Conditional deep generative models for simultaneous simulation and reconstruction of entire events

Etienne Dreyer,[1] Eilam Gross,[1] Dmitrii Kobylianskii,[1,*] Vinicius Mikuni,[2] and Benjamin Nachman[2,†]
[1]*Weizmann Institute of Science, Rehovot, Israel*
[2]*Lawrence Berkeley National Laboratory, Berkeley, California 94720, USA*

(Received 15 April 2025; accepted 23 October 2025; published 9 February 2026)

We extend the particle-flow neural assisted simulations (Parnassus) framework of fast simulation and reconstruction to entire collider events. In particular, we use two generative artificial intelligence tools, continuous normalizing flows and diffusion models, to create a set of reconstructed particle-flow objects conditioned on truth-level particles from CMS Open Simulations. While previous work focused on jets, our updated methods now can accommodate all particle-flow objects in an event along with particle-level attributes like particle type and production vertex coordinates. This approach is fully automated, entirely written in Python, and GPU-compatible. Using a variety of physics processes at the LHC, we show that the extended Parnassus is able to generalize beyond the training dataset and outperforms the standard, public tool Delphes.

DOI: 10.1103/14ph-482n

I. INTRODUCTION

Particle physics is in a precision era where experiments can be digitally emulated from subnuclear scales all the way to the macroscopic size of detectors. Simulated datasets are an integral part of interpreting complex experimental data. Reconstruction algorithms reduce the dimensionality of the inference task by combining real or simulated measurements on many detector components into estimated particle properties. These properties are then the basis for a multitude of downstream analysis tasks. As detectors become more complex and datasets become ever larger, the computational challenges of first-principles simulations and state-of-the-art reconstruction algorithms will become prohibitive [1].

To address this challenge, a suite of fast simulation programs have been developed. Experimental collaborations have developed tailored approaches like those used by ATLAS [2–5] and CMS [6–8] at the LHC. While many pieces of these programs use classical sampling methods, there is a growing interest in generative artificial intelligence (GenAI) for the slowest parts of the simulations, notably calorimeters [9]. These fast surrogate models mimic the output of full detector simulations and are thus processed using the same reconstruction algorithms as the data. While this approach offers dramatic speedup for simulation, it does not alleviate the significant computational cost of reconstruction [10,11].

Outside of experimental collaborations, generic tools like Delphes [12–14] have combined fast simulation with fast reconstruction. While widely used and highly impactful for phenomenological studies, the precision of Delphes is fundamentally limited because it operates by smearing input particles and the smearing functions are hard-coded based on published performance results. Recently, there have been proposals to address these and other related challenges with GenAI. FlashSim [15,16] uses GenAI to generate reconstructed jets and other high-level objects conditioned on the corresponding particle-level inputs. Parnassus [17–19] uses similar tools, operating at the level of individual (reconstructed) particles. These approaches are automatically tunable, can accommodate nontrivial reconstruction effects, and are additionally lightweight, having only Python dependencies and being natively GPU-compatible. Parnassus is proposed to be a general-purpose tool that has a similar output as Delphes. Previous studies with Parnassus have focused on hadronic jets, which have many constituent particles with complex substructure. However, a complete fast simulation and reconstruction tool should also be able to accommodate entire events.

In this paper, we extend Parnassus to full events. While we focus on collider physics, the methods can be applied to any experiment whose data are in an event format and are composed of (reconstructed) particles. While much of the complication from collider events at high energy originates with jets, there are new challenges that arise when extending the model to full events. For example, entire events

*Contact author: dmitry.kobylyansky@weizmann.ac.il
†Contact author: bpnachman@lbl.gov

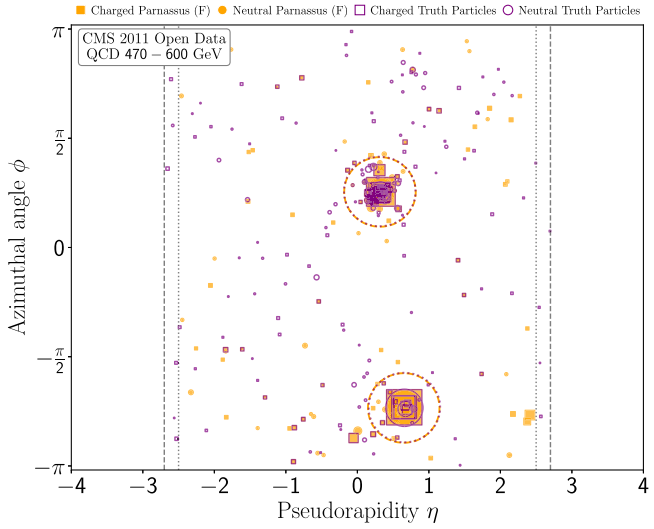


FIG. 1. An event display of a single dijet event from the CMS Open Simulation. The two highest-p_T jets are illustrated with dashed and dotted circles, while all detector-stable and reconstructed particles are shown with markers. The size of the marker reflects the particle energy. The dashed vertical lines denote the training cut $|\eta| < 2.7$, and dotted lines denote the tracker acceptance area $|\eta| < 2.5$. While the Parnassus and the CMS Open Simulation outputs are expected to differ in individual events due to the stochastic nature of the detector response, this display is a qualitative demonstration of Parnassus's ability to capture the full event.

have no obvious anchor to center all of the particles (like the jet axis). Similarly, the identification of $-\pi$ and $+\pi$ in azimuthal angle is now relevant. Furthermore, the overall multiplicity increases with a wide range of energy and length scales that need to be modeled. We address these challenges by extending the neural network models from our previous works. Figure 1 presents a representative event display, with many more detailed results available later in the manuscript. Both Delphes and FlashSim are also capable of modeling full events, but the former uses per-particle, hard-coded smearing functions and the latter is only able to model high-level features of reconstructed objects and not individual particle-flow candidates.

This paper is organized as follows. Section II introduces the CMS Open Simulation dataset for our study. The details of the machine learning methods are introduced in Sec. III. We explore two machine learning approaches for full event generation. Numerical results are presented in Sec. IV, comparing the two methods. The paper ends with conclusions and outlook in Sec. V.

II. DATASET

We use a suite of physics processes in order to build and test the universality of our model. Selected processes include $t\bar{t}$, $gg \to H \to 4\ell$, and QCD $2 \to 2$, as these were available with high statistics in the 2011 CMS Open Simulation. Events were generated with Pythia 6.4.25 [20] using the CMS detector simulation based on Geant4 [21] and then the events were reconstructed using the CMS particle-flow algorithm [22]. We preprocessed these datasets from the AODSIM format, keeping the set of particle-flow candidates (PFCs) and generator-level final-state truth particles (GENs).[1] Each GEN or PFC particle is described by its transverse momentum, pseudorapidity, azimuthal angle, production vertex coordinates, and class ($p_T, \eta, \phi, \vec{v}$, class). Particles are categorized into one of five classes: charged hadron, electron, muon, neutral hadron, or photon.

TABLE I. Information about used Monte Carlo event samples. Usage and parton-level p_T ranges for QCD samples are shown.

Dataset	$p_T^{min} - p_T^{max}$ [GeV]	Type	Training	Testing
$H \to 4\ell$ [25]	⋯	Out-of-distribution		✓
$t\bar{t}$ [26]	⋯	In-distribution	✓	✓
QCD [27]	470–600	In-distribution	✓	✓
QCD [28]	600–800	Out-of-distribution		✓

To constrain our model within the tracker acceptance area ($|\eta| < 2.5$), we apply a relaxed pseudorapidity cut of $|\eta| < 2.7$ for both PFC and GEN particles. During performance evaluation, we enforce a strict cut of $|\eta| < 2.5$. This strategy allows Parnassus to learn the impact of truth particles outside the tracker on those inside. To mitigate tracking inefficiencies and improve momentum reconstruction, we follow Refs. [23,24] and require $p_T > 1$ GeV for PFCs. In order to reduce the cardinality of truth particles, we also require them to have $p_T > 0.25$ GeV.

We used a total of four datasets, with two dedicated to model training and in-distribution performance evaluation and the remaining two for out-of-distribution checks (see Table I). Each training dataset consisted of 1.4×10^6 events, while each testing dataset comprised 50,000 events, resulting in a total of 2.8×10^6 events for training and 200,000 events for testing.

The goal of Parnassus is to take as input the GEN particles and produce PFCs so that the distribution of PFCs matches that of the full CMS detector simulation and reconstruction. As a baseline for comparison, we also create a dataset with Delphes 3.5. In order to eliminate statistical fluctuations, we use exactly the same GEN events for Delphes by converting the GEN events from the CMS dataset into HepMC3 format [29]. We then pass these events through Delphes with the default CMS detector card. As the CMS datasets do not have truth information for the pileup, the pileup is treated as noise (e.g., the model needs to produce pileup PFCs without a starting GEN pileup particle). To match the CMS setup, we add minimum bias events to the Delphes samples following the procedure in Ref. [30], with the

[1]Detector-stable particles with Pythia 6 status code 1.

mean number of pileup vertices per event set to 6.35. We disable smearing of the truth primary vertex in Delphes in order to match the full simulation, while the pileup vertices are smeared with the default values.

III. METHODS

The core of our approach is a point cloud (set of PFCs) generative model that is conditioned on another point cloud (GEN particles). We consider two state-of-the-art methods for achieving this aim—one based on flow matching [Parnassus (F)] and one based on diffusion models [Parnassus (D)]—as described below.

A. Flow and diffusion

Continuous normalizing flows (CNFs) [31] are a competitive generative modeling method. Like other methods, CNFs map a base probability density p_0 to a target probability density p_1. In the case of CNFs, this mapping is defined by a vector field v_θ, parametrized by a neural network, that solves an ordinary differential equation,

$$\mathrm{d}x = v_\theta(t,x)\mathrm{d}t, \tag{1}$$

with $t \in [0,1]$. A common approach to learning v_θ is through a regression task called flow matching (FM) [32],

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t,p_t(x)}||v_\theta(t,x) - u_t(x)||^2, \tag{2}$$

where $\mathcal{L}$ is the loss function for optimizing the parameters θ of the neural network and the function u_t describes the target probability flow field mapping p_0 to p_1. Since u_t is not generally known in closed form, a surrogate objective is to predict an alternative form, $u_t(x|z)$, using the conditional flow matching (CFM) loss function [32–34],

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t,q(z),p_t(x|z)}||v_\theta(t,x) - u_t(x|z)||^2, \tag{3}$$

where the sample-conditional approach uses $z \sim q(z)$, $x \sim p_t(x|z)$. Yao *et al.* [35] and Ke *et al.* [36] showed that additionally steering the direction of the velocity network using a cosine similarity loss term enhances the model's convergence rate,

$$\mathcal{L}_{\mathrm{sim}}(\theta) = \mathbb{E}_{t,q(z),p_t(x|z)}\left(1 - \frac{v_\theta(t,x)\cdot u_t(x|z)}{|v_\theta(t,x)||u_t(x|z)|}\right). \tag{4}$$

In the previous study [30], we identified the condition z with the single sample x_* (in that case, a set of PFCs in the jet) and the target probability path was defined as

$$\begin{aligned} p_t(x|z) &= \mathcal{N}(x|tx_*, (t\sigma - t + 1)^2\mathrm{I}), \\ u_t(x|z) &= \frac{x_* - (1-\sigma)x}{1-(1-\sigma)t}, \end{aligned} \tag{5}$$

which represents a path between a standard normal distribution and a Gaussian distribution centered at x_* with standard deviation σ, which we took to be equal to 10^{-4}.

