A Modular Software Stack for A(i)DAPT

Trevor Reed

Florida International University

Steven Goldenberg, Daniel Lersch
Jefferson Lab Data Science Group
A(i)DAPT Group
CLAS Collaboration

A(i)DAPT: AI to Improve CLAS Simulations

- A(i)DAPT – AI for Data Analysis and PreservaTion
- Broad Goal: Develop a machine learning event generator (simulations)
  - Much faster than traditional simulations
  - Could potentially extend measurements to regions outside of acceptance

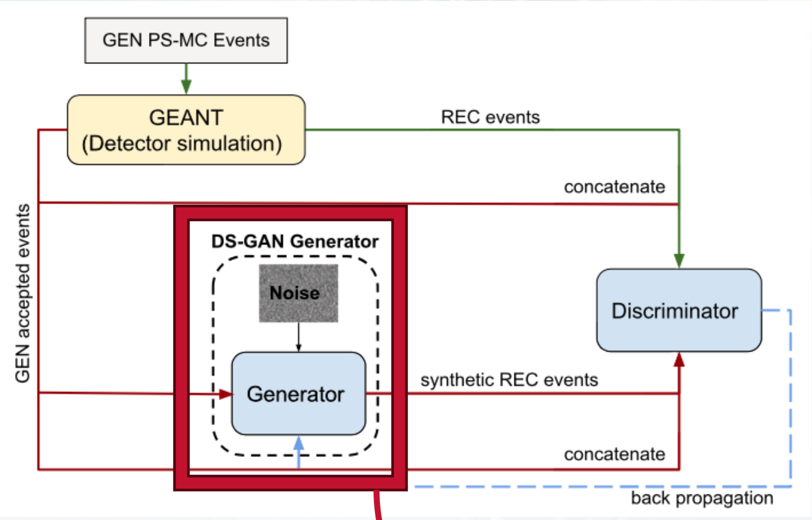
Data Science Group Contributions to A(i)DAPT

- Convert original Jupyter Notebook implementation to modular software stack
  - Improved readability, maintainability, and scalability
  - Stored on GitHub
  - Reduced code redundancies
    - Inner GAN and outer GAN utilize much of the same underlying code
  - Incorporate the Python framework Hydra for configuration management
  - Utilizes registration system
    - Enhance collaboration by allowing for easy swapping of modules
  - Includes unit-tests for testing of individual modules/functions
- Reproduce already achieved results in this new framework
- Optimize/improve training

Using GAN's (Generative Adversarial Networks) to Model Detector Response

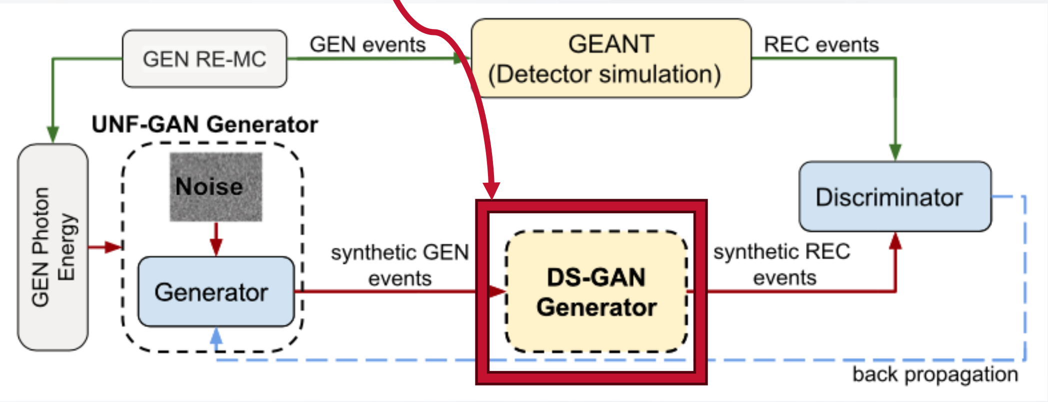
- GAN: Two opposing neural networks
  - Generator: Takes in GEN data and produces "synthetic" REC events
  - Discriminator: Takes the synthetic REC events and "real" REC events and attempts to distinguish them

