AI/ML at JLab

David Lawrence

JLab EPSCI Group Lead

Jan. 28, 2021

EPSCI
Experimental Physics Software and Computing Infrastructure

Artificial Intelligence and Machine Learning

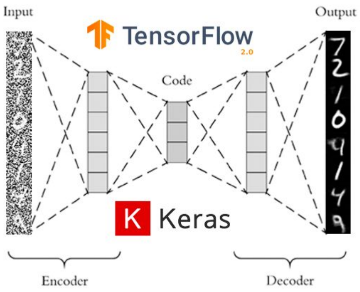
EPSCI
Experimental Physics Software and Computing Infrastructure

- AI Town Hall for JLab ENP+CST on Aug. 8, 2020
  - 24 projects presented: https://docs.google.com/spreadsheets/d/1bAsHq4Zp4pTUMqwUn3600jT3fUrFGlhN5EdCJqn2fr8/edit?usp=sharing
  - Most are current projects, a few recent, and a few planned
  - Broad categories:
    - Detector Design
    - Monitoring
    - Triggering
    - Charged Particle Tracking
    - PID
    - Simulation
    - plus a few more ...

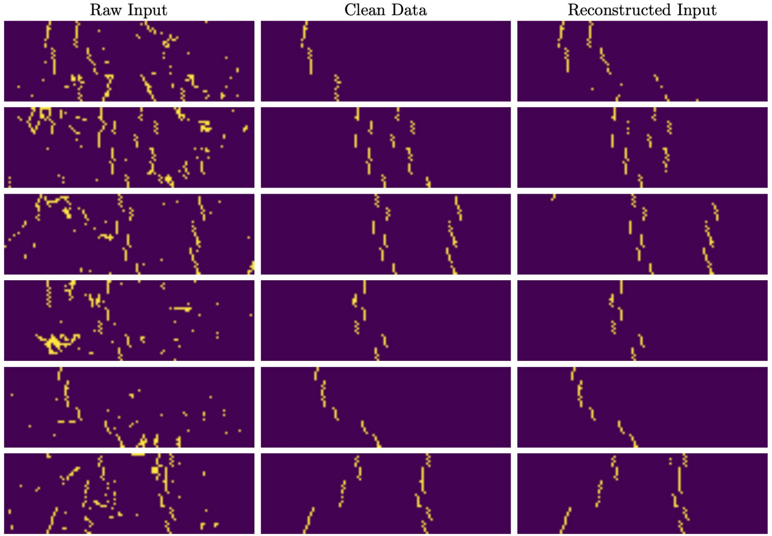
- AI Lunch series
  - Every Wednesday at noon: https://www.jlab.org/AI/lunch_series/2020
  - Internal and external speakers
  - Problem of the Quarter: https://www.jlab.org/AI/quarterly_problem (with prizes!)

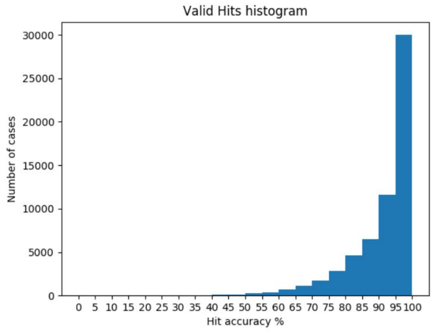
from Gagik Gavalian

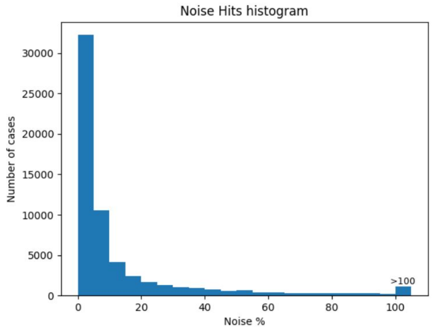
De-Noising CLAS12 Drift Chambers using AI



- Auto-Encoders are used for de-noising raw signal in CLAS12 Drift Chambers
- Network is trained with data representing all hits and hits that represent valid tracks
- Encoder learns coded representation of track segments and reconstructs (decodes) only hits that potentially belong to tracks
- Examples show one and two tracks decoded from noisy input data







- Track reconstruction average accuracy 91%
- Average noise hits ~13%

from Cristiano Fanelli

Model and FoM

$$N\sigma = \frac{||\langle\theta_K\rangle - \langle\theta_\pi\rangle||\sqrt{N_\gamma}}{\sigma_\theta^{1p.e.}} \qquad N_\gamma = (N_\gamma^\pi + N_\gamma^K)/2$$