In contrast, in this work, we identify z with a pair of random variables, one from the data distribution and the other from a standard normal distribution. The path between them is defined by a family of affine Gaussian probability paths, which can be expressed as

$$\begin{aligned} p_t(x|x_*) &= \mathcal{N}(x|\alpha_t x_*, \sigma_t^2\mathrm{I}), \\ u_t(x|x_*) &= \dot{\alpha}_t x_t + \dot{\sigma}_t \varepsilon, \end{aligned} \tag{6}$$

where $\varepsilon \sim \mathcal{N}(0,\mathrm{I})$ and $\alpha_t, \sigma_t \colon [0,1] \to [0,1]$ are smooth functions of t, satisfying[2]

$$\alpha_0 = \sigma_1 = 1,\ \alpha_1 = \sigma_0 = 0, -\dot{\alpha}_t, \qquad \dot{\sigma}_t > 0 \text{ for } t \in (0,1). \tag{7}$$

This case also subsumes probability paths generated by standard diffusion models [38,39] and the denoising score matching (SM) loss can be obtained from the CFM loss [Eq. (3)] by reparametrization [40],

$$\mathcal{L}_{\mathrm{SM}}(\theta) = \mathbb{E}_{t,\varepsilon,x_t}||s_\theta(t,x_t) - \nabla \log p_t(x_t)||^2, \tag{8}$$

where $\nabla \log p_t(x_t) = -\frac{\varepsilon}{\sigma(t)}$. Salimans and Ho in Ref. [41] proposed another parametrization of SM loss, via v-prediction, allowing more stable training,[3]

$$\mathcal{L}_{\mathrm{v}}(\theta) = \mathbb{E}_{t,\varepsilon,x_t}||\mathrm{v}_\theta(t,x_t) - \mathrm{v}_t(x_t)||^2, \tag{9}$$

where $\mathrm{v} = \alpha_t\varepsilon - \sigma_t x_*$.

Data generation with the trained neural network can be done by solving either a probability flow ordinary differential equation (ODE) or a reverse-time stochastic differential equation (SDE),

$$\mathrm{ODE}\colon\ dx_t = v_\theta(t,x_t)dt, \tag{10}$$

$$\mathrm{SDE}\colon\ dx_t = [f(t)x_t - g^2(t)s_\theta(t,x_t)]dt + g(t)d\bar{w}, \tag{11}$$

where

$$f(t) = \frac{\dot{\alpha}_t}{\alpha_t}, \qquad g^2(t) = -2\frac{\dot{\alpha}_t\sigma_t^2 - \alpha_t\sigma_t\dot{\sigma}_t}{\alpha_t}. \tag{12}$$

The exact choice of affine path, training loss, and sampling procedure is summarized in Table II.

[2]Contrary to Ref. [32] we inverted time, as in Ref. [37].

[3]To avoid confusion with the flow vector field v we denote it by straight v.

TABLE II. Parnassus model's parametrization.

Model type	α_t	σ_t	Loss	Reverse process
Parnassus (F)	$1-t$	t	$\mathcal{L}_{\text{CFM}}+\mathcal{L}_{\text{sim}}$	ODE
Parnassus (D)	$\cos\frac{\pi t}{2}$	$\sin\frac{\pi t}{2}$	$\mathcal{L}_{\text{v}}$	ODE

B. Architecture description

We divide the prediction of PFC properties into two stages. The first stage generates event-level quantities: missing energy, the scalar sum of transverse momenta, and the number of PFCs, represented as $\mathcal{E}^{\text{pf}} = (E_x^{\text{miss}}, E_y^{\text{miss}}, H_{\text{T}}, N_{\text{part}})$. The corresponding GEN quantities are denoted $\mathcal{E}^{\text{gen}}$. The second stage predicts the PFCs and their properties, represented as $\mathcal{P}^{\text{pf}} = (p_{\text{T}}^{\text{rel}}, \eta, \phi, \vec{v}, \text{class})$, where $p_{\text{T}}^{\text{rel}} = \frac{p_{\text{T}}}{H_{\text{T}}}$, $\vec{v}$ is the production vertex, and class is the particle type. The corresponding GEN quantities are called $\mathcal{P}^{\text{gen}}$.

Both stages are conditioned on the set of GEN particles and the GEN-level event quantities. The second stage is additionally conditioned on the PFC-level event quantities generated in the first stage. Both the GEN and PFC particle sets are ordered by $p_{\text{T}}^{\text{rel}}$, and their features are normalized using precomputed means and standard deviations. We consider up to 400 PFCs/GENs, which is rarely exceeded. The class feature is one-hot encoded.

The CFM and diffusion models slightly differ in their setup of these two network components:

CFM Architecture

(i) *Event-level network*

We use a ResNet-like [42] model to parametrize $v_\theta^{\mathcal{E}}$. The network processes noisy event quantities $\mathcal{E}^{\text{pf}}$, sampled according to Eq. (5), as well as a set of truth particles with their features $\mathcal{P}^{\text{gen}}$, along with event scaling information and $\mathcal{E}^{\text{gen}}$ serving as global features.

The model architecture is illustrated in Fig. 2. Initially, the current time step is embedded using sine positional encoding followed by a multilayer perceptron (MLP). Global and truth particle features, together with features of the noisy $\mathcal{E}^{\text{pf}}$, are embedded into the same hidden dimension using separate MLPs. Next, the pooled representation of the truth particles is concatenated with the updated global and time step embeddings to form the context (c). The updated noisy features are then processed through an adaptive-layer normalization (adaLN-zero) procedure [43] using c, followed by a sequence of residual blocks. Finally, $v_\theta^{\mathcal{E}}$ is computed via an MLP conditioned on the context c.

(ii) *PFC-level network*

For the PFC-level network, we employ a transformerlike model, similar to that described in Ref. [30], to parametrize $v_\theta^{\mathcal{P}}$. We also split ϕ into two variables $(\sin\phi, \cos\phi)$ to better model the neighborhood of $\pm\pi$. The network receives inputs consisting of a fixed-length set of PFCs with $\mathcal{P}^{\text{pf}}$ sampled according to Eq. (5), as well as a set of truth particles with $\mathcal{P}^{\text{gen}}$ along with event scaling information, $\mathcal{E}^{\text{gen}}$ and $\mathcal{E}^{\text{pf}}$, serving as global features.

The model architecture, depicted in Fig. 3, begins with an initialization similar to that of the event stage. Following this initialization, the hidden representations of PFCs (x) and GEN particles (y) are passed through multiple cross-attention diffusion transformer (DiT) blocks [30,43], where the context c is used as input to the adaptive-layer normalization (adaLN-zero) [43] in each block.

Diffusion Model Architecture

(i) *Event-level network*

Similarly to the CFM architecture, we break down the generation into two components, resulting in two models. The event-level network uses the same architecture as the CFM model to generate the event-level quantities. The main difference is that instead of generating

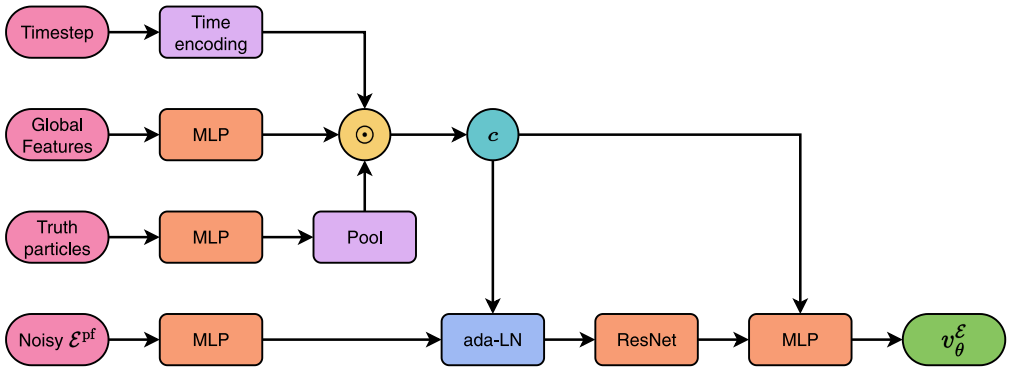


FIG. 2. Architecture of event-level CFM model. Concatenation is indicated by ⊙.

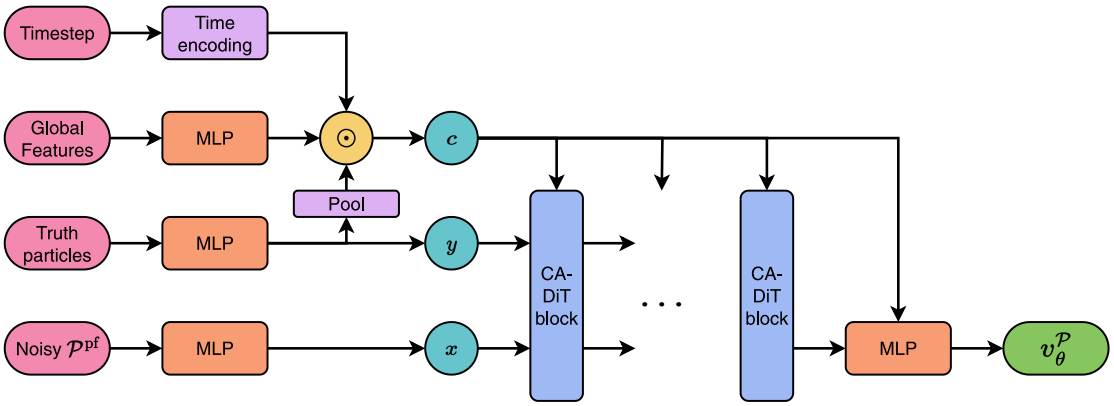


FIG. 3. Architecture of PFC-level CFM model. Concatenation is indicated by ⊙.

