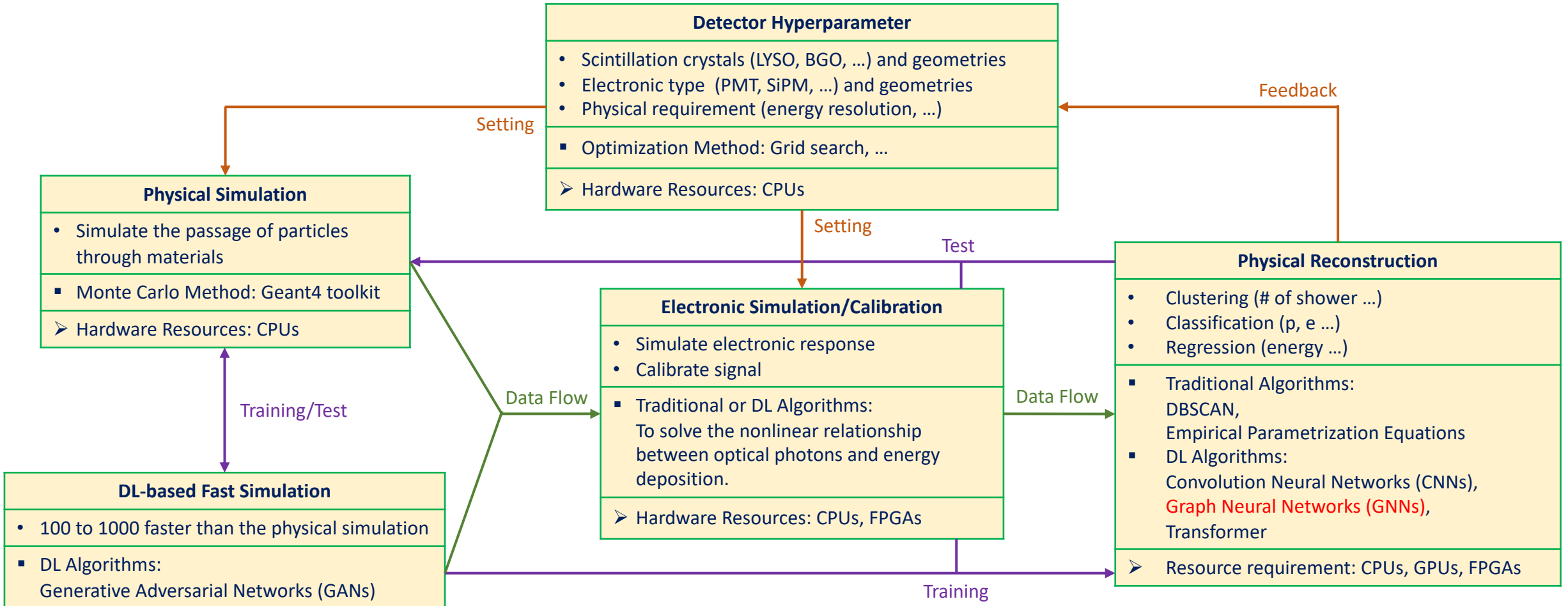
Calorimeter Design Tool in High Energy Physics

