

# Improved particle-flow event reconstruction with scalable neural networks for current and future particle detectors

Joosep Pata<sup>1\*</sup>, Eric Wulff<sup>2</sup>, Farouk Mokhtar<sup>3</sup>, David Southwick<sup>2</sup>,  
Mengke Zhang<sup>3</sup>, Maria Girone<sup>2</sup>, Javier Duarte<sup>3</sup>

<sup>1\*</sup>National Institute of Chemical Physics and Biophysics (NICPB),  
Rävala pst 10, 10143 Tallinn, Estonia.

<sup>2</sup>European Center for Nuclear Research (CERN), CH 1211, Geneva 23,  
Switzerland.

<sup>3</sup>University of California San Diego, La Jolla, CA 92093, USA.

\*Corresponding author(s). E-mail(s): [joosep.pata@cern.ch](mailto:joosep.pata@cern.ch);  
Contributing authors: [eric.wulff@cern.ch](mailto:eric.wulff@cern.ch); [fmokhtar@ucsd.edu](mailto:fmokhtar@ucsd.edu);  
[david.southwick@cern.ch](mailto:david.southwick@cern.ch); [mezhang@ucsd.edu](mailto:mezhang@ucsd.edu); [maria.girone@cern.ch](mailto:maria.girone@cern.ch);  
[jduarte@ucsd.edu](mailto:jduarte@ucsd.edu);

## Abstract

Efficient and accurate algorithms are necessary to reconstruct particles in the highly granular detectors anticipated at the High-Luminosity Large Hadron Collider and the Future Circular Collider. We study scalable machine learning models for event reconstruction in electron-positron collisions based on a full detector simulation. Particle-flow reconstruction can be formulated as a supervised learning task using tracks and calorimeter clusters. We compare a graph neural network and kernel-based transformer and demonstrate that we can avoid quadratic operations while achieving realistic reconstruction. We show that hyperparameter tuning significantly improves the performance of the models. The best graph neural network model shows improvement in the jet transverse momentum resolution by up to 50% compared to the rule-based algorithm. The resulting model is portable across Nvidia, AMD and Habana hardware. Accurate and fast machine-learning based reconstruction can significantly improve future measurements at colliders.

# 1 Introduction

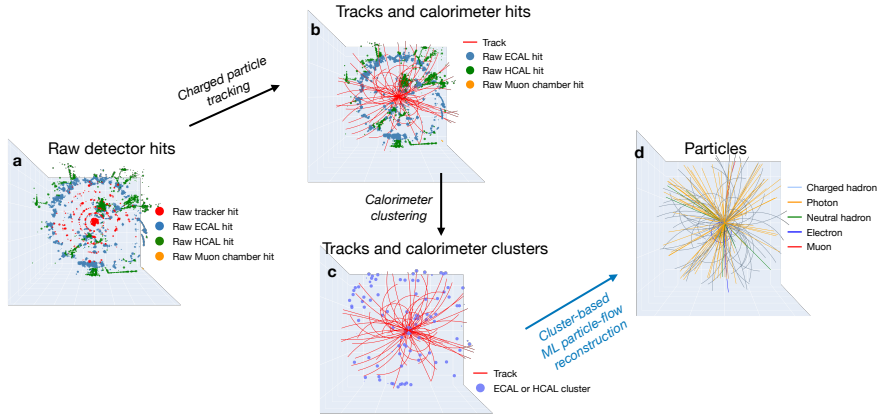
One of the main approaches for event reconstruction at the Large Hadron Collider (LHC) is currently based on the particle-flow (PF) algorithm [1–13], which combines measurements from different subdetectors to produce a holistic particle-based description of the entire event. For the planned High Luminosity LHC (HL-LHC) [14] program, due to the installation of new highly granular detector subsystems such as high-granularity calorimeter (HGCal) for Compact Muon Solenoid (CMS), the reconstruction algorithms have to be revisited or new algorithms have to be developed to fully make use of the data which has a significantly higher complexity. Similarly, for possible future experimental programs such as the Future Circular Collider (FCC) [15, 16], existing algorithms have to be retuned or new ones developed, possibly many times for each new detector scenario under study. Therefore, it is necessary to develop high-fidelity PF reconstruction algorithms that are at the same time computationally efficient, and can be easily extended to new detector concepts without significant manual work. Moreover, if algorithms can be found that can reconstruct events from highly granular detectors with improved fidelity, e.g. in terms of jet response, this may have important implications towards the sensitivity and thus cost-effectiveness of future experiments.

There has been considerable interest in and development of machine learning (ML)-based reconstruction methods, including for PF reconstruction. Models for PF reconstruction based on computer vision were investigated in [17]. In order to train a model that reconstructs events consisting of a variable number of particles, a recipe for a loss function based on attractive and repulsive potentials was given in Ref. [18]. In Ref. [19], a version of this loss was used together with a graph neural network (GNN)-based approach to reconstruct events with high particle multiplicity accurately. This approach was successfully applied in the CMS experiment [20, 21]. Recently, Ref. [22], demonstrated that a network architecture based on learning a hypergraph structure can improve jet reconstruction. In parallel, clustering using ML has been demonstrated for high-granularity calorimeter reconstruction [23]. Extending beyond particle reconstruction, there is considerable interest and progress in reconstructing full decay trees using ML [24, 25].

One of the critical challenges for PF reconstruction is the highly granular nature of the data: events can comprise hundreds of thousands of heterogeneous measurements in various detector subsystems. This motivates studying models that can scale to large input multiplicities and efficiently process full events or batches of events simultaneously for improved throughput. To support such studies, it is beneficial to establish open realistic simulated datasets with sufficient granularity to test various approaches.

In this paper, we utilize an open dataset of electron-positron ( $e^+e^-$ ) collision events at a center of mass energy  $\sqrt{s} = 380$  GeV with full GEANT4 simulation, suitable for detector reconstruction, available in the EDM4HEP [26] format for future studies. The specific center of mass energy was chosen due to being well studied as one of the initial proposed scenarios for CLIC and thus the baseline reconstruction software is available and tuned. However, nothing in our proposed approach is specific to the center of mass energy or the specific detector configuration.

**Fig. 1:** A conceptual overview of the machine-learned particle flow approach based on tracks, hits and clusters on one simulated  $t\bar{t}$  event.



(a) The raw tracker, calorimeter and muon chamber hits, embedded in position space, with the size of the marker proportional to the hit energy. (b) Tracking algorithms reconstruct charged particle tracks from the tracker hits, shown with their extrapolated trajectories. (c) The calorimeter hits are clustered to correspond better to individual particles. (d) The machine-learned particle flow algorithm reconstructs charged and neutral hadrons, photons, electrons and muons based on the tracks and clusters from the previous step, shown with their extrapolated trajectories.

We test two types of scalable ML models as benchmarks that can process full events consisting of tens of thousands of measurements, while avoiding memory allocation or computations that scale quadratically with the input size. Using charged particle tracks and clusters of calorimeter energy deposits, we minimize a particle-based loss function and monitor physics metrics that quantify event reconstruction performance during training. We report the results of an extensive hyperparameter optimization (HPO) of the GNN-based model performed using high performance computing (HPC) resources. We then evaluate the model’s physics and computational performance, as well as its portability and quantization compatibility.

We approach the challenge of high-fidelity full event reconstruction via particle flow using two alternative scalable machine learning models: kernel-based transformers and graph neural networks using locality-sensitive hashing. After a large-scale hyperparameter optimization, we find that the GNN-based model can reconstruct physics events with a higher degree of fidelity compared to the rule-based baseline. At the same time, its computational cost scales linearly with the size of the input event, which is desirable for future deployment scenarios in high-granularity detectors. Moreover,

we demonstrate the portability of the model to several computational hardware processors from Nvidia, AMD, and Intel Habana. The model can also be scaled naturally to lower-level datasets consisting of raw detector hits, if tuned tracking or clustering algorithms are not available. Our proposed approach for particle flow reconstruction is summarized in Fig. 1. We identify steps for future development and also publish the code and datasets following the findable, accessible, interoperable, and reusable (FAIR) principles [27–29] for reproducibility and future development.

## 2 Methods

In this section, we describe the loss function for particle flow reconstruction, the specific neural network (NN) approaches as well as the procedure for dataset generation.

### 2.1 Loss function

The optimization goal for full event particle reconstruction follows the machine-learned particle flow (MLPF) approach previously used in Refs. [19–21], and applies a physics-inspired ansatz to simplify the event-based reconstruction loss to a particle-based classification and regression loss.