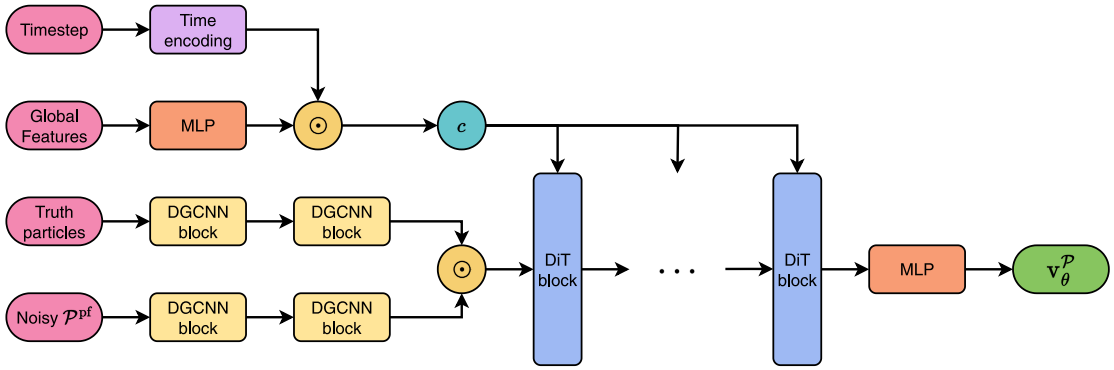


FIG. 4. Architecture of PFC-level diffusion model. Concatenation is indicated by ⊙.

the overall particle multiplicity as a single number, we generate instead the individual multiplicities corresponding to each particle class, with the overall multiplicity to be generated calculated from the sum over classes. The advantage of this strategy is that we do not need to individually generate the particle class using the PFC-level network, but instead we can already condition the generation of particle-flow candidates based on the expected class.[4] The drawback of this approach, as we see later in the results, is that the overall multiplicity per particle identification becomes less accurate. Since the number of features to generate increases, we also increase the internal network representation compared to the CFM event-level network, resulting in more trainable parameters.

(ii) *PFC-level network*

The generation of particles is conceptually similar to the CFM model, where we create a graph- and transformer-based architecture that takes as inputs the information from generator-level particles and event information to guide the diffusion process toward the generation of reconstructed particle candidates. In this case, the architecture used is inspired by OmniLearn [44,45]. The time step and global event information are initially embedded using the same strategy as in the CFM model. The particle information at generation and reconstruction levels is first passed through two dynamic graph convolutional neural network [46] blocks, where k-nearest neighbors with $k = 3$ is used to aggregate the information of nearby particles in $\eta - \phi$ space. This new embedding is then used as inputs to four DiT [43] blocks, where c is again used as inputs to the adaLN operations. The resulting architecture is shown in Fig. 4.

The networks for the CFM training are implemented in PyTorch [47], while the diffusion model is implemented using TensorFlow [48]. Hyperparameters for all models are

[4]To be more explicit, during generation, if we need to generate two electrons, we concatenate the initial Gaussian values of two arbitrary particles with the one-hot encoded value assigned to electrons, requiring the model to generate only the remaining kinematic properties through the reverse diffusion process.

TABLE III. Network hyperparameters of the Parnassus models.

Hyperparameters	CFM	Diffusion
Batch size	128	64
Optimizer	Lion [50]	Lion [50]
Weight decay	0.01	0.01
Max. learning rate	1.2×10^{-4}	0.6×10^{-4}
Min. learning rate	10^{-5}	10^{-7}
Gradient norm clip	1.0	0.0
No. of epochs	70	300
Sampling method	Flow-DPM-Solver [51]	DPM-Solver-2 [52]
No. of time steps (model evaluations)	40 (40)	64 (128)
Max output particles	400	400
Trainable parameters of event-level network	470,532	1,377,352
Trainable parameters of PFC-level network	4,864,427	4,101,703

shown in Table III. The learning rate is modified according to the cosine annealing schedule [49]. We also note that for the CFM model, the number of time steps can be reduced to 25 almost without losing quality.

Table IV compares the timing for the different approaches. We did not specifically optimize for the time, so there is room for improvement. All models are much faster than a full detector simulation and reconstruction [$\mathcal{O}$(min)].

IV. RESULTS

We evaluate the performance of Parnassus on both event-level and jet-level quantities. Jets are clustered using FastJet [53] with the anti-k_t algorithm [54] and $R = 0.5$. To investigate the quality of jet properties, we match PFC jets to GEN jets using the Hungarian matching algorithm [55], with ΔR as a cost function. Since the Hungarian algorithm considers all possible matching, we additionally forbid matches with $\Delta R > 0.2$. To investigate the quality of PFC property generation, we split the PFCs into three sets: all particles, particles assigned to jets, and particles not assigned to jets (henceforth, "background particles"). Inside each category, we additionally perform the same Hungarian-based matching between PFC and GEN particles, forbidding matches with $\Delta R > 0.6$.

TABLE IV. Generation time per shower in seconds for different batch sizes for the $t\bar{t}$ dataset. The average and standard deviation are given for 10 runs per batch size. The GPU studies were run on an NVIDIA A100 with 40 GB VRAM. For Delphes, the time is averaged over 10,000 generated events.

	CFM		Diffusion		Delphes
Batch size	GPU	CPU	GPU	CPU	CPU
1	0.669(3)	4.38(36)	2.784(68)	···	0.0112
10	0.0734(2)	1.59(10)	0.9214(3)	···	···
100	0.0147(1)	1.29(2)	0.769	···	···
1000	0.0136(1)	···	OOM	···	···

We present results at three levels: event, jet, and particle. First, we examine the performance of Parnassus for the processes on which it was trained. For this purpose, we use a hold-out test set that is statistically identical to the training data. Second, we consider the processes that the model did not see during the training.

Figure 5 presents a representative snapshot of results across metrics. We dive into further detail in the subsequent discussion.

A. Event-level performance

As event-level metrics, we consider the number of particles, number of jets, missing transverse energy in the x and y directions, and the scalar sum of particle p_T. The spectra of these observables are presented in Fig. 6. As all subsequent figures follow a similar pattern, we will describe this figure in detail here and then refrain from doing so for the following figures. Each column of Fig. 6 corresponds to a different observable and each row represents a different physics process. The second and third rows show processes that were part of the training, although the results shown are from independent test sets. The first and final rows represent processes that were not part of the training. All histograms are normalized to unity. The filled histograms show the CMS full simulation and reconstruction—the target for both Parnassus and Delphes. The two variants of Parnassus are indicated with an (F) for the flow matching and (D) for diffusion. The goal is for the unfilled histograms to be as close as possible to the filled histograms and for this agreement to span columns (observables) and rows (universality). For the event-level observables, we can see that $H \to 4l$ has the lowest particle and jet multiplicity as well as the narrowest missing energy distribution. In contrast, all of the other processes have multiple, hard jets (more in $t\bar{t}$ than for generic QCD dijets). The Parnassus methods agree well with CMS across all processes and observables, while Delphes is accurate for most of the distributions, but not for the particle multiplicity and H_T.

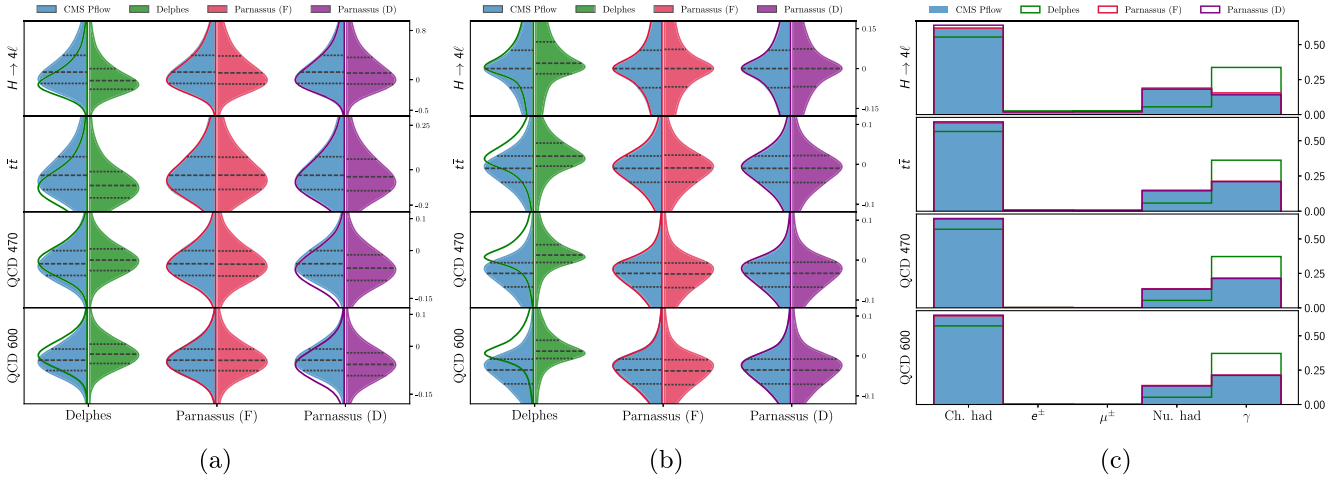


FIG. 5. Three metrics summarizing the performance of the target CMS full simulation in blue compared to Delphes in green and Parnassus in red (flow matching) and purple (diffusion). (a) Violin plots of the fractional residual for the scalar sum of all particles p_{T} in the event (H_{T}), (b) violin plots of the fractional residual for the leading jet C_2 jet substructure observable, and (c) histograms of the reconstructed particle types. Residuals are computed using the difference between the truth and the reconstructed quantities from each algorithm. In all cases, the red and purple distributions agree well with the blue, better than the green ones do. (a) H_{T} residuals. (b) Jet C_2 residuals. (c) Reconstructed particle classes.