Hsin-Yi Chou
July 05, 2024

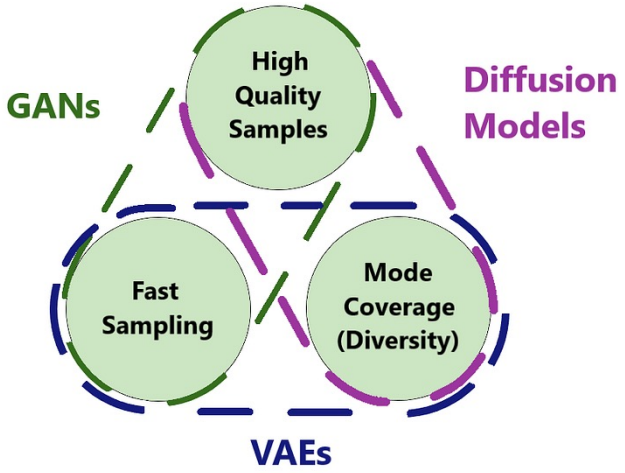


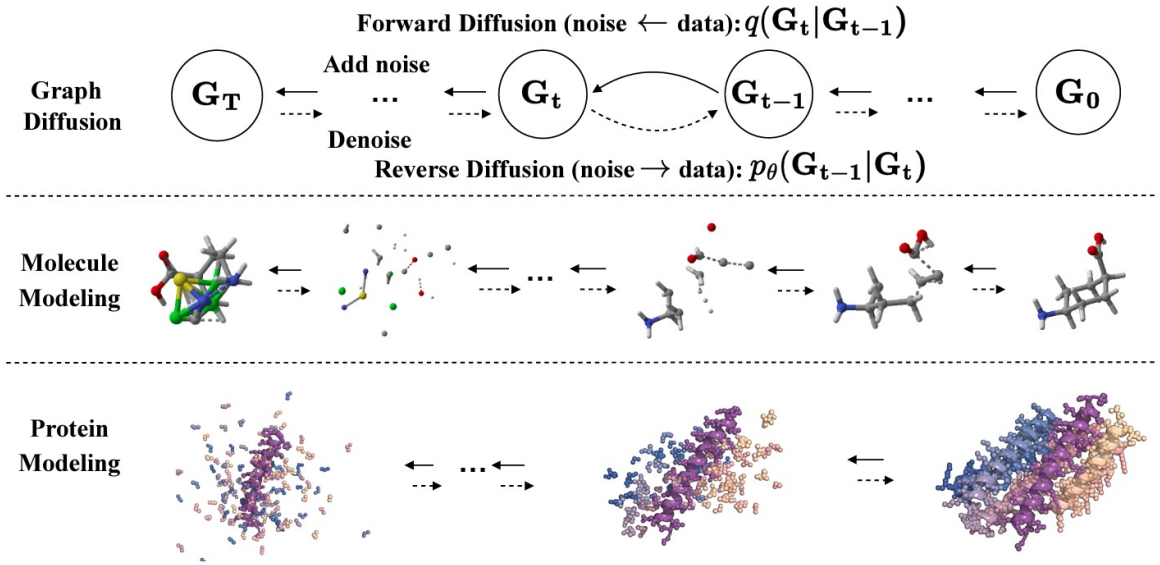
- **Self Supervised Pre-training Method: Contrastive Learning (CL) and Masked Image model (MIM)**
  Contrastive Masked Autoencoders are Stronger Vision Learners
- **Parameter-Efficient Fine-Turning (PEFT):**
  Parameter-Efficient Fine-Tuning Methods for Pre-trained Language Models: A Critical Review and Assessment
- **Data Augmentation: Adding noise, cropping, flipping, …, GANs**

Generative Diffusion Models on Graphs: Methods and Applications

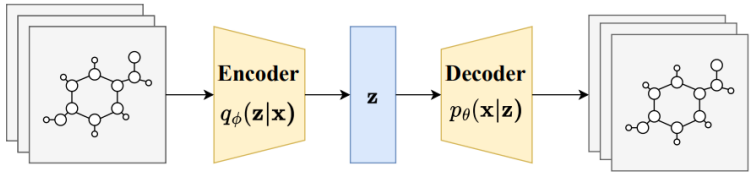
https://arxiv.org/abs/2302.02591

tion [De Cao and Kipf, 2018]. Although these deep generative methods have achieved promising performance, most of them still have several limitations. For example, VAE approaches struggle with the estimation of posterior to generate realistic large-scale graphs and require expensive computation to achieve permutation invariance because of the likelihood-based method [Bond-Taylor *et al.*, 2021]. Most GAN-based methods are more prone to mode collapse with graph-structured data and require additional computation to train a discriminator [De Cao and Kipf, 2018; Wang *et al.*, 2018]. The flow-based generative models are hard to fully learn the structural information of graphs because of the constraints on the specialized architectures [Cornish *et al.*, 2020]. Thus, it is desirable to have a novel generative paradigm for deep generation techniques on graphs.

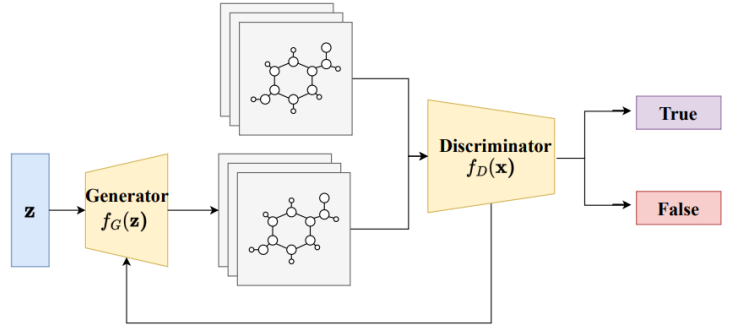




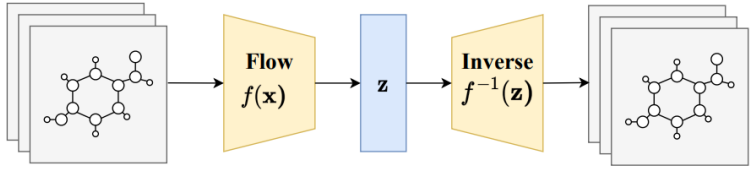
An illustration of diffusion models on molecular and protein generations.



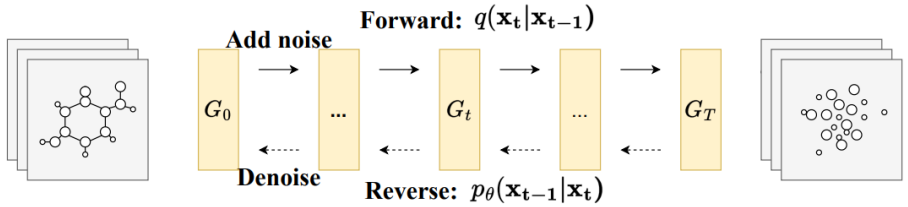
(a) Variational AutoEncoders



(b) Generative Adversarial Networks (GAN)



(c) Normalizing Flows



(d) Diffusion models

Figure 1: Deep Generative Models on Graphs.

