Data-Driven Reweighting for Monte Carlo Simulations

Christian Bierlich [1♠] **Phil Ilten** [2†] **Tony Menzo** [2,⋆] **Stephen Mrenna** [2,3✠]

Manuel Szewc [2,4‖] **Michael K. Wilkinson** [2⊥] **Ahmed Youssef** [2‡] **Jure Zupan** [2,§]

[1] Department of Physics, Lund University, Box 118, SE-221 00 Lund, Sweden
[2] Department of Physics, University of Cincinnati, Cincinnati, Ohio 45221, USA
[3] Scientific Computing Division, Fermilab, Batavia, Illinois 60510, USA
[4] ICAS, ICIFI and ECyT-UNSAM, San Martín, Buenos Aires, 1650, Argentina

[♠]christian.bierlich@hep.lu.se, [†]philten@cern.ch, [⋆]menzoad@mail.uc.edu, [✠]mrenna@fnal.gov,
[‖]szewcml@ucmail.uc.edu, [⊥]michael.wilkinson@uc.edu, [‡]youssead@ucmail.uc.edu,
[§]zupanje@ucmail.uc.edu,

MLHAD

Abstract

This paper introduces a novel method, termed **H**istories and **O**bservables for **M**onte-Carlo **E**vent **R**eweighting (**HOMER**), that can be used for extracting a fragmentation model probability function directly from experimental data without requiring an explicit parametric form. The method consists of three steps: the training of a classifier between simulation and data, the inference of single fragmentation weights, and the calculation of weights for full hadronization chains. We illustrate the use of HOMER on a simplified hadronization problem, a $q\bar{q}$ string fragmenting into pions, and extract a modified Lund string fragmentation function $f(z)$ from binned experimental data.

1 Introduction

Monte Carlo Event Generators (MCEGs), including PYTHIA [1], are fundamental in high-energy particle physics research, supporting both theoretical and experimental efforts. These tools are crucial in collider experiments for generating theoretical predictions that can be tested against observed data. One of the more complex tasks in MCEGs is the simulation of the process by which quarks and gluons combine to form hadrons (*e.g.*, protons), which are the particles that are recorded by detectors in the actual experiments. This process, known as *hadronization*, cannot be calculated from first principles. As a result, MCEGs utilize phenomenologically driven approaches such as the Lund string model [2, 3] or the cluster model [4–6]. Despite the overall success of these models in describing a wide range of experimental data, the models do face limitations when it comes to certain data subsets, motivating a search for alternative, data-driven solutions or augmentations.

Machine Learning (ML) methods provide a new set of tools that may be able to improve current description of the nonpertubative process of hadronization, see, *e.g.*, [7–11] for recent approaches. The fundamental challenge is that the process of hadronization is not directly observable; experiments record event-level information that is then related on a statistical basis to the individual hadron emissions, since the mapping from single emissions to event-level information is non-invertible.

In this paper, we present a new method, the Histories and Observables for Monte-Carlo Event Reweighting (HOMER) method [12], to extract single emission information from event-level observables. HOMER uses phenomenologically motivated hadronization models (*e.g.*, the Lund string model from PYTHIA) as a base starting point, and the augments them appropriately. That is, HOMER first learns the event-level likelihood ratios between the distributions from data and the base hadronization model by training a classifier, a well-established technique in particle physics [13–16]. The main novelty of HOMER lies in using these event-level likelihood ratios to build a modified hadronization model by assigning likelihood ratios for each individual hadron emission. The output of the HOMER method is a data-driven reweighting of the baseline PYTHIA event generator, such that the resulting distributions match the observed training data.

2 Background and Method

HOMER is a framework designed to learn a hadronization model directly from data, without needing a predefined functional form. For the benchmark considered in this work, we take as a starting point the Lund string fragmentation function $f(z)$, which governs hadron formation in PYTHIA. The method is tested on synthetic data generated by PYTHIA as a closure test to evaluate how well the extracted function $f_{\text{HOMER}}(z)$ can approximate the true function, $f_{\text{data}}(z)$.