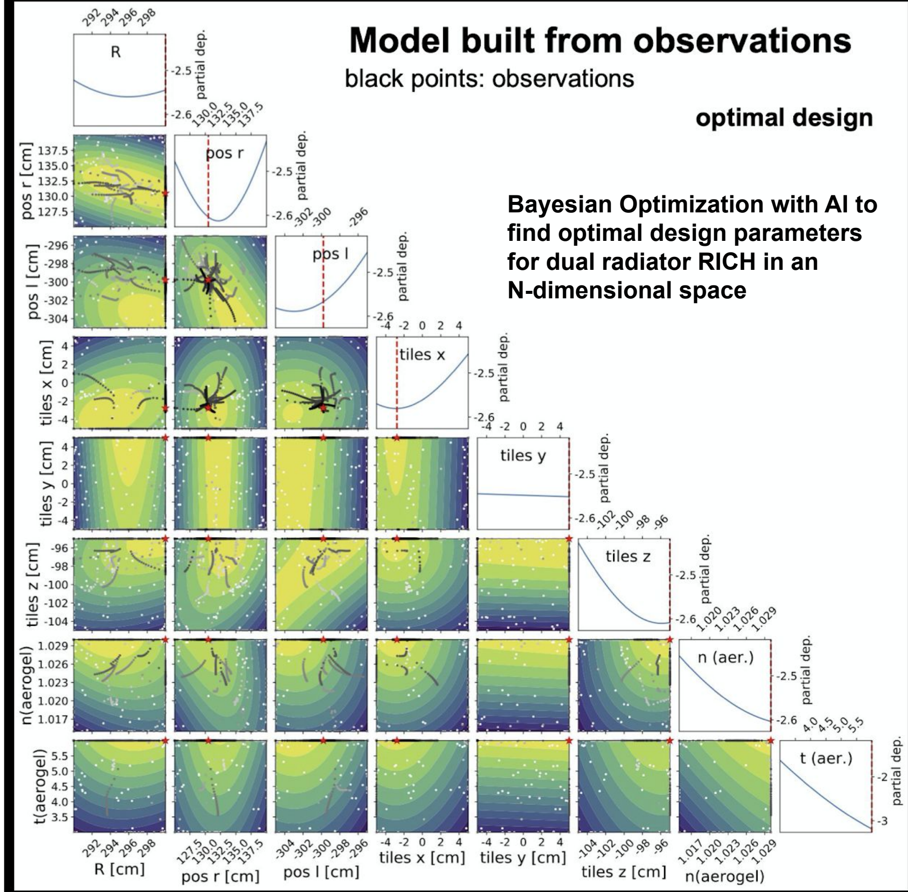
$$\boxed{h = 2 \cdot \left[\frac{1}{(N\sigma)|_1} + \frac{1}{(N\sigma)|_2}\right]^{-1}}$$

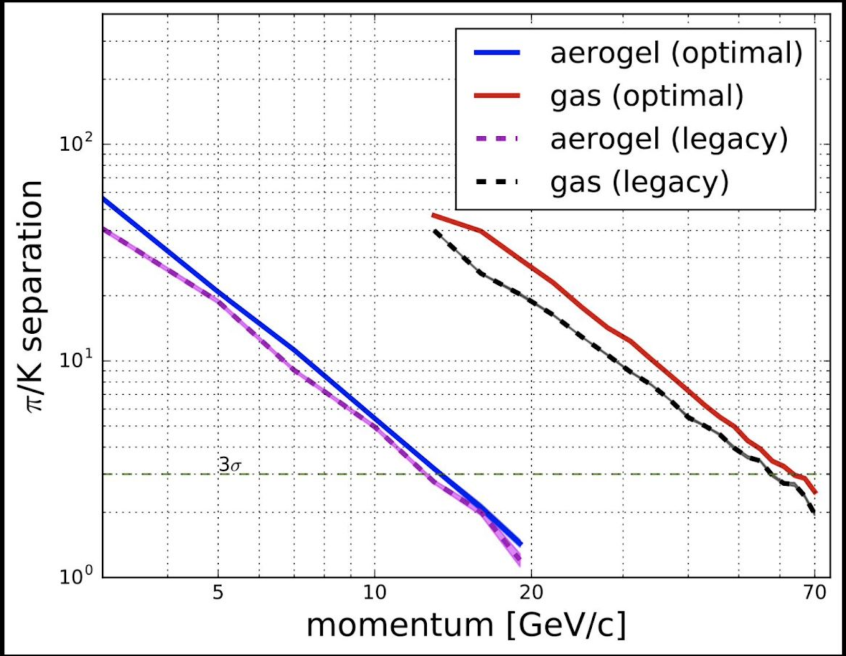
Model built from observations

black points: observations

optimal design

Bayesian Optimization with AI to find optimal design parameters for dual radiator RICH in an N-dimensional space

