# Deep Learning for the Matrix Element Method


Mark Neubauer

University of Illinois at Urbana-Champaign


*International Conference on Computing in High Energy & Nuclear Physics (CHEP)*

*May 9, 2023 in Norfolk, VA USA*

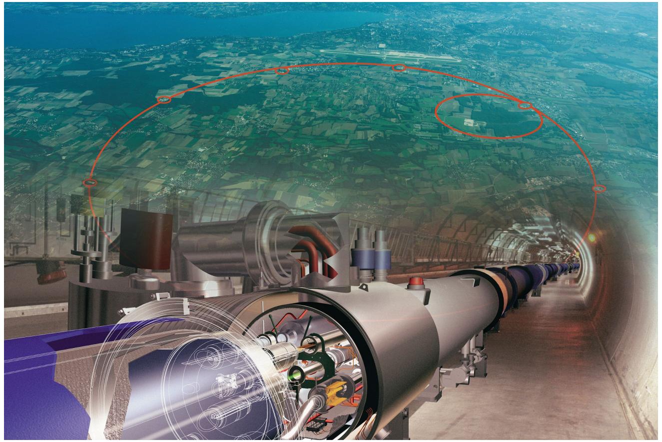
ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

iris hep

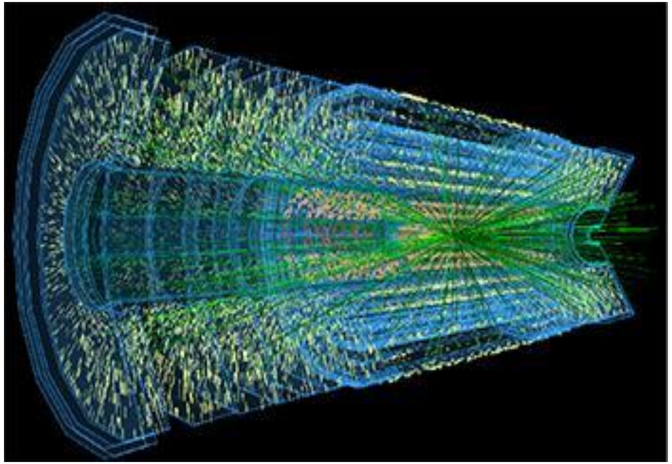
SCAILFIN

# 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
  - High-scale physics can lead to observable, but subtle, kinematic effects in (HL-)LHC data



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



# Matrix Element (ME) Method

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

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

Dynamics from QFT→ Correlations from physics

$\mathcal{P}_{\xi}(\boldsymbol{x}|\boldsymbol{\alpha})$ 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)

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

**Neyman-Pearson Discriminant**

(e.g. process search, hypothesis test)

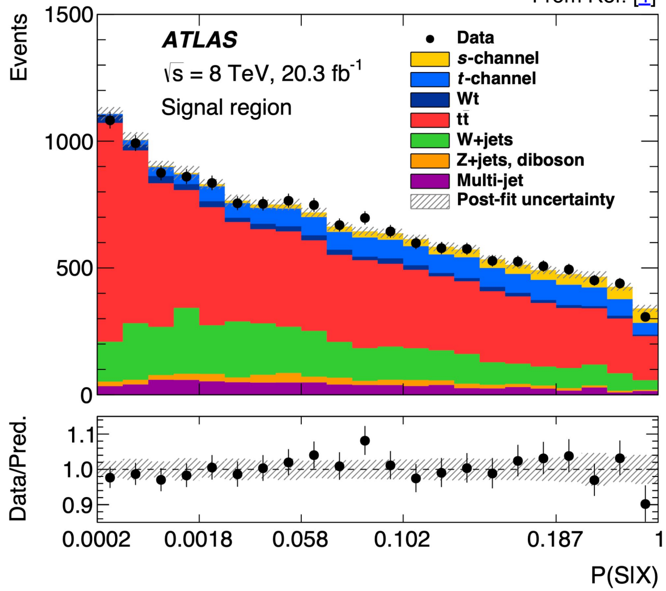
$$p(S|\mathbf{x}) = \frac{\sum_i \beta_{S_i} \mathcal{P}_{S_i}(\mathbf{x}|\boldsymbol{\alpha}_{S_i})}{\sum_i \beta_{S_i} \mathcal{P}(\mathbf{x}|\boldsymbol{\alpha}_{S_i}) + \sum_j \beta_{B_j} \mathcal{P}(\mathbf{x}|\boldsymbol{\alpha}_{B_j})}$$