HOMER starts with a parametric hadronization model that approximates the data and then adjusts it using a two-step process to improve the fit. Here, PYTHIA acts as the baseline simulator with an initial string fragmentation function $f_{\text{sim}}(z)$, using different parameters than those used to generate the synthetic data, $f_{\text{data}}(z)$. The simulator produces **events**, which are compared to data. In our terminology, an event e is a list of observables, $\vec{x}_e$, which describe a *single collision*. A collection of events $\{\vec{x}_{e_1}, \ldots, \vec{x}_{e_n}\}$ is called a **run**. For $\vec{x}_e$ we consider 13 *high-level observables* whose distributions have already been measured by experimental collaborations. We first give a short overview of the Lund string fragmentation model used in PYTHIA, before detailing HOMER and its implementation.

2.1 Lund String Model

The default hadronization model in PYTHIA is the Lund string fragmentation model [2, 17]. In this work we focus on the simple system of a quark-antiquark pair, $q_i\bar{q}_i$, with no attached gluons. The hadronization of this system in the PYTHIA Lund string fragmentation model of hadronization is based on the observation that as the quarks move apart along the z-axis, a flux tube of color field (a string) forms between the $q_i\bar{q}_i$. As the string stretches, the energy stored in it increases until it becomes energetically favorable to produce new quark-antiquark pairs from the vacuum. This causes the string to break into fragments, creating hadrons composed of $q_i\bar{q}_j$ pairs. Each string break is treated probabilistically, and multiple hadrons are produced sequentially, forming a **fragmentation chain**.

The probability of producing a hadron with a longitudinal momentum fraction z is governed by the Lund string fragmentation function:

$$f(z) \propto \frac{(1-z)^a}{z} \exp\left(-\frac{bm_T^2}{z}\right), \tag{1}$$

where $m_T^2 = m_{ij}^2 + p_{\text{T}}^2$ is the transverse mass, a and b are parameters, and m_{ij} is the hadron mass. The transverse momentum $\vec{p}_{\text{T}}$ of the emitted hadron is determined from the transverse momentum of the string, $\vec{p}_{\text{T}}^{\,\text{string}}$. Each string break is described by a seven-dimensional vector:

$$\vec{s}_{hcb} = \{z, \Delta\vec{p}_T, m, \texttt{fromPos}, \vec{p}_T^{\,\text{string}}\}_{h,c,b}, \tag{2}$$

where z is the fraction of the remaining string's lightcone momentum taken by the emitted hadron; $\Delta\vec{p}_T = (\Delta p_x, \Delta p_y)$ is the two-dimensional momentum kick of the emitted hadron; m is the hadron mass; `fromPos` is a boolean, stored within PYTHIA, that encodes whether the string break occurred at the positive or the negative end of the string; and $\vec{p}_T^{\,\text{string}} = (p_x^{\text{string}}, p_y^{\text{string}})$ is the transverse momentum of the string before the breaking. Here, h, c, and b are the indices labeling the history within even sample, the fragmentation chain within the history h, and the string break within that fragmentation chain, respectively.

Each iteration of causally disconnected string fragmentations consists of: randomly selecting one of the two string ends; assigning probabilistically a quark flavor to be pair produced during the string break; generating the transverse momentum of this pair; generating the lightcone momentum fraction of the new hadron; and finally computing the longitudinal momentum of the new hadron, by conserving the total energy and momentum of the system. Iterative fragmentations continue until the energy of the string system crosses a chosen low-energy threshold. The remaining string piece is then combined into a final pair of hadrons. Within the PYTHIA event generator code, this final combination is performed by an algorithm called `finalTwo`. The `finalTwo` method effectively works as a filter by checking whether the final hadrons can be produced on-shell.[1] If this is not the case, the generated fragmentation chain is rejected, and the simulation of hadronization starts anew, taking again the original string as the starting point. This process can be repeated several times, until the `finalTwo` step is successful.

The fragmentation chain of a given string is denoted as $\vec{S}_{hc} = \{\vec{s}_{hc1}, \ldots, \vec{s}_{hcN_{h,c}}\}$, and the history of the simulation, including accepted and rejected chains, is represented as $\vec{\mathbf{S}}_h = \{\vec{S}_{h1}, \ldots, \vec{S}_{hN_h}\}$, where $h = 1, \ldots, N_{\text{data}}$, is the simulation history index, with N_{data} the total number of particle physics events in a run; $c = 1, \ldots, N_h$ is the fragmentation chain index for a particular $h-$th simulation history, which has $N_h - 1$ rejected fragmentation chains and one accepted fragmentation chain; while $b = 1, \ldots, N_{h,c}$ is the string break index, that runs over the c-th fragmentation chain that has a total of $N_{h,c}$ string breaks.