The input to the model is a set of detector elements: charged tracks and calorimeter clusters (or alternatively, raw calorimeter hits), each described by a feature vector  $\mathbf{x} \in X$ . The input features for the tracks consist of the track  $p_T$ , the pseudorapidity of the particle momentum (track tangent)  $\eta$ , the azimuthal angle of the particle momentum  $\phi$ , the track goodness-of-fit  $\chi^2$  and the number of degrees of freedom  $N_{\text{dof}}$ , the slope of the track  $\tan \lambda = p_z / \sqrt{p_x^2 + p_y^2}$ , the signed impact parameter  $D_0$  with respect to the origin  $(0, 0, 0)$  at interaction point in the  $xy$  plane, the signed curvature of the track  $\Omega = \text{sign}(q)/R$  defined via the track radius  $R$  and charge  $q$ , the position of the track along the  $z$ -axis  $Z_0$  with respect to the origin [30]. For calorimeter clusters, the features consist of transverse energy  $E_T$ ,  $\eta$ ,  $\phi$ , electromagnetic calorimeter energy  $E_{\text{ECAL}}$ , hadronic calorimeter energy  $E_{\text{HCAL}}$ , Cartesian spatial coordinates  $x$ ,  $y$ ,  $z$ ; the number of hits and the cluster size, measured in standard deviations of the hit sets in  $x$ ,  $y$ , and  $z$ .

The target of the model is a set of stable particles such as charged and neutral pions, photons, electrons, and muons, defined through sufficient energy matched to detector hits and with a PYTHIA8 8 status code 1. This specific choice of target particles represents a reasonable optimization target, but may not be optimal for all cases. The definition of an optimal ground truth for particle reconstruction is left to future work. Each particle is described by a particle type, charge, and continuous four-momentum values, combined to form a target feature vector  $\mathbf{y} \in Y$ . The goal of the model is, for each event, to reconstruct the set  $Y$ , given the set  $X$ , i.e.  $\Phi(X) \rightarrow Y' \simeq Y$ . To make the set-to-set translation tractable with ML, and to be able to treat particle multiplicities in the range of  $|X|, |Y| \simeq 10^4$  with a single model pass, each target particle is associated with a single unique input element.

The target particles are assigned to input elements by an injective, non-surjective function based on a physics prior. The general approach is based on the object condensation (OC) [18], which includes a potential between inputs and outputs. Here, we only associate charged particles to tracks and neutral particles to the highest-energy calorimeter cluster (or hit), as a tradeoff between expressiveness and computational cost. The event reconstruction loss can be written as a sum over the input elements in each event

$$L(Y, Y') = \sum_i L_{\text{cls}}(y_i, y'_i) + L_{\text{reg}}(y_i, y'_i). \quad (1)$$

Here,  $L_{\text{cls}}(y_i, y'_i)$  is a classification loss between the predicted and target particle type and charge labels. The classification task is imbalanced, so we use the focal loss [31] for particle type classification, which assigns greater weight to samples that are difficult to classify. Similarly,  $L_{\text{reg}}(y_i, y'_i)$  is a regression loss for the momentum components, where we use the Huber loss [32] to reduce the effect of outliers.

We note that in the recent hypergraph-based reconstruction method [22], the strict particle-to-element association is completely avoided, and the associations are instead made an optimization target for an intermediate model. This hypergraph reconstruction approach shows excellent physics performance on small jet-based samples, but further work is needed to extend it to full events with large particle multiplicities and to demonstrate computational feasibility on realistic datasets.

### 2.1.1 Scalable neural network models

We now move to the specific NN structure of the reconstruction model  $\Phi(X)$ . It could be implemented with a simple feedforward network over the individual feature vectors of each input element, i.e.  $\phi(x) \rightarrow y'$ . However, such a model would be unable to consider correlations between related input elements, such as tracks and calorimeter clusters (or hits) arising from a single particle.

An early approach proposed for full-event reconstruction previously used a GNN with a learnable graph structure between the input nodes [19]. This is conceptually similar to a transformer with full self-attention, which has also recently been thoroughly investigated [22].

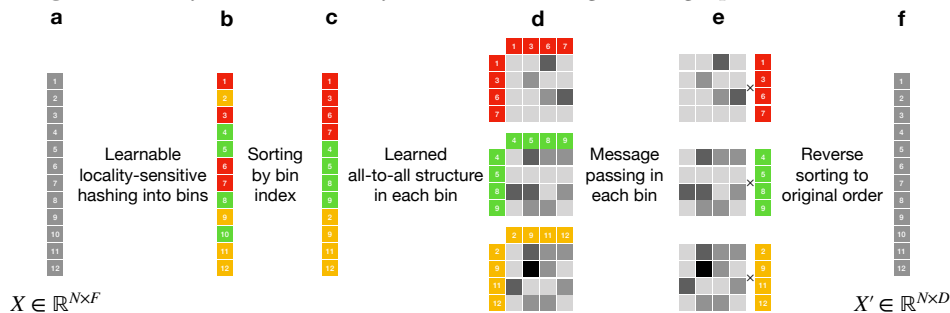
However, the evaluation of full self-attention can be computationally demanding and prohibitive for large input multiplicities ( $\geq 10^4$ ) [33]. Such input multiplicities can be easily reached when considering all tracks and calorimeter clusters in future high luminosity collider events. It is possible that the local nature of the problem can be exploited directly by first pre-partitioning the event and running the full attention-based model only on subsets. However, this requires careful implementation to avoid artifacts from partitioning and subsequent stitching.

Here, we study models that can naturally scale to large input multiplicities by avoiding pairwise memory allocations or computation. Increasing the input multiplicity that ML models can process simultaneously without manual splitting also receives considerable attention in the literature because of its wide range of applications [34, 35], and our approaches are based on existing research:

- The first model uses a dynamically learned graph structure [19], but avoids a full quadratic allocation or computation by using a learnable binning based on locality sensitive hashing (LSH) in each graph building layer [20, 21], inspired by the **Reformer** architecture [36]. This approach divides each event into bins of fixed size based on a learnable function.
- The alternative model is a kernel-based transformer in which the softmax self-attention layer is approximated using positive orthogonal random features [37]. This approach uses a mathematical approximation based on random projections to avoid computing the full attention.

Both of these approaches avoid memory allocations or computations that are quadratic in input multiplicity. The LSH-based approach was first used in [19], initially being paired with a k-nearest neighbors (kNN) graph structure in each bin. In this paper, we instead use an all-to-all fully connected graph in each bin, as we have found it to significantly outperform the kNN based approach computationally as well as in terms of fidelity. Moreover, in this paper, we compare the LSH+GNN approach with the kernel-based transformer alternative systematically. Thus, this paper offers the first detailed comparison of alternative scalable models for particle flow reconstruction.

**Fig. 2:** One layer of the locality sensitive hashing based graph neural network.



(a) The input event of  $N$  tracks and clusters (elements) is described by a list of  $N$  feature vectors, one  $F$ -dimensional vector per element. (b) Each element is first assigned into a bin based on a learnable function, denoted with the color of the box. (c) The elements are then sorted according to the bin, such that elements in the same bin are consecutive. (d) In each bin, a full all-to-all learnable adjacency matrix is constructed between all the elements in the bin, with the learned element-to-element association illustrated by the color of the matrix element. (e) This matrix is used for message passing in each bin, multiplying the corresponding bin feature vectors with the learned adjacency matrix. (f) The output is a list of  $N$  transformed feature vectors, one  $D$ -dimensional vector for each track or cluster. Each input and output vector is represented by a single gray box, the order of the input and output feature vectors is preserved.

The LSH-based GNN is constructed as follows. A full all-to-all graph between  $N$  input elements (tracks and clusters) in the event would have dimensionality  $N^2$ , which for an event with  $N = 10^3$  tracks and clusters would require  $N^2 = 10^6$  individual associations to be stored and computed for each layer. Instead, we split the event dynamically into bins with a fixed and constant size  $B \ll N$ , and define the element-to-element connectivity only in each bin, each bin requiring  $B^2$  associations. For an event with  $N$  input elements, the number of bins is then determined dynamically at runtime by  $N_B = N/B$ , with the last bin being padded if necessary. Therefore, instead of computing an  $N \times N$  adjacency matrix for each layer, we instead compute a three-dimensional  $N_B \times B \times B$  adjacency matrix. This means that an event with  $N = 10^3$  elements and bin size  $B = 10^2$  would consist of  $N_B = 10$  bins, requiring  $BN_B^2 = 10^5$  instead of  $N^2 = 10^6$  individual associations. The graph structure defined by the association matrix consists of  $N_B$  disjoint graphs of  $B$  elements each, which is not a major limitation because the graph building layers can be stacked multiple times, each building different disjoint graphs. From a physics point of view, it is reasonable to expect that nearby (for some learnable definition of neighborhood) inputs should have a stronger association than input elements in highly separated parts of the detector. This graph building layer can be implemented using elementary matrix operations in a fully batched and differentiable way using TENSORFLOW [38, 39]. This is similar to manually partitioning and restitching the event, but achieved directly in the NN model using fully differentiable operations, rather than with a manual heuristic. The GNN is based on stacked layers of graph building and convolution, with the number of layers being a configurable hyperparameter.

