AI for Accelerators: Reinforcement Learning for Booster-GMPS System Documentation

PROJECT BASICS: Training an RL agent to replace a PID controller that currently regulates the Gradient Magnet Power Supply in the Booster complex. This PID controller compensates the minimum and maximum current values to reduce error when compared with set point. Due to the risk of system failure, we had to develop a substantial part of our codebase offline. The surrogate model is created to mimic Booster-GMPS dynamics with high fidelity in order to serve as a safe environment for our RL agent to train in. See this presentation for a brief overview.

CODE:

- Original Repo
- My Simplified Repo
- BOOSTR Dataset Repo

PUBLISHED DATASETS:

- Full Dataset
- Partial Release

RELEVANT PAPERS:

- Dataset Paper
- Preliminary RL Paper
- International Particle Accelerator Conference Paper

MAIN POINT PEOPLE:

- Malachi Schram (wrote most of the original surrogate model and RL code)
- Jason St. John (wrote code that polls the ACNET (Accelerator Control Network) variables, helped building out preliminary pipeline, advised on a lot of the offline ML development)
- Accelerator Experts: Kiyomi Seiya, Brian Schupbach
- Hardware People:
    - Implementation on Board: Bill Pellico
    - Implementation on FPGA/Board: Jovan Mitrevski
- Databricks Support: Sheila Stewart

DATA

Accessing Preprocessed Data on Databricks: see example notebook too!

1. Getting access: Ask one of Jason St. John or Burt Holtzman to add you to the workspace.
2. All the data you want is in /mnt/gmps/final_for_release/+str(var_name) or /mnt/gmps/delta/silver/+str(var_name)

3. Read data like so:
   spark.read.format("parquet").load('/mnt/gmps/final_for_release/'+str("BVIMIN")).orderBy('time')
   a. Join on timestamp column, chain join statements to join multiple devices: BACMNIG.join(BACMNPG.drop("file"), on="time")
   b. Need to order by the timestamp column after joining because joining doesn't preserve ordering for some reason: all_data.orderBy("time")
4. See dataset paper for detailed variable explanation! We collected 54 devices out of a possible 200,000 within the Accelerator Control Network (ACNET) system
5. So far in our development, we have only been using data from March 10, 2020 (even though we have plenty more). You can access the March 10 data using code like:

   BACMNIG = spark.read.format("delta").load('/mnt/gmps/delta/silver/'+str("B:ACMNIG")).select(col("file"), col("goldTS").alias("time"), col("value").alias(str("B:ACMNIG"))).where(col("file")=="MLParamData_1583906408.4261804_From_MLrn_2020-03-10 00:00:00_to_2020-03-11 00:00:00.h5")

   Truncating as follows gives the full data collected for that device:

   BACMNIG = spark.read.format("delta").load('/mnt/gmps/delta/silver/'+str("B:ACMNIG"))

Data Pipeline:

1. Accessing Raw Data on Tape-Backed Storage to Move to the Cloud
   a. Instructions on how to set up Cygwin: https://home.fnal.gov/~jamieson/cygwin.html
   b. Logging into Cygwin with your username and go to relevant directory:
      i. Kinit -f dkafkes@FNAL.GOV
      ii. Klist -f (to create a forwardable ticket)
      iii. SSH dkafkes@accelaigpvm01.fnal.gov (this is your virtual machine)
      iv. Whoami (just to check)
      v. Cd /pnfs/ldrd/accelai/tape or /accelai/app/2afkes/
   c. Check to see if things are staged
      i. One file: cat ".(get)(MLParamData_1576242590.6245875_From_MLrn_2019-12-12+00:00:00_to_2019-12-13+00:00:00.h5)(locality)"
      ii. all files: ls -ltra MLParam*h5 | awk '{print "cat \".(get)("$9")(locality)\""}' | bash
   d. Explicitly stage things

i. cat ".(touch)(' MLParamData_1576242590.6245875_From_MLrn_2019-12-12+00:00:00_to_2019-12-13+00:00:00.h5')(stage)(120)"