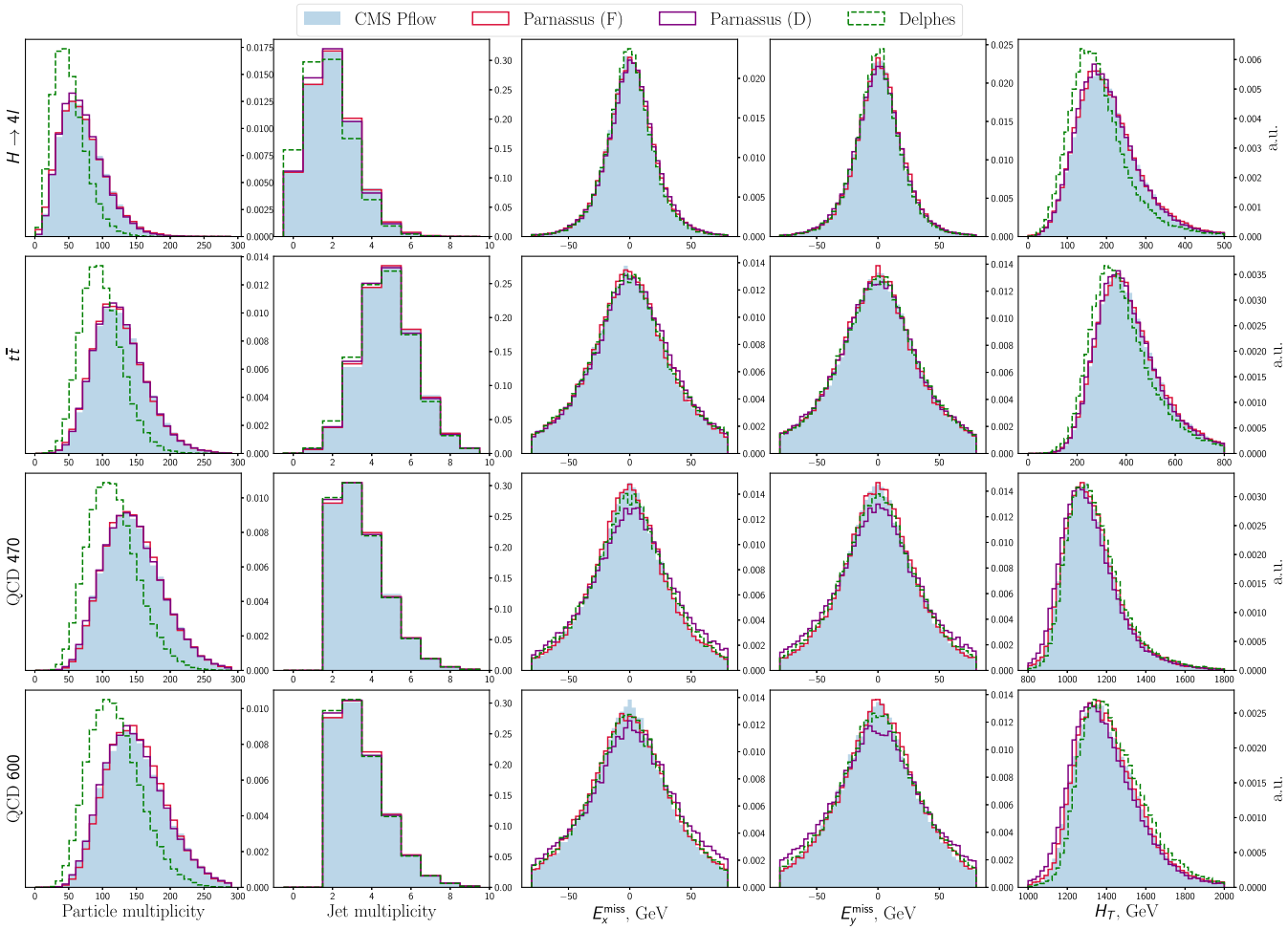


FIG. 6. Histograms of event-level features for $t\bar{t}$, $H \to 4l$, and QCD datasets. Rows correspond to different processes while columns correspond to different observables. All histograms are normalized to unity.

The spectra in Fig. 6 do not tell the full story, since many of these distributions are determined by the particle-level distributions and not by resolution effects. Figure 7 shows the residuals of the observables from Fig. 6, to remove the dependence on the truth quantities (which are the same for all methods). For the discrete quantities (particle and jet multiplicities), the residuals are the difference between the true and reconstructed quantities. For the continuous quantities, the residual is the difference divided by the truth value. The mismodeling in the particle multiplicity observed in Fig. 6 is even more pronounced in the residual. This is also true for H_{T}. As the total particle multiplicity (and the number of jets in $H \to 4l$) is highly sensitive to detector effects beyond just smearing (e.g., fakes), which are not explicitly modeled by Delphes, it is not surprising that these are the observables where the relative advantages with GenAI are most pronounced.

B. Jet-level performance

For the jet-level performance we explore the kinematic properties $(p_{\mathrm{T}}, \eta, \phi)$ as well as substructure quantities, C_2 [56] and D_2 [57]. These latter two observables characterize how consistent the radiation pattern within a jet is with a two-prong hypothesis compared to a one-prong origin. They are widely used for boosted $W/Z/H$ boson tagging. Since we want to accurately reproduce all jets in the event, including low-p_{T} ones, we include all jets in the following figures, except for the jet substructure observables, where we restrict to the leading jets. Figure 8 shows spectra of the

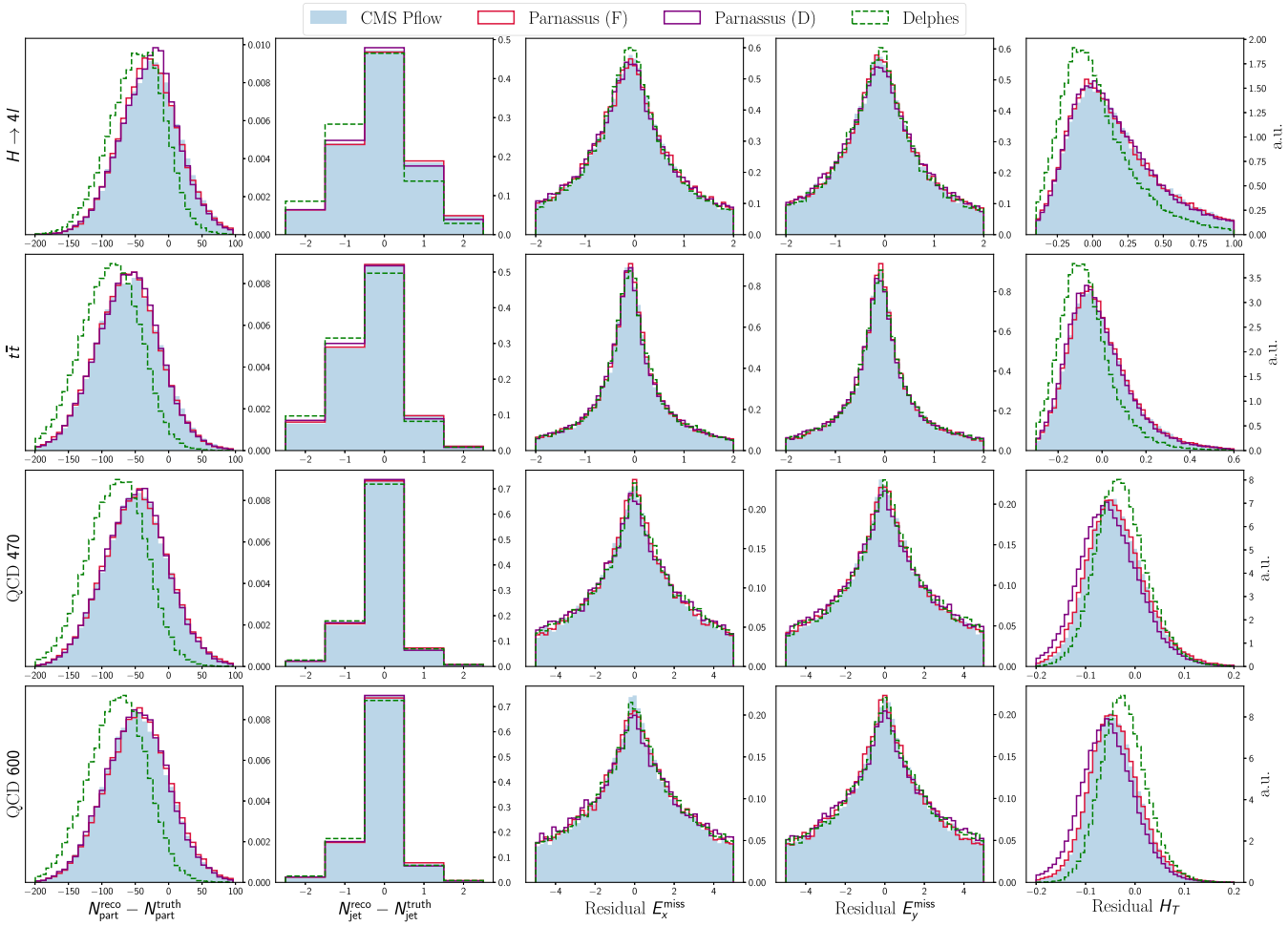


FIG. 7. Histograms of the residuals of event-level features for $t\bar{t}$, $H \to 4l$, and QCD datasets. Rows correspond to different processes, while columns correspond to different observables. All histograms are normalized to unity.

