Instructions on getting from txt files to a trained model with confusion matrices:
The first thing that you want to do is configure your directory properly. Here is the proper directory structure that I used and fits with the code and

- single_track #this is entire project folder name
    - clean_cat_shuffle.py
    - data_preprocessing.py
    - sum_pads_arrays.py
    - txt_to_npy_arrays.py
    - STimplementation.ipynb
    - oldRawDataTxtFiles #here is a folder for storing the txt files
        - output_t.txt
        - output_d.txt
        - output_p.txt
        - output_He3.txt
        - output_He4.txt

All other folders and files will be created automatically by the code. At the beginning of each python script (not the ipynb script) there will be a function and inside that function there will be string constants with the file names for the input and output of that script. It is important to tailor these to your directory structure. The code that appears here is tailored to the above structure.

Before executing any code, set up the proper virtual environment. This is extremely important because my code imports a lot of different things. Look at the **pipfile** to see everything that needs to be added to your virtual environment.

Next, open up the file **entire_pipeline.sh** and edit the part that says “cd /home/DAVIDSON/wiclark1/exercise/single_track”. Make sure that bash navigates to the folder where the 4 python scripts are located.

Also, if you want to get notified via email when entire_pipeline.sh finishes running, add your email to the part that says
--mail-user=wiclark1@davidson.edu

Also, edit the part underneath the comment in **entire_pipeline.sh** that says “#commands for data preprocessing:” according to your choice of power transformed feature scaling or log scaled feature scaling. See thd file data_preprocessing.py for more information.

Otherwise, delete the two lines beginning with
#SBATCH --mail-user
#SBATCH --mail-type

Then, turn **entire_pipeline.sh** into an executable by calling
<chmod +x entire_pipeline.sh>
Then, submit a job to run **entire_pipeline.sh**
<sbatch entire_pipeline.sh>
Then,
<tail -f slurm-1234.out>
with 1234 tailored to your actual job number will show you the click messages and the tdqm progress bars on the for loops. As well as the transitions between different python scripts during the process. Make sure to monitor this occasionally to check for error messages.

Then, after this script finished running, you will have the following directory set-up:

- single_track #this is entire project folder name
    - clean_cat_shuffle.py
    - data_preprocessing.py
    - sum_pads_arrays.py
    - txt_to_npy_arrays.py
    - STimplementation.ipynb
    - oldRawDataTxtFiles #here is a folder for storing the txt files
        - output_t.txt
        - output_d.txt
        - output_p.txt
        - output_He3.txt
        - output_He4.txt
    - smallDataPreprocessingFiles
        - all_events_all_particles_array_shuffled.npy
        - 2021-05-10 21:47:36.372906train_input_log_scale.npy
        - 2021-05-10 21:47:36.372906val_input_log_scale.npy
        - train_targets.npy
        - val_targets.npy
    - smallDataSummedChargeArrays
        - deuteron.npy
        - he3.npy
        - he4.npy
        - proton.npy
        - triton.npy
    - smallDataParticleChargeArrays
        - deuteron_array.npy
        - he3_array.npy
        - he4_array.npy
        - proton_array.npy
        - triton_array.npy

Next, open up STimplementation.ipynb and run it from top to bottom, making sure to edit the parts where you load the training/validation/target data from either the log_scaled or power_transformed data sets, as you can't do both at the same time.

You will also need to edit some of the folder naming for the **call-backs** section when you switch between power transforming and log scaling.

Once you get to the part where you make predictions, go into one of the newly created folders where the checkpoints are stored and see how many epochs the model got through before early stopping nabbed it. Then load these weights into the model and make predictions and visualize the confusion matrices.