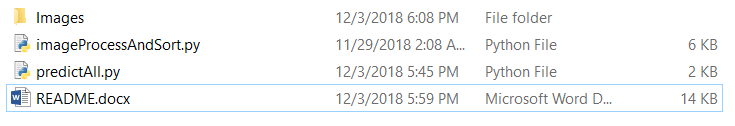
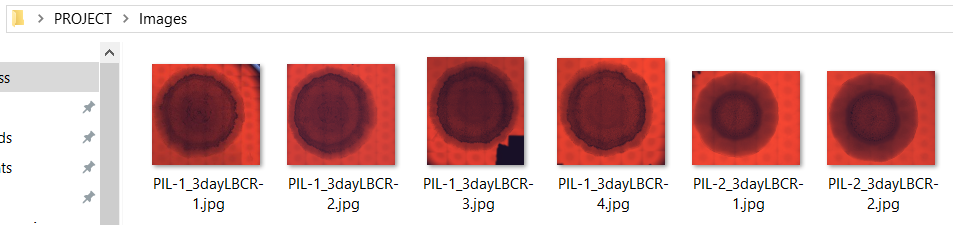
Required Software

Python 3.6.7

Required Packages

OpenCV, shutil, os, Numpy, Glob, requests

Included in the submission is a ‘.zip’ file that has everything needed to run the neural network. The only setup required is moving all raw images into a folder labeled ‘Images’ in the same directory as the extraction:

Run *imageProcessAndSort.py* (this will take some time). This program passes all the images contained in the folder through our preprocessing and sorts the images by their Environment of origin or *env* column in the spreadsheet. The processed and sorted images are then split into folders ~75% in training and ~25% in testing data. After this step you would have to manually upload the training images to a NanoNet and wait for the network to train but we have already trained a NanoNet which is accessible through the *predictAll.py* program. The program returns the probability of the classification for the bacteria colony images in the test folders one at a time.

Note: If the program seems stuck, waiting for the probability data, sometimes pressing <ENTER> may “un-freeze” it.

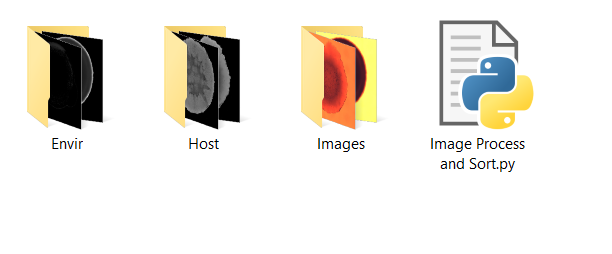
**REASONING:**

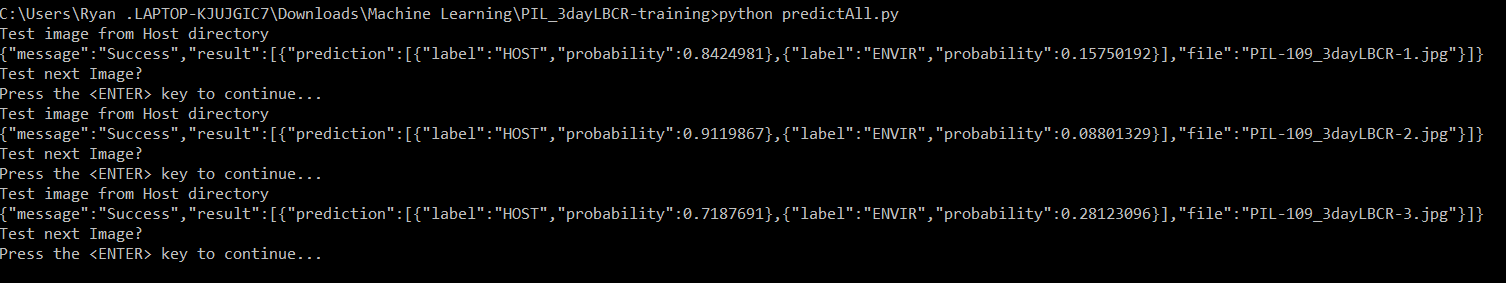
We decided to use NanoNets, a pretrained network, for our research for a few reasons. The network runs faster, it requires less data (our dataset is very small and would be difficult to use to achieve accurate results in its own network), and it is generally more accurate in similar studies that we observed (~ 87.5% Accuracy).

We applied three methods in the preprocessing of our images. The bacteria were segmented in order to remove any additional non-essential information from the image. In some images it appears as though writing on the slide or paper/sticker has been included in the image. The segmentation removes that. The images were converted to grayscale. The red die used on the bacteria is inconsistent between images and offers no useful information because of this. Grayscale removes this ambiguity. The images were resized. Large images take much longer to train with a complex neural network and a smaller image size will process much faster. Furthermore, the NanoNet limits image uploads to 10MB maximum per image.

**SCREENSHOTS:**

**Python Data Segmentation and API Calls**

****

****

**NanoNet API**

****