Write-up for Kaiwen Wei (Team name kevinawei on Kaggle)

I used a RNN model with 150 time steps. The hardest part of getting it to work involved the input of getting 150,000 data points into data that could easily be fed into my model. I found that my model had better performance when using GRU rather than LSTM and with relu activation function. My model has 3 layers, 1 CuDNNGRU layer and 2 dense layers that condense the dimension of the output to 10 and then 1(time to failure). The model was trained with adam optimizer and mae loss function to achieve the best possible results. It was trained for 30 epochs with 1000 steps per epoch and no dropout, as there was no overfitting in any of my tests. One of the other main challenges was trying to optimize the architecture of the layout beyond just the initial architecture. After playing around with many hyper-parameters, it turns out that the initial dimension and amount of parameters led to the best results, with my changes mostly resulting in lower accuracy. Another of the hardest challengers for me was that the results could vary so heavily and thus skew the testing results a lot. One wrong answer that was off by a lot would heavily impact the MSE more so than just an accuracy check on another type of question. That made it imperative that the model was never off by too much, even if it was never exactly accurate. Packages that need to be imported to run my code include matplotlib, keras, tqdm, pandas, and numpy. To recreate my results, put a train.csv file in the same folder as the python module as well as the sample submission so that it can identify the ids that it needs to predict for the submission. Also, put in a test.csv file in that same folder. Then navigate to that folder on the command line and enter rnn-earthquakeprediction.py.