Inner (Folding) GAN



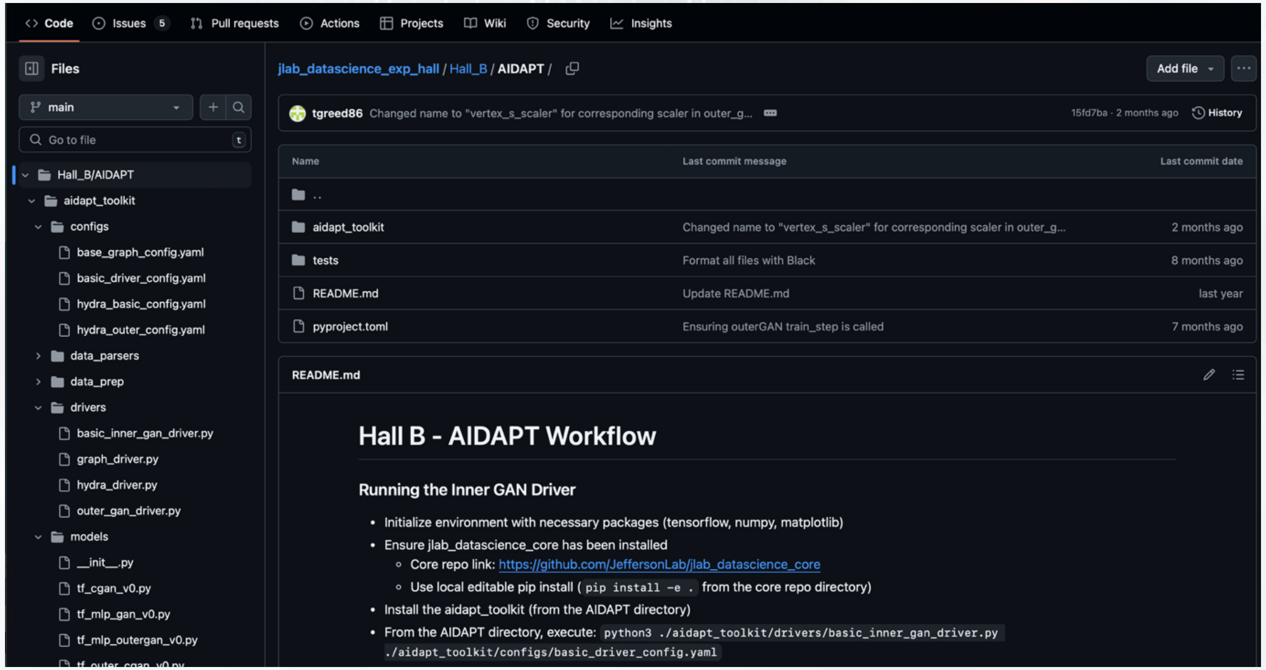
T. Alghamdi et al. Toward a generative modeling analysis of CLAS exclusive 2π photoproduction. Phys. Rev. D, 108:094030, 2023.

Outer (Unfolding) GAN



Running the Software: Getting Started

- Two GitHub repositories you'll need to copy and install:
  - https://github.com/JeffersonLab/jlab_datascience_core.git
  - https://github.com/JeffersonLab/jlab_datascience_exp_hall.git
- There is a tutorial document with detailed instructions
  - Will be made available very soon



Configurations

- Several yaml files in *aidapt_toolkit/configs/*
- For inner GAN, use:
  - *hydra_basic_config.yaml*
- For outer GAN, use:
  - *hydra_outer_config.yaml*