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

# Matrix Element Method: Pros and Cons

- The ME Method has been used for many physics results from collider experiments
- The ME Method has several advantages over machine learning methods
  - Does not require training
  - Incorporates all of the available final state kinematic information, including correlations
  - Has a clear physical meaning in terms of transition probabilities within QFT

From Ref. [1]




- The main limitation of the ME method: ***computationally intensive***
  - E.g. calculating $\mathcal{P}_{\xi}(\boldsymbol{x}|\boldsymbol{\alpha})$ for the process:

$$\boxed{pp \to t\bar{t}H \to W^+bW^-\bar{b}b\bar{b} \to \ell\nu + 6j}$$

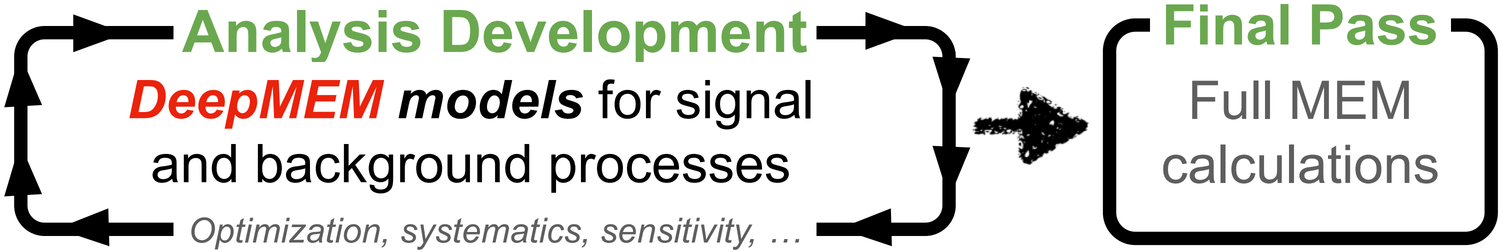
involves high-dimensional integration and can take minutes per event [2]

# ME Method in the Machine Learning Era

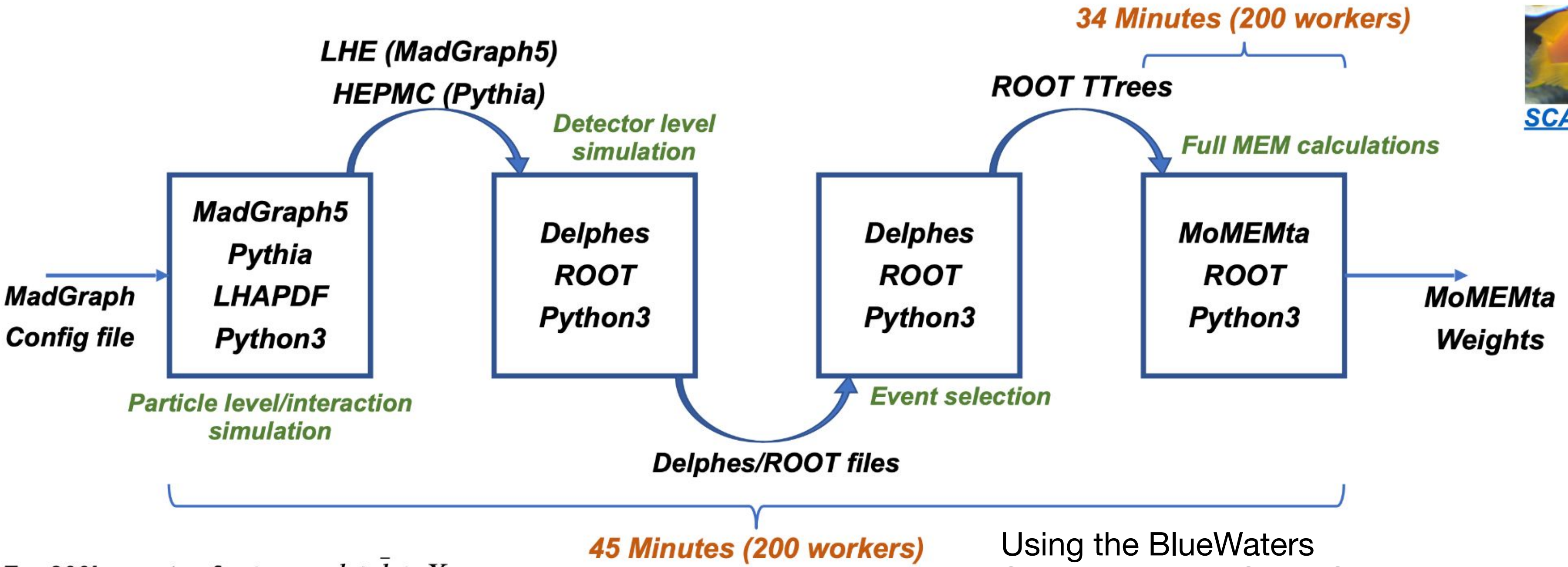
- The use of deep learning for fast and sustainable Matrix Element method calculations was first proposed in [3] (c.f. [4], [5], [6])