A Comprehensive Evaluation of Generative Models in Calorimeter Shower Simulation

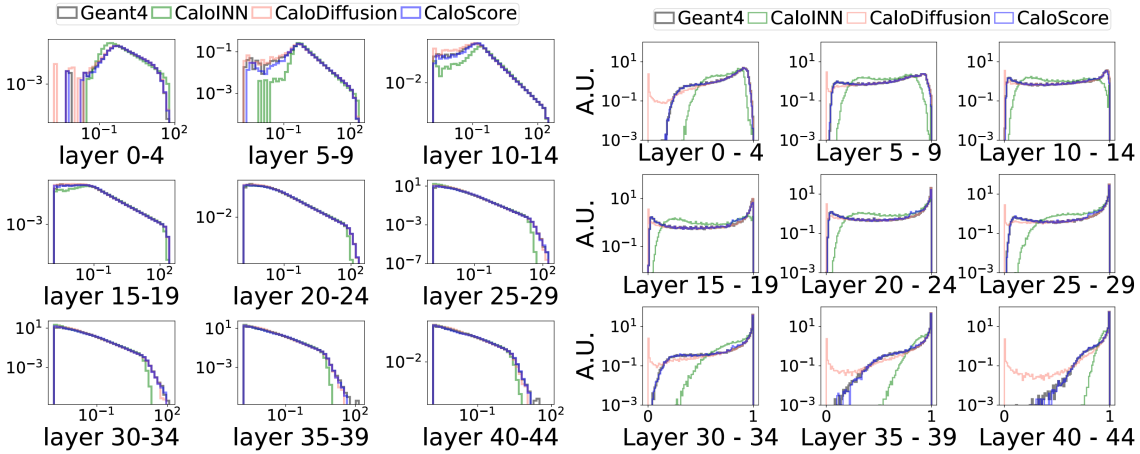
https://arxiv.org/abs/2406.12898

These methods have demonstrated remarkable performance in generating showers faster than Geant4 across various ranges of incident energy for primary particles, although less precise than Geant4. While GAN-based models can generate showers faster than other generative models, it is challenging to train GANs due to the difficulty of converging and 'mode collapse'. VAE-based models can generate samples of calorimeter showers faster than Geant4 and GAN. However, they lack in quality of samples. Nevertheless, VAE-based models are used along with other GAN-based models [9, 10, 20], Normalizing Flow (NF) [25, 17] and Diffusion [49] models. Among the generative methods, NF and Diffusion-based models have shown promising performance in generating shower samples with high fidelity. However, Diffusion-based models are slower in sampling, and NFs require a constraint bijective mapping making flow models restrictive. Refer to Appendix-A for a more comprehensive discussion of these models.

This study fills the gap by evaluating three different state-of-the-art calorimeter shower simulation models using a comprehensive set of metrics. The three models evaluated are CaloDiffusion [5], CaloScore [51] and CaloINN [25]. We chose CaloDiffusion and CaloScore because of their ability to generate high-quality samples and CaloINN because of its faster sample generation capability. Furthermore, these are the three open-sourced implementations that work.

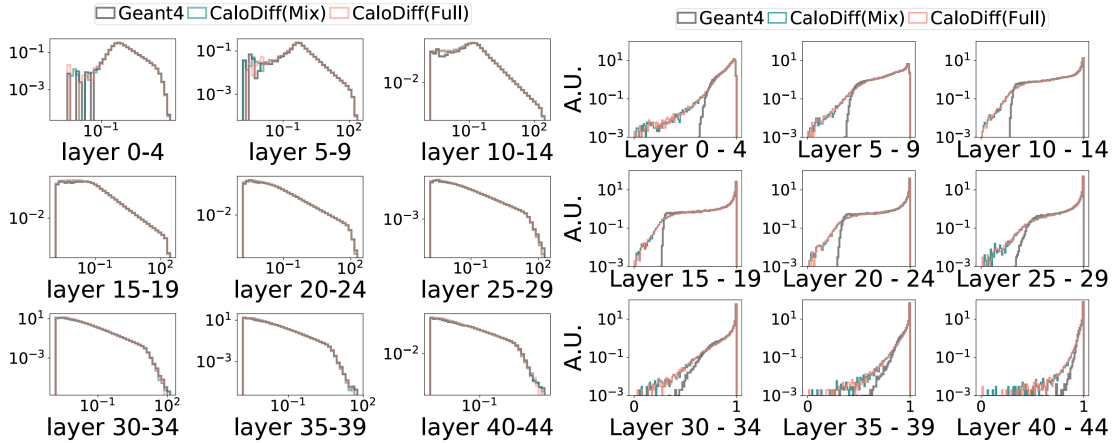
6 Conclusion

"Fast Simulation" using deep-generative models is pivotal in overcoming computational bottlenecks of the calorimeter shower generation. In this study, we conducted a rigorous evaluation of three generative models using standard datasets and a diverse set of metrics derived from physics, computer vision, and statistics. Additionally, we explored the impact of using full vs. mixed precision modes during inference on GPUs. Our evaluation revealed that the CaloDiffusion and CaloScore generative models demonstrate the most accurate simulation of particle showers, yet there remains substantial room for improvement. We identified areas where the evaluated models fell short in



(a) Layer energy distribution (GeV) (b) Distribution of sparsity

Figure 1: Histogram of two physics observables for dataset 2.