- Yaml config files determine inputs, network architectures, epochs, etc.
- Configuration options can also be specified in the command line
  - Configurations files are saved in output directory


```yaml
d_scaler:
  id: numpy_minmax_scaler
  feature_range: &id001
  - -1
  - 1
detector_parser:
  id: aidapt_numpy_reader_v0
  filepaths:
  - ./aidapt_toolkit/data/ps_detector/ps_detector_0.npy
  - ./aidapt_toolkit/data/ps_detector/ps_detector_1.npy
  - ./aidapt_toolkit/data/ps_detector/ps_detector_2.npy
  - ./aidapt_toolkit/data/ps_detector/ps_detector_3.npy
lab2inv:
  id: lab_variables_to_invariants
  MP: 0.93827
model:
  id: tf_cgan_v0
  gan_type: inner
  batch_size: 10000
  discriminator_layers:
  - - Dense
    - units: 256
  - - LeakyReLU
    - negative_slope: 0.2
  - - Dense
    - units: 128
  - - LeakyReLU
    - negative_slope: 0.2
  - - Dense
    - units: 64
  - - LeakyReLU
    - negative_slope: 0.2
  discriminator_optimizer:
  - Adam
  - beta_1: 0.5
    learning_rate: 1.0e-05
  discriminator_loss: 'BinaryCrossentropy'
  epochs: 2
```

```yaml
  generator_layers:
  - - Dense
    - units: 128
  - - LeakyReLU
    - negative_slope: 0.2
  - - BatchNormalization
    - momentum: 0.8
  - - Dense
    - units: 256
  - - LeakyReLU
    - negative_slope: 0.2
  - - BatchNormalization
    - momentum: 0.8
  - - Dense
    - units: 512
  - - LeakyReLU
    - negative_slope: 0.2
  - - BatchNormalization
    - momentum: 0.8
  generator_optimizer:
  - Adam
  - beta_1: 0.5
    learning_rate: 1.0e-05
  generator_loss: 'BinaryCrossentropy'
  image_shape: 4
  label_shape: 4
  latent_dim: 100
v_scaler:
  id: numpy_minmax_scaler
  feature_range: *id001
vertex_parser:
  id: aidapt_numpy_reader_v0
  filepaths:
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_0.npy
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_1.npy
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_2.npy
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_3.npy
driver:
  save_path: ${hydra:runtime.output_dir}
metrics:
  layer_specific_gradients: True
  grad_frequency: 1
  chi2: True
  chi2_frequency: 1
  disc_accuracy: True
  acc_frequency: 1
```

Configurations: Data Input

- Input data paths should be specified according to your inputs
  - The files at these locations were copied from */work/data_science/quantom/aidapt_at_quantom/data/*
- $\pi^+\pi^-p$ photoproduction (g11 simulation configurations)
- See *aidapt_toolkit/data_prep/lab_variables_to_invariants.py* for data file structure


```yaml
d_scaler:
  id: numpy_minmax_scaler
  feature_range: &id001
  - -1
  - 1
detector_parser:
  id: aidapt_numpy_reader_v0
  filepaths:
  - ./aidapt_toolkit/data/ps_detector/ps_detector_0.npy
  - ./aidapt_toolkit/data/ps_detector/ps_detector_1.npy
  - ./aidapt_toolkit/data/ps_detector/ps_detector_2.npy
  - ./aidapt_toolkit/data/ps_detector/ps_detector_3.npy
lab2inv:
  id: lab_variables_to_invariants
  MP: 0.93827
model:
  id: tf_cgan_v0
  gan_type: inner
  batch_size: 10000
  discriminator_layers:
  - - Dense
    - units: 256
  - - LeakyReLU
    - negative_slope: 0.2
  - - Dense
    - units: 128
  - - LeakyReLU
    - negative_slope: 0.2
  - - Dense
    - units: 64
  - - LeakyReLU
    - negative_slope: 0.2
  discriminator_optimizer:
  - Adam
  - beta_1: 0.5
    learning_rate: 1.0e-05
  discriminator_loss: 'BinaryCrossentropy'
  epochs: 2
```

```yaml
  generator_layers:
  - - Dense
    - units: 128
  - - LeakyReLU
    - negative_slope: 0.2
  - - BatchNormalization
    - momentum: 0.8
  - - Dense
    - units: 256
  - - LeakyReLU
    - negative_slope: 0.2
  - - BatchNormalization
    - momentum: 0.8
  - - Dense
    - units: 512
  - - LeakyReLU
    - negative_slope: 0.2
  - - BatchNormalization
    - momentum: 0.8
  generator_optimizer:
  - Adam
  - beta_1: 0.5
    learning_rate: 1.0e-05
  generator_loss: 'BinaryCrossentropy'
  image_shape: 4
  label_shape: 4
  latent_dim: 100
v_scaler:
  id: numpy_minmax_scaler
  feature_range: *id001
vertex_parser:
  id: aidapt_numpy_reader_v0
  filepaths:
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_0.npy
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_1.npy
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_2.npy
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_3.npy
driver:
  save_path: ${hydra:runtime.output_dir}
metrics:
  layer_specific_gradients: True
  grad_frequency: 1
  chi2: True
  chi2_frequency: 1
  disc_accuracy: True
  acc_frequency: 1
```

