Deep Learning for the Matrix Element Method

Mihir Katare

University of Illinois at Urbana-Champaign

Mentors: Matthew Feickert, Mark Neubauer

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

IRIS-HEP Fellowship Presentation

10/18/21

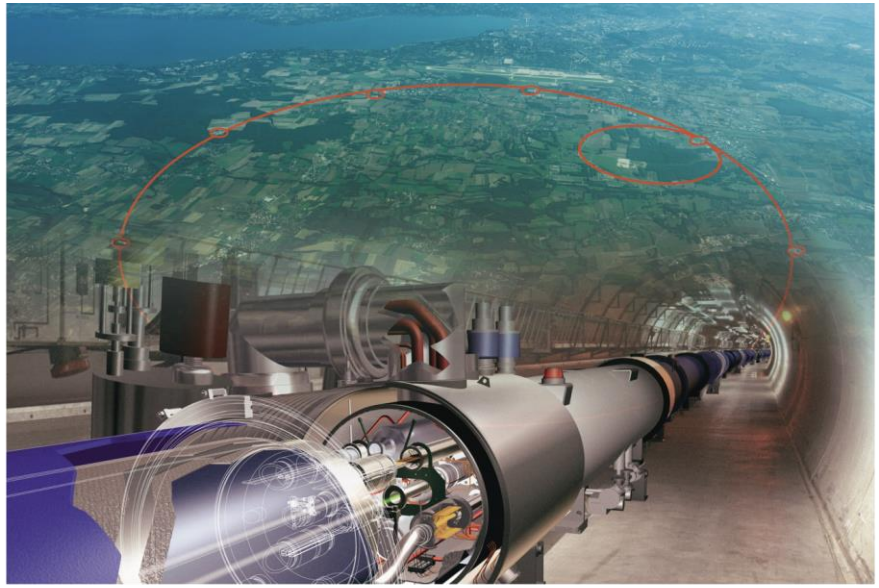
iris hep

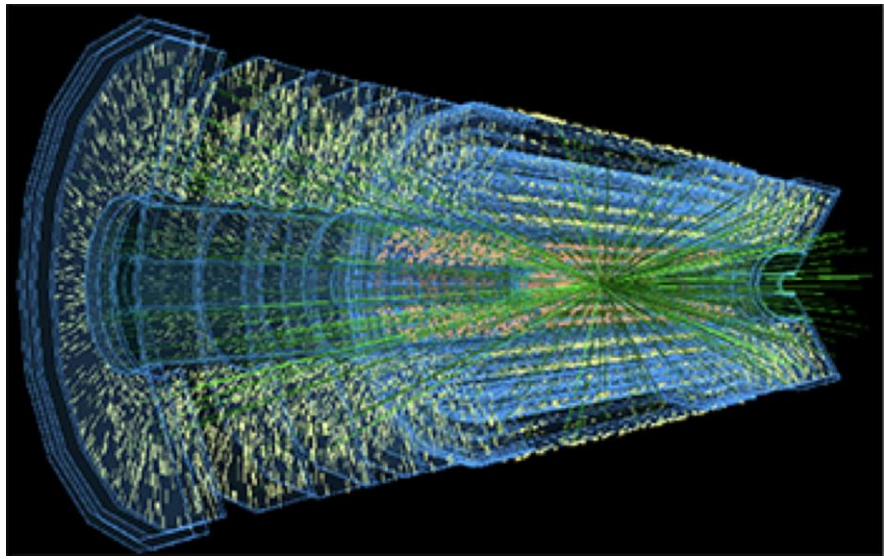
SCAILFIN

Introduction

- The LHC's future is one of a dramatic increase in luminosity rather than energy
  - ➡ Large amount of collision data with complex events expected in future LHC running

- We want to make full use of this data by incorporating and correlating as much of the available information within each event as possible
  - ❖ Methods that employ machine learning are widely used in this context
  - ❖ Alternative: ***Matrix Element Method*** (MEM)





Matrix Element Method (MEM)

Ab initio calculation of an approximate probability density function $\mathscr{P}_\xi(\boldsymbol{x}|\boldsymbol{\alpha})$ for an event with observed final-state particle momenta $\boldsymbol{x}$ to be due to a process $\boldsymbol{\xi}$ with theory parameters $\boldsymbol{\alpha}$

$$\mathscr{P}_\xi(\mathbf{x}\,|\,\boldsymbol{\alpha}) = \frac{1}{\sigma_\xi^{\text{fiducial}}(\boldsymbol{\alpha})} \int d\Phi(\mathbf{y}_{\text{final}})\, dx_1\, dx_2\, \frac{f(x_1)f(x_2)}{2sx_1x_2}\, |\mathscr{M}_\xi(\mathbf{y}\,|\,\boldsymbol{\alpha})|^2\, \delta^4(\mathbf{y}_{\text{initial}} - \mathbf{y}_{\text{final}})\, W(\mathbf{x},\mathbf{y})$$

Dynamics from QFT→ Correlations from physics

$\mathscr{P}_\xi(\boldsymbol{x}|\boldsymbol{\alpha})$ can be used in a number of ways to search for new phenomena at particle colliders