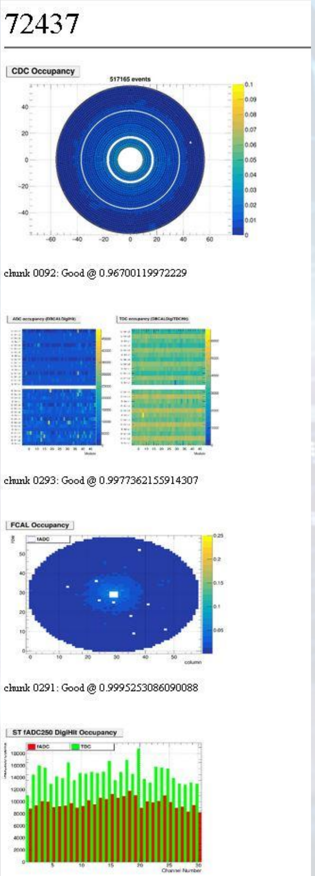




EPSCI
Experimental Physics Software and Computing Infrastructure

Hydra

A.I. Data Quality monitoring



A portion of the real-time web dashboard of Hydra inference

- Traditionally, scientists working shifts must frequently scan dozens of plots to ensure the quality of incoming data
- Plots are themselves just pictures. A.I.'s are now very good at classifying pictures.
- This is applied A.I. since it uses models already designed for image classification such as Google's Inception_v3 network
- Between 93 and 99% accurate when compared to expert labeling
    - Has found mislabeling by human experts indicating an irreducible error that is expert dependent
- Currently capable of analyzing an image in under 200ms
    - This equates to a throughput in excess of 10,000 images a day when running. (far more than a human)

from Chris Tennant

ML-based Superconducting RF Cavity Fault Classification

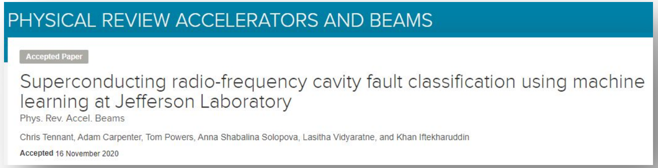
- developed and deployed an online machine learning system in CEBAF to <u>automatically classify C100 cavity faults</u>
- avoids time-intensive labeling from subject matter experts
- results are useful for:

  <u>Post-Run Analysis</u>

  use aggregate statistics for data-driven guidance for maintenance and/or upgrade activities

  <u>Post-Fault Analysis</u>

  provides critical feedback to control room operators
- system provided feedback to subject matter experts (SME) in the summer 2020 physics run


PHYSICAL REVIEW ACCELERATORS AND BEAMS

Accepted Paper

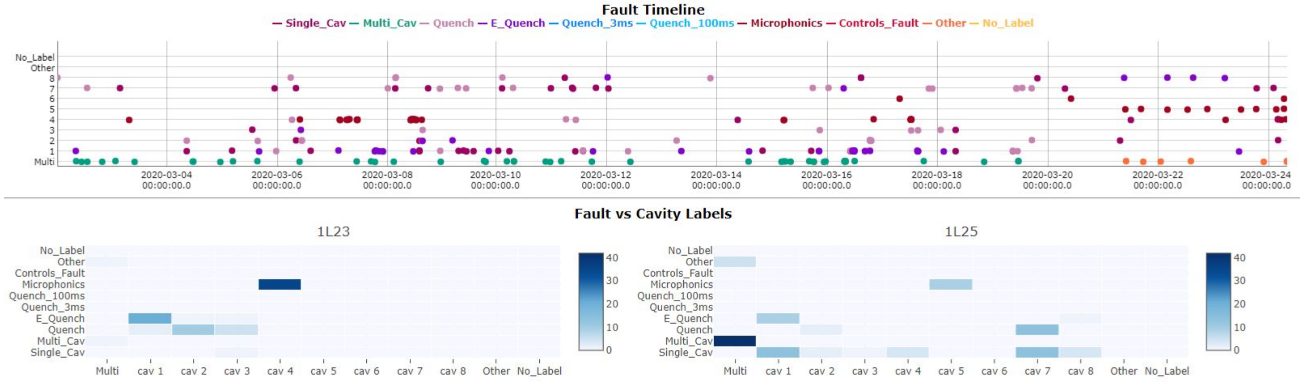
Superconducting radio-frequency cavity fault classification using machine learning at Jefferson Laboratory

Phys. Rev. Accel. Beams

Chris Tennant, Adam Carpenter, Tom Powers, Anna Shabalina Solopova, Lasitha Vidyaratne, and Khan Iftekharuddin

Accepted 16 November 2020

- online visualization tools communicate ML results clearly and concisely to operators and SMEs
- 3-year FOA awarded to extend work to address other SRF challenges



Experimental Detector Control/Calibration with A.I.