## MEM Model Development


## Use in Analysis



# Current ME Method Calculation Pipeline




SCAILFIN

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

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

Using the BlueWaters Supercomputer @ UIUC


*Dockerized Container*

*Data Flow*

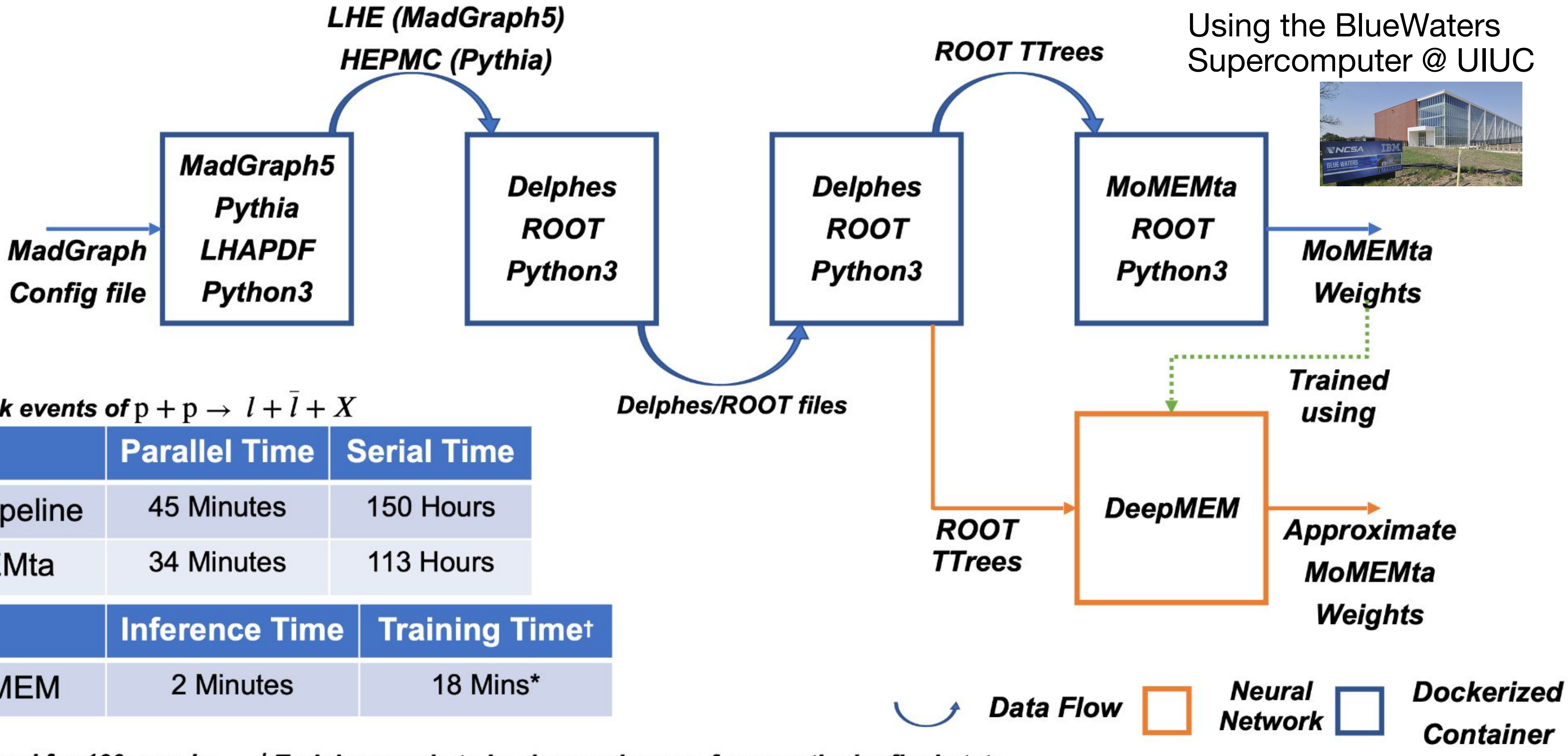
# DeepMEM Objectives

- Address challenges of the ME Method while retaining the benefits:
  - Retain the transparency and accuracy of the ME method calculations, while at the same time dramatically reducing their computational time
- Exploit ***Deep Neural Networks (DNNs)*** which are arbitrary function approximators that scale well with data → DeepMEM Ref [8]
  - Replace the calculations performed by ME method frameworks like MadWeight and MoMEMta with DNNs trained to learn these calculations (i.e. *learn maps such as:* $\boldsymbol{x} \rightarrow \mathcal{P}_{\xi}(\boldsymbol{x}|\boldsymbol{\alpha})$ or $\boldsymbol{x} \rightarrow \mathcal{P}_{\xi 1}(\boldsymbol{x}|\boldsymbol{\alpha}) / \mathcal{P}_{\xi 2}(\boldsymbol{x}|\boldsymbol{\alpha})$ )
  - Final calculations used in an analysis would be performed using the full pipeline for publication-quality accuracy → DeepMEM expedites calculations during research and development, and for quick studies
- Make MEM pipeline open and easy to use (e.g. via containerization) toward MEMaaS [3] & FAIR AI models

FAIR4HEP

# MEM Pipeline using DNN Approximations

Using the BlueWaters Supercomputer @ UIUC