Utilizing Prebuilt Container: Running Interactively

- **NOTE: A container is not required to run software**
  - Provided as a convenience
- Already-built container for both interactive and batch running
  - /work/clas12/reedtg/data_science/aidapt_10-14-24-update/jlab_datascience_exp_hall/Hall_B/AIDAPT/TFContainers/build_1/tensorflow-2.16.1-gpu.sif
- Run container interactively

```
> singularity run --bind/path/to/jlab_datascience_exp_hall:/jlab_datascience_exp_hall
/path/to/container_image/tensorflow-2.16.1-gpu.sif
Apptainer> cd /jlab_datascience_exp_hall/Hall_B/AIDAPT
Apptainer> python3 ./aidapt_toolkit/drivers/hydra_driver.py
```

  - hydra_driver.py is the primary executing file for the inner GAN

Utilizing Prebuilt Container: Running on Batch Farm

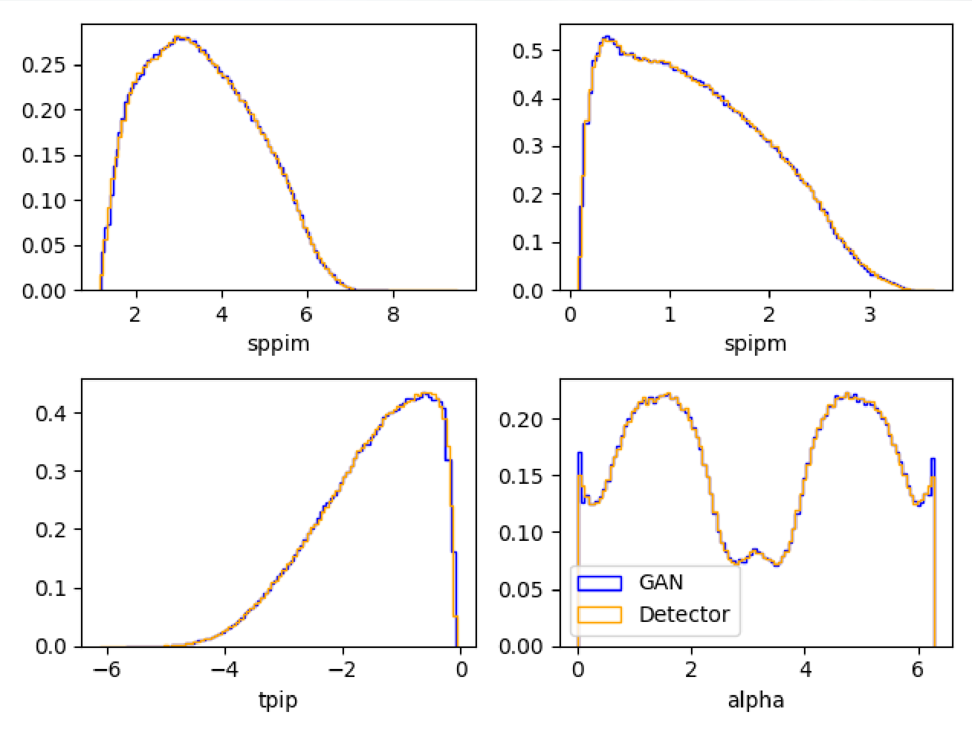
- Command:

```
singularity exec --nv \
    --bind /absolute/path/to/jlab_datascience_exp_hall:/jlab_datascience_exp_hall \
    /absolute/path/to/container_image/tensorflow-2.16.1-gpu.sif \
    sh -c "cd /jlab_datascience_exp_hall/Hall_B/AIDAPT && \
    python3 aidapt_toolkit/drivers/hydra_driver.py”
```

- Provide command in SLURM (or SWIF2) submission script
  - Can request to run on GPU
  - Example SLURM submission script at: */work/clas12/reedtg/data_science/aidapt_10-14-24-update/jlab_datascience_exp_hall/Hall_B/AIDAPT/container_slurm_sub_script*