The alternative kernel-based transformer model avoids quadratic scaling using the following approach. For  $N$  elements, given queries  $Q \in \mathbb{R}^{N \times d_q}$  and keys  $K \in \mathbb{R}^{N \times d_k}$ , the attention mechanism encodes a value matrix  $V \in \mathbb{R}^{N \times d_v}$  as

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \quad (2)$$

Here, the  $\text{softmax}(QK^\top)$  operation creates a full  $N \times N$  matrix. As in Ref. [37], we define a transformation  $\psi(\mathbf{x}) \rightarrow \mathbf{x}'$  that transforms an input feature map  $\mathbf{x}$  using predetermined random projections to a new feature space  $\mathbf{x}'$ . For a sufficiently large number of random projections, attention can be approximated as

$$\text{Attn}(Q, K, V) \simeq Q'(K'^\top V) \quad (3)$$

where  $Q'$  and  $K'$  are the query and key matrices after the random feature mapping  $\psi$ , respectively. Allocation of the entire  $N \times N$  matrix is avoided, since the order of operations is changed to first multiply keys with values and then subsequently with queries. In the special case of self-attention,  $Q$ ,  $K$ , and  $V$  are all derived from  $X$  through a linear layer, and the self-attention mechanism can be seen as an analogy to graph building and message passing. A visual overview of the supervised learning setup, as well as the LSH model structure, is shown in Fig. 2.

The complete model for PF reconstruction is then implemented based on stacked layers of LSH-based GNN or self-attention and feedforward networks, with the number of layers being a Hyperparameter (HP).

### 2.1.2 Dataset

Having defined NN models for PF reconstruction that avoid  $\mathcal{O}(N^2)$  memory allocation and computational scaling, we train and test them on a realistic dataset. We generate  $e^+e^-$  collision events with PYTHIA8 (v8.306) [40] and carry out a complete detector simulation with GEANT4 (v11.0.2) using the Key4HEP framework (v2023-01-15) [41]. In particular, we use the Compact Linear Collider (CLIC) detector model [42, 43], along with the Marlin reconstruction code [44], and the Pandora package [45–47] for a baseline PF implementation. The CLIC detector model is chosen because it is publicly available, well documented and realistic, and similar to detector concepts that are in use at LHC, or under consideration for either HL-LHC or FCC.

The CLIC detector model is based on the CMS detector at CERN. It features a superconducting solenoid with an internal diameter of 7 m, providing a magnetic field of 4 T in the center of the detector. Silicon pixel and strip trackers, the electromagnetic (ECAL) and hadron calorimeters (HCAL) are embedded within the solenoid. Each subdetector is divided into a barrel and two endcap sections. The ECAL is a highly granular array of 40 layers of silicon sensors and tungsten plates. The HCAL is built from 60 layers of plastic scintillator tiles, read out by silicon photomultipliers, and steel absorber plates. The muon system surrounding the solenoid consists of 6 and 7 layers of resistive plate chambers interleaved with yoke steel plates in the endcap and barrel respectively. Two smaller electromagnetic calorimeters, LumiCal and BeamCal, cover the very forward region of the detector on either side of the interaction point [42, 43].

Collision events are generated with different physics processes to test the out-of-distribution performance of the model. In particular, we use PYTHIA8 to generate  $\simeq 10^6$  inclusive  $t\bar{t}$ , ZH, fully hadronic WW each, and  $\simeq 2 \times 10^6$   $q\bar{q}$  events. Furthermore, we generate  $\simeq 7 \times 10^5$  of  $t\bar{t}$  PU10 events with a beam-induced hadronic overlay  $gg \rightarrow q\bar{q}$  interactions, corresponding to an average of 10 additional interactions per event, known as pileup (PU), to test the stability under varying conditions. We also generate single  $e^\pm$ ,  $\mu^\pm$ ,  $K_L^0$ ,  $\pi^0$ ,  $\pi^\pm$ , neutron, and photon particle gun samples, with a uniform energy distribution in  $E \in [1, 100]$  GeV, generated using the DDSIM [48] package,  $\simeq 10^6$  events each for performance tests. The  $q\bar{q}$  and  $t\bar{t}$  were split in an 80/20 ratio to form train/test samples, while the ZH, WW,  $t\bar{t}$  PU10 and single particle gun datasets were never used in training.

The datasets with generator particles; reconstructed tracks, hits and calorimeter clusters; as well as reconstructed particles from the baseline Pandora algorithm are saved in the EDM4HEP format, including all the relevant associations. Overall, the size of the dataset is approximately 2.5 TB before preprocessing to the ML-specific format using the TFDS library [49]. The raw datasets in EDM4HEP format, along with the scripts and configurations to generate the data, are available at [50].

There are about 50–500 tracks or calorimeter clusters per event, while the multiplicity for raw calorimeter hits is considerably larger at  $5\text{--}15 \times 10^3$  per event. In this paper, we analyze the datasets at the level of the tracks and calorimeter clusters to

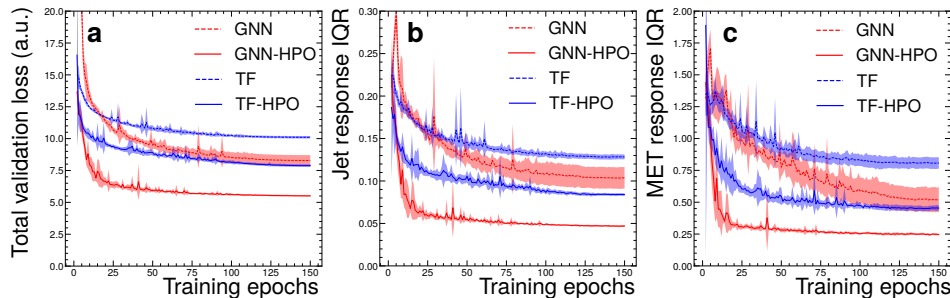


study the physics performance of models. We note that it is straightforward to apply the model on tracks and raw calorimeter hits, however, we leave this analysis for a follow-up paper. The input elements and target particles have different underlying signatures in different samples, depending on the physics process, therefore the models will have to demonstrate out-of-distribution generalization.

Jets are defined by clustering particles with the generalized  $k_T$  algorithm ( $R = 0.7, p = -1$ ) for  $e^+e^-$  colliders [51, 52] with a minimum  $p_T \geq 15$  GeV. No additional quality cuts are applied on the jets at this stage. We also evaluate the  $\vec{p}_T^{\text{miss}}$  and total 3D momentum of the reconstructed and generated particles. The missing transverse momentum vector  $\vec{p}_T^{\text{miss}}$  is calculated as the negative vector sum of the transverse momenta of all particles in an event, and its magnitude is denoted as  $p_T^{\text{miss}}$ . We use these quantities to assess the performance of MLPF and the baseline PF reconstruction with respect to the ground truth defined by the generator particles.

### 3 Results and discussion

**Fig. 3:** The model validation loss and physics performance throughout training for the graph neural network and kernel-based transformer, before and after hypertuning.



The validation loss (a), jet (b) and  $p_T^{\text{miss}}$  resolution (c), parameterized by the interquartile range (IQR) of the response distribution, for the graph neural network (GNN, red) and kernel-based transformer (TF, blue) models before (dashed lines) and after hyperparameter optimization (HPO, solid lines), evaluated using 10 trainings, each trained on four Nvidia A100 devices in a distributed data-parallel manner. Lower values correspond to better physics performance. Only the loss is explicitly minimized, the jet and  $p_T^{\text{miss}}$  resolution improvement emerges from the minimization of the loss function. We show the evolution of these quantities during the full training process, where only the random seed differs in each run. The shaded regions show the standard deviation of the metrics across the runs.

Several steps were taken to ensure efficient training on large-scale datasets. The models can be implemented using matrix operations in native `TENSORFLOW`. Given the varying size of events, the model implements variable-sized batching, supporting

