Note: When inspecting pml files, make sure to open in a code editor, such as Atom. Your computer will want to open the pml files within PyMol, which will automatically run the script. Seeing as scripts add to or delete things, you don’t want to run them accidently.

Once you are ready to run, drop the pml file to PyMol to start the script.

At the beginning of every run, the folder “train” and “test” and all contents are deleted. This is to avoid accidently added images from multiple runs to the same folders. If you are happy with your run and don’t want anything deleted, rename the test and train folders. Uncomment ‘shutil.rmtree” if you want to accept this functionality

It’s important to update the current example input file names to the real file names. Each step will use the output file of the previous step.

1. get\_clusters\_from\_pdb.pml

Create initial list of metals with resi, resn and chain

Current input files (update): fe\_small\_example.txt, fes\_small\_example.txt, fe2\_small\_example.txt (update starting at line 13)

* This script runs for each protein in a txt file
* This script looks into a PyMol selection and returns either the list of metals (metal\_list) in a tuple of (resi, resn, chain) or all the items in the selection (full\_list), again in a tuple, with the function listselection (line 21)
* First the script uses the function listselection to return a list of metals in the entire protein
* Then the script loops through the list of metals and select resn within 3.0 radius of each metal. For each metal, it uses the full\_list of items within a 3.0 radius (line 71), to classify as a Fe, Fes, or Reiske.
* The script saves the protein, resi, resn, chain, and resn\_list within 3.0 radius to a csv file

1. Run the webscrapper\_current.py in python

Confirm classifications with MolQL

Current input files (update): record\_images\_example\_input.csv (update at line 19)

* Output file from the last step in line 19.
* Update your download location in line 25.
* Set up environment:

<https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#activating-an-environment>

conda create -n webscrape\_env python=3.7

pip install selenium==4.1.1

pip install pandas

pip install webdriver-manager

1. create\_png\_metal\_by\_protein.pml

Create the images for the neural network training and testing

Current input files (update): test\_record\_images\_molql\_example\_input.csv (added lines to the previous output from webscrapper to include all the metals, not just FES) (update at line 19)

* Set number of images per folder at line 13 and 14
* Indicate if you want to hide the metal at line 16
* This file will use the webscraper output, so update the filename in line 19
* Make sure the last column, with the proper cluster classification is called “folder2” and that the “protein” column is in the third column down in the input file
* The script will divide proteins per metal into test and train sets and randomize the order of the proteins
* The script creates a dictionary “protein\_dict” that keeps track of how many unique resi there are per protein, and which one you recently used.
* The script cycles through each protein and uses the next possible resi. Restart at the original resi once all resi per protein have been used.
* For training data, each protein/resi is assigned a rotation from angle\_comb variable. You’ll want to update angle\_comb if you want to use different angles
* For test data, each protein/resi is assigned a random angle
* The data for each image is saved to a dictionary and written out in a csv starting with record\_images\_metal\_hidden\_ and ending with the time stamp

1. create\_png\_metal\_from\_excel.pml.

Using the output excel file, you can re-create the same images with the same rotations, but with the metal hidden

Current input files (update): record\_images\_metal\_hidden\_False\_example\_input.csv (update at line 20)

1. Run a modification of https://pytorch.org/tutorials/beginner/finetuning\_torchvision\_models\_tutorial.html
2. create\_aa\_pymol.pml

Creates the amino acid image. It works the same way as the rotations for metals, and just needs to be run in PyMol.

Input files: amino\_list.csv