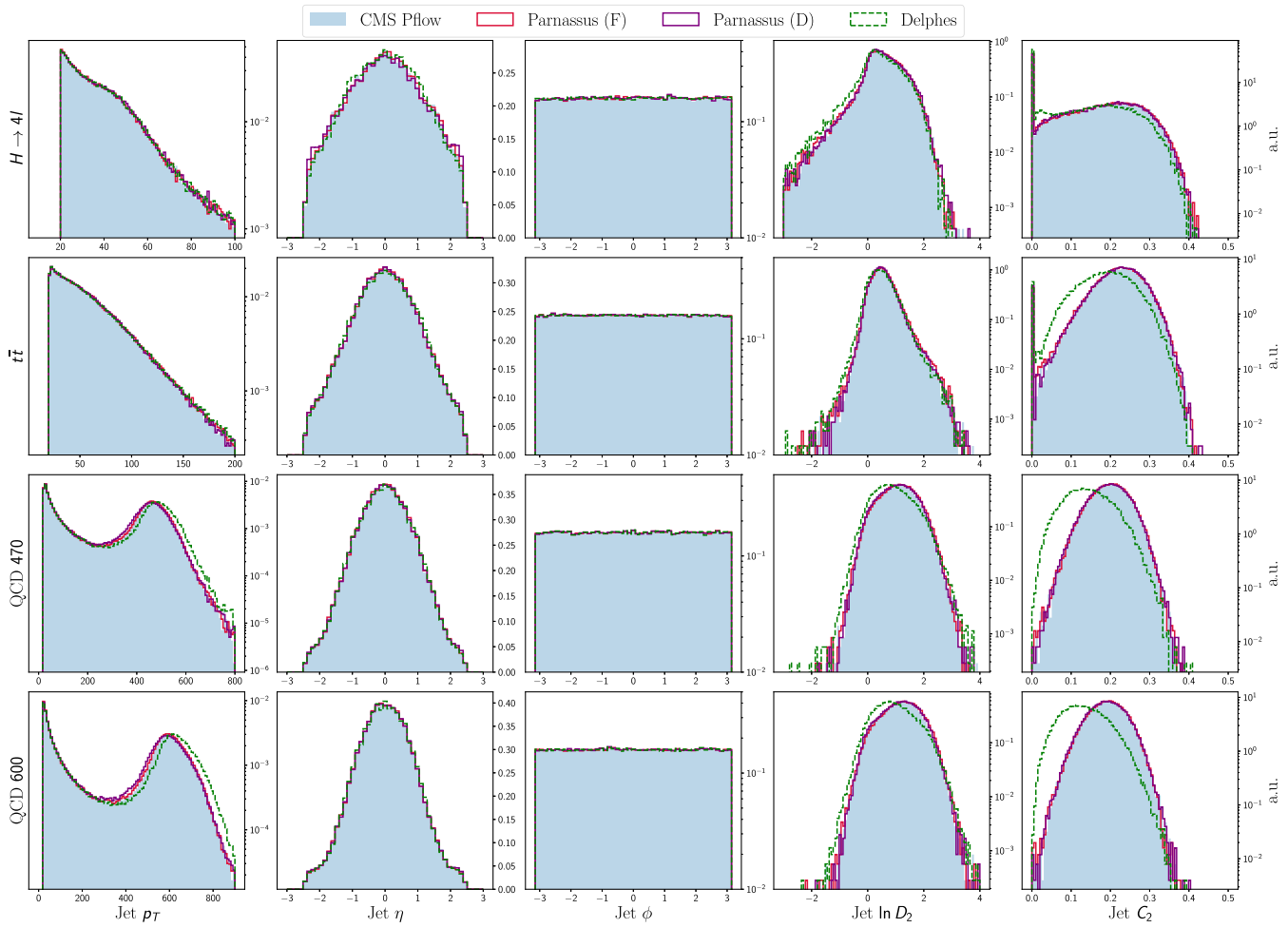


FIG. 8. Histograms of jet-level features for $t\bar{t}$, $H \to 4l$, and QCD datasets. Rows correspond to different processes while columns correspond to different observables. All histograms are normalized to unity.

jet-level quantities. The jet p_T is steeply falling for the Higgs boson and top quark pair processes and bimodal for QCD. This is because the QCD events are composed of two high-p_T jets (the second peak) in addition to a number of additional jets superimposed (first peak). The jet kinematic properties are mostly determined by the particle-level quantities and agree well across methods. In contrast, the jet substructure observables are subject to significant detector effects and are much more accurate for Parnassus compared to Delphes.

The corresponding residuals for jets are shown in Fig. 9. These residuals are calculated between PFC jets and the matched GEN jet. Jets without a match are not included here. Across the board, Delphes presents a narrower resolution than the CMS response, which is well captured by both Parnassus models. For the angular coordinates, the difference between Delphes and the other models is a pure variance effect, while there is also a bias for the other observables. This is most prominent for C_2. The Parnassus models also capture the improvement in resolutions of kinematic properties with p_T, seen most clearly by comparing the third and fourth rows of Fig. 9.

C. Particle-level performance

The overall spectra of PFCs are presented in Fig. 10. The details of the varying detector subsystems are noticeable in the η distribution of particles, which is unimodal at particle level, but develops peaks at transition regions at detector level. As the detector model in Delphes has the same transition regions as the real CMS detector, the locations of these features are well modeled, even if the fine details of the spectrum at those locations is not as precise as Parnassus. All fast simulation methods average over the small

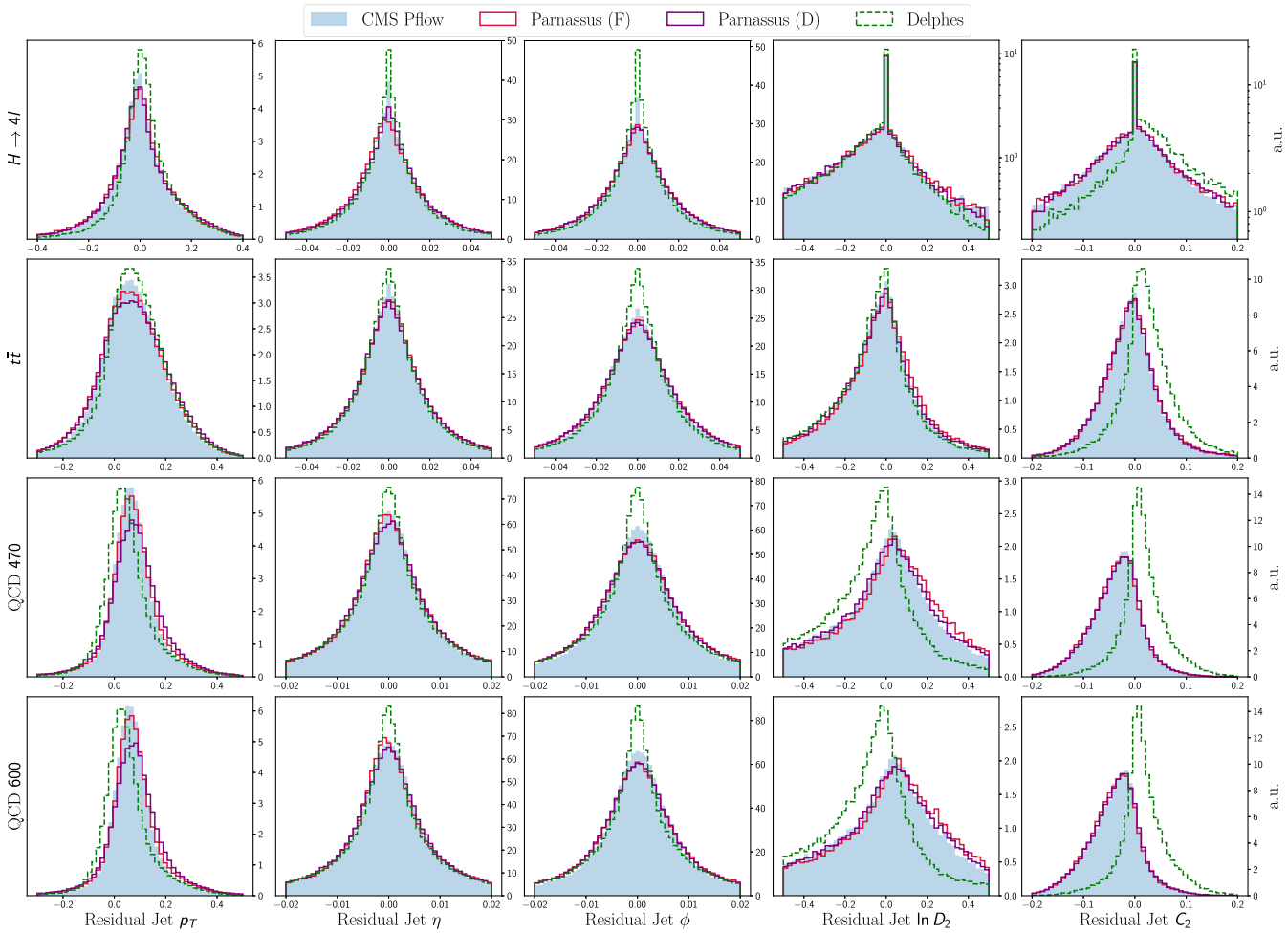


FIG. 9. Histograms of residuals of jet-level features for $t\bar{t}$, $H \to 4l$, and QCD datasets. Rows correspond to different processes while columns correspond to different observables. All histograms are normalized to unity.

modulations in ϕ, which are only barely visible even with the fine binning chosen for Fig. 10. The diffusion model is more accurate at high η compared to the flow model, while the flow model is more accurate at central η compared to the diffusion model. The vertex information is used for pileup rejection and flavor tagging—including this low-level information allows for these quantities to be included in downstream high-level taggers. Delphes is not designed to precisely model the resolution in v_x and v_y, but is tuned to approximately capture the spread in v_z. In contrast, both Parnassus models reproduce all of the properties of the CMS data. The performance of Parnassus in the first and last rows is strong evidence that the models are learning universal properties of the detector response, as these events are quite different from those seen during training.

For PFCs that are matched to GEN particles, the relative residual distributions are presented in Fig. 11. The particle p_T residual shows a narrow peak around zero and then a broad shoulder on both sides of this peak. Since we do not track energy deposits through the detector, it is only possible to match PFC and GEN particles based on geometry and energy. We interpret the shoulders in the first column of Fig. 11 as actually corresponding to either truly unmatched particles that happen to be "close" or cases where a GEN particle created more than one PFC or one PFC is due to more than one GEN particle. This can result from material interactions, confusion in reconstruction, or the finite granularity of the detector. As in the jet case, the resolutions from Delphes tend to be narrower than CMS, while the Parnassus models seem to capture both the bias and variance of the resolution functions, for both in-distribution and out-of-distribution events. As the tracker and calorimeter granularity are similar in η and ϕ, the resolutions in the two angular coordinates are similar. The p_T dependence for