Sample Likelihood

(e.g. $\boldsymbol{\alpha}$ measurements via max. likelihood)

$$\mathscr{L}(\boldsymbol{\alpha}) = \prod_i \sum_k f_k \mathscr{P}_{\xi_k}(\mathbf{x}_i\,|\,\boldsymbol{\alpha})$$

Neyman-Pearson Discriminant

(e.g. process search, hypothesis test)

$$p(\mathbf{x}\,|\,S) = \frac{\sum_i \beta_{S_i} \mathscr{P}_{S_i}(\mathbf{x}\,|\,\boldsymbol{\alpha}_{S_i})}{\sum_i \beta_{S_i} \mathscr{P}(\mathbf{x}\,|\,\boldsymbol{\alpha}_{S_i}) + \sum_j \beta_{B_j} \mathscr{P}(\mathbf{x}\,|\,\boldsymbol{\alpha}_{B_j})}$$

For the purpose of this talk: $\mathscr{P}_\xi(\boldsymbol{x}|\boldsymbol{\alpha})$ is a function that can be computed numerically and provides physics-driven information useful for measurements, hypothesis tests and searches

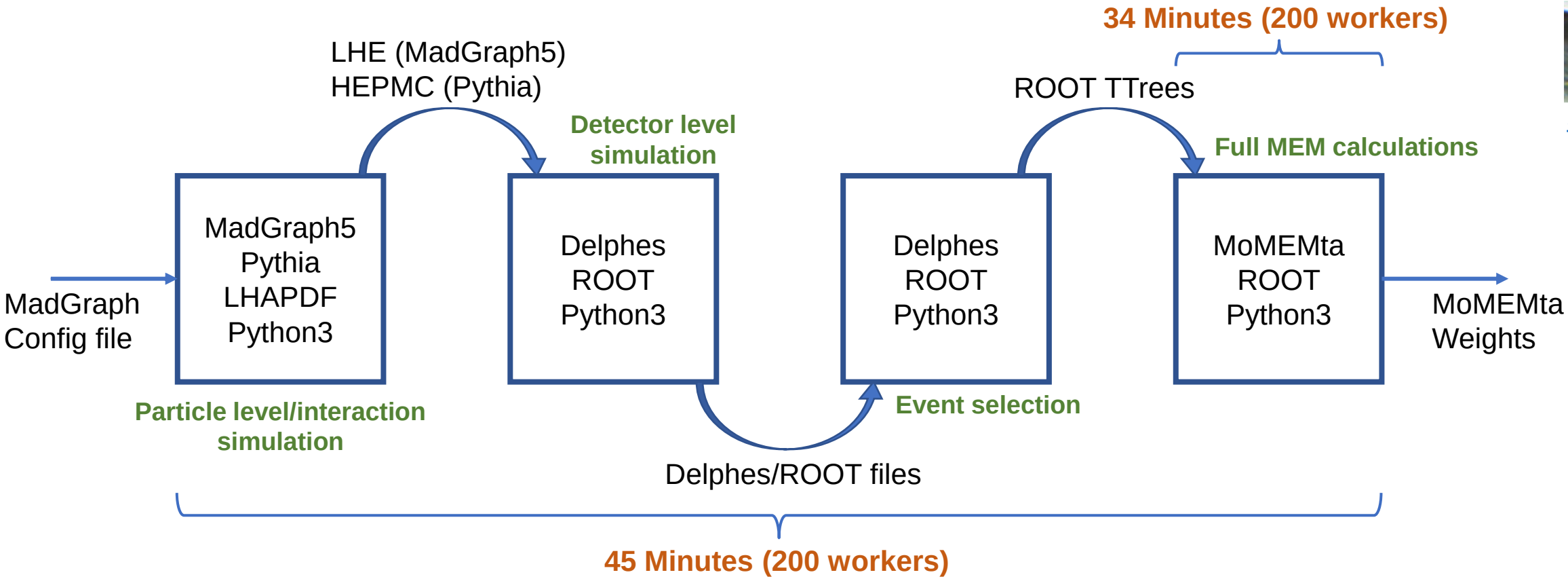
Matrix Element Method in the Machine Learning Era

- MEM has numerous advantages over ML-based methods
  - ❖ Does not require training
  - ❖ Incorporates all available kinematic information, including correlations
  - ❖ Has a clear physical meaning in terms of transition probabilities in QFT
- Main limitation of the MEM is that it is ***computationally intensive***
  - ❖ E.g. Calculating $\mathscr{P}_{\boldsymbol{\xi}}(\boldsymbol{x}|\boldsymbol{\alpha})$ for $pp \to t\bar{t}H \to W^+bW^-\bar{b}b\bar{b} \to \ell\nu + 6$ jets involves high-dimensional integration and can take minutes per event
- It was proposed by Neubauer, *et al.* in [1] (cf. [2], [3], [4]) to use ML methods to approximate MEM calculations so they are *sustainable*

Refs: [1], [2], [3], [4]

Current MEM Calculations Pipeline


SCAILFIN



	Parallel Time	Serial Time
Entire Pipeline	45 Minutes	150 Hours
MoMEMta	34 Minutes	113 Hours