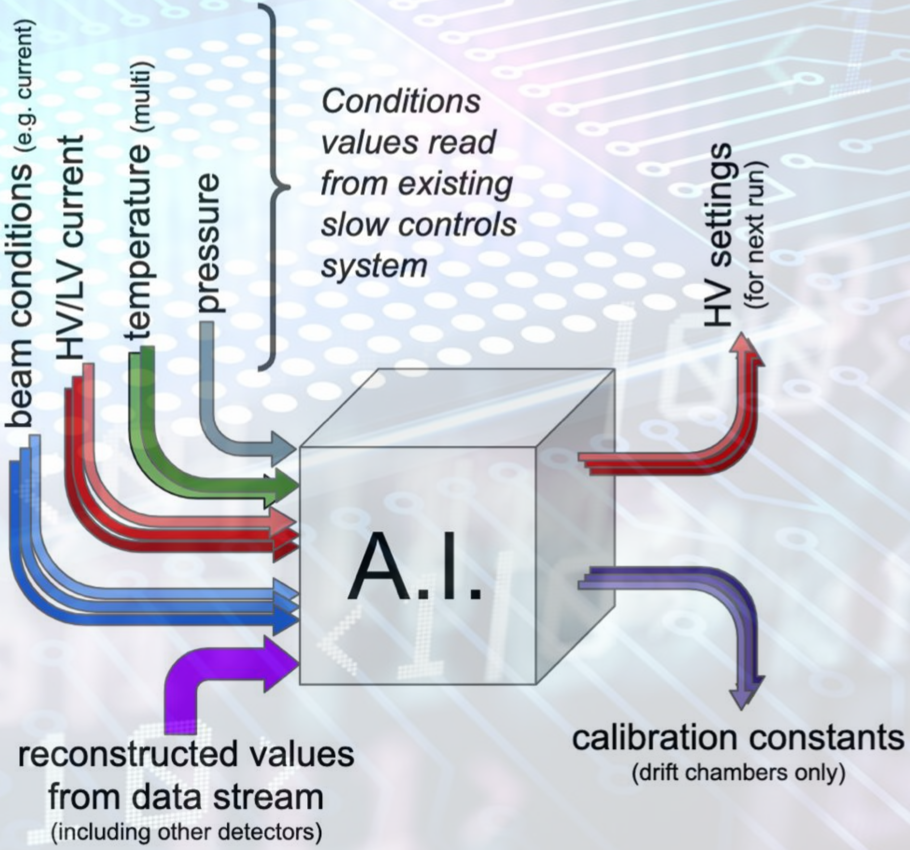
EXPERIMENTAL PHYSICS SOFTWARE AND COMPUTING INFRASTRUCTURE

Project Goals

- Adjust detector controls in near-real time to reduce or eliminate need for offline calibration
- Stabilize the GlueX CDC gain to within 5% over a 2 week period with no measurable degradation of the timing resolution.
- Reduce time to process data and therefore to publication by 3-6 months

Lab20_2261 grant awarded for $0.81M

David Lawrence - JLab PI
Thomas Britton - JLab Co-PI
Naomi Jarvis - CMU Co-PI
New hires: 1 Post-doc and 1 Computer Scientist



Available Hardware at JLab

3x 2019 nodes, each with 4 Nvidia Titan RTX 2080 GPUs

- 5k cores
- 24GB

These are available in the SciComp farm, but you must use slurm to login or submit jobs.

get interactive session with 2 GPUS

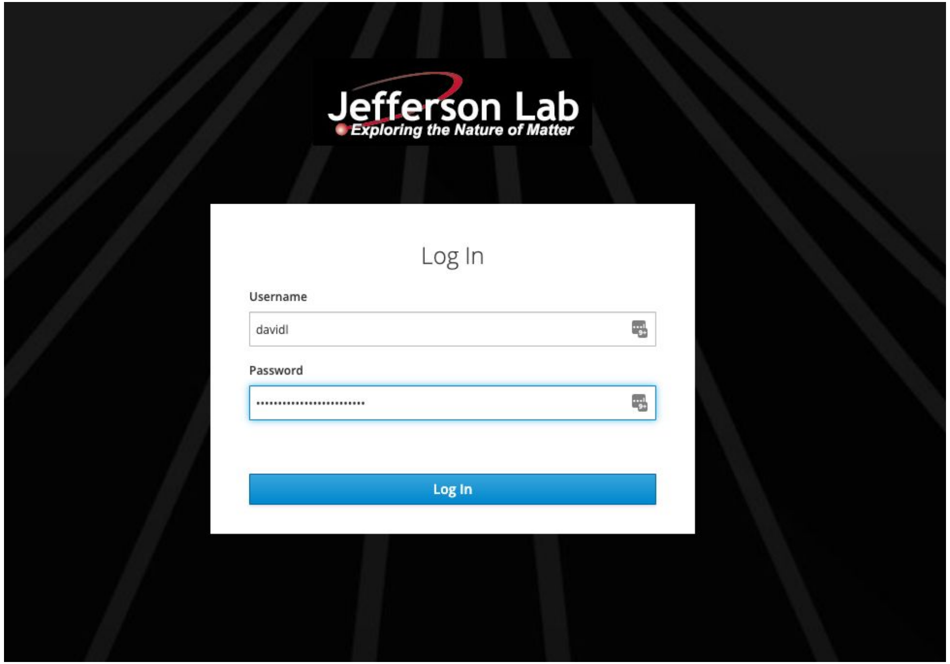
```
salloc --gres gpu:TitanRTX:2 --partition gpu --nodes 1 --time=12:00:00 --mem=24GB
srun --pty bash
```