(a) Layer energy distribution (GeV) (b) Distribution of sparsity

Figure 5: Comparison of histogram of two physics observables with dataset 3 between full precision and mixed precision mode using CaloDiffusion.

Reconstruction of electromagnetic showers in calorimeters using Deep Learning

https://arxiv.org/abs/2311.17914

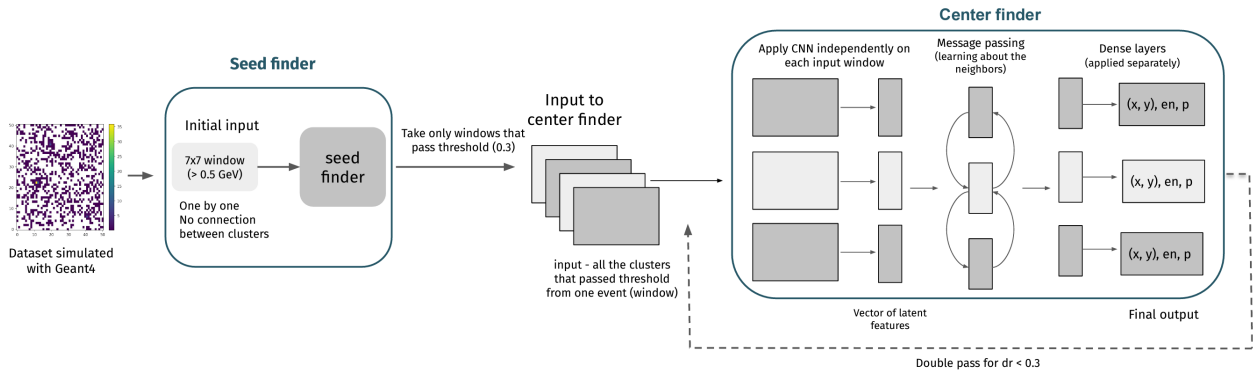


Fig. 6 Flow chart of the DeepCluster model. 7×7 seed windows are first selected around all possible seeds (>0.5 GeV) in the event. They are separately passed as input to the seed-finder NN predicting P_{seedSF} for each seed window. Selected seed windows with $P_{\text{seedSF}} > P^{\text{thr}}_{\text{seedSF}}$ are combined into groups of 4 with their neighbors and passed to center-finder NN which predicts the coordinates x_{reco}, y_{reco}, the energy E_{reco} and a new seed score P_{seedCF} for each seed window.

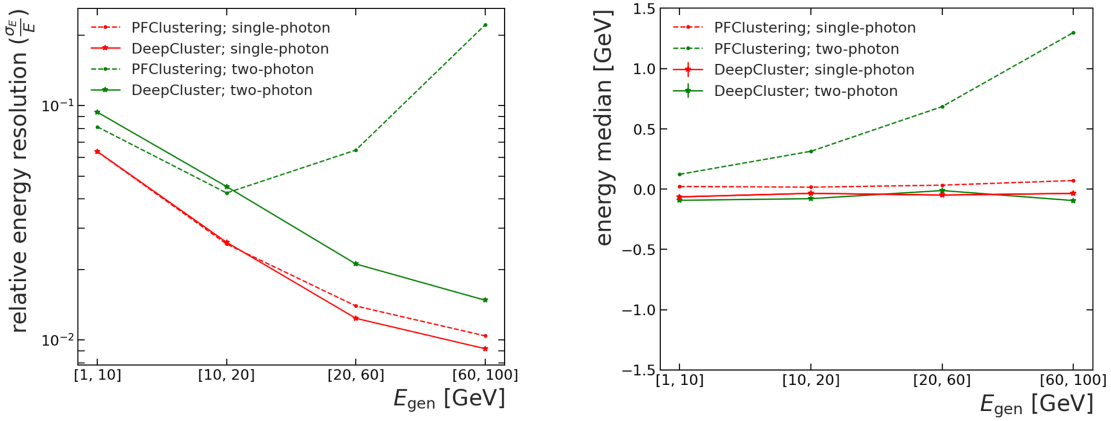


Fig. 9 Relative energy resolution (left) and energy median (right) obtained with the DeepCluster model and PFClustering algorithm applied on the single- and two-photon test datasets. The results are shown in the bins of generated energy E_{gen}.