2.2 HOMER Method

The HOMER method aims to correct the Lund string model in PYTHIA to match experimental data via reweighting individual emissions. The fragmentation probability for a string break, $\vec{s}_{hcb}$, depends on the transverse momentum $\vec{p}_T{}^{\text{string}}$ of the string. HOMER learns a weight $w_{\text{s}}^{\text{data}}(\vec{s}_{hcb})$ that modifies the baseline fragmentation function from PYTHIA:

$$p_{\text{sim}}(\vec{s}_{hcb}) \to w_{\text{s}}^{\text{infer}}(\vec{s}_{hcb}) p_{\text{sim}}(\vec{s}_{hcb}).$$

These weights are learned by matching the reweighted samples to experimental data. The method is carried out in three steps:

1) **Event Classifier:** A classifier is trained to distinguish between events generated by the baseline model and experimental data. The classifier assigns an event-level weight, $w_{\text{class}}(e_h)$, which approximates the true event weight $w_{\text{exact}}(e_h)$:

$$w_{\text{class}}(e_h) \approx w_{\text{exact}}(e_h) = \frac{p_{\text{data}}(e_h)}{p_{\text{sim}}(e_h)},$$

where $w_{\text{class}}(e_h) = \frac{y(\vec{x}_h)}{1-y(\vec{x}_h)}$, with $y(\vec{x}_h)$ the classifier output that takes the observables $\vec{x}_h$ as the input. In this paper, we present results for binned data, using only information already available from LEP measurements in HEPDATA, as used in the Monash tune [18]. The classifier is a feed-forward NN implemented with the PYTORCH library[19]. To avoid over-fitting, we consider a small NN composed of two inner layers with 13 and 26 neurons each, and with `ReLU` activation functions. The final layer has a `Sigmoid` activation function to ensure $y(\vec{x}_h) \in [0, 1]$. The loss function itself is

$$\mathcal{L} = \sum_{\vec{x}_i} \frac{N_{\text{data}}}{n_i} \sum_{k=1}^{n_i} \frac{\left(p_k^{\vec{x}_i} - \bar{p}_k^{\vec{x}_i}(y)\right)^2}{p_k^{\vec{x}_i}}\,, \tag{3}$$

where the summation is over all the observables and $p_k^{\vec{x}_i}$ ($\bar{p}_k^{\vec{x}_i}$) are the measured (expected) fractions of events per bin. In particular, the expected fractions are estimated from events simulated with the baseline simulation model and weighted with $w_{\text{class}}(e_h) = y(\vec{x}_h)/\big(1 - y(\vec{x}_h)\big)$, so that $\mathcal{L}$ is minimized for $w_{\text{class}}(e_h) \approx w_{\text{exact}}(e_h)$.

2) **Fragmentation Weights:** In this step, the event-level weights $w_{\text{class}}(e_h)$ from Step 1 are translated into emission-level weights $w_{\text{s}}^{\text{infer}}(\vec{s}_{hcb})$ for each string break, which adjust the baseline fragmentation probability to match experimental data. In order to do so, we need to account for the fact that PYTHIA simulates both accepted and rejected fragmentation chains, while only the final hadron momenta

[1]An on-shell particle satisfies the relation $E^2 = p^2 + m^2$, i.e., it can exist as a real particle.