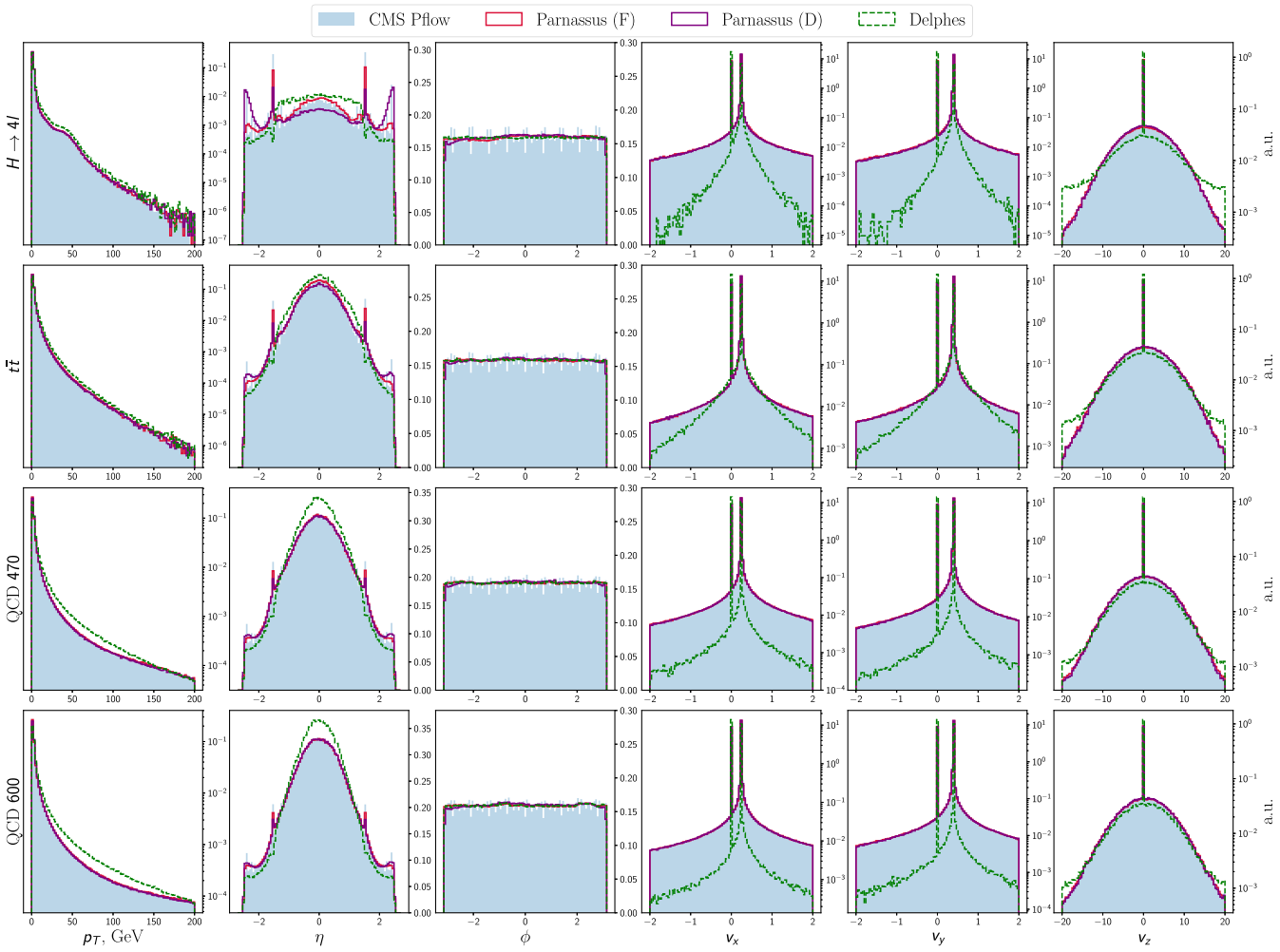


FIG. 10. Distributions of particle features for $t\bar{t}$, $H \to 4l$, and QCD datasets. Rows correspond to different processes while columns correspond to different observables. All histograms are normalized to unity.

the energy resolution is not as prominent as for the angles because of the tradeoff between the tracker resolution that grows with energy and the calorimeter resolution that decreases with energy.

Next, we show the performance of the particle class reconstruction in Fig. 12. Possible options include charged hadrons (Ch. had), electrons, muons, neutral hadrons (Nu. had), and photons. The first four columns of the figure depict confusion matrices between truth particles (vertical axis) and reconstructed particles (horizontal axis). The matching is the same as in Fig. 10 and carries the same limitations (e.g., there is no unique match—charged PFCs can be matched to neutral GENs, etc.). The goal of the fast simulation methods is to match the CMS performance, not to make the matrix as diagonal as possible. The Parnassus models reproduce all of the trends well, while the Delphes simulation is much more diagonal than CMS. The different properties of the four event types are also visible in the matrices. For example, the probability of an event containing an electron is small except in Higgs and $t\bar{t}$ events, where electrons are part of the hard scatter. Interestingly, Parnassus accurately models the muon classification accuracy (about 30%) even in the QCD events, where the muons are mostly from decays-in-flight from light- or heavy-flavor hadrons. The last column of Fig. 12 contains histograms of the frequency for reconstructing particles of a given type.

These last sets of plots have focused on inclusive PFCs matched to GEN particles. It is also interesting to see if the performance is the same for particles inside jets (the subject of Ref. [19]) and outside jets ("background"). Figures corresponding to the previous figures for these

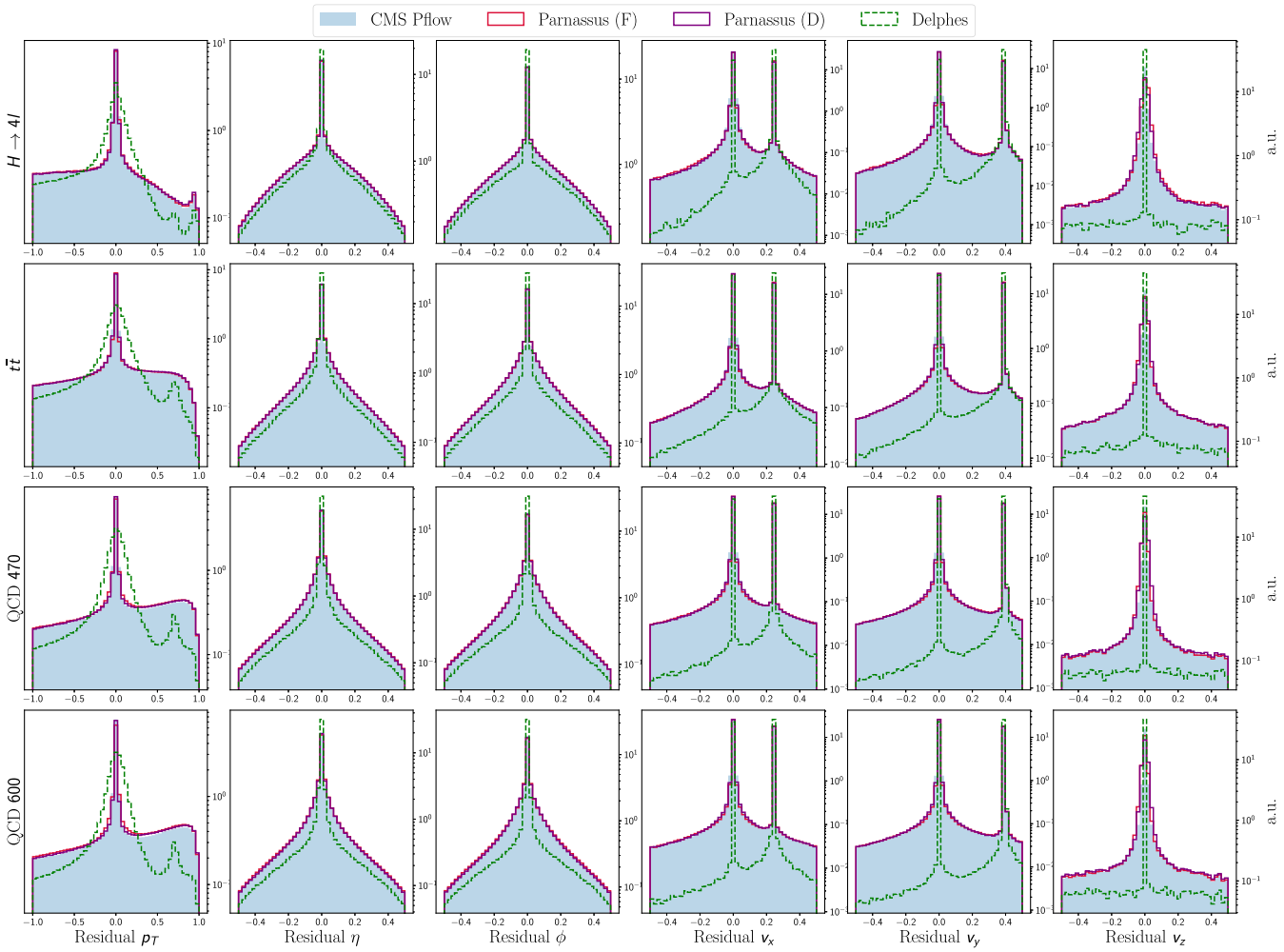


FIG. 11. Residual distributions of matched particle features for $t\bar{t}$, $H \to 4l$, and QCD datasets. Rows correspond to different processes while columns correspond to different observables. All histograms are normalized to unity.

classes of particles are presented in Figs. 13 and 14. While the spectra of in-jet and out-of-jet particles are quite different (higher p_T and more central inside jets), the Parnassus models are able to match the CMS resolution functions. The overestimation of the amount of smearing by Delphes is present inside and outside of jets.

Lastly, we explore the properties of PFCs not matched to GEN particles in Fig. 15. These PFCs consist of genuine fake particles as well as PFCs not properly associated with their GEN particle from the matching algorithm (e.g., because more than one PFC or more than one GEN belong together). The fakes are mostly due to pileup (since we lack pileup GEN particles) and not random combinations of hits within the detector. Compared to matched particles, the unmatched PFCs are softer and more forward. These trends are captured by both Delphes and Parnassus, with a better level of agreement with CMS from the machine learning models. For the kinematic properties, this is most prominent at high p_T, where Delphes significantly overestimates the rate in $H \to 4\ell$ and $t\bar{t}$ events and significantly underestimates the rate in QCD dijet events. The two Parnassus models are about the same, with slightly better performance from the diffusion model in QCD and at high $|\eta|$ and slightly better performance from the flow model at central pseudorapidity. Delphes is not able to model the vertex distributions for these PFCs, while Parnassus provides an accurate model of their vertex distributions. By construction, Delphes is not able to produce pure fakes, since it only smears GEN particles and cannot create new particles. In contrast, the Parnassus models are able to create new, fake particles from scratch.

