**How to Use The “RNN Data Prepper” Jupyter Notebook**

**Packages:**

If you can run the first code block with no problems, this section can be ignored.

First and foremost, OpenCV (and potentially other packages depending on your Anaconda version) may not be in the Jupyter Notebook environment by default. The easiest way I’ve found to handle this is to use the code I’ve commented out in the “imports” block (the first one). The commented out code does the following:

1. Imports sys, a package that I believe all recent Python versions have that allows for virtual environment manipulation
2. Uses sys to install opencv-python, which is OpenCV’s package for python

The line immediately following the commented out section imports cv2, which is what we need from opencv-python.

The code only needs to be run once and future running of this commented out code will cause an error along the lines of “you already have this package.” I know of no further issues this will cause, but I wouldn’t test it as environments can be tricky (also it makes doing the imports take substantially longer).

**Data Processing:**

The code begins by accessing the data within the images folder. This directory is assumed to be collections of folders with the following assumptions:

1. Each directory within images represents one culture growing over time. The first image is the beginning of the culture, and each following image is the culture after a certain amount of time. The “certain amount of time” should be the same for every sample, but I doubt it will be too much of an issue if it isn’t.
2. The images are saved as NEFs, but this is relatively easy to change in the code (simply CTRL-F “NEF”)

The data begins as a list of images from each directory, with a “|” as a delimiter between folders. Next, any sequential images are paired and these 2-tuples are saved in a list. We then use the encoder to convert each of the image tuples to latent vector tuples using the encoder, and save the training data as “train\_data\_rnn.z” in the images folder.

This whole process shouldn’t take too long, it took less than 10 minutes on my machine

**Future Work:**

Carlos has spoken about the RNN taking into consideration the action taken when predicting future growth, and while I can see this being an effective strategy for different domains, the extreme reduction in dimensions from 128x128 images (16384 dimensions) to 2048 dimensions (1/8th reduction exactly) coupled with the relatively short time between photos (30 minutes if I am not mistaken) indicate to me that this would complicate the problem for the RNN for minimal benefit. However, if the latent vector dimensions were increased to say 4096 (1/4th reduction) or 8192 (1/2 reduction), I could see this potentially being a worthwhile endeavor. Whether actions are considered or not, increasing the dimensions of the latent vector would allow for the RNN to make more accurate and nuanced predictions.