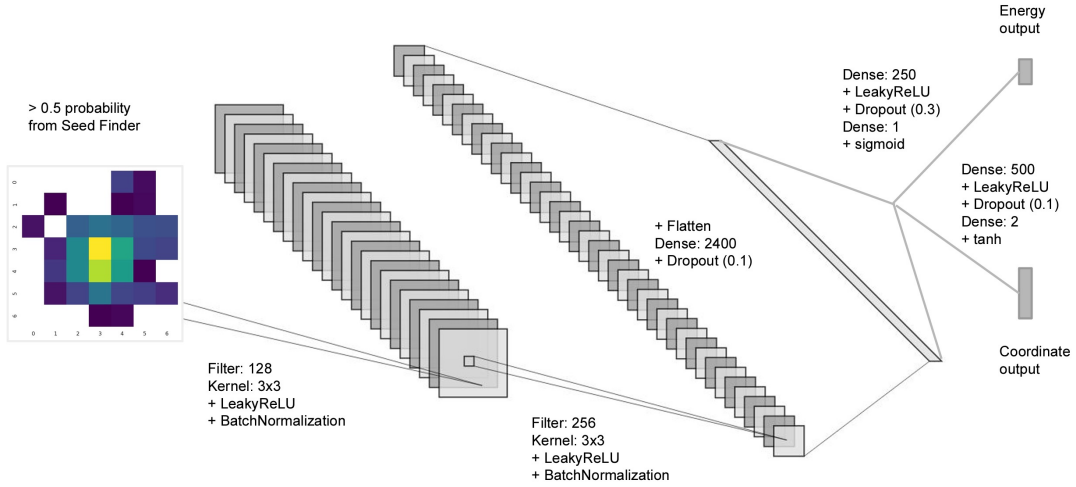


Fig. 3 Center-finder NN architecture. The seed windows are passed separately as input to the network. Each input is processed by two convolutional layers until the vector of summary features is extracted. This vector is passed through one dense layer and further sent separately to two different branches (coordinate and energy predictions). In each branch, it passes through two additional dense layers. Detailed information on the number of nodes at each layer is presented in the figure.

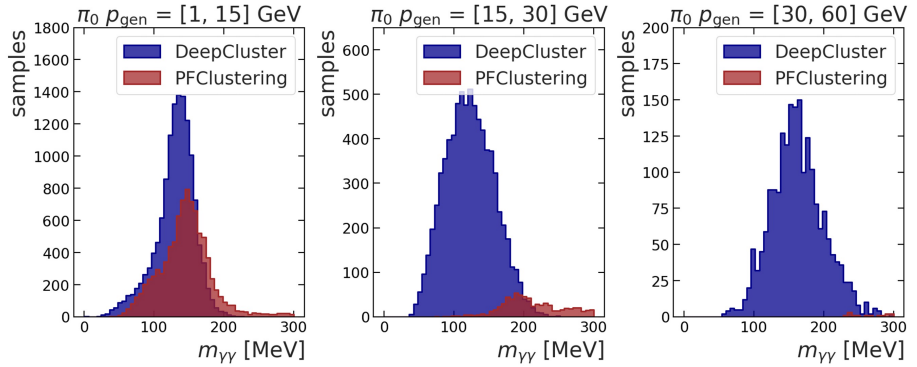


Fig. 12 $m_{\gamma\gamma}$ mass distribution reconstructed with DeepCluster model and PFClustering algorithm on the π_0 dataset. The results are shown in bins of the generated momentum p_{gen} of the π_0.

Design of a SiPM-on-Tile ZDC for the future EIC, and its Performance with Graph Neural Networks

https://arxiv.org/abs/2406.12877

Abstract

We present a design for a high-granularity zero-degree calorimeter (ZDC) for the upcoming Electron-Ion Collider (EIC). The design uses SiPM-on-tile technology and features a novel staggered-layer arrangement that improves spatial resolution. To fully leverage the design's high granularity and non-trivial geometry, we employ graph neural networks (GNNs) for energy and angle regression as well as signal classification. The GNN-boosted performance metrics meet, and in some cases, significantly surpass the requirements set in the EIC Yellow Report, laying the groundwork for enhanced measurements that will facilitate a wide physics program. Our studies show that GNNs can significantly enhance the performance of high-granularity CALICE-style calorimeters by automating and optimizing the software compensation algorithms required for these systems. This improvement holds true even in the case of complicated geometries that pose challenges for image-based AI/ML methods.