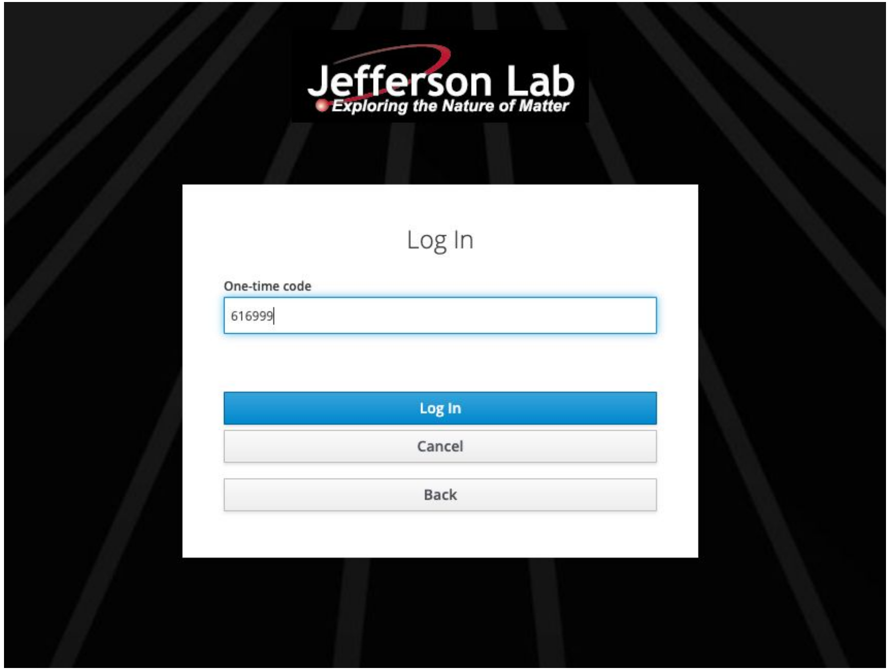
- Some additional funding has been allotted for more GPUs this year from ENP
- Details being worked out an exactly what to buy

Documentation:
https://scicomp.jlab.org/docs/Access_GPUs
https://scicomp.jlab.org/docs/farm_slurm_gpu_jobs

EPSCI
Experimental Physics Software and Computing Infrastructure

JupyterHub https://jupyterhub.jlab.org/

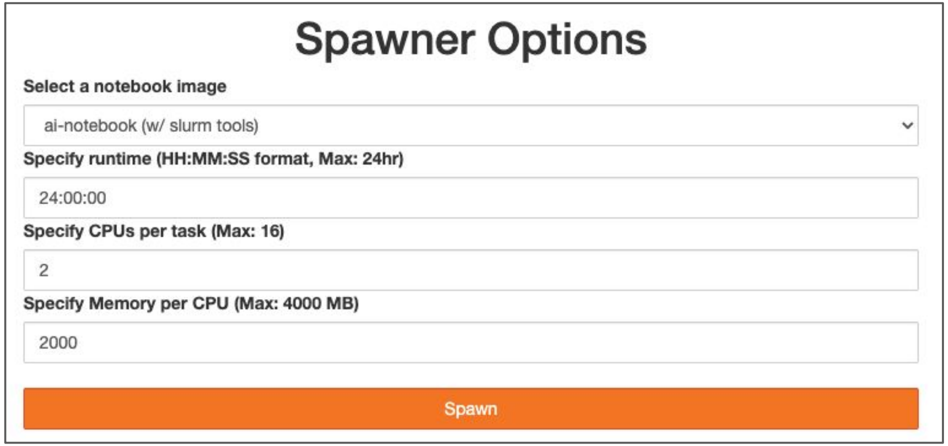




Jefferson Lab Key-Cloak 2-Factor
Use Google Authenticator app on your phone (*n.b. this is different from the MobilePass app!)*

https://jupyterhub.jlab.org/

JupyterHub


Spawner Options

Select a notebook image

ai-notebook (w/ slurm tools)

Specify runtime (HH:MM:SS format, Max: 24hr)

24:00:00

Specify CPUs per task (Max: 16)

