Development of Novel Unfolding Techniques using Normalizing Flows

Applicant: Jack P. Rodgers

Mentor: Dr. Massimiliano Galli

Project Duration: 12 weeks (12 May 2025 - 28 July 2025)

Problem Statement:

The main problem that we try to solve is to find a mapping of the detector signatures that we observe to the true observables of the particles in the detector. In the LHC community, this procedure often manifests as converting the "reco-level" distributions of particles to the "gen-level" distributions (also phrased as "data" and "simulation" in the context of Monte Carlo) and is the whole problem of unfolding [3]. Traditional approaches to unfolding have involved creating response matrices to model the effect of detectors [5] or using traditional statistical tests to try to approximate a reweighting scheme. However, with the advent of machine learning, physicists involved in research at CERN can create models with learnable weights trained directly on the enormous amount of data produced at the LHC. Two prominent examples of this are Bayesian Networks and OmniFold [1], which train to sample from a learned distribution or use classifier predictions about gen or reco particles to create weights, respectively.

Solution:

Normalizing flows have become very popular in HEP/ML applications due to their ability to take a standard distribution, such as a Gaussian, and transform it into a much more complicated distribution. However, implementations of normalizing flows for this task are generative [4] by design and sample from the learned unfolded distribution rather than directly mapping. The implementation proposed in this project is unique in that the normalizing flow is conditioned on whether the input comes from simulation or data [2], and since the network is conditioned on both of these processes, it can be used to produce a two-way mapping making the need for two or more flows obsolete which is a relevant way to treat the matter given the high interest in foundational models (model-fits-all) for HEP tasks in recent years. I have a significant interest in being able to develop the framework that Dr. Galli has developed, as it holds a vast amount of potential to increase performance, efficiency, and most importantly, interpretability (which modern blackbox neural networks lack) in such an important problem in HEP.

Timeline:

I intend to help develop the codebase for Dr. Galli's project using Python/Pytorch according to the timeline below, and testing/adding to the framework to answer important questions:

Week 1-2: Read papers on recent developments in unfolding and how these may be relevant to our work, but more importantly, establish a space on the Purdue cluster to work on the project for HPC capability and get oriented with the current status of the code.

Week 3-4: Take performance metrics on more observables. Currently, the network only attains good performance on the 4-5 observables (features) given to it. How many more quantities can it unfold on simultaneously before we observe significant performance degradation?

Week 5-6: Compare ML process performance for derived quantity. For some derived quantity that depends on a more fundamental quantity, will unfolding before or after applying the relevant transformation on the original/derived quantity give us the same result? If not, why?

Week 7-8: Distribution-level metrics are important, but how does the flow perform on a single event? Is the event's relevant observable quantity displaced at the cost of satisfying a distribution measure?

Week 9-10: How dependent is the model we have on a particular process? We can test interaction generalizability by training on one process and performing inference on a different process. How generalizable are the mappings learned by the network?

Week 11-12: 2 questions: How does the model perform when applying negative weights? How do we perform with unfolding on signal and background together? These tasks may both require custom samples to answer and are therefore placed at the end of the fellowship to give time for notoriously lengthy sample production.

References:

[1] OmniFold: A Method to Simultaneously Unfold All Observables
[https://arxiv.org/pdf/1911.09107]

[2] One Flow to Correct Them All
[https://arxiv.org/pdf/2403.18582]

[3] CERN Unfolding Brief
[https://s3.cern.ch/inspire-prod-files-a/abf1ed80cf0adea41154f9726a18ba13]

[4] Modern Unfolding Techniques
[https://arxiv.org/pdf/2404.18807]

[5] Traditional Unfolding
[https://arxiv.org/abs/hep-ph/9509307]