For **300k events of $p + p \rightarrow l + \bar{l} + X$

Dockerized Container

Data Flow

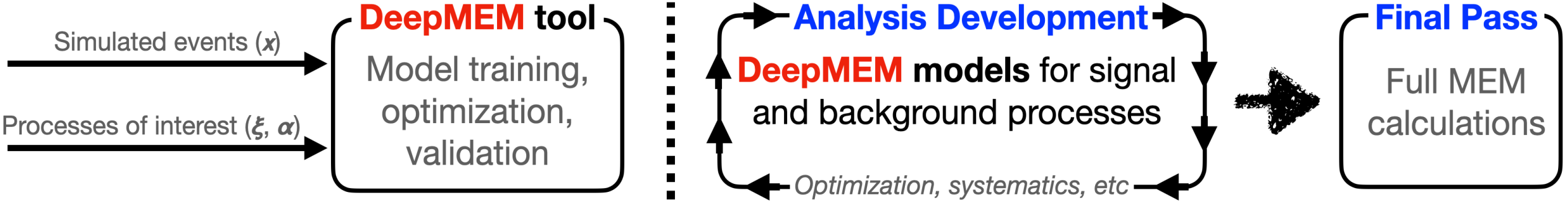
DeepMEM Objective

- Address the challenges of MEM while retaining the benefits:
  - Retain the transparency of MEM calculations
  - Reduce the time required by MEM calculations

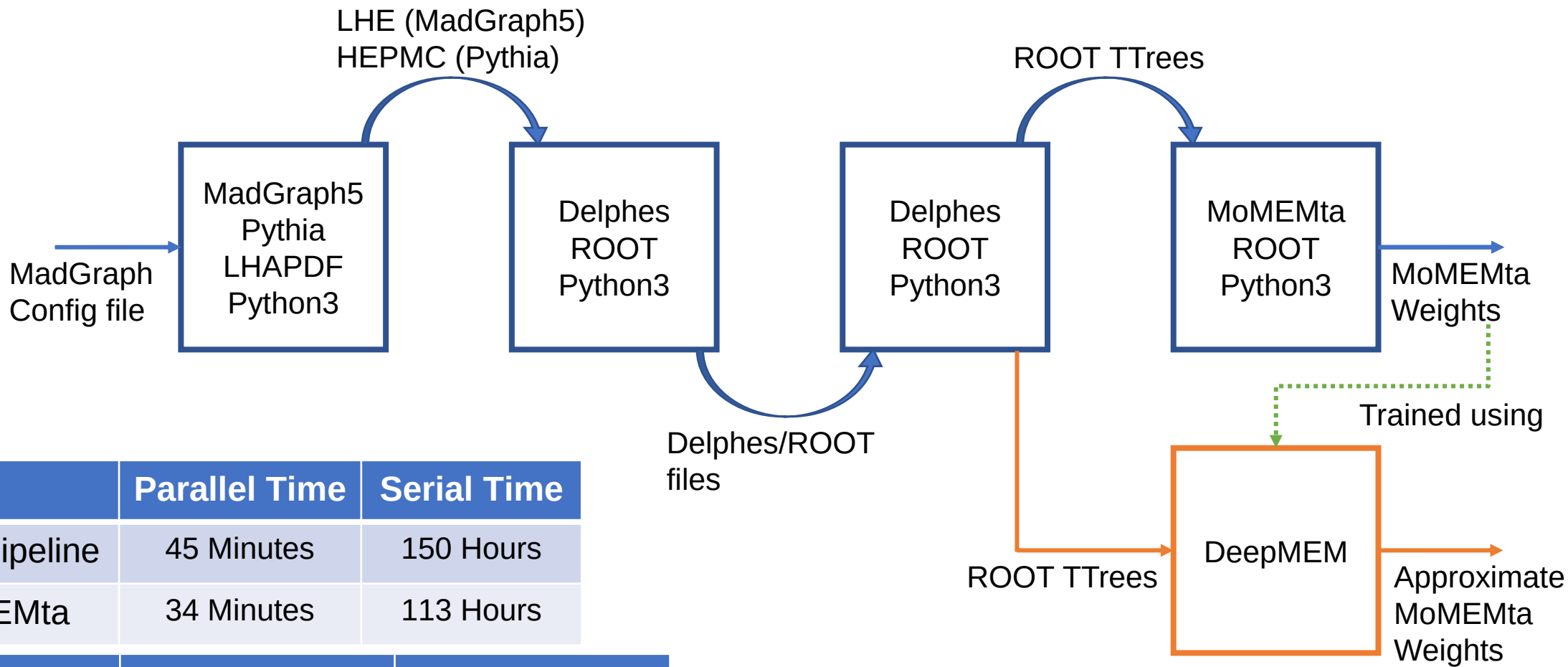
- **Deep Neural Networks:** arbitrary function approximators that scale well with data

- Replace the calculations performed by MoMEMta with a neural network trained using MoMEMta outputs

- Final calculations will be performed using the full pipeline for accuracy – Using DeepMEM expedites calculations during research