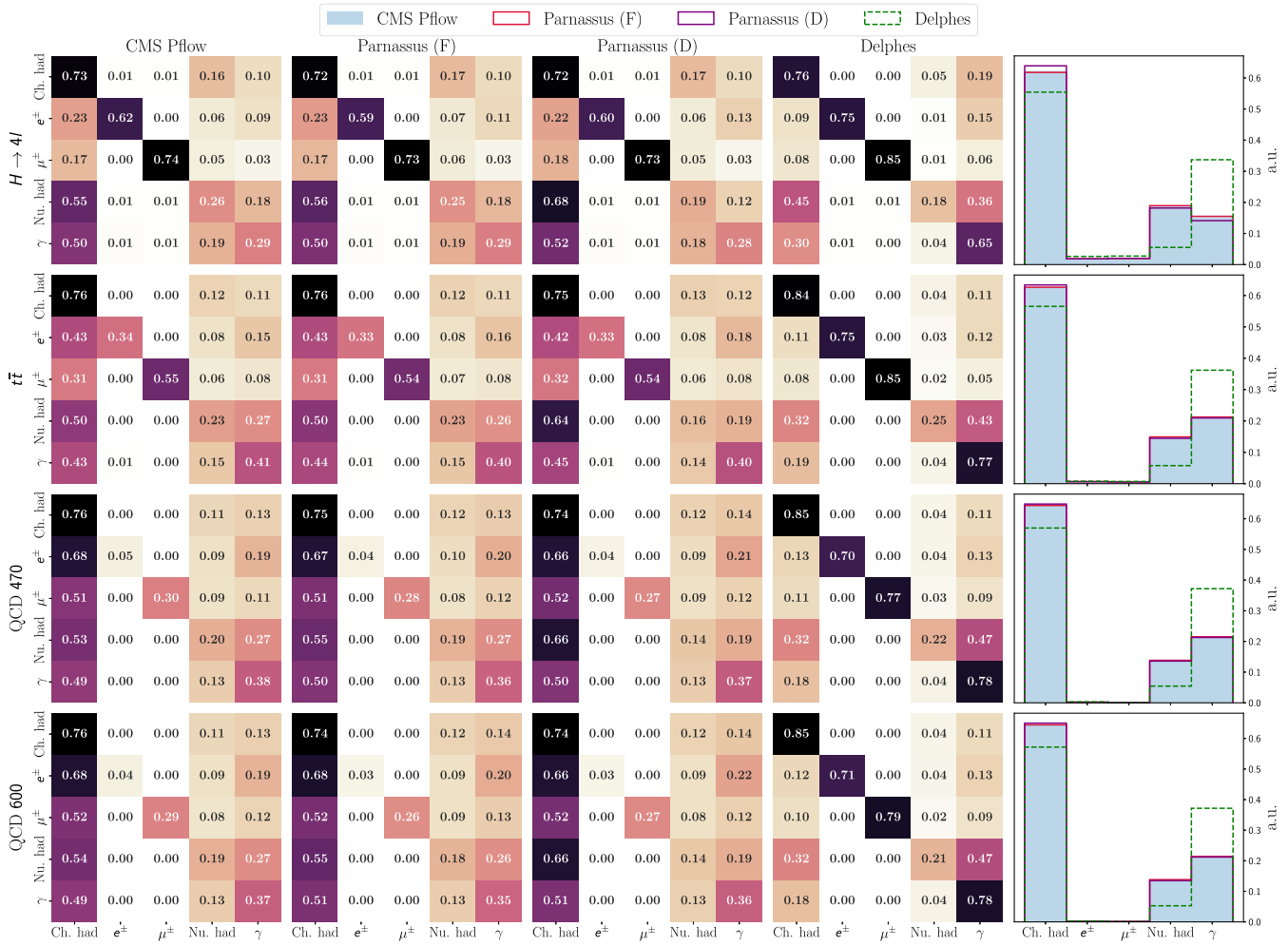


FIG. 12. Class confusion matrices of matched particles (first four columns) and all particle distributions (rightmost column) for $t\bar{t}$, $H \to 4l$, and QCD datasets. For the confusion matrices, the truth particle class is on the vertical axis and the reconstructed class is on the horizontal axis. The frequency distributions are normalized to unity.

V. CONCLUSION AND OUTLOOK

In this paper, we have extended the Parnassus fast simulation and reconstruction approach to entire events. This has required us to accommodate much larger particle multiplicities than from single jets. Furthermore, we now model the particle type and vertex information. Using the CMS detector as an example, we have shown that Parnassus is an accurate surrogate model for simulation and reconstruction, with excellent performance even on physics processes outside of the training dataset. The training of Parnassus is fully automated and the inference is purely Pythonic and GPU-compatible. We have provided two machine learning models with comparable performance. The framework is flexible and it is straightforward to swap out one for another. In practice, it may be useful to have multiple models with comparable accuracy for determining uncertainties.

Our vision is that Parnassus will provide an accurate, portable, and low-barrier-to-entry framework for fast simulation and reconstruction across particle physics experiments. This paper marks the first setup that is ready for deployment. As next steps, we will pursue three directions. First, we will continue to improve the accuracy, precision, stability, and speed of the machine learning models. The current level of accuracy is sufficient to go beyond the capabilities of Delphes and it may one day be able to provide a direct complement to collaboration-internal fast simulation and reconstruction tools. Second, we will build a software interface to make it easy for users to connect particle-level Monte Carlo event generators with their own detector-level analysis, including calibrations. Lastly, we will facilitate the training of a suite of detector models (starting with the two CMS ones in this paper), so that users can study diverse particle physics experiments at colliders and beyond.

ACKNOWLEDGMENTS

V. M. and B. N. are supported by the U.S. Department of Energy (DOE), Office of Science, under Contract

No. DE-AC02-05CH11231. E. G., E. D., and D. K. are supported by the Minerva Grant No. 715027, and the Weizmann Institute for Artificial Intelligence grant program, Ref. 151676.

DATA AVAILABILITY

The code for this paper can be found at [58]. The postprocessed CMS dataset and the generated Delphes dataset can be found on Zenodo at [59].

APPENDIX: ADDITIONAL PLOTS

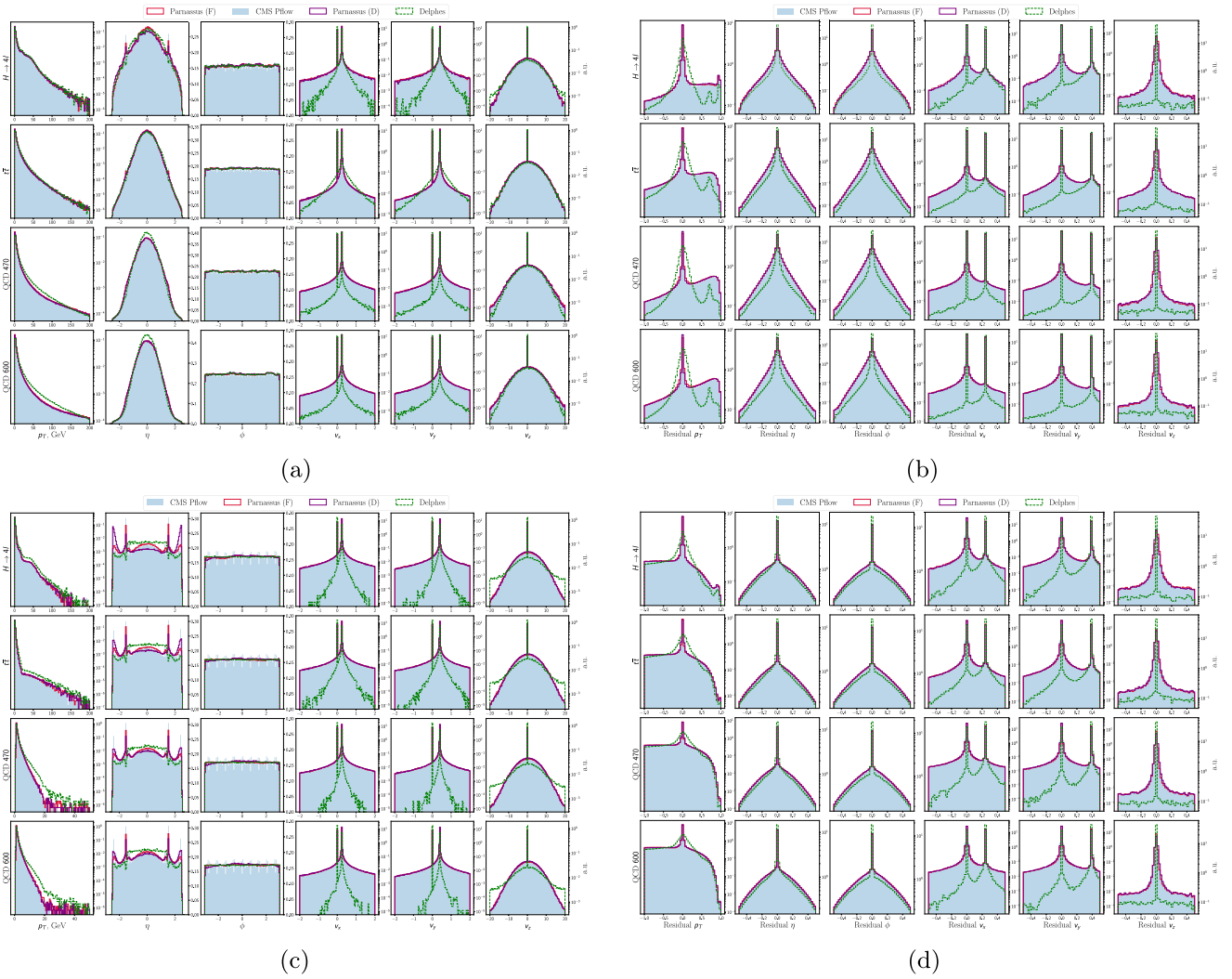
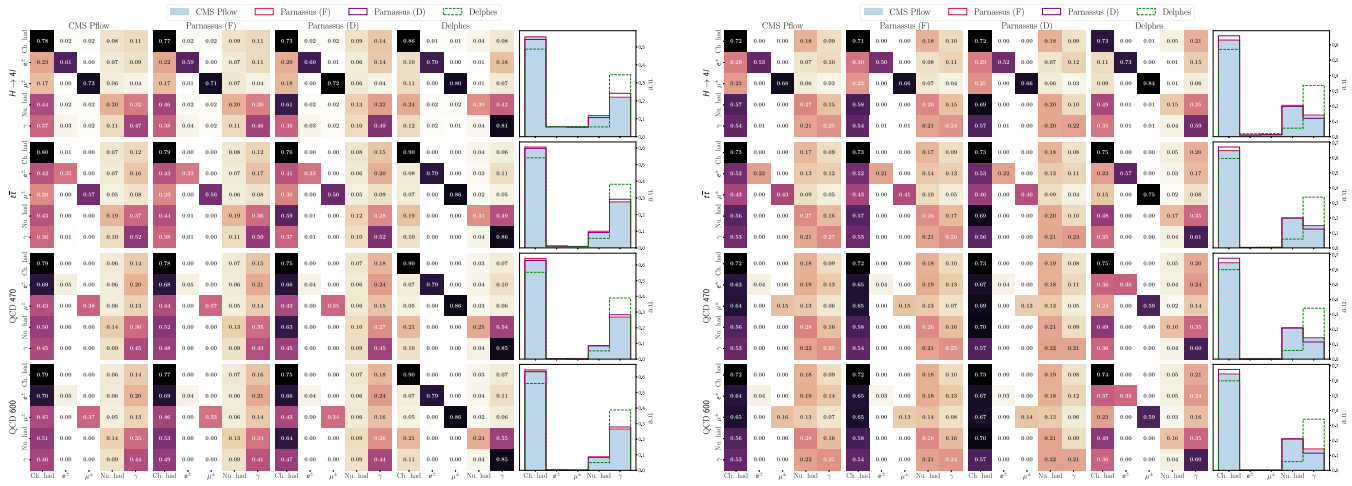
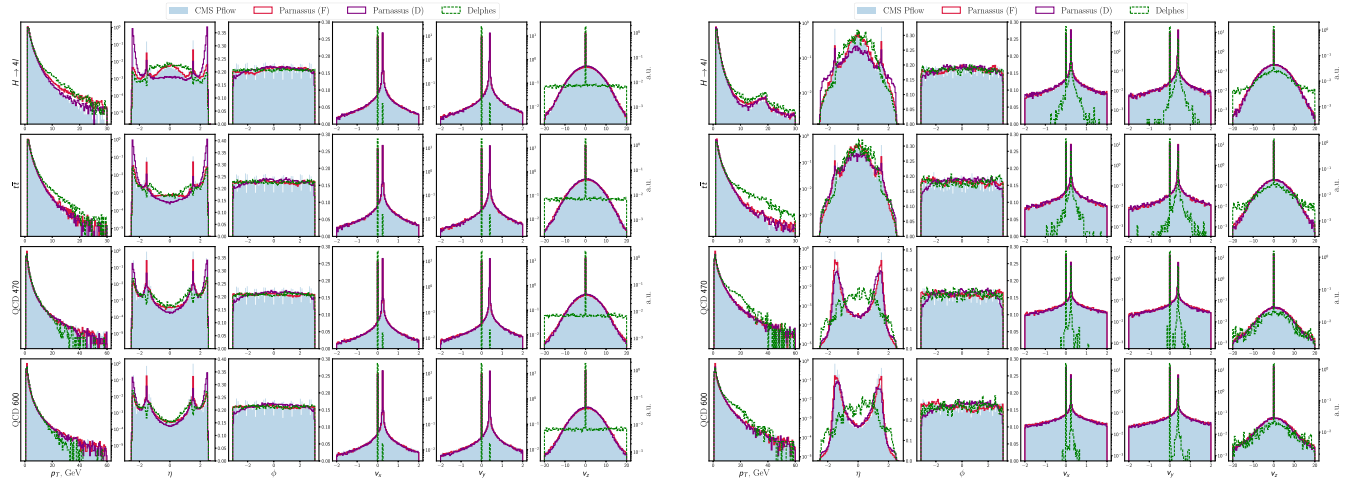


