CaloConsistency

Douglas Williams
University of Chicago
douglasmsw@uchicago.edu

1 Introduction

Calorimeters are an array of sensors immediately surrounding a particle collision chamber which detect deposited energy and limited directional information of secondary particles produced via interactions with the collider material. This early group of sensors is the most data-rich but simulating detected energies requires simulation of primary and secondary particle interactions, making it extremely computationally expensive.

Programs at the LHC are looking to scale up calorimeter detection with more sensors, precision, and of course experiments. But doing so will strain computational and time budgets for the entire project. There is a search for new methods which can speed up the simulation process without a loss in precision.

CaloDiffusion [1], released in October of 2023, trained a diffusion denoising model on shower data released by the Fast Calorimeter Simulation Challenge (CaloChallenge) [2]. While results are strong, sampling is still a very slow and expensive process. That said, CaloDiffusion proves the capability for neural methods to implicitly model particle dynamics as detected in calorimeters, opening up a new work of fast generative modeling techniques for application.

This work seeks to explore whether consistency distillation training, guided by a pre-trained CaloDiffusion model, can achieve similar levels of precision while dramatically speeding up sampling. This project was done over finals week at a quarter-system university so the presented systems are far from optimized. Evaluations of this naive implementation show the system to be approximating target distributions while providing a 10x speedup in sampling time.

2 Related Work

2.1 Diffusion Models

Diffusion models [3] configure image generation as a latent variable modeling problem, in which a model learns to iteratively transform image data from randomly sampled Gaussian noise one from its training data distribution. The intermediate noised image posteriors are described by the forward process, which is a Markov chain which progressively noises the input data according to a variance schedule over a fixed total number of noising steps T: β_i where $i \in [1, T]$.

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1})$$
$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

The reverse process is correspondingly a Markov chain with learned Gaussian transitions starting from the fully noised distribution $p(x_T) = \mathcal{N}(x_T; 0, I)$ and moving to $p(x_0)$. The noise prediction model $s_\theta(x_t, t)$ outputs what it believes to be noise in the image, such that if said noise is removed we are left with an image from the data distribution.

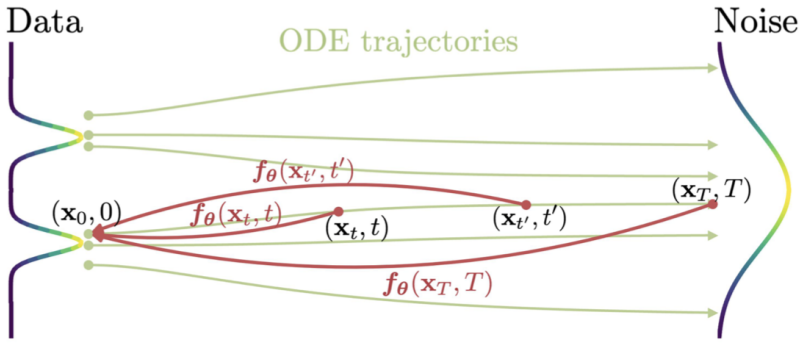
$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1|x_t})$$
$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}}} s_\theta(x_t, t))$$
$$\alpha_t := 1 - \beta_t \text{ and } \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$$

Note that this scheme assumes T discrete time steps and builds out β_t from these integer steps.

2.2 Consistency Models

Consistency models [4] work under the assumption that the sampling path from original data distribution to noise distribution is a consistent and relatively smooth mapping. These trajectories between the two distributions can be learned by a model which, given some initial noise and direction, can "identify" the relevant trajectory and travel along it straight to the data distribution.



This theoretically allows for one-step or small multi-step sampling. Consistency distillation is a training process that assumes there exists an accurate trained diffusion model which can give information about these trajectories. The below training and sampling algorithm statements from the paper outlines the process.