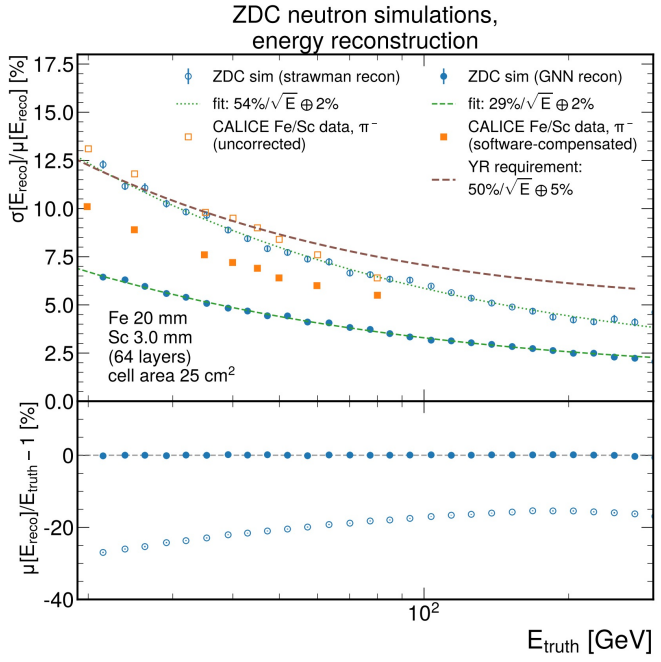


Figure 3: Energy resolution (top row) and scale (bottom row) obtained with the strawman (open symbols) and GNN (filled symbols) for simulated single neutrons. The resolutions are compared to those of the CALICE beamtest [17] (orange squares) with (filled) and without (open) software compensation.

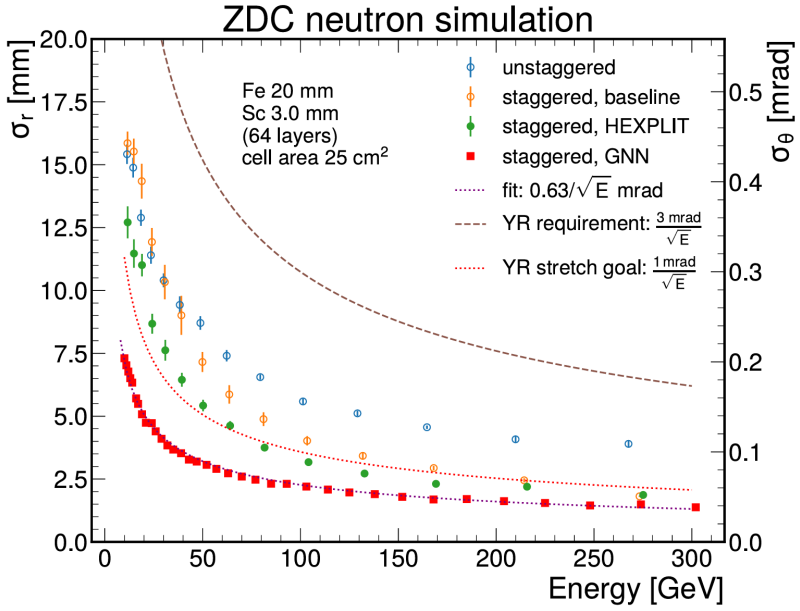


Figure 5: Position resolution for neutrons as a function of the generated energy. Results are shown with an unstaggered layout (blue) and staggered layout with the positions reconstructed with the baseline (orange), HEXPLIT (green), and GNN (red) reconstruction algorithms.

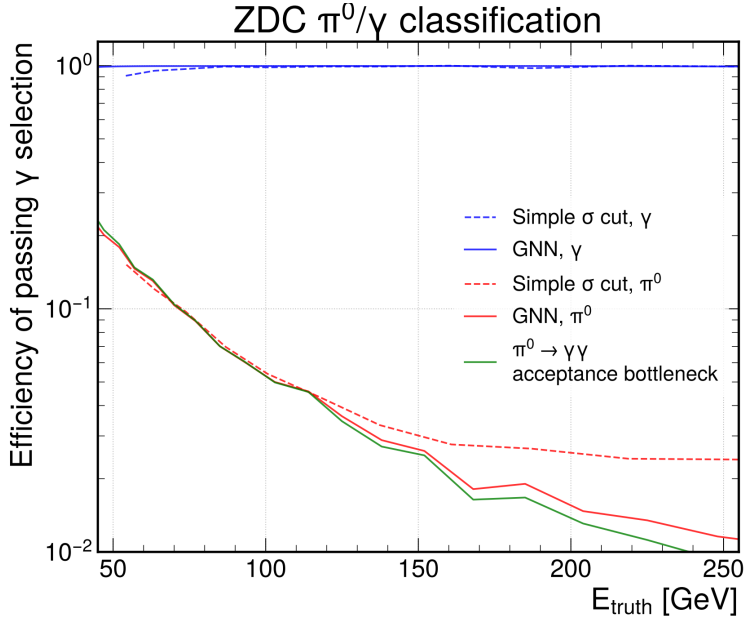


Figure 9: Efficiency of classifying γ (blue) and π^0 (red) as γ. The π^0 rejection efficiency has a bottleneck (green) at lower energies due to events with only one photon hitting the ZDC. The green curve quantifies the fraction of events where both decay photons hit the ZDC, and it hence shows the theoretical best classification performance.

GenGNN: A Generic FPGAFramework for Graph Neural Network Acceleration

https://arxiv.org/abs/2201.08475

1 Introduction

Graph neural networks (GNNs) have become a powerful tool for applying deep learning to solve tasks infilling graph structures. Learning tasks for graphs can be applied at node-level (e.g., presence of protein [1]), edge-level (e.g., drug-drug interactions [2]), and graph-level (e.g, molecular property prediction [3]). Representative applications of GNNs include analysis of social networks and citation networks, recommendation systems, traffic forecasting, LIDAR point cloud segmentation for autonomous driving, high energy particle physics, and molecular representations [4].