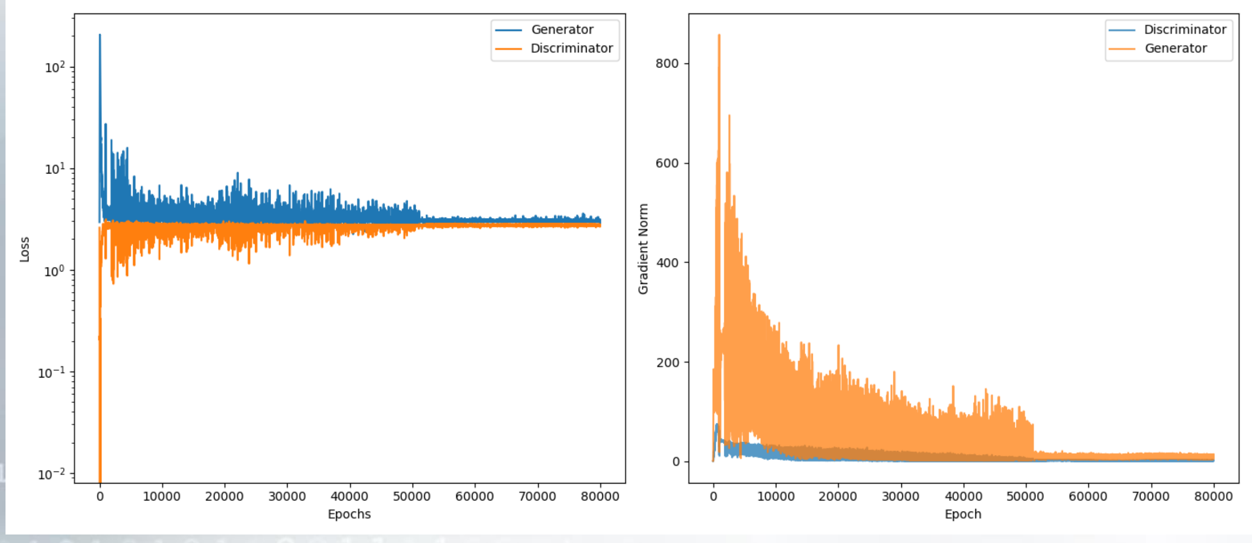
Inner GAN Training Results

- Model is trained on 4 kinematic variables (shown on right)
- Input for inner GAN is phasespace simulation of these 4 variables
- Plots show results after 80,000 epochs
- These plots are saved automatically in output directory
  - distributions.png (right)
  - training_analysis.png (below)