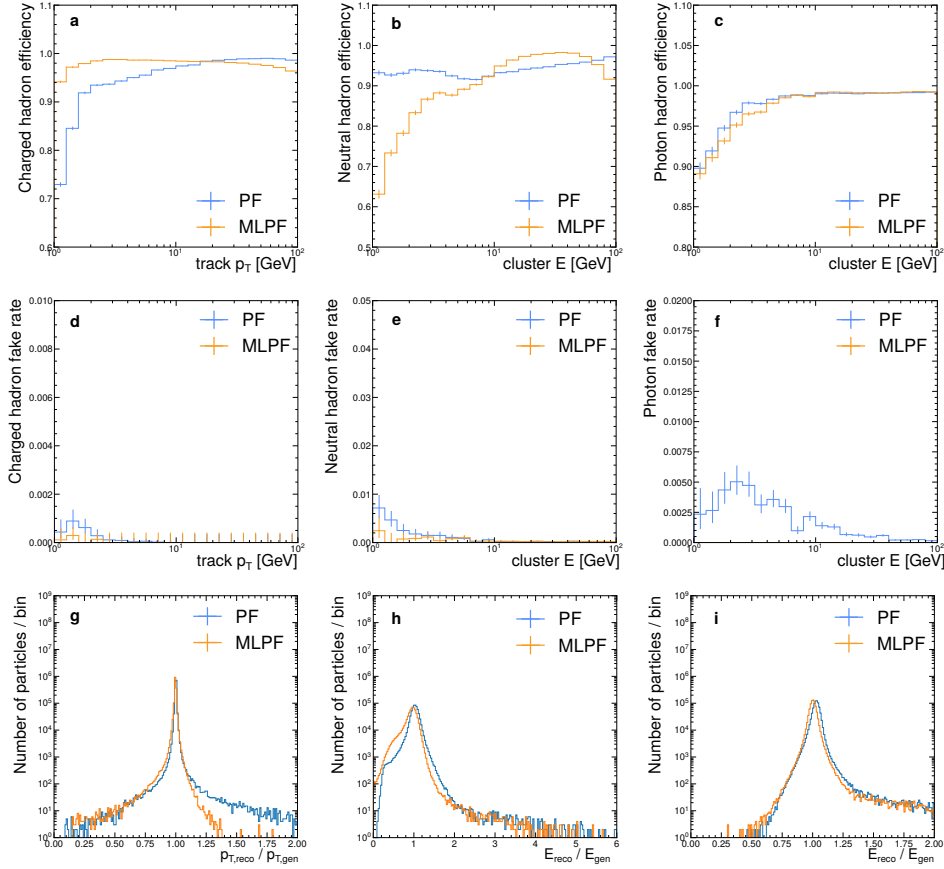
batching events at fixed batch size increments, with the batch size inversely proportional to the size of the events in the batch. This was necessary to reduce the amount of zero-padding, and to enable automatic kernel compilation, so that a limited number of kernels have to be compiled for each bucket size increment. Since the total size of the datasets can reach several terabytes, efficient larger-than-RAM training is enabled through dynamic dataset loading and interleaving using the `TENSORFLOW DATASETS` library. The model supports mixed-precision training in `BFLOAT16` and `FLOAT16`. The training in `BFLOAT16` results in loss values that are stable and largely compatible with that of `FLOAT32`, while we find that the dynamic loss scaling for `FLOAT16` results in NAN gradients and does not converge. At the time of writing, software support for `BFLOAT16` in `TENSORFLOW` is limited, such that the operations are not placed on tensor cores and the use of `BFLOAT16` does not result in increased training throughput, while training with `FLOAT16` results in moderate speedups of 30–40% for the GNN model. Improving training throughput using `BFLOAT16` is the focus of future work.

An extensive HPO was performed for both the GNN- and kernel-based transformer model using the JURECA supercomputer [53]. JURECA is a pre-exascale modular supercomputer operated by Jülich Supercomputing Centre at Forschungszentrum Jülich. The system consists of a flexible Data Centric (DC) module, based on the Atos BullSequana XH2000. It has, among others, 192 accelerated compute nodes with four NVIDIA A100 GPUs and two AMD EPYC 7742 CPUs each. We employ a similar approach to HPO as in Ref. [54], using Bayesian Optimization in combination with the ASHA [55] algorithm. The optimization was performed in a distributed manner on 96 GPUs spread over 24 compute nodes, consuming roughly 7000 and 5000 GPU hours for the optimization of the kernel-based transformer and the GNN model respectively. The hyperparameter optimization details can be found in Supplementary Note 1: Hyperparameter optimization.

The performance improvements achieved from HPO are presented in Fig. 3. We find that the optimized versions of both the GNN and the kernel-based transformer significantly outperform the unoptimized versions. The relevant evaluation criterion for model selection is the reconstruction of jets and  $p_T^{\text{miss}}$  compatible with those at the generator level, a target that is not explicitly trained for, but can be reached by minimizing the particle-level loss. The optimized version of the GNN significantly outperforms the kernel-based transformer, although both use a similar number of trainable parameters ( $\simeq 5 \times 10^6$ ). As the GNN-based model has significantly improved validation loss and physics performance, we focus on it for the rest of this paper.

First, we study the performance of the chosen GNN model on single particle gun samples that were never used in training. In Fig. 4, we see that the performance of the baseline PF and our proposed MLPF algorithm is broadly similar for charged hadrons, while the efficiency and fake rate for neutral hadrons and photons are higher for PF, especially at low calorimeter cluster energies. Apart from differences in acceptance thresholds that we observe here, we do not expect or observe significant physical differences on single particle gun samples. The  $p_T$  and energy response distributions are broadly similar for all particle types. From this, we conclude that while not perfect, the baseline PF algorithm is reasonably well tuned and represents a meaningful comparison point.

**Fig. 4:** Performance of the particle flow (PF) and machine-learned particle flow (MLPF) algorithms on single particle gun samples.



Charged hadron efficiency in (a), neutral hadron efficiency in (b), photon efficiency in (c). The efficiencies are parameterized as a function of track  $p_T$  or cluster energy. The charged hadron fake rate is shown in (d), the neutral hadron fake rate in (e), the photon fake rate in (f). For efficiency and fake rate, we show the binomial statistical uncertainties from limited samples. We show the  $p_T$  response of charged hadrons in (g), and the energy response of neutral hadrons in (h) and photons in (i).

We also report the single-particle truth and reconstructed distributions for all held-out samples in Figs. 5 and 6, as well as for jets in Fig. 7.

The physics performance in event-level quantities is summarized in Fig. 8. The jet response distribution of the baseline PF algorithm is somewhat asymmetric, as the baseline jets are biased towards higher  $p_T$  values with respect to the matched generator jet, while the distribution from our proposed algorithm is significantly more symmetric around unity. This is due to the overall momentum regression being optimized on the samples directly in MLPF.

We use ZH, WW and  $t\bar{t}$  PU10 events to evaluate out-of-distribution performance. We evaluate the jet response by clustering reconstructed particles into jets, by matching the reconstructed and generator-level jets, and computing the ratio of reconstructed to generator-level jet  $p_T$ . In all samples, the fraction of reconstructed jets from the GNN-based MLPF is the same or higher than for PF, with generally an improved jet response width, quantified by the interquartile range (IQR) and median compared to the rule-based baseline. This improvement over the baseline was not observed before hyperparameter tuning. We also evaluate the total 3-momentum response for either PF or MLPF particles, and find that the MLPF model improves both the median and IQR of the total 3-momentum response distributions for all samples.

To compare the momentum resolutions between the different approaches, while accounting for the differences in the momentum scales, we evaluate the metric of the IQR divided by the median of the response distributions. We quantify the evolution of this metric for the jet  $p_T$  (total event 3-momentum) in bins of generator-level jet  $p_T$  (generated total event 3-momentum) on the  $t\bar{t}$  with 10 PU interactions ( $t\bar{t}$  PU10) sample in Fig. 9. The baseline PF and proposed MLPF algorithms behave qualitatively similarly, with improved response IQR over median values at higher generator-level jet  $p_T$  (total 3-momentum), while the MLPF algorithm consistently outperforms the baseline PF on this sample by up to 50%. This improvement has important implications for the sensitivity of key measurements at future colliders, such as those involving Higgs bosons that decay to bottom quark-antiquark pairs [56].

An important factor in the development of scalable ML-based full event reconstruction models is improving the computational throughput in future deployment scenarios. Possible approaches to improve inference throughput could include more efficient model formulations and implementations [57], sparsity and quantization. Therefore, we compare the inference scalability of the rule-based PF implementation on CPU and the proposed scalable GNN implementation on GPU with respect to increasing input multiplicity, with the results summarized in Fig. 10. Inference scaling tests of the GNN model are tested on a small, 8 GB consumer GPU (Nvidia RTX2060S) to emulate a resource-constrained scenario in edge deployment.

The rule-based version is run with 25, 50, 100 and 200 generated  $\pi^-$  particles per event for 10 events. We generated three different sets of these events varying the random seed each time, and measured the runtime of the particle-flow module in Key4HEP. The runtime of the baseline increases nonlinearly with increasing particle multiplicity, and segmentation faults occur for more than approximately 200 particles per event, possibly due to the baseline code and configuration not being tuned for such a high number of particles. Currently, it is only possible to track memory usage of the full baseline reconstruction chain, not individual algorithms. However, we find that the maximum memory requirements increase approximately linearly from about 2 GB for 25 particles to about 8 GB for 200 particles.

The GNN model is run with a varying input size on an 8 GB consumer GPU multiple times to average over random fluctuations. Both the batch size  $B$ , i.e., the number of batched events ( $B \in \{1, 2, 4, 8, 16\}$ ) and the event size  $N$ , i.e., the number of input elements per event ( $N = 256n$  for  $n \in [1, 40]$ ) are varied independently. The inference runtime scales approximately linearly with  $N$ . We note that  $B \gg 1$  is required

to saturate the GPU, but this is highly model and device specific. The maximum GPU memory of our proposed MLPF algorithm varies between about 300 MB for the smallest tested configurations ( $B = 1, N = 256$ ) to about 4.5 GB for the largest tested configuration ( $B = 16, N = 10240$ ), with allocations scaling in a stepwise manner due to the LSH binning.