Algorithm 2 Consistency Distillation (CD)

Input: dataset $\mathcal{D}$, initial model parameter $\boldsymbol{\theta}$, learning rate η, ODE solver $\Phi(\cdot, \cdot; \boldsymbol{\phi})$, $d(\cdot, \cdot)$, $\lambda(\cdot)$, and μ
$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
repeat
  Sample $\mathbf{x} \sim \mathcal{D}$ and $n \sim \mathcal{U}[\![1, N-1]\!]$
  Sample $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t_{n+1}^2 \boldsymbol{I})$
  $\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n} \leftarrow \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi})$
  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) \leftarrow$
    $\lambda(t_n) d(\boldsymbol{f_\theta}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f_{\theta^-}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n))$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$
  $\boldsymbol{\theta}^- \leftarrow \text{stopgrad}(\mu \boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta})$
until convergence

Algorithm 1 Multistep Consistency Sampling

Input: Consistency model $\boldsymbol{f_\theta}(\cdot, \cdot)$, sequence of time points $\tau_1 > \tau_2 > \cdots > \tau_{N-1}$, initial noise $\hat{\mathbf{x}}_T$
$\mathbf{x} \leftarrow \boldsymbol{f_\theta}(\hat{\mathbf{x}}_T, T)$
for $n = 1$ **to** $N-1$ **do**
  Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\hat{\mathbf{x}}_{\tau_n} \leftarrow \mathbf{x} + \sqrt{\tau_n^2 - \epsilon^2}\mathbf{z}$
  $\mathbf{x} \leftarrow \boldsymbol{f_\theta}(\hat{\mathbf{x}}_{\tau_n}, \tau_n)$
end for
Output: $\mathbf{x}$

In this setting, we set N to be the number of time steps we want to iterate over during sampling. During consistency distillation training, sample steps $n \in [1, N-1]$ are converted to time steps for the trained diffusion model $t \in [0, T]$ using a conversion function $\Delta(n) = [\epsilon^{1/\rho} + \frac{n-1}{N-1}(T^{1/\rho} - \epsilon^{1/\rho})]^\rho$. For simplicity we will denote time steps using t_n, which refers to the output of $\Delta(n)$.

The training process samples some noised image $x_{t_{n+1}}$, then use an ODE solver (typically Euler or 2nd order Heun) to predict another image nearby along the trajectory x_{t_n}.

$$\hat{x}^\theta_{t_n} := x_{t_{n+1}} + (t_n - t_{n+1}\Phi(x_{t_{n+1}}, t_{n+1}; \theta))$$
When using Euler: $\Phi(x, t; \theta) = -t s_\theta(t, s)$

3 Method

Multiple adaptations needed to be made to adjust consistency distillation to the CaloDiffusion setting. First, the ddpm noise schedule as implemented in CaloDiffusion enforces discrete time steps, meaning all sampled t_n values needed to be rounded to the nearest integer value. While the model still gets samples across time steps, it does reduce the granularity with which it explores trajectories.

Another change is in the calculation of the neighboring point on a trajectory. When calculating $\hat{x}^{\theta}_{t_n}$, we are actually just trying to find a point along the denoising trajectory. One way to accomplish this is to calculate $\nabla_{x_{t_n}} log[p_\theta(x_{t_n})]$ under the assumption that the noise change is normally distributed.

The denoising trajectory during sampling is constructed by iteratively predicting and removing time step noise. This means the trajectory is really a piece-wise function made up of $T-1$ segments representing the trajectories from $s_\theta(x_t, t)$. In an ideal world, we would like to leverage the information available in s_θ to predict the direction of the next step along this piece wise trajectory. Below is the ddpm prediction step without the stochastic scaled noise added back on afterwards (ODE formulation).

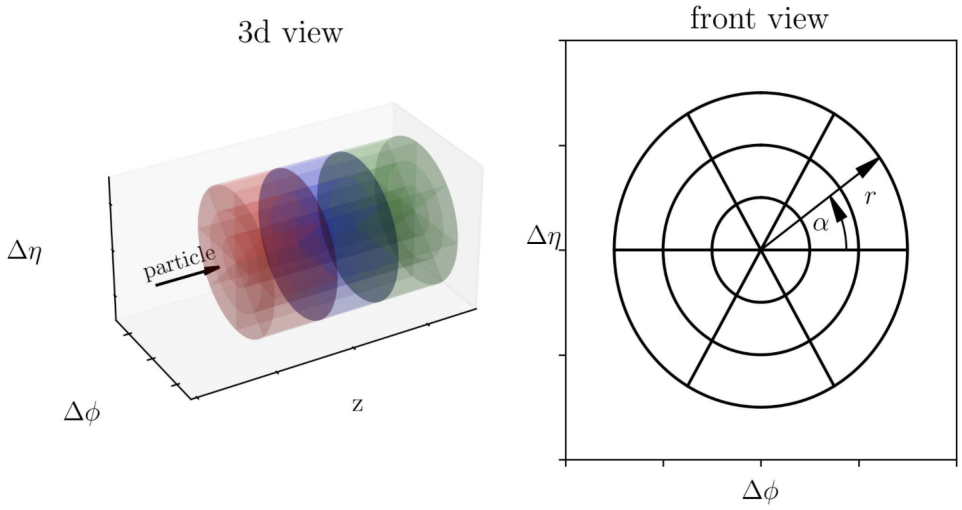
$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}}} s_\theta(x_t, t))$$