FIG. 13. Kinematic properties of particles in jets (top) and out of jets (bottom). (a) Distributions of matched to jets particle features. (b) Residual distributions of matched to jets particle features. (c) Distributions of background particle features. (d) Residual distributions of background particle features.



(a) (b)

FIG. 14. Particle identification performance for in-jet (a) and out-of-jet (b) particles. (a) Class confusion matrices of matched to jets particles. (b) Class confusion matrices of background particles.



(a) (b)

FIG. 15. Distributions of PFCs not matched to GEN particles. (a) All particles. (b) Particles matched to jets.

[1] V. D. Elvira *et al.*, in *Snowmass 2021* (2022).

[2] G. Aad *et al.* (ATLAS Collaboration), Eur. Phys. J. C **70**, 823 (2010).

[3] M. Beckingham, M. Duehrssen, E. Schmidt, M. Shapiro, M. Venturi, J. Virzi, I. Vivarelli, M. Werner, S. Yamamoto, and T. Yamanaka (ATLAS Collaboration), The simulation principle and performance of the ATLAS fast calorimeter simulation FastCaloSim (2010), https://cds.cern.ch/record/1300517.

[4] G. Aad *et al.* (ATLAS Collaboration), Comput. Software Big Sci. **6**, 7 (2022).

[5] ATLAS Collaboration, Comput. Software Big Sci. **8**, 9 (2024).

[6] S. Abdullin, P. Azzi, F. Beaudette, P. Janot, and A. Perrotta (CMS Collaboration), J. Phys. Conf. Ser. **331**, 032049 (2011).

[7] A. Giammanco, J. Phys. Conf. Ser. **513**, 022012 (2014).

[8] S. Sekmen (CMS Collaboration), Proc. Sci. ICHEP2016 (**2016**) 181 [arXiv:1701.03850].
[9] C. Krause *et al.*, Rep. Prog. Phys. **88**, 116201 (2025).
[10] ATLAS Software and Computing HL-LHC Roadmap, Technical Report, CERN, Geneva, 2022.
[11] C. O. Software and Computing, CMS Phase-2 Computing Model: Update Document, Technical Report, CERN, Geneva, 2022.
[12] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi (Delphes 3 Collaboration), J. High Energy Phys. 02 (2014) 057.
[13] M. Selvaggi, J. Phys. Conf. Ser. **523**, 012033 (2014).
[14] A. Mertens, J. Phys. Conf. Ser. **608**, 012045 (2015).
[15] F. Vaselli, A. Rizzi, F. Cattafesta, and G. Cicconofri (CMS Collaboration), Report No. CERN-CMS-NOTE-2023-003.
[16] F. Vaselli, F. Cattafesta, P. Asenov, and A. Rizzi, Mach. Learn. Sci. Technol. **5**, 035007 (2024).
[17] N. Soybelman, N. Kakati, L. Heinrich, F. A. Di Bello, E. Dreyer, S. Ganguly, E. Gross, M. Kado, and J. Shlomi, Mach. Learn. Sci. Tech. **4**, 045036 (2023).
[18] D. Kobylianskii, N. Soybelman, N. Kakati, E. Dreyer, B. Nachman, and E. Gross, Phys. Rev. D **110**, 092013 (2024).
[19] E. Dreyer, E. Gross, D. Kobylianskii, V. Mikuni, B. Nachman, and N. Soybelman, Phys. Rev. Lett. **133**, 211902 (2024).
[20] T. Sjöstrand, S. Mrenna, and P. Z. Skands, J. High Energy Phys. 05 (2006) 026.
[21] S. Agostinelli *et al.* (Geant4 Collaboration), Nucl. Instrum. Methods Phys. Res., Sect. A **506**, 250 (2003).
[22] A. M. Sirunyan *et al.* (CMS Collaboration), J. Instrum. **12**, P10003 (2017).
[23] P. T. Komiske, R. Mastandrea, E. M. Metodiev, P. Naik, and J. Thaler, Phys. Rev. D **101**, 034009 (2020).
[24] A. Tripathee, W. Xue, A. Larkoski, S. Marzani, and J. Thaler, Phys. Rev. D **96**, 074003 (2017).
[25] CMS Collaboration, Simulated dataset GluGluToHToZZTo4L_M-125_7 TeV-minloHJJ-pythia6-tauola in AODSIM format for 2011 collision data (SM Higgs), 10.7483/OPENDATA.CMS.G4SB.Y2BX (2016).
[26] CMS Collaboration, Simulated dataset TTJets_MSDecays_central_TuneZ2_7 TeV-madgraph-tauola in AODSIM format for 2011 collision data (SM Inclusive), 10.7483/OPENDATA.CMS.VNS2.C226 (2016).
[27] CMS Collaboration, Simulated dataset QCD_Pt-470to600_TuneZ2_7 TeV_pythia6 in AODSIM format for 2011 collision data (SM Exclusive), 10.7483/OPENDATA.CMS.BKTD.SGJX (2016).
[28] CMS Collaboration, Simulated dataset QCD_Pt-600to800_TuneZ2_7 TeV_pythia6 in AODSIM format for 2011 collision data (SM Exclusive), 10.7483/OPENDATA.CMS.EJT7.KSAY (2016).
[29] A. Buckley, P. Ilten, D. Konstantinov, L. Lönnblad, J. Monk, W. Pokorski, T. Przedzinski, and A. Verbytskyi, Comput. Phys. Commun. **260**, 107310 (2021).
[30] E. Dreyer, E. Gross, D. Kobylianskii, V. Mikuni, B. Nachman, and N. Soybelman, Phys. Rev. Lett. **133**, 211902 (2024).
[31] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, arXiv:1806.07366.
[32] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, arXiv:2210.02747.
[33] M. S. Albergo and E. Vanden-Eijnden, arXiv:2209.15571.
[34] X. Liu, C. Gong, and Q. Liu, in *The Eleventh International Conference on Learning Representations, ICLR 2023* (OpenReview.net, Kigali, Rwanda, 2023).
[35] J. Yao, C. Wang, W. Liu, and X. Wang, in *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, edited by A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang (Curran Associates Inc., Red Hook, NY, USA, 2024).
[36] L. Ke, H. Xu, X. Ning, Y. Li, J. Li, H. Li, Y. Lin, D. Jiang, Y. Yang, and L. Zhang, arXiv:2503.04824.
[37] N. Ma, M. Goldstein, M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, and S. Xie, arXiv:2401.08740.
[38] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, arXiv:2011.13456.
[39] T. Karras, M. Aittala, T. Aila, and S. Laine, in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, edited by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Curran Associates Inc., Red Hook, NY, USA, 2022).
[40] Y. Lipman, M. Havasi, P. Holderrieth, N. Shaul, M. Le, B. Karrer, R. T. Q. Chen, D. Lopez-Paz, H. Ben-Hamu, and I. Gat, arXiv:2412.06264.
[41] T. Salimans and J. Ho, in *International Conference on Learning Representations* (2022).
[42] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Las Vegas, NV, USA, 2016), pp. 770–778.
[43] W. Peebles and S. Xie, arXiv:2212.09748.
[44] V. Mikuni and B. Nachman, Phys. Rev. D **111**, L051504 (2025).
[45] V. Mikuni and B. Nachman, Phys. Rev. D **111**, 054015 (2025).
[46] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, ACM Trans. Graphics (tog) **38**, 1 (2019).
[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, New York, 2019).
[48] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, in *OSDI*, Vol. 16 (USENIX Association, Savannah, GA, USA, 2016), pp. 265–283.
[49] I. Loshchilov and F. Hutter, arXiv:1608.03983.
[50] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, and Q. V. Le, in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, edited by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Curran Associates Inc., Red Hook, NY, USA, 2023).

[51] E. Xie, J. Chen, J. Chen, H. Cai, H. Tang, Y. Lin, Z. Zhang, M. Li, L. Zhu, Y. Lu, and S. Han, arXiv:2410.10629.
[52] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, edited by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Curran Associates Inc., Red Hook, NY, USA, 2022).
[53] M. Cacciari, G. P. Salam, and G. Soyez, Eur. Phys. J. C **72**, 1896 (2012).
[54] M. Cacciari, G. P. Salam, and G. Soyez, J. High Energy Phys. 04 (2008) 063.
[55] H. W. Kuhn, Nav. Res. Logist. Quart. **2**, 83 (1955).
[56] A. J. Larkoski, G. P. Salam, and J. Thaler, J. High Energy Phys. 06 (2013) 108.
[57] A. J. Larkoski, I. Moult, and D. Neill, J. High Energy Phys. 12 (2014) 009.
[58] https://github.com/parnassus-hep/cms-flow-evt.
[59] https://zenodo.org/records/15083495.