are experimentally measurable. The event weight $w_{\text{exact}}(e_h)$ is thus an average over all possible fragmentation chains leading to the same event:

$$w_{\text{exact}}(e_h) = \frac{p_{\text{sim}}^{\text{acc}}}{p_{\text{data}}^{\text{acc}}} \left\langle w_{\text{exact}}(\vec{S}_{hN_h}) \right\rangle ,$$

where $w_{\text{exact}}(\vec{S}_{hN_h})$ is the product of the weights for individual string breaks, and $p_{\text{sim}}^{\text{acc}}$ ($p_{\text{data}}^{\text{acc}}$) is the fraction of total simulated (reweighted) chains that are accepted by `finalTwo`. A neural network g_θ is used to parameterize $w_{\text{s}}^{\text{infer}}(\vec{s}_{hcb}, \theta)$ for each string break. It takes the string break vector $\vec{s}_{hcb}$ as input and outputs the weight w_s^{infer} for this break. To achieve this, we use a Message-Passing Graph Neural Network (MPGNN) implemented in the PYTORCH GEOMETRIC library [20] and represent each fragmentation chain as a particle cloud with no edges between the nodes. The nodes carry string break vectors $\vec{s}_{hcb}$, where $b = 1, \ldots, N_{h,c}$. We identify an edge function that is evaluated on each node and produces updated features for that node with $\ln g_\theta$, so that the updated weight for the whole fragmentation chain is obtained by summing $\ln g_\theta$ over all the nodes and exponentiating the sum.

We further parameterize the edge function $\ln g_\theta$ as the sum of two different functions, $\ln g_\theta = g_1 - g_2$, to better encode the conditional structure of the problem. These two functions are fully connected neural networks with 3 layers of 64 neurons each and rectified-linear-unit, `ReLU`, activation functions. The inputs are either string break vectors $\vec{s}_{hcb}$ for g_1, or the string variables $\{\vec{p}_T^{\,\text{string}}\}_{h,c,b}$ for g_2. The output of each neural network is a real number with no activation function applied.

The network is trained using a loss function with two components: a cross-entropy loss, which matches the learned event weights $w_{\text{infer}}(e_h, \theta)$ to the event weights from Step 1, $w_{\text{class}}(e_h)$; and a regularization term, which ensures that the learned weights respect the conditional structure of the fragmentation model. We used the `Adam` optimizer with an initial learning rate of 10^{-3}, which decreases by a factor of 10, if no improvement is found after 10 steps. We train for 100 epochs with batch sizes of 10^4. To avoid over-fitting, we apply an early-stopping strategy with 20 step patience. Minimizing this loss enables the network to reweight fragmentation chains, aligning the simulations with experimental data at the hadron level.

3) HOMER Output: Once the fragmentation weights, $w_{\text{s}}^{\text{infer}}(\vec{s}_{hcb}, \theta)$, are learned, the final step of the HOMER method is to compute the total weight for each simulated event. The total weight for a single event history is the product of the weights for all string breaks in that event's fragmentation chain, including both accepted and rejected fragmentations:

$$w_{\text{HOMER}}(\vec{\mathbf{S}}_h) = \prod_{c=1}^{N_h} \prod_{b=1}^{N_{hc}} w_{\text{s}}^{\text{infer}}(\vec{s}_{hcb}, \theta),$$

where N_{hc} is the number of string breaks in each chain c. Unlike the event-level weights $w_{\text{exact}}(e_h)$, which average over all possible histories leading to the same event, $w_{\text{HOMER}}(\vec{\mathbf{S}}_h)$ assigns a weight to each individual simulation history. This allows us to efficiently reweight the simulated events using Monte Carlo samples, updating the fragmentation model to better match the measured experimental data.

3 Results

The learned model and its fit to data are shown in Fig 1, where in the right panel we display the fit for the total multiplicity (*i.e.*, the number of hadrons per event). The results for the other observables can be found in App. A. The fit to data in Step 1 is performed using the binned distributions of 13 high-level observables. These observables, already considered in the standard Monash tune, indirectly constrain the hadronization model. In these plots, we compare different distributions that benchmark the performance of the model:

- **Simulation**: Simulated distributions from the baseline model (Pythia).
- **Data**: Experimentally measured distributions, using synthetic data from Pythia with different values of the Lund parameter.
- **Homer**: Distributions reweighted from Simulation using Homer method weights $w_{\text{Homer}}(e_h)$.