Gradient Norm

Loss



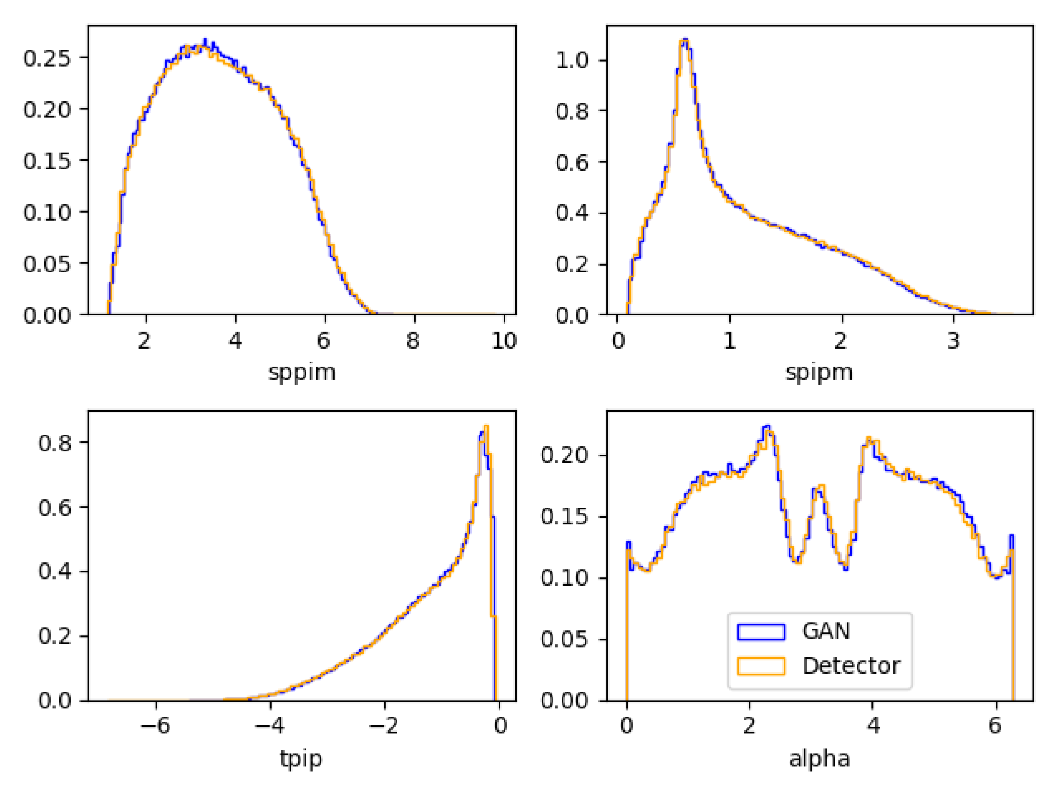
Training Outer GAN

- Once inner GAN is trained, the same steps can be used to train outer GAN
- Outer GAN reads from a different yaml file, hydra_outer_config.yaml
- Model architecture is a bit different
- Additional configurations:
  - *folding_id*
  - *folding_path*
- These point to the already trained and saved inner GAN model
- Appropriate driver to run the outer GAN training is outer_gan_driver.py


```yaml
d_scaler:
  id: numpy_minmax_scaler
  feature_range: &id001
  - -1
  - 1
detector_parser:
  id: aidapt_numpy_reader_v0
  filepaths:
  - ./aidapt_toolkit/data/Realistic_detector/realistic_detector_0.npy
  - ./aidapt_toolkit/data/Realistic_detector/realistic_detector_1.npy
  - ./aidapt_toolkit/data/Realistic_detector/realistic_detector_2.npy
  - ./aidapt_toolkit/data/Realistic_detector/realistic_detector_3.npy
lab2inv:
  id: lab_variables_to_invariants
  MP: 0.93827
model:
  id: tf_outer_cgan_v0
  folding_id: tf_cgan_v0
  folding_path: ./outputs/trained_inner_gan_models/2024-12-05/cgan_model
  gan_type: outer
  batch_size: 10000
  discriminator_layers:
  - - Dense
    - units: 1024
  - - LeakyReLU
    - negative_slope: 0.2
  - - Dense
    - units: 512
  - - LeakyReLU
    - negative_slope: 0.2
  - - Dense
    - units: 256
  - - LeakyReLU
    - negative_slope: 0.2
  - - Dense
    - units: 128
  - - LeakyReLU
    - negative_slope: 0.2
  - - Dense
    - units: 64
  - - LeakyReLU
    - negative_slope: 0.2
```

Outer GAN Training Results

- "Realistic" simulation (i.e. not phasespace)
  - Considers the 3 dominant intermediate resonances
    - $p\rho^0$
    - $\Delta^{++}\pi^-$
    - $\Delta^0\pi^+$
- Input to outer GAN generator is only 1 variable
  - Mandelstam s ($\propto E_\gamma$)
- Plots show results after 120,000 epochs
  - Outer GAN typically takes a bit longer to converge



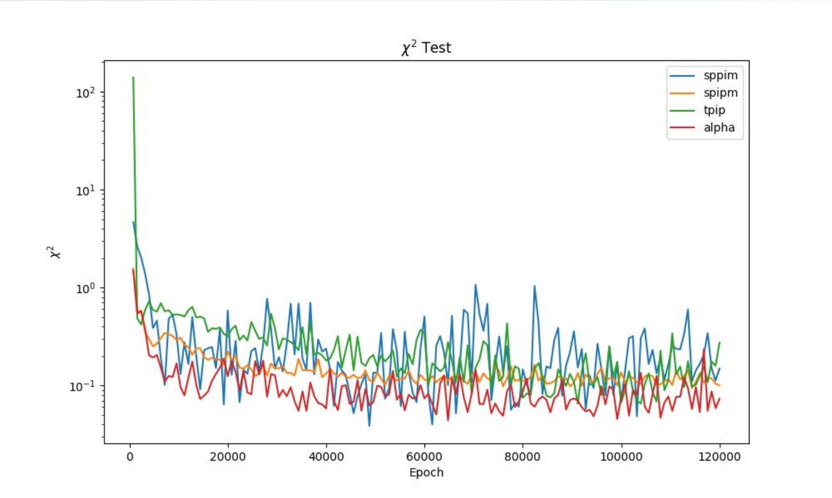
Additional Training Feedback

- At the end of the configuration (yaml) files is a section called “metrics”
- Here, the user can turn on/off additional training metrics
  - Layer-specific gradient norms
  - χ^2 test
  - Discriminator accuracy test
- The “*<metric_name>_frequency*” line determines how frequently (in terms of epoch) the metrics will be calculated and saved