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

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

| | Inference Time | Training Time† |
|---|---|---|
| DeepMEM | 2 Minutes | 18 Mins* |

* Trained for 100 epochs † Training needs to be done only once for a particular final state

# Data and Selection Description

- As a proof of principle, we studied the simple Drell-Yan process:

$$\boxed{pp \to \ell + \ell + X}$$

- Parsing the ROOT Trees produced after event selection, we use the 4-momentum of the final state particles and MET
- Mass is a very good discriminant, so we keep the neural network blind to mass by excluding it (following the approach of [6])

- Inputs:
  - $p_T$, $\eta$, $\phi$ of leptons & jets
  - Magnitude, $\phi$ of MET
  - $\rightarrow$ 14 input parameters
- Outputs:
  - Log-transformed MoMEMta weight values for each hypothesis

- Final dataset contains ~300k events

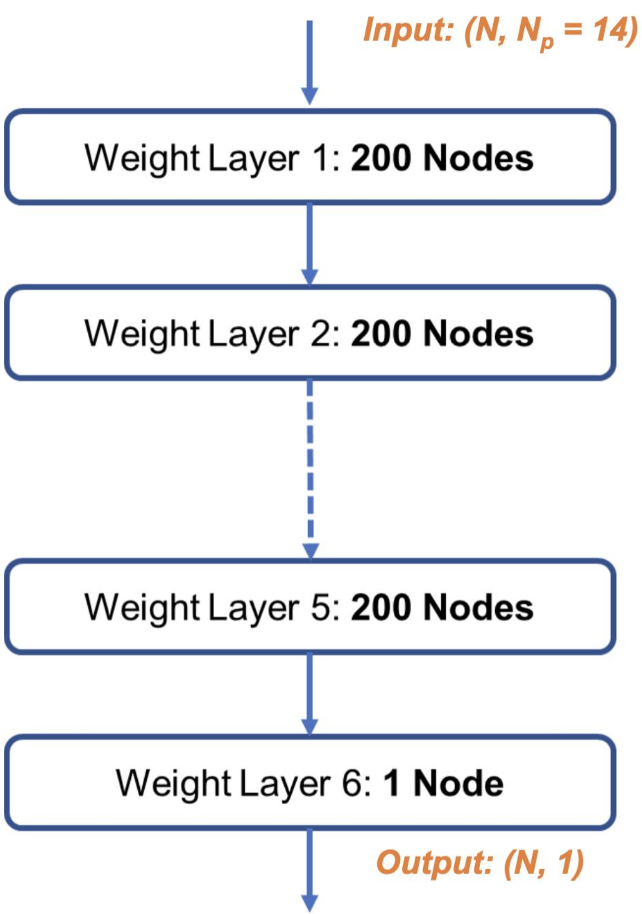
# Multiprocessing Data Loader

- PyTorch built-in Data Loader is designed for image/computer vision data - loads individual data based on use mappings
  - Inefficient for contiguous, tabular data
- No out-of-the-box Data Loader that can address the issues
- Data Managing and Loading Module
  - Parse ROOT Trees based on user input
  - Use Python Multiprocessing library constructs for data "cache"
  - Spawn processes using PyTorch to load data from the cache
  - Load next chunk of data and replace "cache"
- We get significantly faster data loading for our application than built-in Data Loader