We confirm that the ML-based reconstruction based on the LSH-binned GNN for full event reconstruction avoids the quadratic scaling present in a typical rule-based full event reconstruction algorithm. It is likely that additional effort will be able to significantly improve the performance of both algorithms. For example, in CMS, a  $k$ -dimensional tree [60] is used to avoid quadratic scaling [11]. With aggressive quantization, the throughput of ML models can be significantly improved with a negligible performance degradation [61].

At this point, it is not meaningful to compare the absolute throughput of the rule-based model on a CPU and the ML-based model on a GPU, as neither method is particularly optimized for throughput, and the comparison is strongly affected by the specific choice of hardware.

The training scalability is also tested, with results presented in Fig. 10, on three different HPC centers with different accelerator hardware: Nvidia H100 GPUs from Flatiron Institute’s CoreSite cluster [62], AMD MI250 GPUs from the LUMI supercomputer [63], and Intel Habana Gaudi HPUs from the Voyager supercomputer [64]. The LUMI supercomputer features GPU nodes with 64-core AMD Trento CPUs and four AMD MI250X cards, each card consisting of two accelerator chips. Voyager is an NSF-funded supercomputer with 42 first-generation Intel Habana Gaudi (training) nodes, each with eight cards, two first-generation Intel Habana Goya (inference) nodes, a 400 GbE Arista switch, and 3 PB of Ceph file system available at the San Diego Supercomputer Center located at the University of California San Diego. Intel Habana also provided additional access to eight Gaudi2 nodes in an HLS-Gaudi2 Deep Learning Server [65]. For the training tests, there are some differences in the configuration on the HPCs. For the AMD processors, multi-card training was implemented with a mirrored worker configuration, while for the Nvidia and Habana processors, Horovod [58] was used. Events were zero-padded to a regular size of 512 elements per event. A batch size of 250 events per device was used for the Nvidia, AMD, and Habana Gaudi2 processors, while a batch size of 100 per device was used for the Habana Gaudi processors. We observe nearly linear scaling or better for all processors. The scaling for the Habana Gaudi1 (Gaudi2) processors is enhanced by the all-to-all non-blocking intra-node network connection, where each processor has a 100 (300) Gb network connection to every other processor [64, 65].

## 4 Conclusions

We have used a realistic simulation for the CLIC detector, conceptually similar to existing and future detector designs, to develop scalable machine learning models for full event particle flow reconstruction. We compare two scalable machine learning (ML) models: a graph neural network (GNN) model that uses locality sensitive hashing (LSH), and a kernel-based transformer using large-scale hyperparameter optimization

(HPO). Both of these models avoid quadratic scaling in memory and computation, thus allowing the processing of events from highly granular detectors. We find that a GNN model significantly outperforms the kernel-based transformer alternative and the baseline Pandora-based particle-flow (PF) on the basis of individual particles as well as event-level quantities such as jets and total 3-momentum. Improved particle-level and event-level reconstruction can result in significant improvements for future flagship analyses such as those involving the Higgs boson decaying to bottom quarks. At the same time, the flexible and learnable LSH approach allows one to process complex events expected at future detectors with up to  $10^4$  particles per event efficiently, while supporting various existing hardware accelerators such as Nvidia, AMD and Habana without additional porting effort. Our work contributes to the existing body of research by proposing a new, challenging open dataset for particle flow reconstruction studies, by defining relevant benchmarks and by identifying efficient models that can solve these benchmarks better than existing rule-based algorithms without additional tuning.

Further research is possible in several directions. First, it would be useful to repeat this exercise using the simulation from detectors that are taking data in Run 3 of the Large Hadron Collider (LHC), e.g. in CMS, to study the performance of the reconstruction in more realistic conditions, and also study the reconstruction performance on real data. Second, we are currently using a simple particle-based loss, while the use of contrastive-adversarial learning methods may allow one to account for event-level discrepancies more effectively [66]. Third, the hypergraph model [22] shows promising physics performance but currently only supports small input sequences. It may be interesting to extend the hypergraph construction over dynamically binned events using the LSH approach. While the current model works at the level of tracks and calorimeter clusters, our proposed approach scales naturally to cases where one considers the raw detector hits directly as an input, possibly allowing direct event reconstruction without having to tune clustering or tracking algorithms. It may also be useful to construct features using semi-supervised or unsupervised learning from real data, to reduce the reliance on simulated datasets for supervised learning [67]. Furthermore, large-context models are continuously improving, and it may be interesting to apply the latest developments such as FLASHATTENTION [68, 69] on the models in this paper. For improving throughput on e.g. CPUs, quantization has shown promise, and it may be interesting to investigate if this can be repeated for the type of models proposed for PF reconstruction. Finally, it is important to integrate the proposed ML-based reconstruction models into reconstruction frameworks such as CMSSW and KEY4HEP.

## Data availability

Our datasets are published following the findable, accessible, interoperable, and reusable principles. The datasets are available in [50], the results including the trained model weight files in [70]

## Code availability

Our code is published following the findable, accessible, interoperable, and reusable principles. The code used for analysis is available in [71].

## Authorship statement

The authors contributed according to the contributor roles taxonomy (CRediT) categories as follows. **Joosep Pata**: conceptualization, methodology, software, validation, data curation, writing (original draft), funding acquisition, resources. **Eric Wulff**: methodology, software, data curation, writing (review and editing), resources. **Farouk Mokhtar**: methodology, software, investigation, supervision. **David Southwick**: software, investigation. **Mengke Zhang**: validation, investigation. **Maria Girone**: funding acquisition, resources. **Javier Duarte**: conceptualization, software, resources, funding acquisition, supervision, writing (review and editing).

## Acknowledgements

Eric Wulff and David Southwick were supported by CoE RAISE. The CoE RAISE project has received funding from the European Union’s Horizon 2020 – Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733.

Joosep Pata was supported by the grant PSG864 of the Estonian Research Council.

Javier Duarte, Farouk Mokhtar, and Mengke Zhang were supported by U.S. Department of Energy (DOE) Award Nos. DE-SC0021187 and DE-SC0021396 (FAIR4HEP) and U.S. National Science Foundation (NSF) Cooperative Agreement OAC-2117997 (A3D3). Farouk Mokhtar was also supported by a UCSD HDSI fellowship and an IRIS-HEP fellowship through NSF Cooperative Agreement OAC-1836650. The use of Intel Habana processors and additional support for Javier Duarte was facilitated by NSF Award No. 2005369 (Voyager). We also thank Amit Majumdar, Mahidhar Tatineni, Paul Rodriguez, and Marty Kandes at the San Diego Supercomputer Center and Leonid Goldgeisser, Charles Chen, Chen Levkovich, Leon Si Tran, and Chenna Bayapureddy at Intel Habana for technical support and assistance in producing results for the Gaudi2 processors.

The authors gratefully acknowledge the computing time granted through JARA on the supercomputer JURECA [53] at Forschungszentrum Jülich. We acknowledge the Estonian Scientific Compute Infrastructure and NICPB for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium through Estonian Scientific Compute Infrastructure.

Datasets and software published following the findable, accessible, interoperable, and reusable (FAIR) principles support open, transparent, and reusable research, and therefore we thank our colleagues at the relevant collaborations for releasing the Key4HEP, Marlin, EDM4HEP and CLICdp software to the wider community, and encourage the wider adoption of the FAIR principles.

We are grateful to the reviewers and the editor for their careful reading of the manuscript and their comments which significantly improved the clarity of this paper.