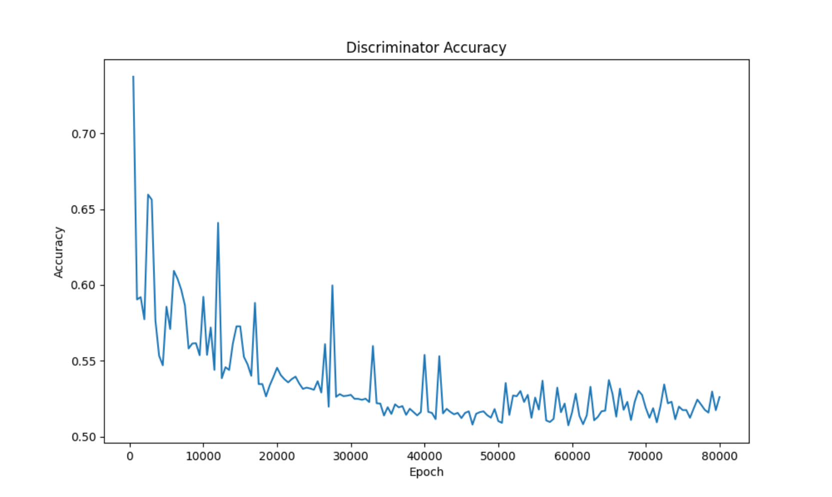
```
  generator_layers:
  - - Dense
    - units: 128
  - - LeakyReLU
    - negative_slope: 0.2
  - - BatchNormalization
    - momentum: 0.8
  - - Dense
    - units: 256
  - - LeakyReLU
    - negative_slope: 0.2
  - - BatchNormalization
    - momentum: 0.8
  - - Dense
    - units: 512
  - - LeakyReLU
    - negative_slope: 0.2
  - - BatchNormalization
    - momentum: 0.8
  generator_optimizer:
  - Adam
  - beta_1: 0.5
    learning_rate: 1.0e-05
  generator_loss: 'BinaryCrossentropy'
  image_shape: 4
  label_shape: 4
  latent_dim: 100
v_scaler:
  id: numpy_minmax_scaler
  feature_range: *id001
vertex_parser:
  id: aidapt_numpy_reader_v0
  filepaths:
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_0.npy
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_1.npy
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_2.npy
  - ./aidapt_toolkit/data/ps_vertex/ps_vertex_3.npy
driver:
  save_path: ${hydra:runtime.output_dir}
metrics:
  layer_specific_gradients: True
  grad_frequency: 1
  chi2: True
  chi2_frequency: 1
  disc_accuracy: True
  acc_frequency: 1
```