***Load times are for 100 epochs of the MoMEMta test dataset***

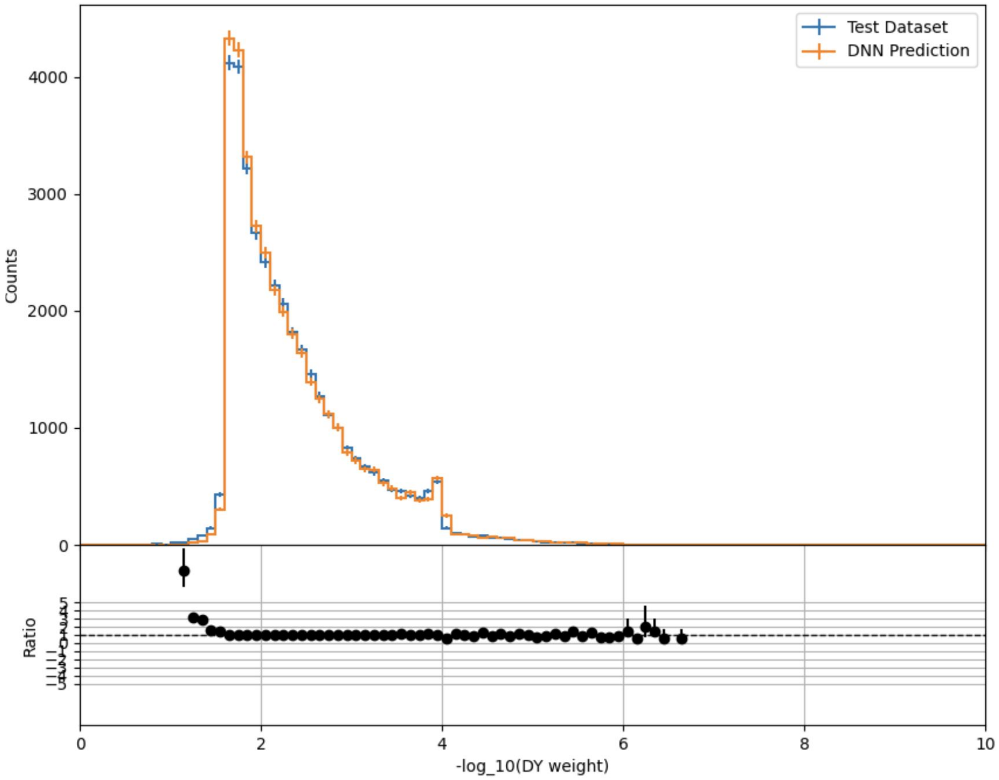
| | Load Time |
|---|---|
| In-Built | 506 s |
| Our Implementation | 55 s |

# Network Architecture




- We use a fully-connected Deep Neural Network with 5 deep (200 nodes) layers
- Adam optimizer with learning rate = 0.001
- 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 on an NVIDIA DGX A100

# Results using DNN




- Testing on unseen data gives a good by-eye fit between the DeepMEM predictions and the MoMEMta test data
- Mean Absolute % Error = **1.6%**

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

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

# Residual Networks

- Residual networks (ResNets) are neural network architectures that incorporate skip connections into the network architecture
- Eases training for deep networks by providing shortcuts for backpropagation, while gaining accuracy from the depth of the network (see ref [7])
- ResNets have empirically shown to perform well for aggressively deep networks (ILSVRC'15) [7]
- **Why do ResNets work**?
  - Address vanishing gradient problem
  - Smaller loss values can successfully transmit through a deep network and be used to update the precursor layers

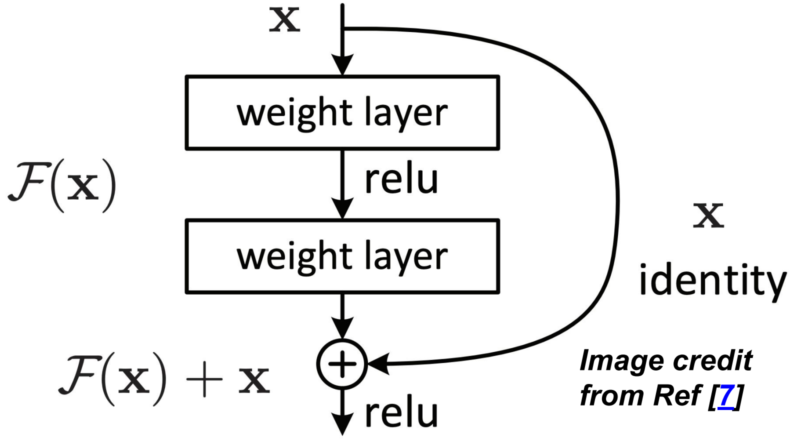
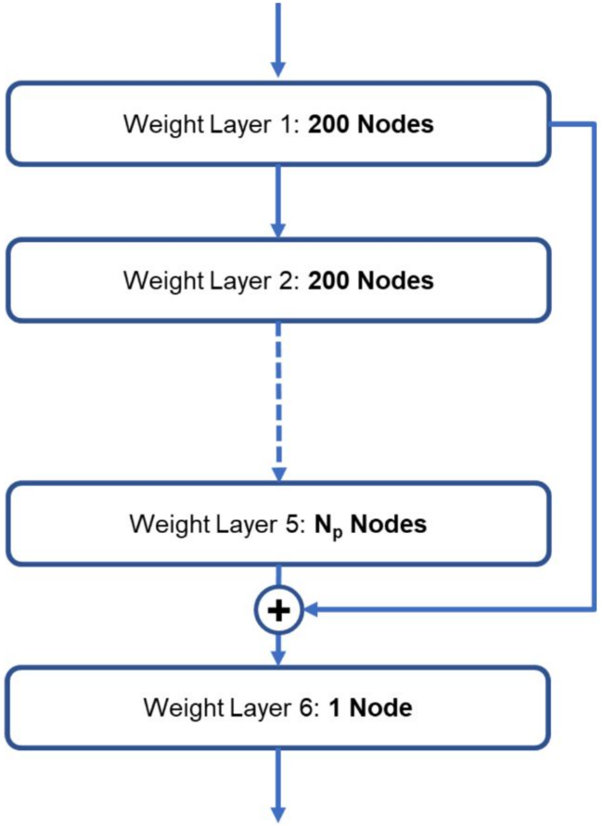