MEM Pipeline using Deep Learning Approximations



	Parallel Time	Serial Time
Full Pipeline	45 Minutes	150 Hours
MoMEMta	34 Minutes	113 Hours

	Inference Time	Training Time†
DeepMEM	2 Minutes	18 Mins*

For **300k events of $p + p \rightarrow l + \bar{l} + X$

Data Flow Neural Network Dockerized Container

* Trained for 100 epochs

† Training needs to be done only once for a particular final state

Explanation of Data

We consider the simple Drell-Yan Process with lepton pair final state:

$$\mathrm{p} + \mathrm{p} \rightarrow l + \bar{l} + X$$

- Parsing the ROOT TTrees produced after event selection, we use the four-momentum of the final state particles and missing transverse energy (MET).
- Particle mass is a very good discriminant, and we keep the neural network blind to the mass by excluding it [1]
- Inputs:
  - Pt, Eta, Phi components of both leptons and produced jets
  - Magnitude and Phi of the MET
  - 14 input parameters
- Outputs:
  - Log transformed MoMEMta weight values
- Final Dataset contains ~300K events

[1] F. Bury, C. Delaere, Matrix Element Regression with Deep Neural Networks -- breaking the CPU barrier

Multiprocessing Dataloader

- PyTorch In-built dataloader is built for image/computer vision data – loads individual samples based on user mappings
  - This is inefficient for contiguous, tabular data

- No out-of-the-box solution that can address the issues
  - Tabular data is loaded faster in chunks
  - The dataset might be too large to fit in memory at once

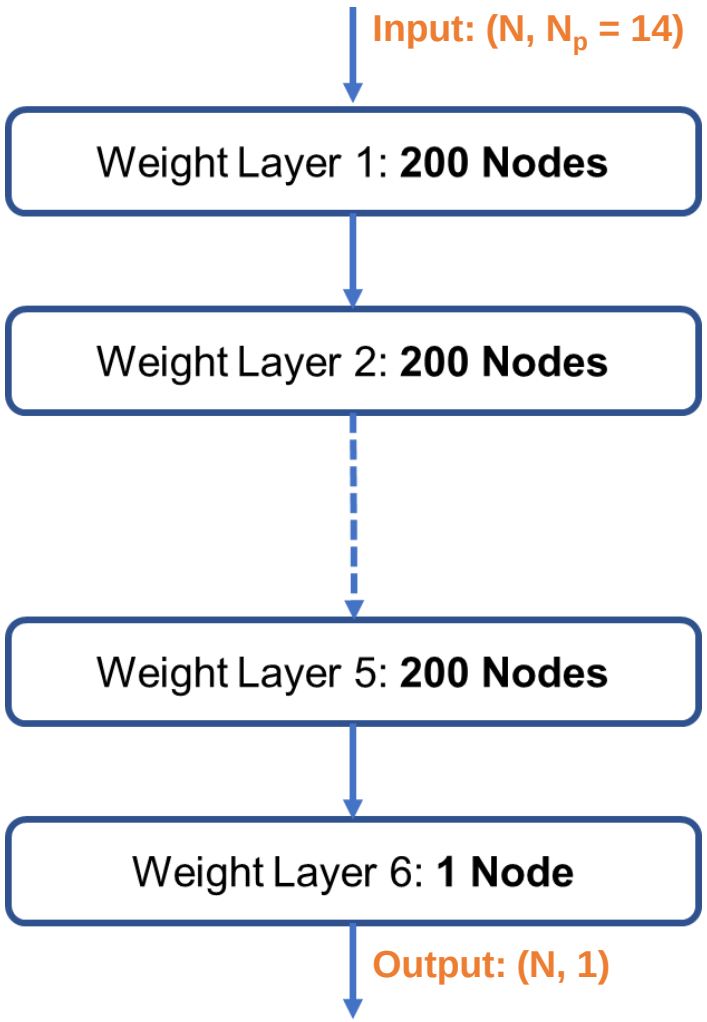
- Data Managing and Loading Module:
  1. Parse ROOT TTrees based on user-input
  2. Use Python Multiprocessing library constructs to store a “cache” of data
  3. Spawn processes using PyTorch to load data from cache
  4. Load the next chunk of data and replace the “cache”

- We get significantly faster data loading in comparison to the in-built dataloader