e. Guide to Python files in data pipeline folder
   i. Relevant analytics files: stdev_finder.py, metadata.py
   ii. Stage and move to the cloud: move_over.py
f. Other tips if you aren't familiar with Linux things
   i. Check to see what is still running
      1. Ps aux | grep dkafkes
   ii. Using a screen: https://linuxize.com/post/how-to-use-linux-screen/
      1. Screen
      2. Ctrl, a, d to detach
      3. More commands: https://www.blackhillsinfosec.com/wp-content/uploads/2017/02/Gtftf1qMeUiq_kP_bQVa5QOBWO5yFsU5cjt96PJBeB4ToW2BOcxclotSKW5pIKh6uQDeiAs2048.jpg
   iii. Moving files from local → server (you have to be local on Cygwin)
      1. scp /cygdrive/c/Users/dkafkes/Desktop/fermi/accelerator-reinforcement-learning/for-Jason/stdev_finder.py dkafkes@accelaigpvm01.fnal.gov:/accelai/app/kafkes/
   iv. Moving files from server → local
      1. scp dkafkes@accelaigpvm01.FNAL.gov:/accelai/app/stjohn/workflow/MLParamData_1606939767.061787_From_ArkIv_2020-03-11+00:00:00_to_2020-03-12+00:00:00.h5 /cygdrive/c/Users/dkafkes/Desktop
   v. To install python modules (now you have to run with python3)
      1. echo export PYTHONPATH="$PYTHONPATH:~/lib/python2.7/site-packages/">>~/.bash_profile
      2. source~/.bash_profile
      3. pip3 install –user pandas

2. *Preprocessing Instructions on Databricks: all in old workspace folder!*
   a. Getting access: Ask one of Jason St. John or Burt Holtzman to add you to the workspace.
   b. Creating a parquet file from all the h5 files: save_as_parquet
   c. Building a Delta (Databrick's special version of a Data) Lake: create_db
   d. Time alignment preprocessing: the way the devices were polled resulted in them being sampled at different rates, resulting in a lag from the desired 15 Hz "heartbeat" of the Booster. These differences were not constant across or within devices. See slide 9 of this presentation for a brief overview of how we solved that problem.
      i. time_align_parallel which is run by ScheduleDevices (utilizing parallel_run Scala code to set up concurrently running notebooks on the cluster)

ii. Notice the import of a csv file mapping parameter to maximum lag time in ScheduleDevices. You will need to recalculate this for new data using the optimal_interval notebook (in outputs folder).

SURROGATE MODEL

1. Mlflow surrogate current in surrogate model folder is probably all you need. The cell at the top ensures the API doesn't close the interaction between the run and MLFlow if run time > 3 days.
    a. MLFlow Experiment is already set up! See all experiments here. I've left notes on some of the runs, but also in the RL section (below) you will see I've pulled out some relevant ones.
        i. There's a really nifty time travel feature that lets you revert the notebook back to how it was at a particular date and time when a version was run. You might find this useful.
        ii. See mlflow_quick_start_track_server (1) for an interactive introduction to MLFlow.
    b. Koalas mlflow surrogate is everything I got done before I left trying to ~sparkify~ the code so we could train on more than one day of data.
2. ML-UQ to check what the CD should be set to in inference mode.
    a. See the overlapping distribution plots at the bottom. You want the training reconstruction to match the historical data as closely as possible.

@dkafkes: needs to be sparkified

REINFORCEMENT LEARNING

Follow the installation instructions here: https://github.com/fermilab-accelerator-ai/control-for-accelerators-in-hep ~~~Don't forget to build the egg!~~~

Most Relevant Files: You will spend most of your time changing small things here and there/running from these.

- Training/Testing:
    - surrogate_accelerator_v1.py
        - Control your action space with self.nactions
        - Be sure to make sure self.batch_id is set to something orthogonal (not overlapping) for train and test!
    - run_dqn_surrogate_accelerator.py
    - test_dqn_surrogate_accelerator.py
        - I created another file for testing just so I didn't confuse myself
- Visualizing:
    - PlotTotalReward.ipynb has the train/test reward plots, the explore-exploit "curve", histogram of actions (based on action space), and a way of

quantifying the differences between the RL agent's actions and the historical PID compensations data

Relevant Surrogate Models: I've organized these all in the ./models/'digital twin or surrogate model' folder.