*Image credit from Ref [7]*

# Residual Network Architecture



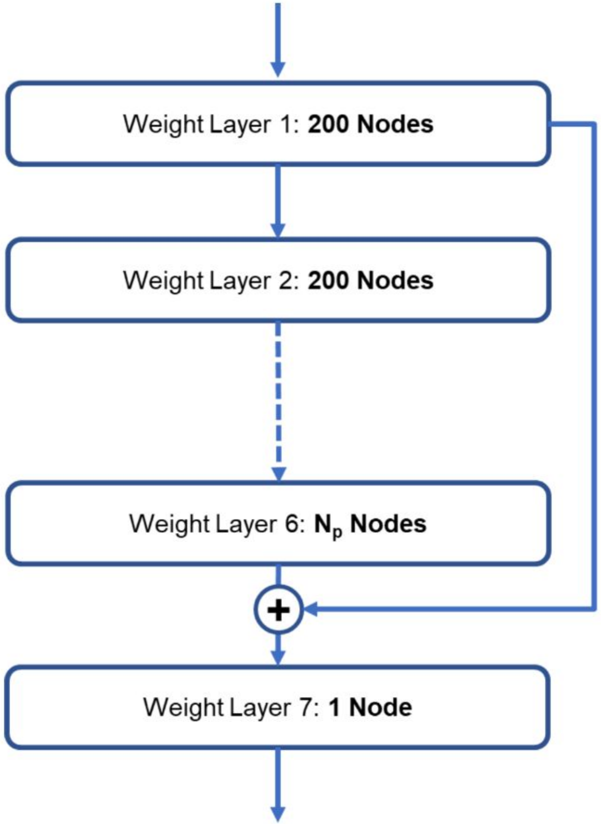

***ResNet A:*** *5 Deep Layers followed by a skip connection*

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

***(This Network is less complex than DNN A)***

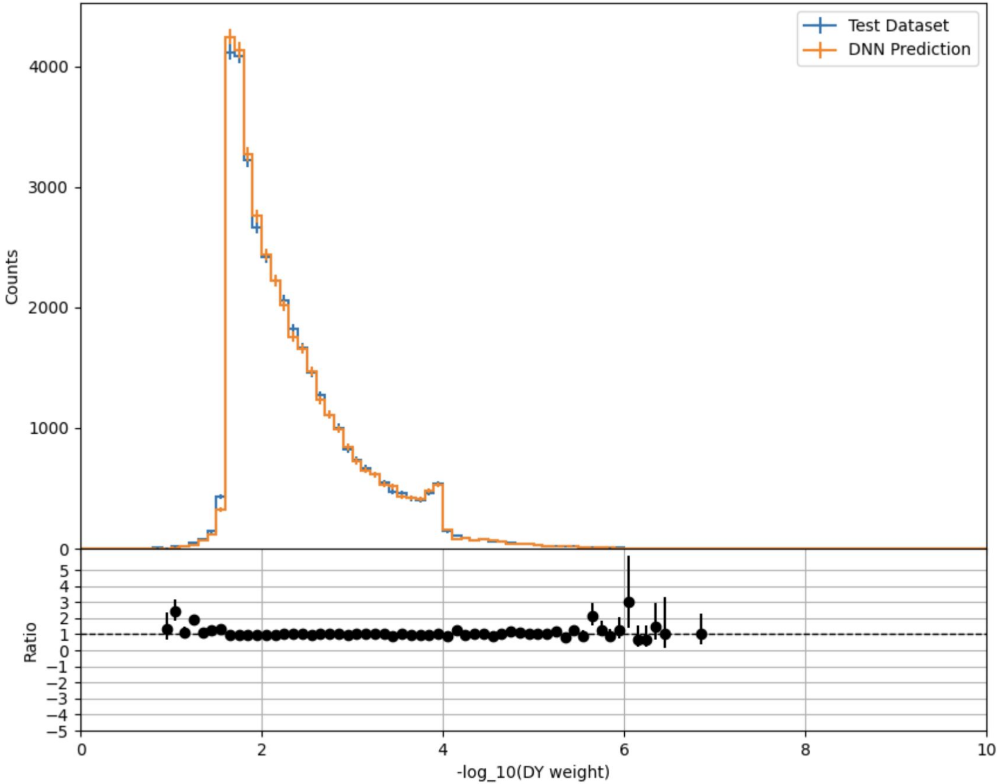
***We include a skip connection into the original DNN A by adding an extra layer to the depth***