	Load Time
In-Built	506 s
Our Implementation	55 s

Load times are for 100 epochs of the MoMEMta test dataset

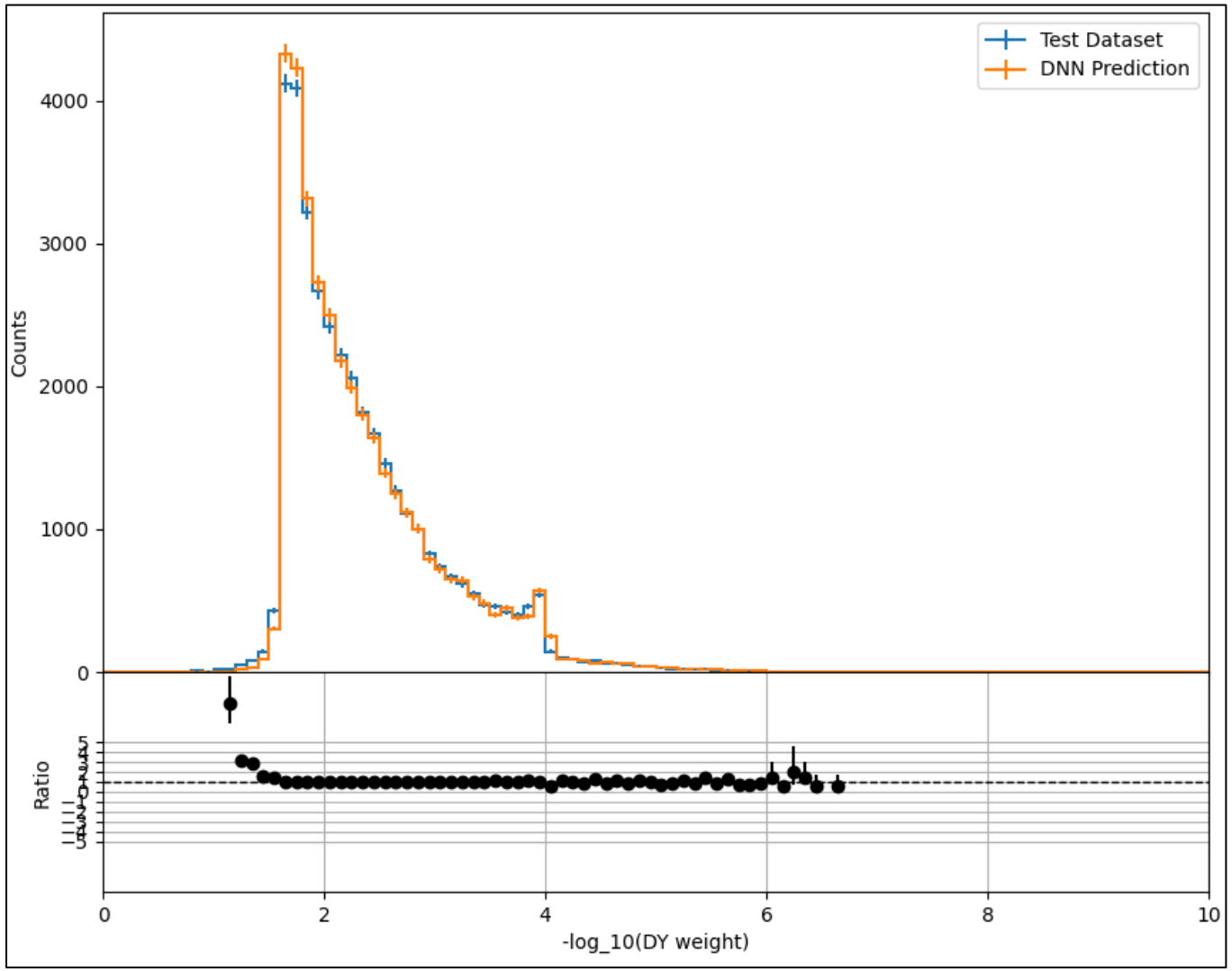
Network Architecture



DNN A: 5 Deep and fully-connected Layers

- We use a Fully-Connected Deep Neural Network with 5 deep layers of 200 Nodes each
- We split the data 8:1:1 for training, validation and testing purposes
- The output is the approximate transformed MoMEMta weights for N = ~270k training and validation events
- The network is trained for 100 epochs

Results using DNN



Results from DNN A: 5 Deep and fully-connected Layers

- Testing on **unseen data** gives us a good visual fit between DeepMEM predictions and the test data
- Ratio = $\frac{\#\ of\ predicted\ events\ in\ bin}{\#\ of\ actual\ events\ in\ bin}$
- Mean Absolute % Error = **1.6%**

where $\mathrm{MAPE} = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$

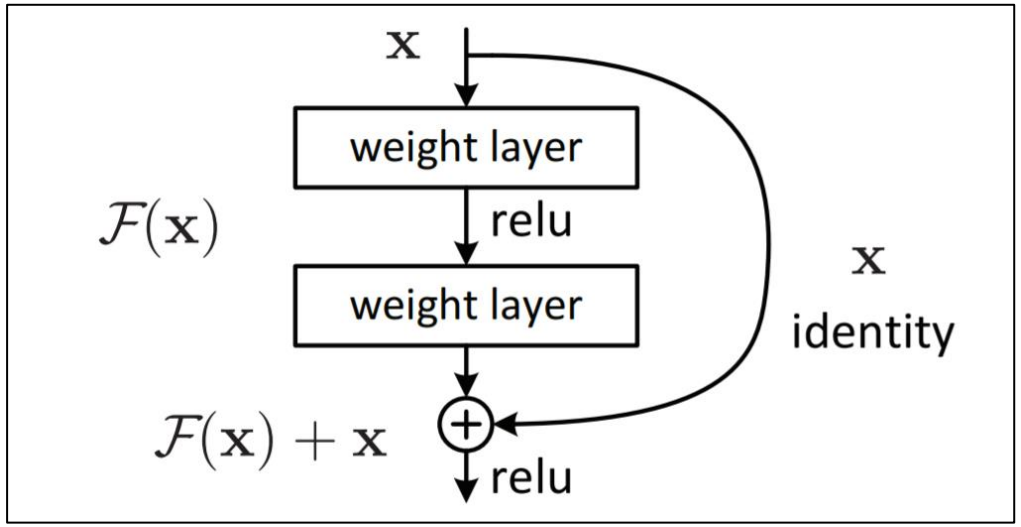
- However, we can see that the network cannot generalize well on bins that do not contain a lot of events