2

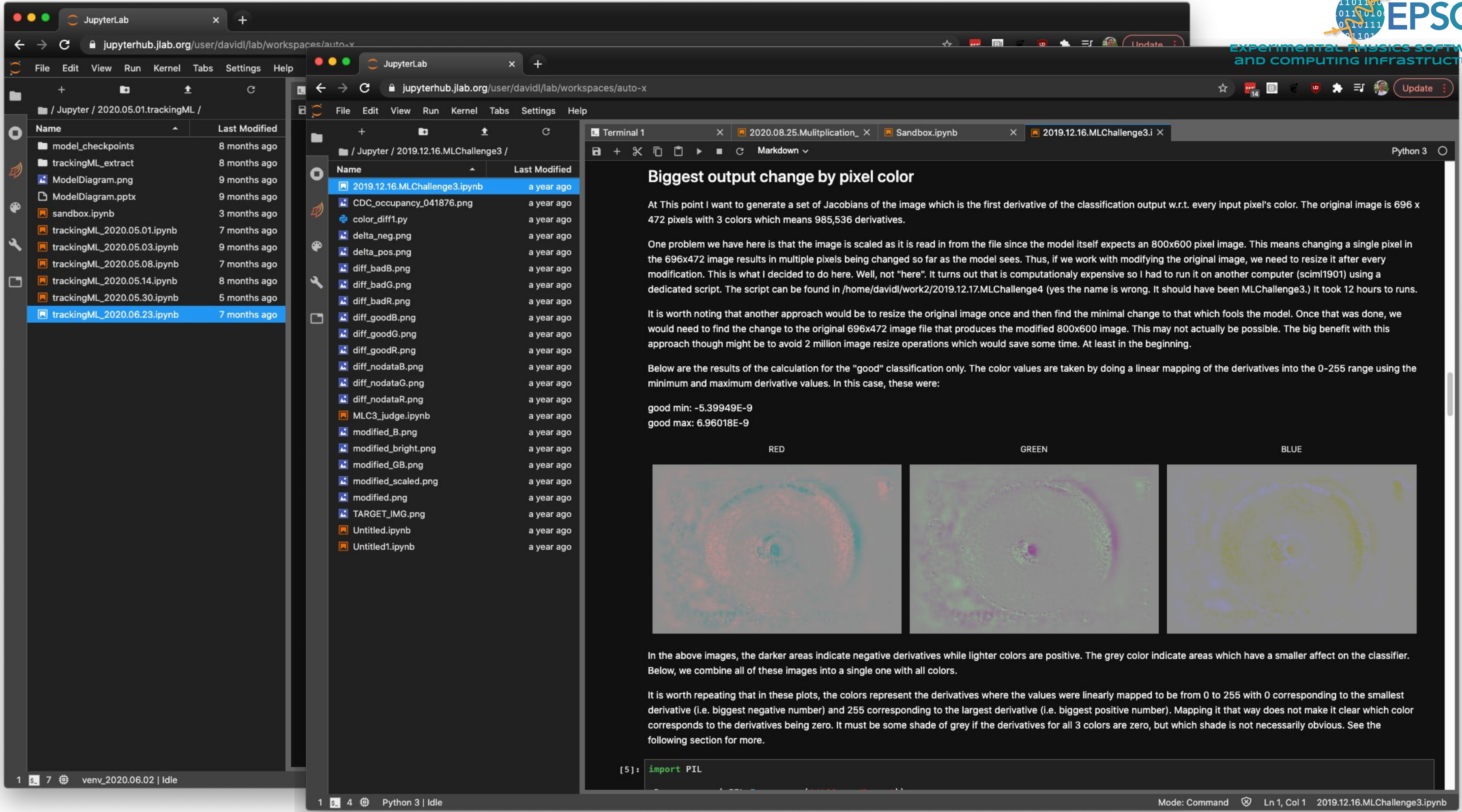
Specify Memory per CPU (Max: 4000 MB)