***(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



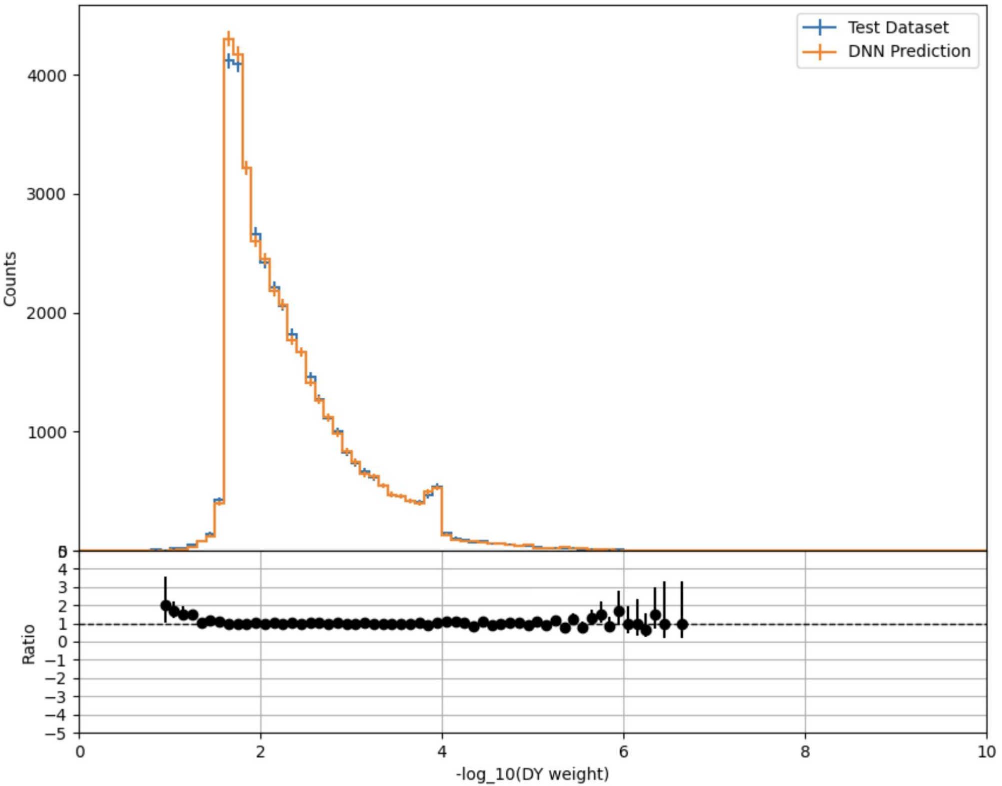

- We see better generalization as compared to the original DNN with this architecture

- Mean Absolute % Error = **1.4%**

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

- We argue that adding a skip connection improved the results since ResNet A is less complex than the original DNN