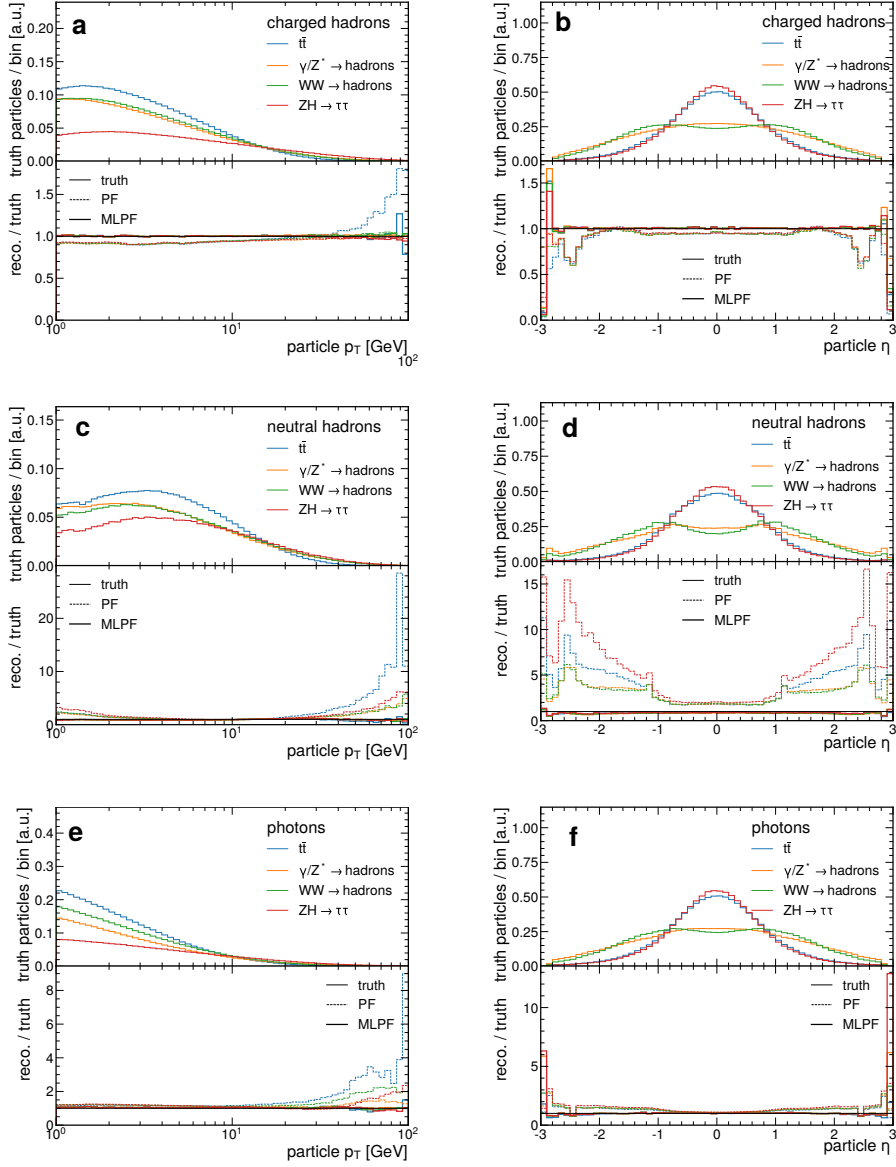
We would also like to thank Nilotpall Kakati and Etienne Dreyer for cross-checks of our published code and datasets.

## **Competing interests**

The authors declare no competing interests.

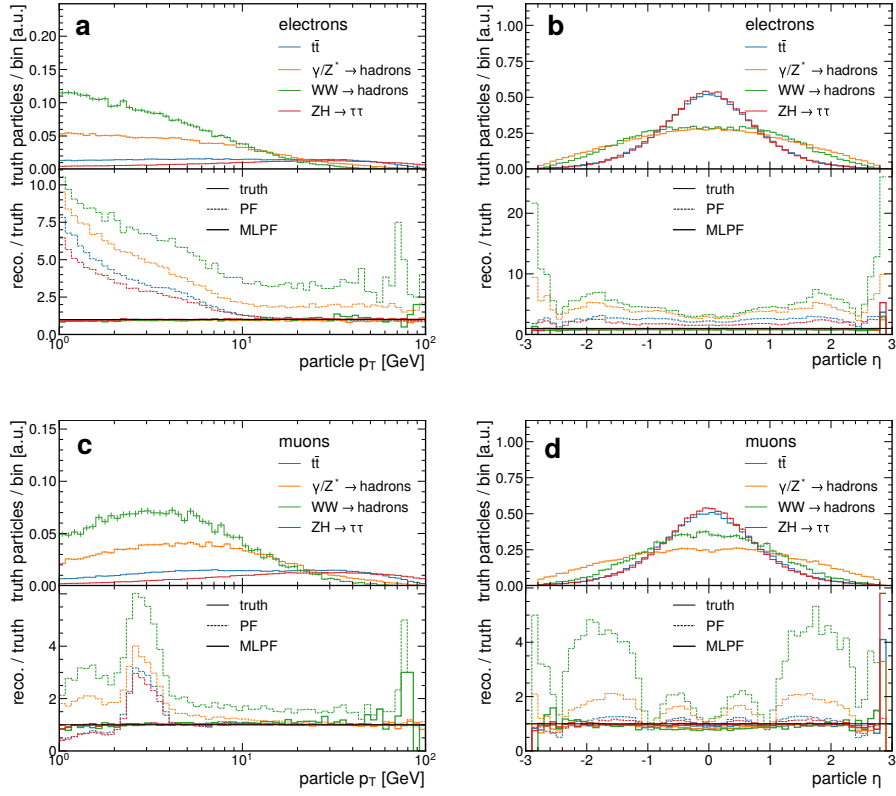


**Fig. 5:** The generated (truth) and reconstructed kinematic distributions for baseline particle flow (PF) and the proposed machine-learned particle flow (MLPF) algorithm.



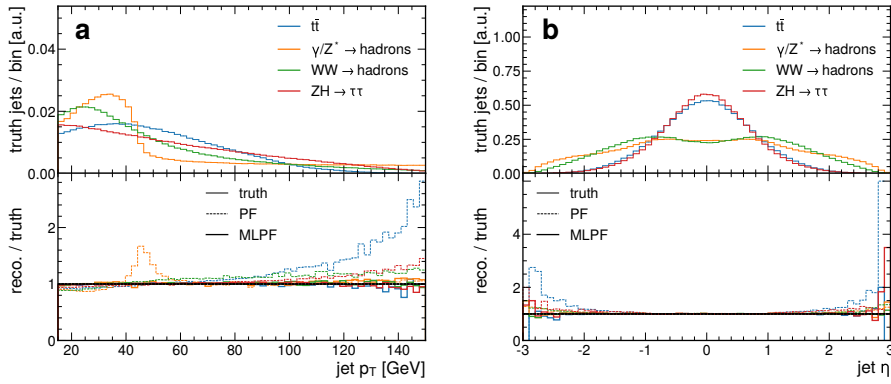
The distribution of generated particles for  $t\bar{t}$ ,  $q\bar{q}$ ,  $WW$  and  $ZH$ , and the ratio between the reconstructed and generated particle distributions for the baseline algorithm (dashed line) and the machine-learned algorithm (solid line) for charged hadrons in (a) and (b), neutral hadrons in (c) and (d), photons in (e) and (f).

**Fig. 6:** The generated (truth) and reconstructed kinematic distributions for baseline particle flow (PF) and the proposed machine-learned particle flow (MLPF) algorithm for electrons and muons.



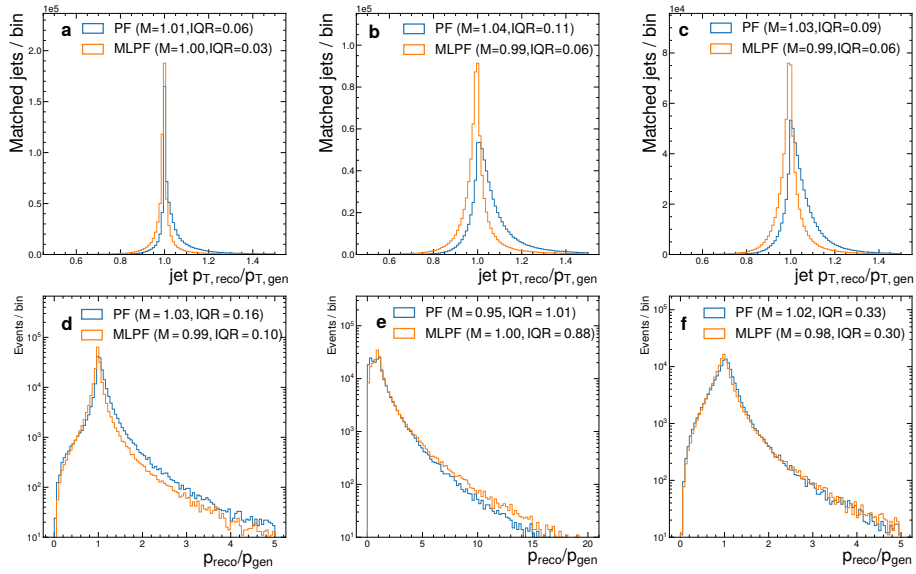
The distribution of generated particles for  $t\bar{t}$ ,  $q\bar{q}$ ,  $WW$  and  $ZH$ , and the ratio between the reconstructed and generated particle distributions for the baseline algorithm (dashed line) and the machine-learned algorithm (solid line) for electrons hadrons in (a) and (b) and muons in (c) and (d).

**Fig. 7:** The generated (truth) and reconstructed kinematic distributions for baseline particle flow (PF) and the proposed machine-learned particle flow (MLPF) algorithm for jets.



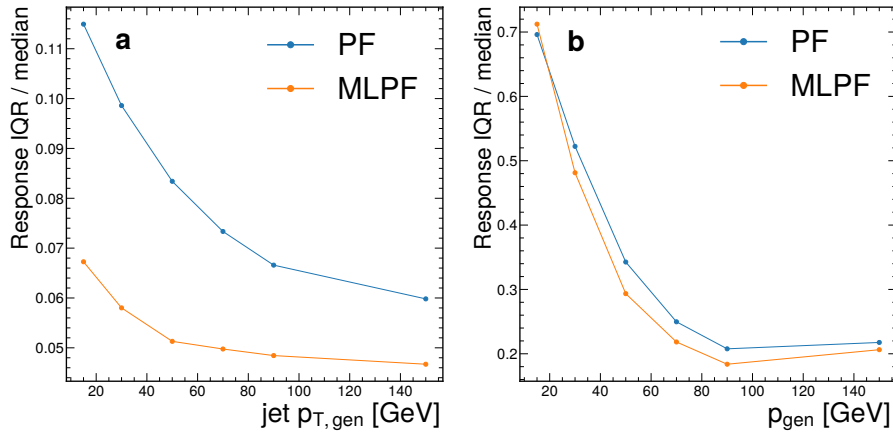
The  $p_T$  (a) and  $\eta$  (b) distributions of generated jets for  $t\bar{t}$ ,  $q\bar{q}$ ,  $WW$  and  $ZH$ , and the ratio between the reconstructed and generated jet distributions for the baseline algorithm (dashed line) and the machine-learned algorithm (solid line).