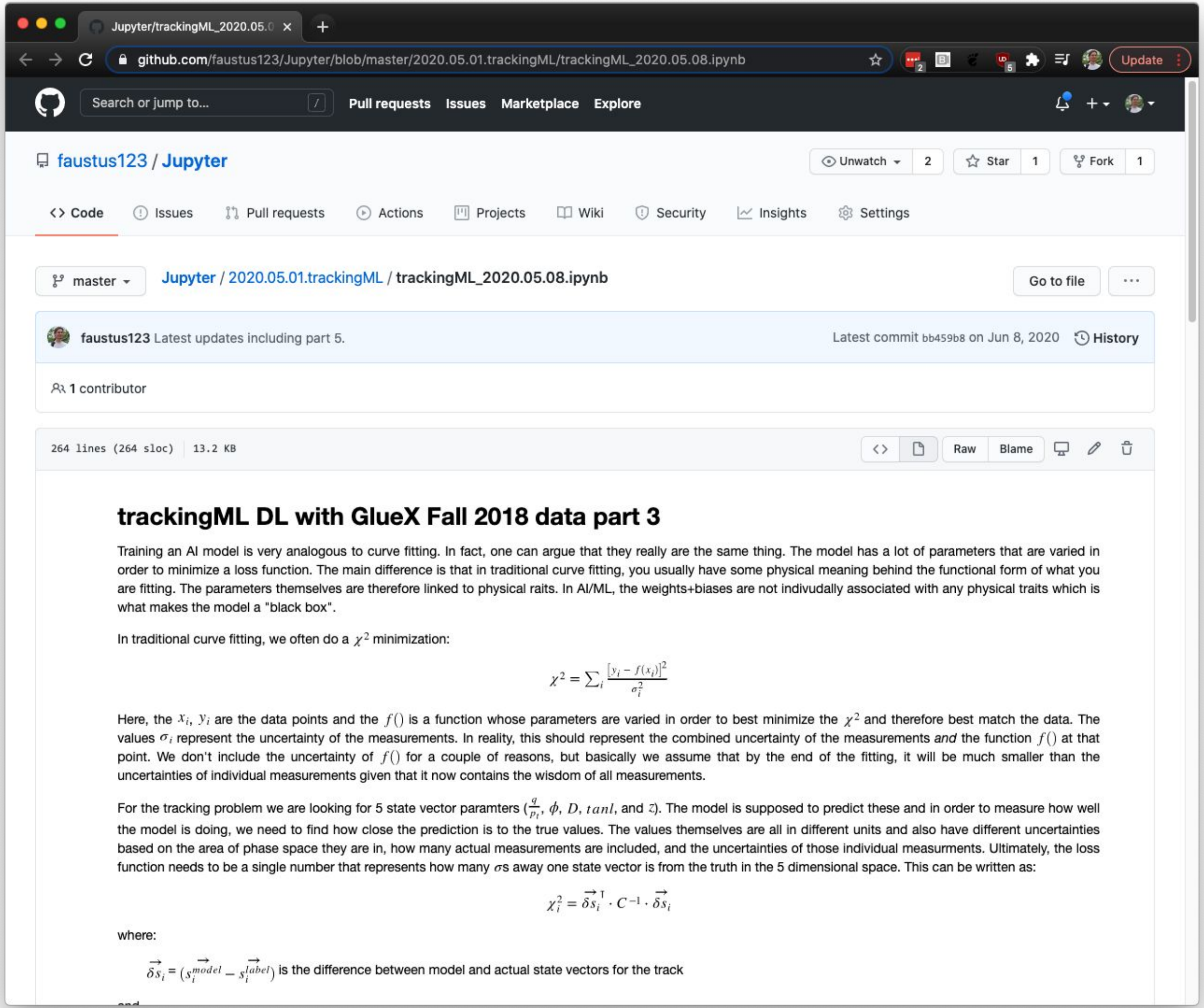
2000

Spawn

- Only Python supported at the moment
- Can run with your own virtual environment and *pip install* any packages you want

- Notebooks are run on a scicomp node that is allocated when it is "spawned"
- Edits are automatically saved so work is not lost when farm job expires
- Notebooks are saved as .ipynb files in your home directory and can be copied or backed up for safe keeping
- Currently can't allocate on node with GPU, but that is being worked on
- Really good for interactive or developing scripts.
- Better off exporting to regular python script if you want to run batch type job or with GPU