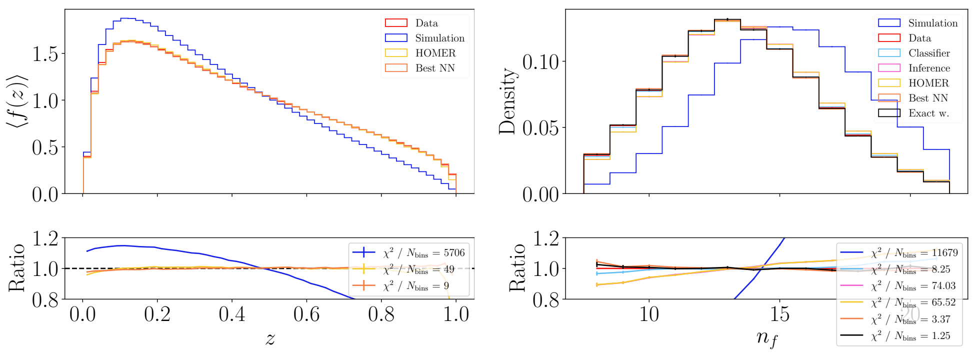
Figure 1: The learned fragmentation function (left panel) and the resulting hadron multiplicity n_f (right panel).

- **Classifier**: Distributions reweighted with Step 1 classifier weights $w_{\text{class}}(e_h)$.
- **Inference**: Distributions reweighted with Step 2 inference weights $w_{\text{infer}}(e_h, \theta)$.
- **Exact Weights**: Simulation distributions reweighted using exact weights, including rejected chains. This represents an upper limit on HOMER's fidelity.
- **Best NN**: The neural networks g_1 and g_2 are directly trained on a dataset composed of individual emissions to learn the true single emission weights. This represents a more realistic upper limit for HOMER, which incorporates biases from architectural choices in g_1 and g_2.

We explicitly quantify the agreement between Data and all the other distributions per observable $\vec{x}$, both by plotting the ratio between distributions and computing a goodness-of-fit metric χ^2/N_{bins} similar to 3.

The results collected in Figs. 1,2,3 demonstrate that the HOMER method is able to reweigh base distributions to provide a good approximations to the Data distributions, although not quite at the level of the optimal Exact Weights. This difference is also present for the results of the two intermediate steps, the Classifier and Inference distributions. The Classifier distributions (Step 1 weights) demonstrate the highest fidelity, while there is a slight decrease in performance when progressing to Step 2. This reduction arises from imperfections in the fragmentation function learned in Step 2, which then translates to a modest loss of the reweighing performance. Note, however, that the differences between the reweighted distributions and data are at a level that is expected to be below other sources of uncertainties in any realistic experimental analysis.

The resulting HOMER average fragmentation function is shown in the left panel of Fig. 1, where we compare it to the "Data" and "Best NN" average fragmentation functions. We observe how the functions agree at the percent-level, indicating how the inverse problem of hadronization is in principle solvable. Even better results can be obtained using event-by-event information, as shown in [12].

4 Conclusions and outlook

In this paper, we presented the HOMER method for learning the fragmentation function $f(z)$ directly from data by reweighting hadron emissions generated with a baseline PYTHIA simulation. The method successfully reproduced the true fragmentation function with high accuracy, achieving percent-level agreement. The accuracy with binned observables is well within expected experimental systematic errors, making this approach suitable for real-world applications. More details about the method can be found in [12], while the public code for this work can be found at https://gitlab.com/uchep/mlhad in the `HOMER/` subdirectory.

Looking ahead, we envision extensions of the HOMER method such that it can be used also for more complex fragmentation processes, such as $q\bar{q}$ strings with gluons, the full flavor structure of PYTHIA, and hadron decays.

Acknowledgments. AY, JZ, MS, MW, PI, SM, and TM acknowledge support in part by NSF grants OAC-2103889, OAC-2411215 and OAC-2417682. AY, JZ, MS, and TM are also in part funded by DOE grant DE-SC101977. JZ acknowledges support in part by the Miller Institute for Basic Research in Science, University of California Berkeley. AY acknowledges support in part by The University of Cincinnati URC Graduate Support Program. SM is supported by the Fermi Research Alliance, LLC under Contract No. DE-AC02- 07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. CB acknowledges support from Vetenskapsrådet contracts 2016-05996 and 2023-04316. MW and PI are also supported by NSF grant NSF-PHY-2209769. This work is supported by the Visiting Scholars Award Program of the Universities Research Association. This work was performed in part at Aspen Center for Physics, which is supported by NSF grant NSF-PHY-2210452.