**Fig. 8:** Jet and total 3-momentum response in the validation samples, comparing the baseline particle flow (PF) and the machine-learned particle flow (MLPF).



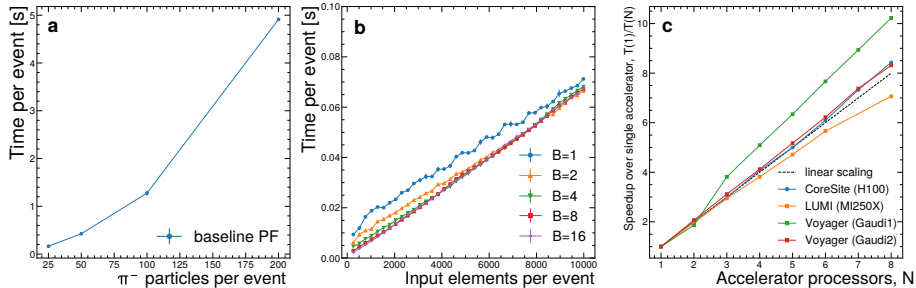
The jet response for ZH samples in (a), for WW samples in (b) and for  $t\bar{t}$  with hadronic overlay on (c). The total 3-momentum response is shown for ZH samples in (d), for WW samples in (e) and for  $t\bar{t}$  with hadronic overlay on (f). We also show the median of the distribution ( $M$ ), for which values closer to unity are better, and the interquartile range (IQR), for which lower values are better.

**Fig. 9:** The jet and total 3-momentum resolution, comparing the baseline particle flow (PF) with the machine-learned algorithm (MLPF).



The jet  $p_T$  response (a) and total 3-momentum response (b), comparing the baseline particle flow (PF) with the machine-learned algorithm (MLPF), parameterised as the interquartile range (IQR) divided by the median the  $t\bar{t}$  sample with hadronic overlay, evaluated in bins of generated jet  $p_T$  (total 3-momentum). Lower values correspond to better resolution.

**Fig. 10:** Computational performance of the baseline particle flow (PF) and the proposed machine-learned algorithm in inference and training.



Absolute timing of the baseline PF (a) and the proposed GNN-based algorithm (b), illustrating the scaling with respect to input particle or element (track and cluster) multiplicity. The absolute processing time of the baseline algorithm on a single CPU thread is approximately 1,s/event at the reference point of 100 charged pions, which corresponds to approximately  $96 \pm 3$  tracks and  $170 \pm 20$  clusters. The runtime of the machine-learned particle flow (MLPF) algorithm on the Nvidia RTX2060S GPU at the reference point of  $N = 256$  elements, batch size  $B = 16$  is 2 ms/event. This should not be interpreted as a complete, exhaustive and final computational benchmark, nor should it be used to claim 500-fold improvement in throughput from MLPF, as the absolute timing of any algorithm is heavily dependent on optimizations and hardware. In [21], the runtime of a heavily optimized version of the baseline algorithm was measured at 9 ms/event, whereas the MLPF model at 320 ms/event, both on a single CPU thread. On panel (c), we demonstrate the scaling of the training performance across multiple devices on a single machine on Nvidia, AMD, and Habana processor cards from the CoreSite, LUMI, and Voyager supercomputers, respectively. For the multi-device scaling test, Horovod [58] was used on CoreSite and Voyager, while the TENSORFLOW MirroredStrategy [59] was used on LUMI. The batch size was adjusted to fit a single device and the dataset was fully cached in RAM.

## References

- [1] CELLO Collaboration. An analysis of the charged and neutral energy flow in  $e^+e^-$  hadronic annihilation at 34 GeV, and a determination of the QCD effective coupling constant. *Phys. Lett. B* **113**, 427 (1982). URL [https://doi.org/10.1016/0370-2693\(82\)90778-X](https://doi.org/10.1016/0370-2693(82)90778-X).
- [2] ALEPH Collaboration. Performance of the ALEPH detector at LEP. *Nucl. Instrum. Meth. A* **360**, 481 (1995). URL [https://doi.org/10.1016/0168-9002\(95\)00138-7](https://doi.org/10.1016/0168-9002(95)00138-7).
- [3] DELPHI Collaboration. Performance of the DELPHI detector. *Nucl. Instrum. Meth. A* **378**, 57 (1996). URL [https://doi.org/10.1016/0168-9002\(96\)00463-9](https://doi.org/10.1016/0168-9002(96)00463-9).
- [4] ZEUS Collaboration. Measurement of the diffractive structure function  $F_2(D(4))$  at HERA. *Eur. Phys. J. C* **1**, 81 (1998). URL <https://doi.org/10.1007/s100520050063>.
- [5] ZEUS Collaboration. Measurement of the diffractive cross-section in deep inelastic scattering using ZEUS 1994 data. *Eur. Phys. J. C* **6**, 43 (1999). URL <https://doi.org/10.1007/PL00021606>.
- [6] Bocci, A. *et al.* Study of jet energy resolution at CDF. *Int. J. Mod. Phys. A* **16S1A**, 255 (2001). URL <https://doi.org/10.1142/S0217751X01006632>.
- [7] Connolly, A. L. A Search for Supersymmetric Higgs Bosons in the Di-tau Decay Mode in  $p\bar{p}$  Collisions at 1.8 TeV (2003). URL <https://doi.org/10.2172/15017134>.
- [8] CDF Collaboration. Measurement of  $\sigma(p\bar{p} \rightarrow Z) \cdot \text{Br}(Z \rightarrow 2\tau)$  in  $p\bar{p}$  collisions at  $\sqrt{s} = 1.96$  TeV. *Phys. Rev. D* **75**, 092004 (2007). URL <https://doi.org/10.1103/PhysRevD.75.092004>.
- [9] D0 Collaboration. Measurement of  $\sigma(p\bar{p} \rightarrow Z + X) \text{Br}(Z \rightarrow \tau^+\tau^-)$  at  $\sqrt{s} = 1.96$  TeV. *Phys. Lett. B* **670**, 292 (2009). URL <https://doi.org/10.1016/j.physletb.2008.11.010>.
- [10] CMS Collaboration. The CMS Experiment at the CERN LHC. *JINST* **3**, S08004 (2008). URL <https://doi.org/10.1088/1748-0221/3/08/S08004>.
- [11] CMS Collaboration. Particle-flow reconstruction and global event description with the CMS detector. *JINST* **12**, P10003 (2017). URL <https://doi.org/10.1088/1748-0221/12/10/P10003>.
- [12] ATLAS Collaboration. Jet reconstruction and performance using particle flow with the ATLAS detector. *Eur. Phys. J. C* **77**, 466 (2017). URL <https://doi.org/10.1140/epjc/s10052-017-5031-2>.

- [13] H1 Collaboration. Measurement of charged particle multiplicity distributions in DIS at HERA and its implication to entanglement entropy of partons. *Eur. Phys. J. C* **81**, 212 (2021). URL <https://doi.org/10.1140/epjc/s10052-021-08896-1>.
- [14] High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report V. 0.1 **4/2017** (2017).
- [15] Selvaggi, M. Physics requirements for the FCC-hh calorimeter system. *J. Phys. Conf. Ser.* **1162**, 012010 (2019).
- [16] Abada, A. *et al.* FCC-hh: The hadron collider. *Eur. Phys. J. ST* **228**, 755 (2019).
- [17] Di Bello, F. A. *et al.* Towards a Computer Vision Particle Flow. *Eur. Phys. J. C* **81**, 107 (2021).
- [18] Kieseler, J. Object condensation: one-stage grid-free multi-object reconstruction in physics detectors, graph and image data. *Eur. Phys. J. C* **80**, 886 (2020). URL <https://doi.org/10.1140/epjc/s10052-020-08461-2>.
- [19] Pata, J. *et al.* MLPF: Efficient machine-learned particle-flow reconstruction using graph neural networks. *Eur. Phys. J. C* **81**, 381 (2021). URL <https://doi.org/10.1140/epjc/s10052-021-09158-w>.
- [20] Pata, J. *et al.* Machine Learning for Particle Flow Reconstruction at CMS. *J. Phys. Conf. Ser.* **2438**, 012100 (2023). URL <https://doi.org/10.1088/1742-6596/2438/1/012100>.
- [21] Mokhtar, F. *et al.* Progress towards an improved particle flow algorithm at CMS with machine learning (2023). URL <https://doi.org/10.48550/arXiv.2303.17657>.
- [22] Di Bello, F. A. *et al.* Reconstructing particles in jets using set transformer and hypergraph prediction networks. *Eur. Phys. J. C* **83**, 596 (2023). URL <https://doi.org/10.1140/epjc/s10052-023-11677-7>.
- [23] Bhattacharya, S. *et al.* GNN-based end-to-end reconstruction in the CMS Phase 2 High-Granularity Calorimeter. *J. Phys. Conf. Ser.* **2438**, 012090 (2023). URL <https://doi.org/10.1088/1742-6596/2438/1/012090>.
- [24] Kahn, J. *et al.* Learning tree structures from leaves for particle decay reconstruction. *Mach. Learn. Sci. Tech.* **3**, 035012 (2022).
- [25] Pardinas, G. *et al.* GNN for Deep Full Event Interpretation and Hierarchical Reconstruction of Heavy-Hadron Decays in Proton–Proton Collisions. *Comput. Softw. Big Sci.* **7**, 12 (2023).
- [26] Gaede, F. *et al.* EDM4hep and podio – The event data model of the Key4hep project and its implementation **251**, 03026 (2021). URL <https://doi.org/10.1051/epjconf/202125103026>.