Residual Networks

Residual Networks are neural network architectures that **incorporate skip connections** in the network architecture

Ease training for deep networks by **providing shortcuts for backpropagation**, while gaining accuracy from the depth of the network. [1]

ResNets have empirically shown to have better results for aggressively deep networks (ILSVRC 2015) [1]

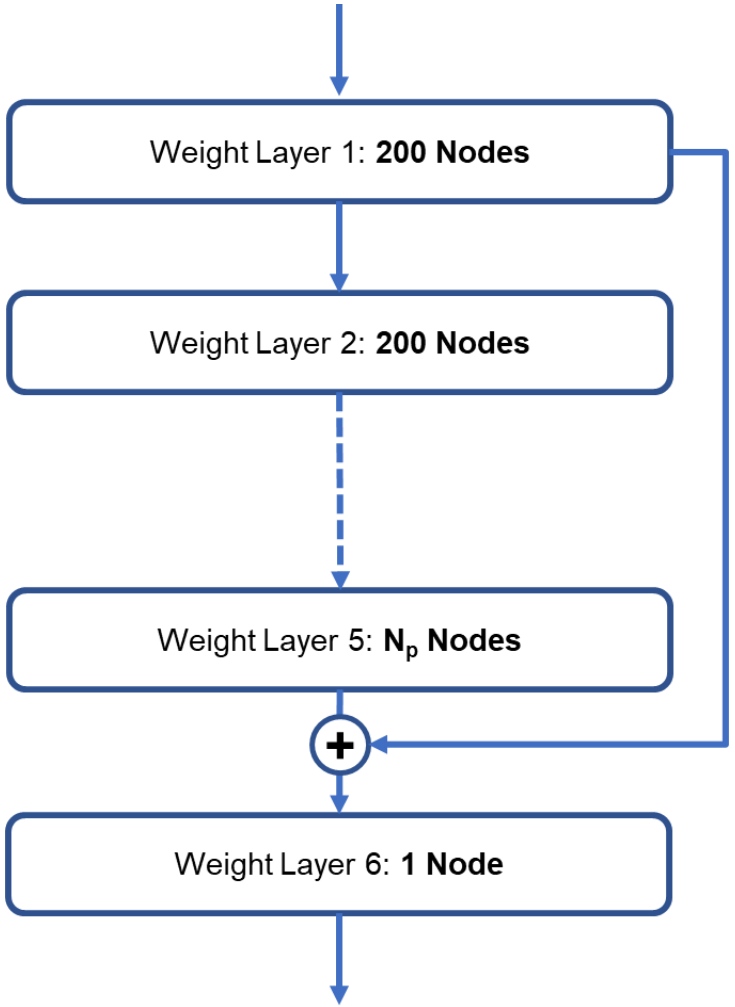


Why do ResNets work?

- They address the gradient vanishing phenomenon
- Smaller loss values can successfully transmit through the deep network and update earlier layers

Image credit & reference: [1] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition

Residual Network Architecture



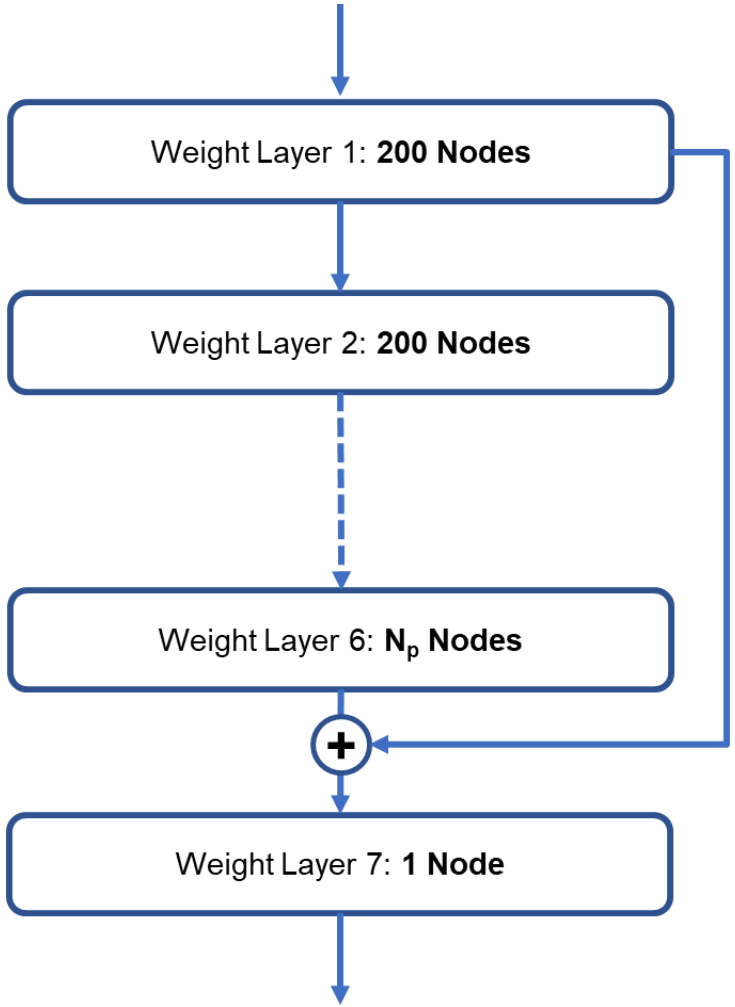
ResNet A: 5 Deep Layers followed by a skip connection