- Current Surrogate: 6→2 is '../models/digital twin or surrogate model/databricks models/6to2_UQ_prelim/fullbooster_noshift_e25_bs32_k_invar6_outvar2_axis1_mmscaler_t0_D03012021-T173721_kfold4__final.h5'
  - This is good for the first 250,000 timestamps of data from March 10, 2020! It's current because that's what we had been using for the RL.
- Next move (eventually): 6→2 FULL DAY is '../models/digital twin or surrogate model/databricks models/6to2_UQ_full_day'
- Preliminary Paper Surrogate: 5→3 is '../models/digital twin or surrogate model/original paper model/fullbooster_noshift_e250_bs99_nsteps250k_invar5_outvar3_axis1_mmscaler_t0_D10122020-T175237_kfold2__e16_vl0.00038.h5'
  - NOTE: In case you ever have to go back to this, you also need to revert to the old way of scaling with MinMax scalers (see where it says transform and inverse transform?). I sort of doubt you'll have to do that though…
- I'm giving you all historical surrogates too!

Relevant Policy Models: I've organized these all in the ./models/'trained RL-DQN' folder. You can copy and paste these file paths into the PlotTotalReward notebook.

- Preliminary Paper Results:
  - Train: '../results/results_dqn_MLP_1_n128_gamma85_250warmup_train5_surrogate1_in5_out3_D04142021-T114217_v1/'
  - Test: '../test/play_results_dqn_surrogate1_D04152021-T165301_v1/'
- IPAC Paper Results (includes CD=Concrete Dropout)
  - Train: '../results/results_dqn_MLP_1_n128_gamma85_250warmup_train5_surrogate1_in6_out2_UQ_D05132021-T201402_v1/'
  - Test: '../test/UQ_play_results_dqn_surrogate1_D05172021-T142933_v1/'
- 15 Action Space results
  - With CD in Surrogate Model
    - Train: '../results/results_dqn_MLP_1_n128_gamma85_250warmup_train5_surrogate1_in6_out2_UQ_D05262021-T104923_v1/'
    - Test: '../test/UQ_play_results_dqn_surrogate1_D05272021-T102210_v1/'
  - Sans CD

- Train: '../results/sansUQ_results_dqn_MLP_1_n128_gamma85_250warmup_train5_surrogate1_in6_out2_D05282021-T113137_v1/'
- Test: '../rest/sansUQ_play_results_dqn_surrogate1_D06012021-T134031_v1/'

##

I'm going to provide some brief details on all of the notebooks in my Databricks workspace just in case you need them. I apologize for it being so disorganized, but hopefully you can organize it in any way you see fit.

File	Previous Use
Example	I made this to demonstrate loading data for the published dataset paper. You might find it useful too.
Final metadata for nature scidat	Used this for the table you see in the dataset paper that contains all of the metadata for each recorded variable.
Mlflow surrogate for IPAC	I had to revert and rerun one of the surrogate model versions for our proceedings paper that went to the International Particle Accelerator Conference.
New	This was what I was basing concrete dropout for surrogate model off of.
Overview of previous experiments	There isn't much in here, but maybe you'll need to use the commented out displayHTML command at some point so I'm leaving it.
/surrogate model/___init____	Nothing here
/surrogate model/2 variables	Originally, I was setting up code separately for different models. This was for a 2 input model. I don't think you need to revisit this code, since the mlflow surrogate current is set up.
/surrogate model/5 variables	Ditto for this.
/surrogate model/analysis	These are plotting methods.
/surrogate model/Data Generator	Work done for running on ExaLearn. I was not involved in this and suspect I imported it when I originally pulled down the repo to upload it to Databricks.
/surrogate model/dataset	Dataset methods that I later put inside digital_twin_utils.
/surrogate model/DataViewEEDM	We briefly tried decomposing our surrogate model input variables using Empirical Mode Decomposition, but we decided it would be too