Broader Impact

A data-driven hadronization model is expected to have a significant impact on a large range of collider experiments, allowing for more accurate theoretical predictions while also providing checks on theoretical assumptions such as factorization and universality. Moreover, resolving a non-trivial inverse problem with the help of empirical forward simulations is a general problem beyond the specifics of particle physics and we expect HOMER to be impactful in other areas where a compromise between model bias and fit quality is needed.

References

[1] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Peter Z. Skands. An introduction to PYTHIA 8.2. *Comput. Phys. Commun.*, 191:159–177, 2015. doi: 10.1016/j.cpc.2015.01.024.

[2] Bo Andersson, G. Gustafson, G. Ingelman, and T. Sjostrand. Parton Fragmentation and String Dynamics. *Phys. Rept.*, 97:31–145, 1983. doi: 10.1016/0370-1573(83)90080-7.

[3] Bo Andersson. The Lund model. *Camb. Monogr. Part. Phys. Nucl. Phys. Cosmol.*, 7:1–471, 1997.

[4] Richard D. Field and Stephen Wolfram. A QCD Model for e+ e- Annihilation. *Nucl. Phys. B*, 213:65–84, 1983. doi: 10.1016/0550-3213(83)90175-X.

[5] Thomas D. Gottschalk. An Improved Description of Hadronization in the {QCD} Cluster Model for e^+e^- Annihilation. *Nucl. Phys. B*, 239:349–381, 1984. doi: 10.1016/0550-3213(84)90253-0.

[6] B.R. Webber. A QCD Model for Jet Fragmentation Including Soft Gluon Interference. *Nucl. Phys. B*, 238:492–528, 1984. doi: 10.1016/0550-3213(84)90333-X.

[7] Phil Ilten, Tony Menzo, Ahmed Youssef, and Jure Zupan. Modeling hadronization using machine learning. 3 2022.

[8] Aishik Ghosh, Xiangyang Ju, Benjamin Nachman, and Andrzej Siodmok. Towards a deep learning model for hadronization. *Phys. Rev. D*, 106(9):096020, 2022. doi: 10.1103/PhysRevD.106.096020.

[9] Jay Chan, Xiangyang Ju, Adam Kania, Benjamin Nachman, Vishnu Sangli, and Andrzej Siodmok. Fitting a Deep Generative Hadronization Model. 5 2023.

[10] Christian Bierlich, Phil Ilten, Tony Menzo, Stephen Mrenna, Manuel Szewc, Michael K. Wilkinson, Ahmed Youssef, and Jure Zupan. Towards a data-driven model of hadronization using normalizing flows. 11 2023.

[11] Jay Chan, Xiangyang Ju, Adam Kania, Benjamin Nachman, Vishnu Sangli, and Andrzej Siodmok. Integrating Particle Flavor into Deep Learning Models for Hadronization. 12 2023.

[12] Christian Bierlich, Phil Ilten, Tony Menzo, Stephen Mrenna, Manuel Szewc, Michael K. Wilkinson, Ahmed Youssef, and Jure Zupan. Describing hadronization via histories and observables for monte-carlo event reweighting, 2024. URL `https://arxiv.org/abs/2410.06342`.

[13] A. Rogozhnikov. Reweighting with Boosted Decision Trees. *J. Phys. Conf. Ser.*, 762(1):012036, 2016. doi: 10.1088/1742-6596/762/1/012036.

[14] Anders Andreassen and Benjamin Nachman. Neural Networks for Full Phase-space Reweighting and Parameter Tuning. *Phys. Rev. D*, 101(9):091901, 2020. doi: 10.1103/PhysRevD.101.091901.

[15] Sascha Diefenbacher, Engin Eren, Gregor Kasieczka, Anatolii Korol, Benjamin Nachman, and David Shih. DCTRGAN: Improving the Precision of Generative Models with Reweighting. *JINST*, 15(11):P11004, 2020. doi: 10.1088/1748-0221/15/11/P11004.

[16] Krish Desai, Benjamin Nachman, and Jesse Thaler. Moment Unfolding. 7 2024.