- [27] Wilkinson, M. D. *et al.* The FAIR guiding principles for scientific data management and stewardship. *Sci. Data* **3**, 160018 (2016).
- [28] Chen, Y. *et al.* A FAIR and AI-ready Higgs boson decay dataset. *Sci. Data* **9**, 31 (2022).
- [29] Duarte, J. *et al.* FAIR AI Models in High Energy Physics. *Mach. Learn.: Sci. Technol.* **4** (2023) 045062 (2022).
- [30] Kramer, T. Track parameters in LCIO. Tech. Rep. LC-DET-2006-004 (2006).
- [31] Lin, T.-Y. *et al.* Focal loss for dense object detection 2980–2988 (2017). URL <https://doi.org/10.48550/arXiv.1708.02002>.
- [32] Huber, P. J. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics* **35**, 73 – 101 (1964). URL <https://doi.org/10.1214/aoms/1177703732>.
- [33] Wang, S. *et al.* Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (2020).
- [34] Brown, T. *et al.* Language models are few-shot learners **33**, 1877 (2020).
- [35] Touvron, H. *et al.* LLaMA: Open and efficient foundation language models (2023). [2302.13971](https://arxiv.org/abs/2302.13971).
- [36] Kitaev, N. *et al.* Reformer: The efficient transformer (2020). URL <https://doi.org/10.48550/arXiv.2001.04451>.
- [37] Choromanski, K. *et al.* Rethinking attention with performers (2020). URL <https://doi.org/10.48550/arXiv.2009.14794>.
- [38] Abadi, M. *et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015). [1603.04467](https://arxiv.org/abs/1603.04467).
- [39] TensorFlow Developers. TensorFlow (2023). URL <https://doi.org/10.5281/zenodo.7753622>.
- [40] Bierlich, C. *et al.* A comprehensive guide to the physics and usage of PYTHIA 8.3. *SciPost Phys. Codeb.* (2022). URL <https://doi.org/10.21468/SciPostPhysCodeb.8>.
- [41] Ganis, G., Helsen, C. & Völkl, V. Key4hep, a framework for future HEP experiments and its use in FCC. *Eur. Phys. J. Plus* **137**, 149 (2022). URL <https://doi.org/10.1140/epjp/s13360-021-02213-1>.
- [42] CLIC Collaboration. CLICdet: The post-CDR CLIC detector model. CLIC Detector and Physics Study Note (2017). URL <https://cds.cern.ch/record/>

2254048.

- [43] Arominski, D. *et al.* A detector for CLIC: main parameters and performance (2018). [1812.07337](https://doi.org/10.1016/j.nima.2005.11.138).
- [44] Gaede, F. Marlin and LCCD: Software tools for the ILC. *Nucl. Instrum. Meth. A* **559**, 177 (2006). URL <https://doi.org/10.1016/j.nima.2005.11.138>.
- [45] Marshall, J. S. *et al.* The Pandora software development kit for particle flow calorimetry. *J. Phys. Conf. Ser.* **396**, 022034 (2012). URL <https://doi.org/10.1088/1742-6596/396/2/022034>.
- [46] Marshall, J. S. *et al.* Performance of particle flow calorimetry at CLIC. *Nucl. Instrum. Meth. A* **700**, 153 (2013). URL <https://doi.org/10.1016/j.nima.2012.10.038>.
- [47] Marshall, J. S. *et al.* The Pandora software development kit for pattern recognition. *Eur. Phys. J. C* **75**, 439 (2015). URL <https://doi.org/10.1140/epjc/s10052-015-3659-3>.
- [48] Petrič, M. *et al.* Detector simulations with dd4hep. *J. Phys.: Conf. Ser.* **898**, 042015 (2017).
- [49] TensorFlow Datasets, a collection of ready-to-use datasets. URL <https://github.com/tensorflow/datasets/>.
- [50] Pata, J. *et al.* Simulated datasets for detector and particle flow reconstruction: CLIC detector (2023). URL <https://doi.org/10.5281/zenodo.8260741>.
- [51] Cacciari, M. *et al.* The anti- $k_T$  jet clustering algorithm. *JHEP* **04**, 063 (2008).
- [52] Cacciari, M. *et al.* FastJet user manual. *Eur. Phys. J. C* **72**, 1896 (2012).
- [53] Jülich Supercomputing Centre. JURECA: Data Centric and Booster Modules implementing the Modular Supercomputing Architecture at Jülich Supercomputing Centre. *J. Large-Scale Res. Facilities* **7** (2021). URL <https://doi.org/10.17815/jlsrf-7-182>.
- [54] Wulff, E. *et al.* Hyperparameter optimization of data-driven AI models on HPC systems. *J. Phys. Conf. Ser.* **2438**, 012092 (2023).
- [55] Li, L. *et al.* A system for massively parallel hyperparameter tuning **2**, 230 (2020).
- [56] Dawson, S. *et al.* Report of the Topical Group on Higgs Physics for Snowmass 2021: The Case for Precision Higgs Physics (2022).
- [57] Gu, A. *et al.* Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752* (2023).

- [58] Sergeev, A. *et al.* Horovod: fast and easy distributed deep learning in TensorFlow (2018). [1802.05799](https://arxiv.org/abs/1802.05799).
- [59] Tensorflow Developers. Tensorflow MirroredStrategy (2024). URL <http://web.archive.org/web/20240114172228/https://www.tensorflow.org/api-docs/python/tf/distribute/MirroredStrategy>.
- [60] Bentley, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM* **18**, 509 (1975).
- [61] Dettmers, T. *et al.* GPT3.int8(): 8-bit matrix multiplication for transformers at scale **35**, 30318 (2022). URL <https://doi.org/10.48550/arXiv.2208.07339>.
- [62] Flatiron Institute. CoreSite Cluster (2023). URL <http://web.archive.org/web/20231025020233/https://www.simonsfoundation.org/2022/11/14/new-flatiron-institute-supercomputer-the-most-power-efficient-ever-built/>.
- [63] LUMI Consortium. LUMI Supercomputer (2023). URL <http://web.archive.org/web/20231230052719/https://www.lumi-supercomputer.eu/lumis-full-system-architecture-revealed/>.
- [64] Amaro, R. E. *et al.* Voyager – An Innovative Computational Resource for Artificial Intelligence & Machine Learning Applications in Science and Engineering. *Practice and Experience in Advanced Research Computing* 278 (2023). URL <https://doi.org/10.1145/3569951.3597597>.
- [65] Intel Habana. HLS-Gaudi2 Deep Learning Server (2022). URL <http://web.archive.org/web/20221228142747/https://habana.ai/wp-content/uploads/2022/09/HLS-Gaudi2-Datasheet-Aug-2022.pdf>.
- [66] Huang, T. *et al.* Learning to measure the point cloud reconstruction loss in a representation space 12208–12217 (2023). URL [10.1109/CVPR52729.2023.01175](https://arxiv.org/abs/10.1109/CVPR52729.2023.01175).
- [67] Kishimoto, T. *et al.* Pre-training strategy using real particle collision data for event classification in collider physics (2023). URL <https://doi.org/10.48550/arXiv.2312.06909>.
- [68] Dao, T. *et al.* FlashAttention: Fast and memory-efficient exact attention with IO-awareness (2022). URL <https://doi.org/10.48550/arXiv.2205.14135>.
- [69] Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning (2023). URL <https://doi.org/10.48550/arXiv.2307.08691>.
- [70] Pata, J. *et al.* MLPF results on the simulated CLIC dataset, v2024.01 (2023). URL <https://doi.org/10.5281/zenodo.10567397>.
- [71] Pata, J. *et al.* jpata/particleflow: MLPF training with CLIC simulation (2023). URL <https://doi.org/10.5281/zenodo.10893930>.