JupyterLab
jupyterhub.jlab.org/user/davidl/lab/workspaces/auto-x
File Edit View Run Kernel Tabs Settings Help
/ Jupyter / 2020.05.01.trackingML /
Name
Last Modified
model_checkpoints 8 months ago
trackingML_extract 8 months ago
ModelDiagram.png 9 months ago
ModelDiagram.pptx 9 months ago
sandbox.ipynb 3 months ago
trackingML_2020.05.01.ipynb 7 months ago
trackingML_2020.05.03.ipynb 9 months ago
trackingML_2020.05.08.ipynb 7 months ago
trackingML_2020.05.14.ipynb 8 months ago
trackingML_2020.05.30.ipynb 5 months ago
trackingML_2020.06.23.ipynb 7 months ago
venv_2020.06.02 | Idle
JupyterLab
jupyterhub.jlab.org/user/davidl/lab/workspaces/auto-x
Update
File Edit View Run Kernel Tabs Settings Help
Terminal 1
2020.08.25.Mulitplication_
Sandbox.ipynb
2019.12.16.MLChallenge3.i
Markdown
Python 3
/ Jupyter / 2019.12.16.MLChallenge3 /
Name
Last Modified
2019.12.16.MLChallenge3.ipynb a year ago
CDC_occupancy_041876.png a year ago
color_diff1.py a year ago
delta_neg.png a year ago
delta_pos.png a year ago
diff_badB.png a year ago
diff_badG.png a year ago
diff_badR.png a year ago
diff_goodB.png a year ago
diff_goodG.png a year ago
diff_goodR.png a year ago
diff_nodataB.png a year ago
diff_nodataG.png a year ago
diff_nodataR.png a year ago
MLC3_judge.ipynb a year ago
modified_B.png a year ago
modified_bright.png a year ago
modified_GB.png a year ago
modified_scaled.png a year ago
modified.png a year ago
TARGET_IMG.png a year ago
Untitled.ipynb a year ago
Untitled1.ipynb a year ago
Biggest output change by pixel color
At This point I want to generate a set of Jacobians of the image which is the first derivative of the classification output w.r.t. every input pixel's color. The original image is 696 x 472 pixels with 3 colors which means 985,536 derivatives.
One problem we have here is that the image is scaled as it is read in from the file since the model itself expects an 800x600 pixel image. This means changing a single pixel in the 696x472 image results in multiple pixels being changed so far as the model sees. Thus, if we work with modifying the original image, we need to resize it after every modification. This is what I decided to do here. Well, not "here". It turns out that is computationaly expensive so I had to run it on another computer (sciml1901) using a dedicated script. The script can be found in /home/davidl/work2/2019.12.17.MLChallenge4 (yes the name is wrong. It should have been MLChallenge3.) It took 12 hours to runs.
It is worth noting that another approach would be to resize the original image once and then find the minimal change to that which fools the model. Once that was done, we would need to find the change to the original 696x472 image file that produces the modified 800x600 image. This may not actually be possible. The big benefit with this approach though might be to avoid 2 million image resize operations which would save some time. At least in the beginning.
Below are the results of the calculation for the "good" classification only. The color values are taken by doing a linear mapping of the derivatives into the 0-255 range using the minimum and maximum derivative values. In this case, these were:
good min: -5.39949E-9
good max: 6.96018E-9
RED
GREEN
BLUE
In the above images, the darker areas indicate negative derivatives while lighter colors are positive. The grey color indicate areas which have a smaller affect on the classifier. Below, we combine all of these images into a single one with all colors.
It is worth repeating that in these plots, the colors represent the derivatives where the values were linearly mapped to be from 0 to 255 with 0 corresponding to the smallest derivative (i.e. biggest negative number) and 255 corresponding to the largest derivative (i.e. biggest positive number). Mapping it that way does not make it clear which color corresponds to the derivatives being zero. It must be some shade of grey if the derivatives for all 3 colors are zero, but which shade is not necessarily obvious. See the following section for more.
[5]: import PIL
Python 3 | Idle
Mode: Command
Ln 1, Col 1
2019.12.16.MLChallenge3.ipynb



trackingML DL with GlueX Fall 2018 data part 3

Training an AI model is very analogous to curve fitting. In fact, one can argue that they really are the same thing. The model has a lot of parameters that are varied in order to minimize a loss function. The main difference is that in traditional curve fitting, you usually have some physical meaning behind the functional form of what you are fitting. The parameters themselves are therefore linked to physical raits. In AI/ML, the weights+biases are not indivudally associated with any physical traits which is what makes the model a "black box".

In traditional curve fitting, we often do a χ^2 minimization:

$$\chi^2 = \sum_i \frac{[y_i - f(x_i)]^2}{\sigma_i^2}$$

Here, the x_i, y_i are the data points and the $f()$ is a function whose parameters are varied in order to best minimize the χ^2 and therefore best match the data. The values σ_i represent the uncertainty of the measurements. In reality, this should represent the combined uncertainty of the measurements *and* the function $f()$ at that point. We don't include the uncertainty of $f()$ for a couple of reasons, but basically we assume that by the end of the fitting, it will be much smaller than the uncertainties of individual measurements given that it now contains the wisdom of all measurements.

For the tracking problem we are looking for 5 state vector paramters ($\frac{q}{p_t}$, ϕ, D, $tanl$, and z). The model is supposed to predict these and in order to measure how well the model is doing, we need to find how close the prediction is to the true values. The values themselves are all in different units and also have different uncertainties based on the area of phase space they are in, how many actual measurements are included, and the uncertainties of those individual measurments. Ultimately, the loss function needs to be a single number that represents how many σs away one state vector is from the truth in the 5 dimensional space. This can be written as:

$$\chi_i^2 = \vec{\delta s_i}^{\,T} \cdot C^{-1} \cdot \vec{\delta s_i}$$

where:

$\vec{\delta s_i} = (\vec{s_i}^{model} - \vec{s_i}^{label})$ is the difference between model and actual state vectors for the track

Github will render notebooks displaying the results from when you ran them.

EPSCI
Experimental Physics Software and Computing Infrastructure

Summary

- AI Activity and resources are growing at JLab
  - GPUs
  - Jupyterhub
- A few big projects and several smaller projects related to experimental program
  - Detector Design
  - Reconstruction
  - Triggering
  - Monitoring
  - Controls
  - …
- If help needed, reach out to EPSCI group

  kishan@jlab.org
  davidl@jlab.org