# Results using Residual Network B



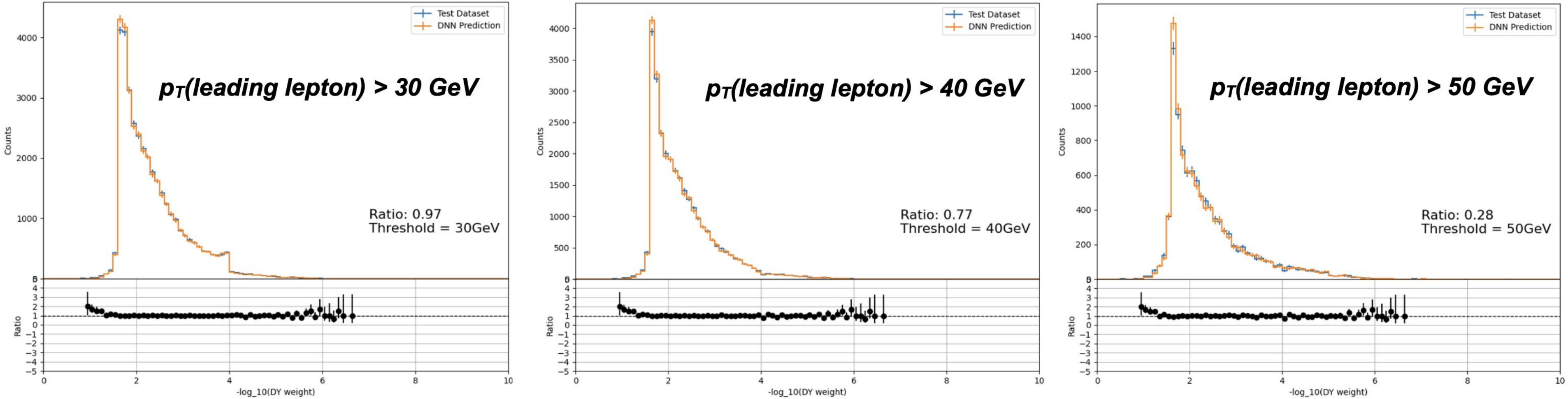

- We see better generalization as compared to the original DNN and similar to ResNet A with this architecture
- Mean Absolute % Error = **1.2%**

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

- A more complex network with a skip connection gives us slightly better results by leveraging its depth

# Generalization in Kinematic Phase Space

- We checked the modeling (ResNet B) on different kinematic subsets of the test data (No Retraining!)




- Good modeling retained → DeepMEM modeling of MEM weights robust against subsamples defined by leading lepton $p_T$ cut
  - Similar good results observed for subsamples through jet $p_T$ cuts

# Summary

- Implemented deep learning methods to approximate ME Method calculations and demonstrated the viability of this approach
- Implemented a Residual Network for better generalization; showed the model to be robust against kinematics variations w/o retraining

# Future Work

- ❖ Study processes with more complex decays and final state particles
- ❖ Explore other ML architectures, include adding physics constraints
- ❖ Generate simulated data and models adhering to FAIR principles and exploit novel tools developed for AI model intepretability
  - ➢ See CHEP23 talks: FAIR AI Models in HEP, FAIR4UFO Models, Interpretability for DNN Top Taggers

DeepMEM is an open-source python library distributed on PyPI that available for similar studies: `python -m pip install deepmem`

# Acknowledgements


- The key ideas were developed through discussions w/ **Philip Chang**
- This work was performed by **Mihir Katare** and **Matthew Feickert**, with guidance from **Avik Roy**


*Philip Chang*

*Mihir Katare*

*Matthew Feickert*

*Avik Roy*


- This work was supported through grants from the National Science Foundation under IRIS-HEP ([OAC-1836650]) and SCAILFIN ([OAC-1841456])

# References


[1] ATLAS Collaboration, "Evidence for single top quark production in the *s*-channel in *pp* collisions at √8 TeV with ATLAS using the Matrix Element Method", *PLB* 756, 228 (2016)

[2] A. Bayse, "A search for the *ttH* (*H*→*bb*) channel at the LHC with the ATLAS detector using a matrix element method", Ph.D. Thesis, UIUC (2015)

[3] P. Chang, S. Gleyzer, M. Neubauer, D. Zhong, "Sustainable Matrix Element through Deep Learning", HSF-CWP-018, 10.5281/zenodo.4008241 (2017)

[4] Albrecht, J. *et al*. (The HEP Software Foundation), "A Roadmap for HEP Software and Computing R&D for the 2020s", *Comput. Softw. Big. Sci.* (2019) 3, 7

[5] F. Bury, C. Delaere, "Matrix Element Regression – Breaking the CPU Barrier", *JHEP* 20 (2021)

[6] A. Butter, T. Heimel, T. Martini, S. Peitzsch, T. Plehn, "Two Invertible Networks for the Matrix Element Method" (2022)

[7] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition", ILSVRC 2015, arXiv:1512.03385 [cs.CV] (2015)

[8] DeepMEM Github repository

[9] M. Katare, M. Neubauer, M. Feickert, A. Roy, "Deep Learning for the Matrix Element Method", *Proceedings of Science* ICHEP2022, 246 (2022)