Training Metrics Plots

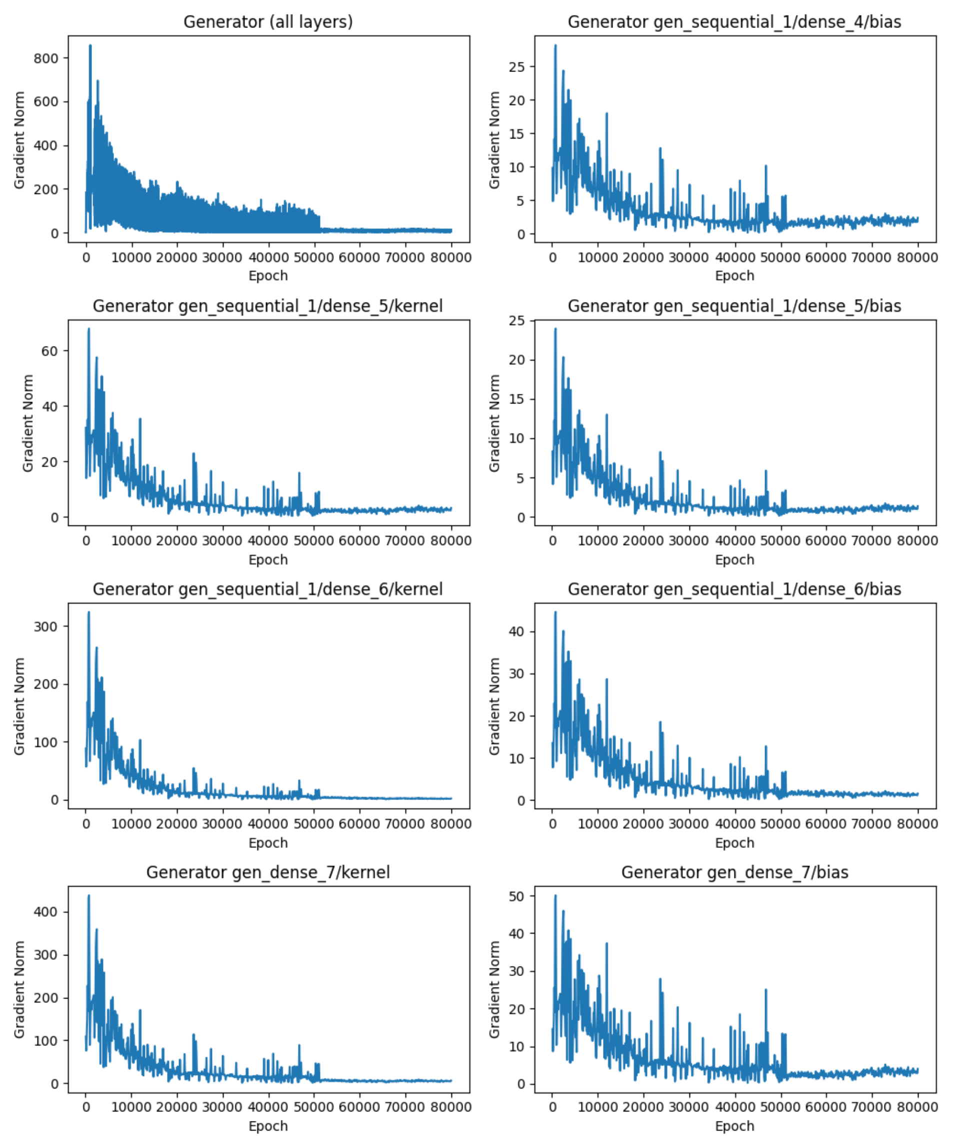
χ^2



Accuracy



Layer-specific Gradient Norm



Final Remarks

- This reworked software stack should improve usability, reproducibility, debugging, readability, etc.
- The model, metrics plots, and configuration files are saved to the output folder after training.
- Some hyperparemeters (epochs, network architecture, learning rate, etc.) will likely need to be adjusted for various training datasets
  - This newer framework makes it relatively simple to change these values from the configuration files
  - Should be no need to go into the bulk of the code
- We've produced a document to serve as an overview and guide for using this software produced by the JLab Data Science Group for A(i)DAPT
  - The tutorial document should be followed, especially when initially setting up the software
- This work was done to make the jobs of those presently in A(i)DAPT easier
  - We also encourage others who are interested to try it out

Thank You

QUESTIONS?

Backup Slides

Utilizing Prebuilt Container: Running on Batch Farm

- Command:
  - singularity exec --nv \

    --bind /absolute/path/to/jlab_datascience_exp_hall:/jlab_datascience_exp_hall \

    /absolute/path/to/container_image/tensorflow-2.16.1-gpu.sif \

    sh -c "cd /jlab_datascience_exp_hall/Hall_B/AIDAPT && \

    python3 aidapt_toolkit/drivers/hydra_driver.py”
- Provide command in SLURM (or SWIF2) submission script
  - Can request to run on GPU
  - Example SLURM submission script at: */work/clas12/reedtg/data_science/aidapt_10-14-24-update/jlab_datascience_exp_hall/Hall_B/AIDAPT/container_slurm_sub_script*

Utilizing Prebuilt Container: Running Interactively

- **NOTE: A container is not required to run software**
  - Provided as a convenience
- Already-built container for both interactive and batch running
  - /work/clas12/reedtg/data_science/aidapt_10-14-24-update/jlab_datascience_exp_hall/Hall_B/AIDAPT/TFContainers/build_1/tensorflow-2.16.1-gpu.sif
- Run container interactively
  - > singularity run --bind/path/to/jlab_datascience_exp_hall:/jlab_datascience_exp_hall /path/to/container_image/tensorflow-2.16.1-gpu.sif
  - Apptainer> cd /jlab_datascience_exp_hall/Hall_B/AIDAPT
  - Apptainer> python3 ./aidapt_toolkit/drivers/hydra_driver.py
    - hydra_driver.py is the primary executing file for the inner GAN