We include a skip connection into the original DNN A while retaining Depth

(This Network is less complex than DNN A)

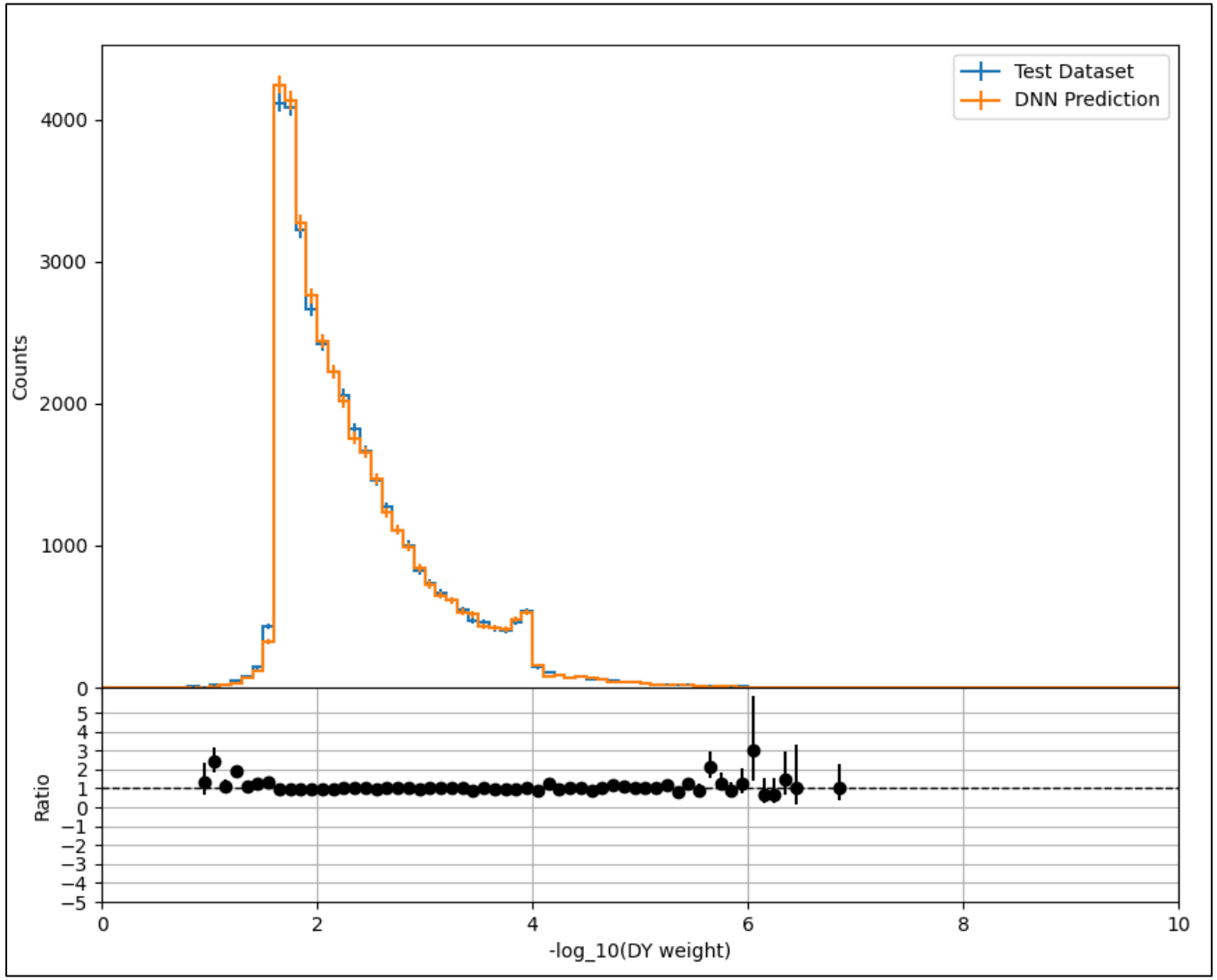
We include a skip connection into the original DNN A by adding an extra layer to the depth

(This Network is more complex and deeper than DNN A)



