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

**Packages:**

Potential issues with OpenCV have been discussed in Readme’s for files that need to be ran before this one. Check those docs for information.

**Data Processing:**

The code begins by making a list of all images within directories of the “images” directory. See the assumptions made for the VAE trainer. It then puts all these images through the trained encoder to get their latent vectors (z) and saves them. It then puts all these latent vectors through the RNN to get their predicted growth over time vector (z’) and saves these.

**Action Processing and Future Work:**

Next, the script assigns an action to all the z-z’ pairs. Since the data we have is difficult to process algorithmically, right now the script simply assigns 22 to all the actions. The actions can be made with higher dimensions easily, as we’d just need to make a dataset that has all the dimensions as information and then alter the current action assignment section of code to format this information into the way we want (dimensionally speaking) into a list that can be appended to the “actions” list. This is what is currently done with the one dimension being action taken, making it a list of lists.

The easiest, but potentially less robust way, of matching images to actions is to just create one large single-column excel sheet/csv of actions in the chronological order they were taken. This method requires no delimiters between sessions or anything like that, as the data of z and z’ vectors are already sorted chronologically by the script (due to the images being sorted chronologically in the directories by alphanumeric ordering, the naming convention used for the images is perfect for that). This would mean you could load the action dataset in using pandas and simply go line by line of the dataset and assign that action information row to each z-z’ pair (requiring only one code block modification and a pretty minor one at that). However, using this method would make it hard for the RNN to consider actions one day, but I’ve discussed in the “RNN Data Prepper Readme” why I think that might not be worthwhile and thus not an important consideration anyway.

You could also make an excel sheet for each dataset with each image and make the initial image loading also load in the actions taken. This would require more refactoring of the code written than the previous method, as both the action pairing cell and the image importer cell would need to be changed. However, this method would allow the RNN to consider actions more easily, but that code would need to be refactored to consider the actions to make this worthwhile as well.