Targeting various applications, there is a huge demand for GNN inference acceleration with diverse **requirements**. For instance, point cloud segmentation and detection for autonomous driving [5] and high energy particle physics [6] require *real-time processing*; in a particular example, the collision data from particle collider are collected every 25ns and thus must be processed using GNNs within nanoseconds with raw input graphs [7, 8]. For social network applications, the size of the graphs to be processed is usually extremely huge and the computation time and memory cost are significant; therefore, such applications are in demand for GNN accelerators for *large-scale* graphs. Consequently, hardware acceleration is critical to apply GNNs to these applications and address real-time or large-scale computation.

The **challenges** and the **limitations** of existing accelerators, however, are significant. First, GNN computation is both communication-intensive and computation-intensive, as also noted by previous literature [9–11], involving massive irregular memory access for message passing and heavy computation for embedding transformation. Second, novel GNN models are rapidly emerging while the accelerator innovation is lagging behind. For instance, most state-of-the-art GNN accelerators are tailored for graph convolutional networks (GCNs) [9, 12, 13], which can be conveniently expressed as sparse matrix multiplications (SpMM). However, the majority of GNNs are not suitable for SpMM because of complicated operations such as edge embedding, attention, mixed neighborhood aggregation, etc. Therefore, *generic, extensible, and flexible acceleration frameworks* are required to rapidly adapt to evolving GNN models. Third, some accelerators adopt graph preprocessing to employ data locality [9, 14–17], while some apply graph partitioning relying on the property of a fixed input graph (e.g., by analyzing the adjacency matrix sparsity [13]). Such preprocessing or graph-specific techniques *are not feasible* for real-time applications with millions of input graphs with varied structures.

Self Supervised Pre-training Method: Contrastive Learning (CL) and Masked Image Model (MIM)

Contrastive Masked Autoencoders are Stronger Vision Learners

https://www.computer.org/csdl/journal/tp/2024/04/10330745/1SrO8maFmFy

https://arxiv.org/abs/2207.13532v2

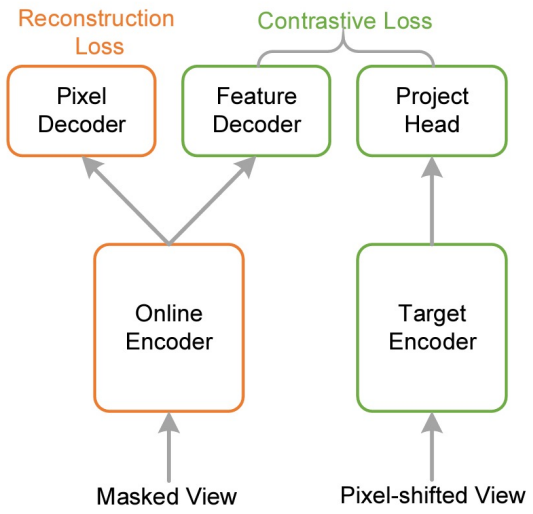


Figure 1: Overview of CMAE. CMAE improves over its MIM counterpart by leveraging contrastive learning through novel designs.