ResNet B: 6 Deep Layers followed by a skip connection

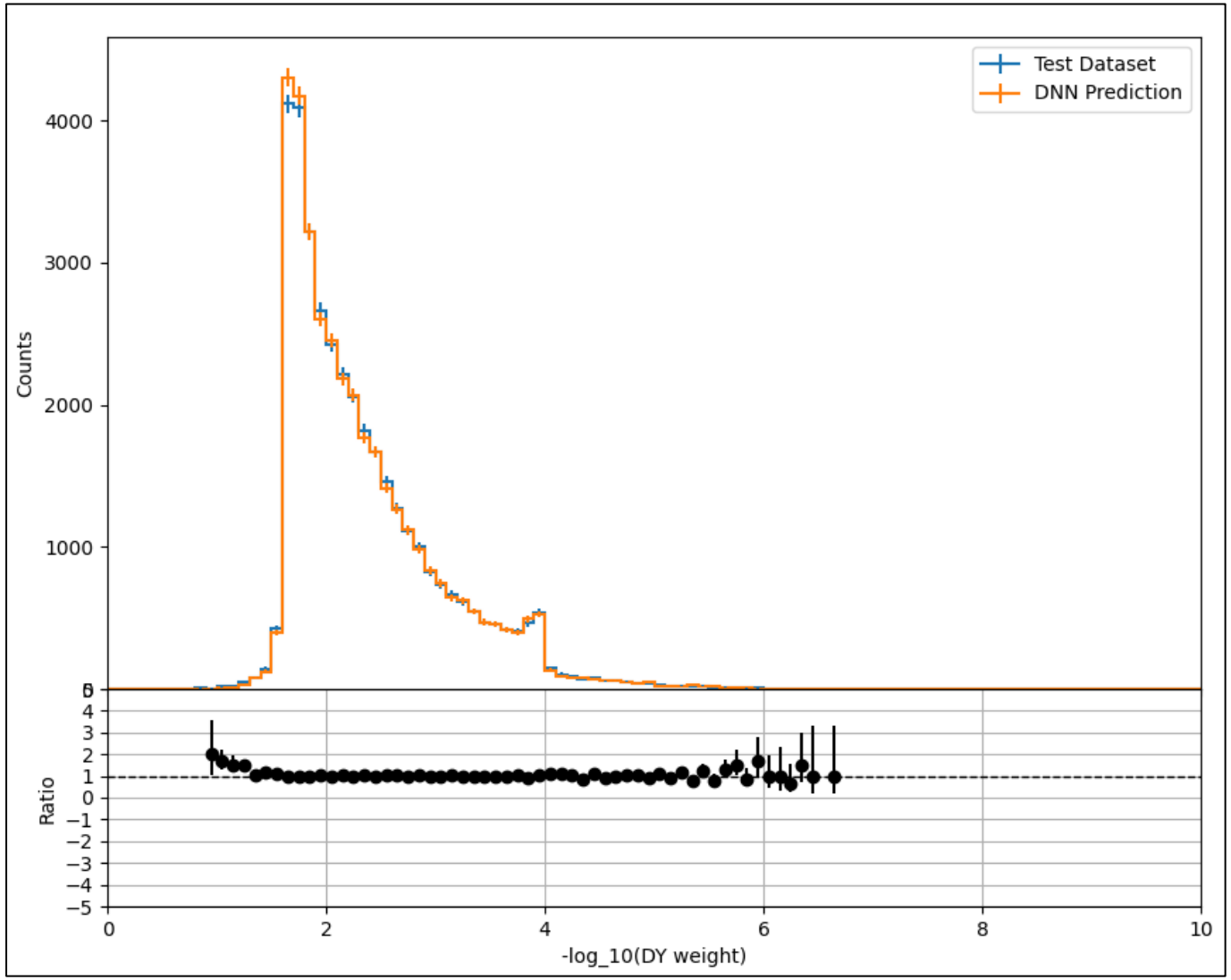
Results using Residual Network A



Results from ResNet A: 5 Deep Layers followed by a skip connection

- We see much better generalization using this architecture
- Mean Absolute % Error = **1.4%**
- We can argue that adding a skip connection improved the results since ResNet A was less complex than DNN A

Results using Residual Network B



Results from ResNet B: 6 Deep Layers followed by a skip connection

- We see even better generalization using this architecture
- Mean Absolute % Error = **1.2%**
- A more complex network with a skip connection gives us even better results by leveraging its depth

Summary

- Implemented ML methods to approximate MEM calculations and demonstrated the viability of this approach on a simple DY process
- Implemented a versatile Multiprocessing Dataloader to efficiently load data from ROOT files
- Implemented Residual Network architecture for better generalization

❖ A next step in this effort is to go beyond the simple DY process to other processes with more complex decays and more final state particles

❖ DeepMEM is an **open-source python library distributed on PyPI** that can be used on similar datasets:

- `python -m pip install deepmem`

Acknowledgements

I would like to thank both my mentors **Dr. Matthew Feickert** and **Dr. Mark Neubauer** for their incredible help throughout the project.

Thank you to IRIS-HEP and the NSF for funding this fellowship.