This serves as the basis for our ODE step, by sampling a noisy image x_{t_n}, we can apply the noise schedule scaling to move backwards along the trajectory. We treat this as our first order approximation of the trajectory from our point, and then scale it to the size step we would like based on the $\Delta(n)$ function. So our final ODE step comes in the form:

$$\hat{x}^{\theta}_{t_n} = \frac{1}{\sqrt{\alpha_{t_{n+1}}}}(x_{t_{n+1}} - \frac{(t_{n+1}-t_n)\beta_{t_{n+1}}}{\sqrt{1-\bar{\alpha}_{t_{n+1}}}} s_\theta(x_{t_{n+1}}, t_{n+1}))$$

4 Results

Before delving into results, it is important to clarify the data we are working with. The figure below gives side and front views of a calorimeter, which is essentially a 3D cylinder divided into radially distributed bins which we refer to as voxels. Each bin contains a sensor which can detect energy. The left 3D image shows how this cylinder can be sliced into a series of layers, each one voxel thick, for which we have energy readings over the course of an experiment.



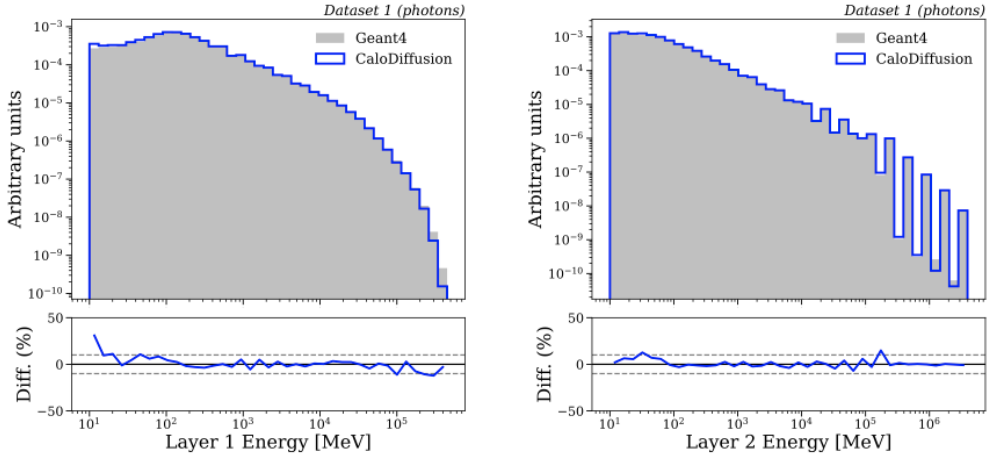
Our goals are a significant speed up in inference sampling without loss of precision. All experiments were done on the photons dataset in dataset one because it is the smallest and quickest to train. We observe a significant speed up in sampling compared to CaloDiffusion. Our CaloConsistency sampled from our evaluation dataset in 412.908 seconds while CaloDiffusion times out after running for 12 hours.

We effectively generate a sort of scaled heat map over these bins, with stronger voxel values referring to higher detected energies. Because of the inherent stochasticity in this experimental system, we cannot just compare one state-of-the-art Geant4 simulation and another CaloConsistency one because they will never be the same. Instead we look for the aggregate statistical properties to be consistent.