[17] Bo Andersson. *The Lund Model*, volume 7 of *Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology*. Cambridge University Press, 7 2023. ISBN 978-1-00-940129-6, 978-1-00-940125-8, 978-1-00-940128-9, 978-0-521-01734-3, 978-0-521-42094-5, 978-0-511-88149-7. doi: 10.1017/9781009401296.

[18] Peter Skands, Stefano Carrazza, and Juan Rojo. Tuning PYTHIA 8.1: the Monash 2013 Tune. *Eur. Phys. J. C*, 74(8):3024, 2014. doi: 10.1140/epjc/s10052-014-3024-y.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[20] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019.

A Results for All Observables

In this appendix, we present the results for all 13 high-level observables used in the analysis. These results provide a comprehensive comparison of the distributions obtained from the baseline PYTHIA simulation, the "data" (derived from PYTHIA using a different parameter set), and the output of the HOMER method. Intermediate results from Step 1 (classifier reweighting) and Step 2 (inference) are also included for comparison.

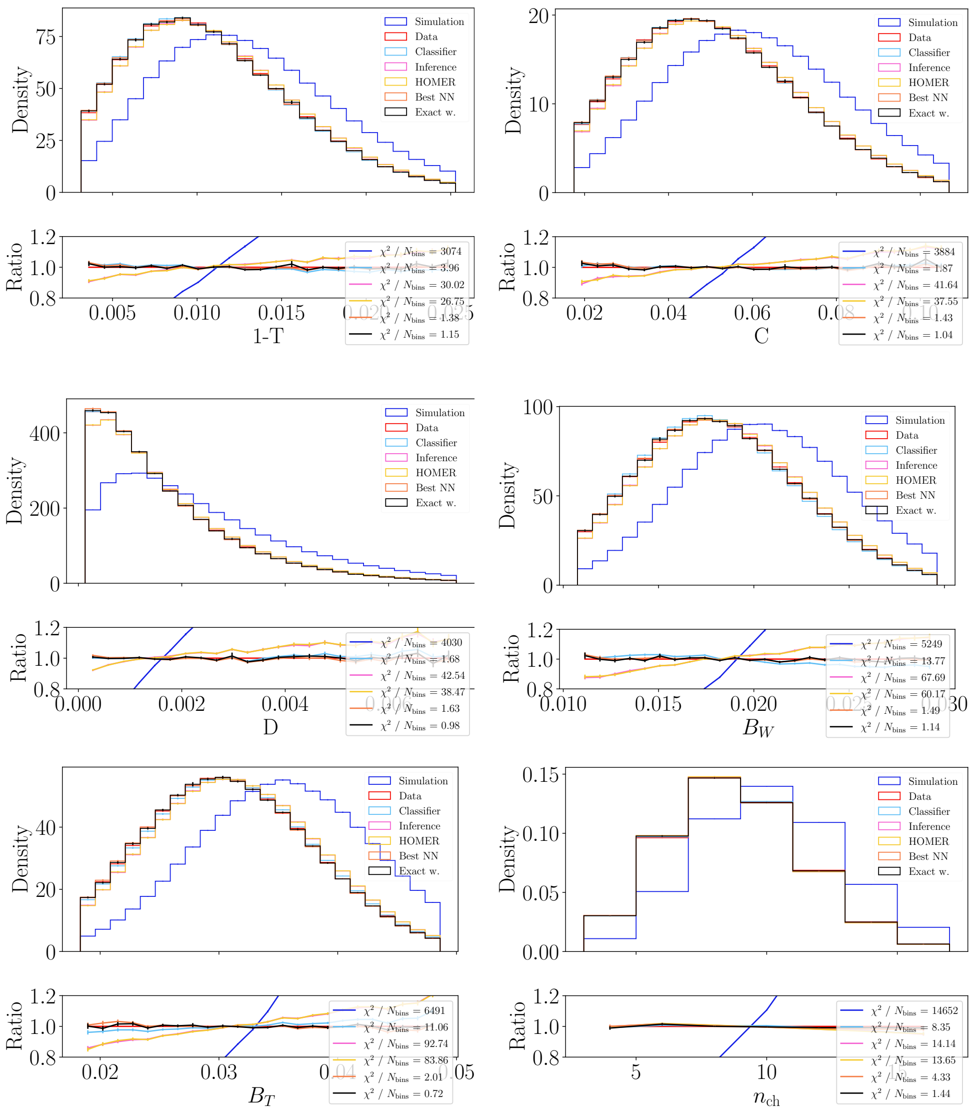

Figure 2: Distributions of high-level observables $1-T$, C, D, B_W, B_T and n_{ch} for the case where Step 1 of the HOMER method is performed on binned high-level observables.