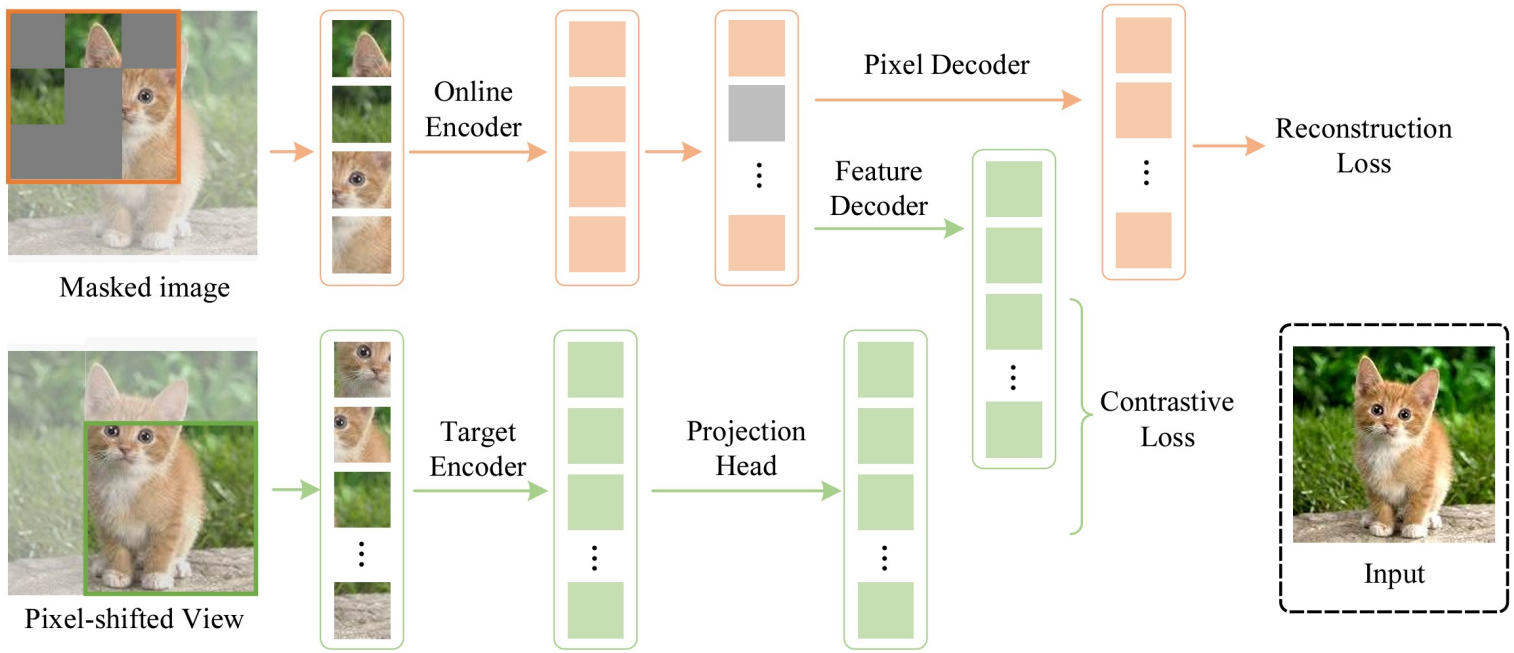
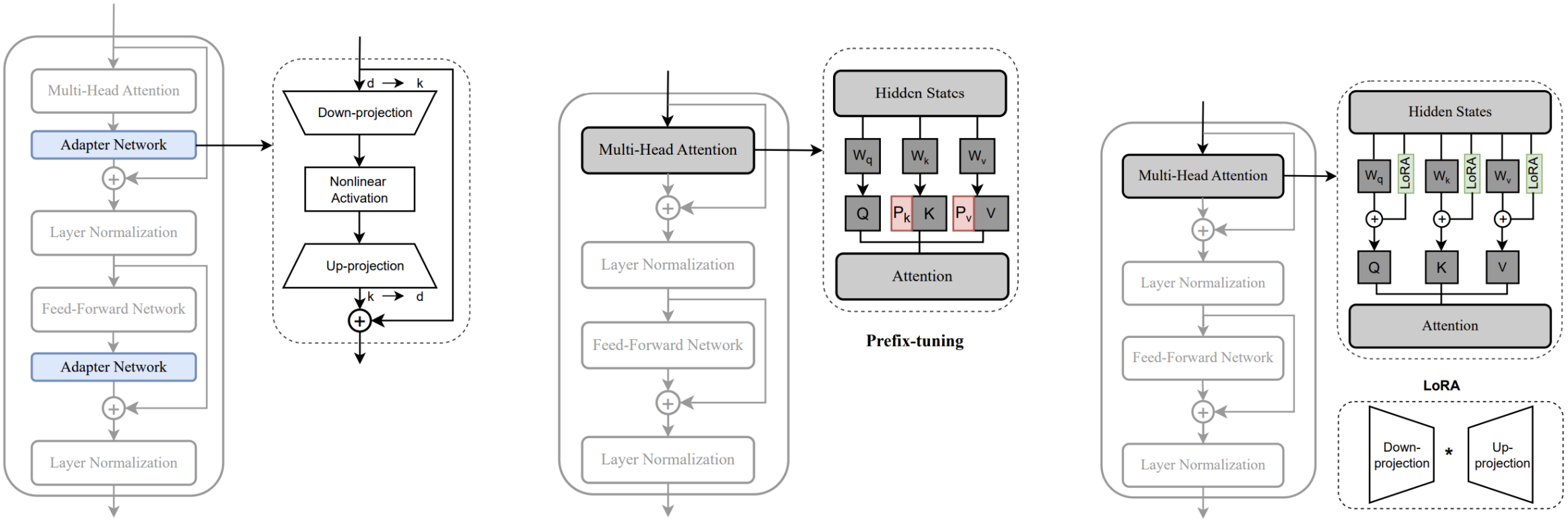


Figure 3: Overall pipeline. Our method contains three components: the online encoder, target encoder and online decoder. Given a training image, it applies *pixel shifting* to generate different views, which are then fed into the online and target encoders respectively. The online encoder randomly masks a fraction of the image patches and operates on the visible ones. The target encoder operates on the whole view after pixel shifting. The pixel decoder learns to reconstruct the input image from the image tokens (along with MASK tokens) provided by the online encoder, while the *feature decoder* learns to predict the features of the input image for contrastive learning with the target encoder output features. After the pre-training, only the online encoder is kept for downstream applications.

Parameter-Efficient Fine-Turning (PEFT):

Parameter-Efficient Fine-Tuning Methods for Pre-trained Language Models: A Critical Review and Assessment

https://arxiv.org/abs/2312.12148



(a) Sequential Adapter (b) Prefix-tuning (c) LoRA

Fig. 3: The detailed architecture of (a) **Sequential Adapter**, (b) **Prefix-tuning**, and (c) **LoRA**.