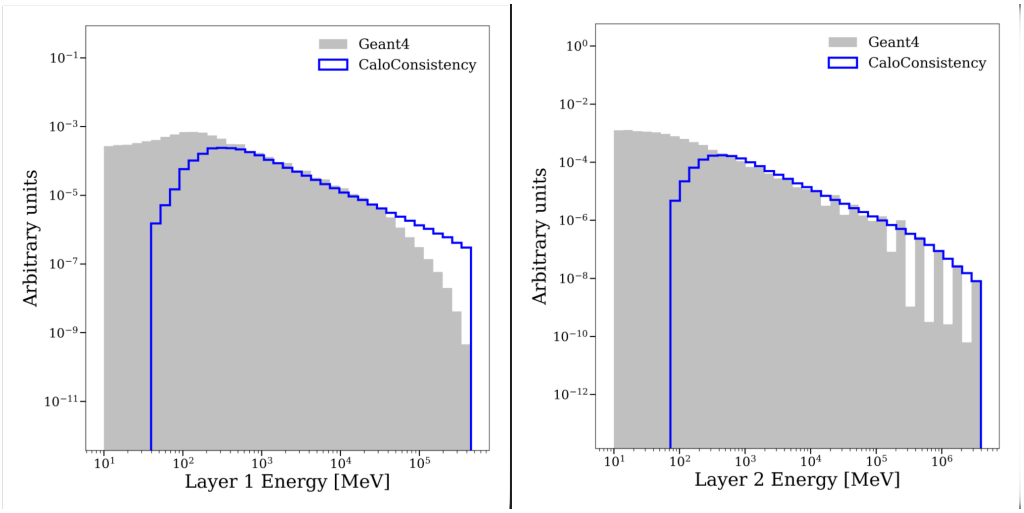
Below are a series of plots from the CaloDiffusion paper demonstrating their model's accuracy. These are normalized histograms of the energies detected in different layers or slices of the calorimeter as

simulated by Geant4 (grey) and CaloDiffusion (dark blue line). The goal is for these two to line up as close to perfectly as possible. Here we present a comparison between our naive CaloConsistency model and the well trained CaloDiffusion model.