	hard to implement that decomposition on the board funneling inputs to the FPGA.
/surrogate model/digital_twin_utils	These should be all the methods you need to run mlflow surrogate current.
/surrogate model/Jan29	For whatever reason, I needed to go back to look at code that was run during an MLFlow experiment from January 29. I used the time travel feature and then saved a new notebook because I didn't want to override the current version.
/surrogate model/just in case	A bunch of code for how to load the data before I decided that using the csv was faster for the purposes of training <= one day's data from March 10.
/surrogate model/koalas mlflow surrogate	I'm trying to set up sparkified training before I leave for you to come to, but I expect that this won't be done :< You will need to utilize koalas (pandas but with Apache Spark on the backend) to rework the data loading and passing such that we can efficiently train with more data.
/surrogate model/make_movie	I think this is for the RL part to stitch together different consecutive pictures into a continuous loop.
/surrogate model/method test	I was testing the loading of data from the original h5 files.
/surrogate model/ML-UQ	Another intermediate file on the journey to debugging the addition of concrete dropout to the surrogate model code. The code to plot the overlapping reconstruction and historical data is at the bottom.
/surrogate model/ML-UQ-Diana	This file was my baseline for the above one. We had weird problems with this process, which is why it was duplicated so I had something to work off of, but they've all been sorted out and the mlflow surrogate current code implemented CD so you have nothing to worry about.
/surrogate model/mlflow surrogate current	I cleaned up this notebook so it should be easy to run. This is currently how you run an MLFlow experiment and train. Once you've sparkified the training, you'll either move to koalas mflow surrogate as your go to or replace this file with the koalas version.
/surrogate model/models	These are the models mlflow surrogate current imports. Concrete dropout is already

	implemented. You might come here to change the amount of dropout.
/old workspace/causality	I used this for the Granger causality study to see how two variables related/were potentially causal to one another and, if they were, at which lag.
/old workspace/create_db	After parquet files have been created from the initial h5 files, run this to create your database and Delta Lake.
/old workspace/double_check_for_malachi	You don't need this file any longer. I was checking something from the old code for Malachi.
/old workspace/FermiBronzeSilver	This was an intermediate file that we used to help develop the code we needed for preprocessing. Everything we needed was distilled into time_align_parallel.
/old workspace/full data as parquet	I used this to create the parquet files that were published in the full release. This required me to save parquets with only one partition.
/old workspace/lags	I also tried to leverage some available packages to run the Granger causality study in R, but it didn't work well.
/old workspace/mlflow_quick_start_tracking	This notebook was gifted to me by Sheila and provides a good introduction into how to use MLFlow on Databricks.
/old workspace/parallel_histogram	More on granger causality study, I guess I wanted a histogram based on lag values and certain variables.
/old workspace/parallel_lags	This is run via ScheduleLags to calculate different information criterion based on varying lags between two potentially causal variables.
/old workspace/parallel_run	I was gifted this by Sheila. It's Scala code that helps you run multiple notebooks at once via one of the Schedule notebooks (see below).
/old workspace/save_as_parquet	Use this to accumulate parameter tables within each h5 file into parquet files that can then be made into a database and datalake using create_db. It differs from full data as parquet file because it allows for multiple partitions (which is what you want on the backend if you aren't leaving the Databricks environment).
/old workspace/ScheduleDevices	This is the code you will run to preprocess all of the data concurrently. It imports the parallel_run code and then calls time_align_parallel.

/old workspace/ScheduleLags	This is the code you will run for a Granger causality study (if you have to do another one). It imports the parallel_run code and then calls parallel_lags.
/old workspace/ScheduleParquet	This is the code you will run for saving all data as a parquet with only one partition. It imports the parallel_run code and then calls full data as parquet.
/old workspace/time_align_parallel	This is the cleaned up preprocessing code that aligns the device timestamp to the 15 Hz heartbeat frequency of the Booster. You should not need to change this, and only minimally if the data formatting changed. It is run via ScheduleDevices.
/old workspace/utils	This is the methods file for the Granger causality study.
/old workspace/validate	I used this as a part of the validation process and also to create figures based off of the Booster technician log for the dataset paper.
/old workspace/visualize	I used this to create the metadata figures for the dataset paper.
/old workspace/outputs/optimal_interval	Run this to find metadata regarding the lags between timestamps within each device. Notably, you need to make a csv mapping param name to maximum lag values which is then imported and used in ScheduleDevices.
/old workspace/surrogate model	I'd ignore this whole folder! We already have the surrogate model fixed. This is what it was like before.