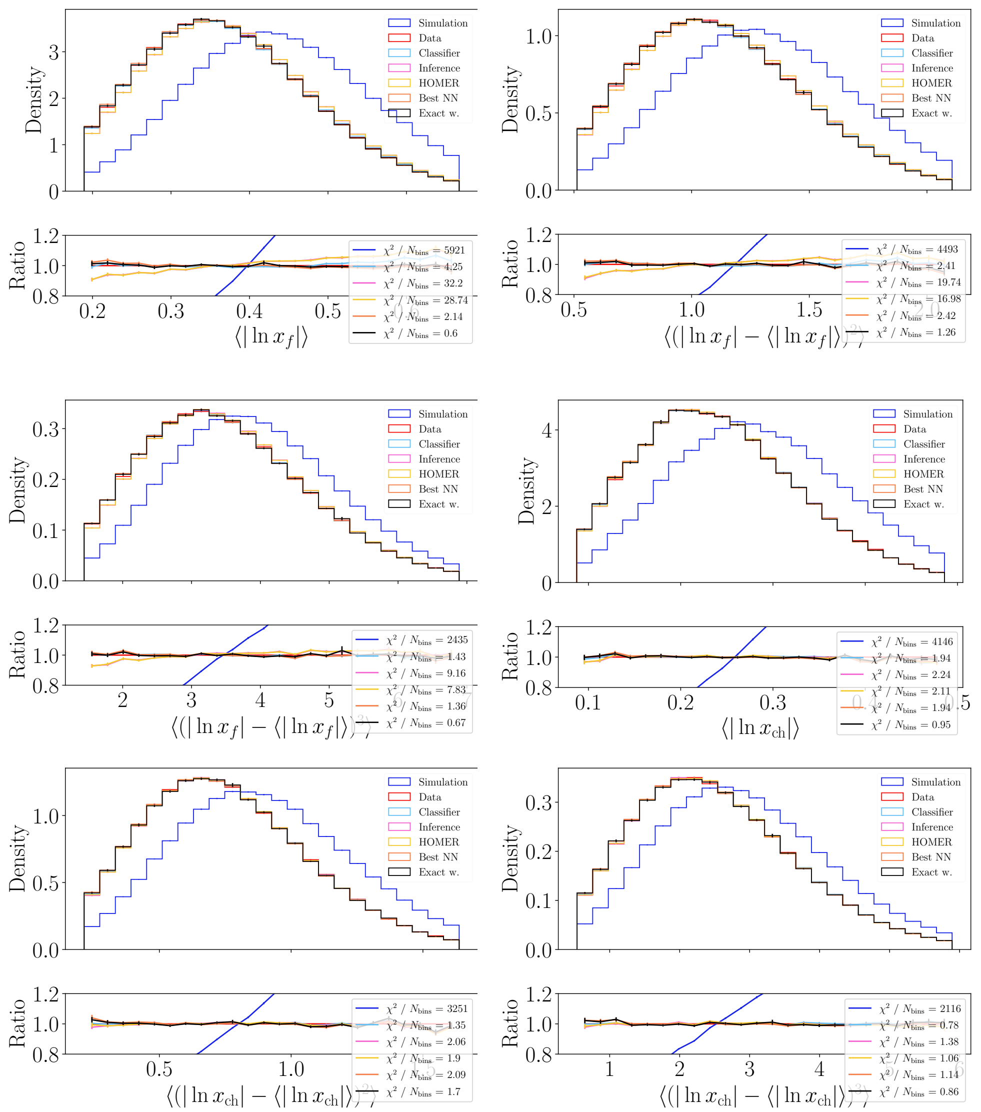

Figure 3: The distributions of the first three moments of $\ln x$, where $x = 2|\vec{p}|/\sqrt{s}$, for both hadron and charged hadron distributions for each event.