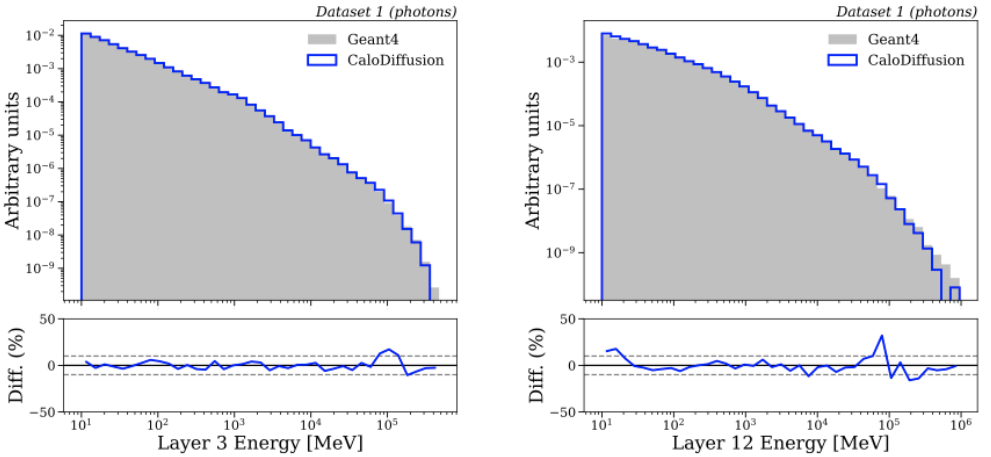
CaloDiffusion



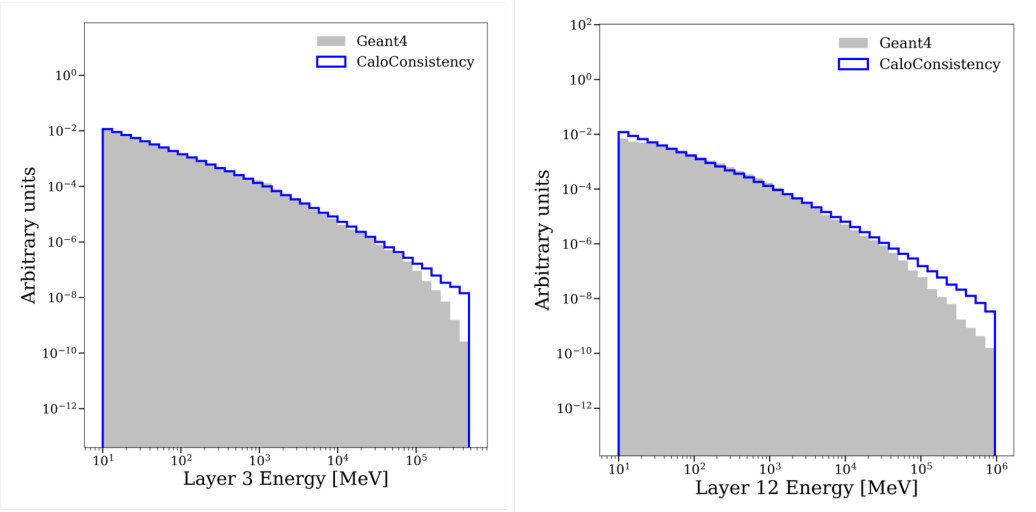
CaloConsistency



CaloDiffusion



CaloConsistency

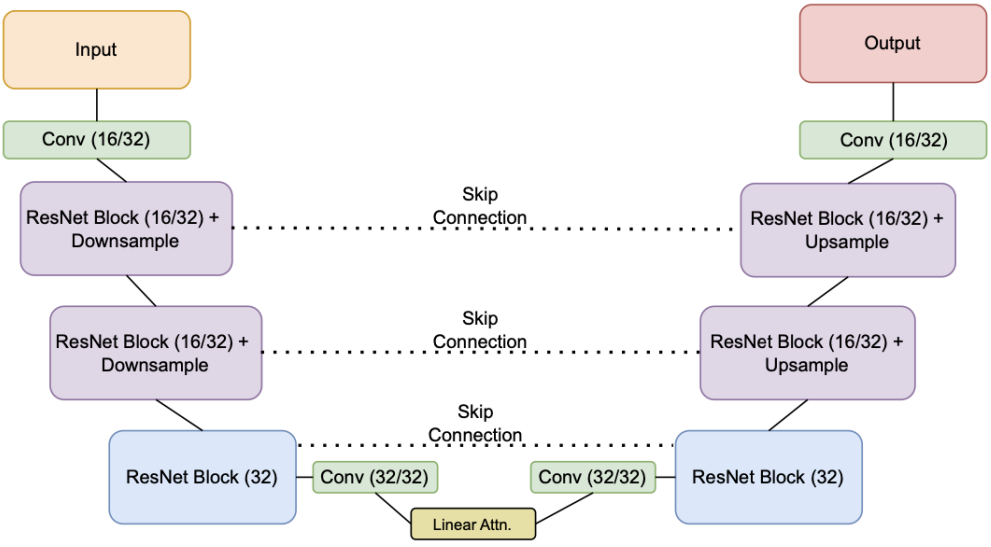


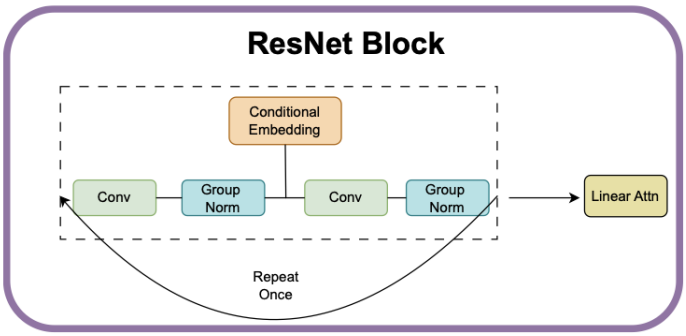
5 Discussion

It is clear from the above plots that my consistency model is nowhere near precise enough for application. That said, there are signs of life. The distribution of energy across layers is approaching the Geant4 target for each of the plots. The model used to generate the plotted samples was trained for 300 epochs. Longer training may yield stronger results.

There are doubts to be had about my sampling and trajectory estimation process, particularly sampling. Working out the best way to scale predicted noise such that it approximates the diffusion model's trajectories is difficult. Time was a major limitation in generating these results so I was not able to implement continuous time sampling which would allow for more accurate and closer paired trajectory points.

There is also room to improve the architecture used. For this project I used the CaloDiffusion UNet, which compresses and reconstructs outputs. There are other potential architectures possible to test if given more time.





References

All code can be found at: https://github.com/Douglasmsw/CaloConsistency

[1] Amram, Oz & Pedro, Kevin (Oct, 2023). Denoising diffusion models with geometry adaptation for high fidelity calorimeter simulation. https://arxiv.org/abs/2308.03876

[2] Fast Calorimeter Simulation Challenge 2022 (2022). https://calochallenge.github.io/homepage/

[3] Ho, Jonathan; Jain, Ajay; Abbeel, Pieter (Dec, 2020). Denoising Diffusion Probabalistic Models. https://arxiv.org/abs/2006.11239

[4] Yang, Song et al. (May, 2023). Consistency Models. https://arxiv.org/abs/2303.01469