Google Summer of Code 2025

Learning the Latent Structure with Diffusion Models

Details:

GSoC Contributor Name: Shouvik Dey
Degree: Integrated B.Tech+M.Tech in IT
College: Indian Institute of Information Technology, Gwalior, India
E-Mail ID: shouvik.dey21@gmail.com
Contact Number: +91-9402375921
Timezone: India/UTC+5:30, Indian Standard Time
Languages: English, Hindi, Bengali, Assamese, German
Social Handles: **LinkedIn profile**: https://www.linkedin.com/in/shouvik-dey-a50b7222b/: ,
Gitter: @shouvik_2112

TABLE OF CONTENTS

Index of Contents

1. Overview

1.1 Project Synopsis

High-fidelity simulation of billions of high-energy collision events is crucial for new physics at the Large Hadron Collider (LHC). Due to the computational cost of traditional simulation methods, effective generative models that can precisely sample from the latent space distribution are required. In order to create realistic multidimensional point cloud distributions of hits resulting from particle interactions with the detectors, this project will create diffusion models for learning the latent structure of these events.
By leveraging diffusion models, the project seeks to:

- **Enhance event generation efficiency:** Simulate high-fidelity collision events with reduced computational overhead.
- **Accurately model latent space distributions:** Learn the underlying structure of **point cloud data** from particle interactions.
- **Benchmark generative performance:** Compare diffusion models with other approaches, such as **VAEs** and **GANs**, using evaluation metrics like **FID, SSIM**, and **Inception Score**.

1.2 Impact on Scientific Community

- Advanced and better classification of jet events, contributing to better event differentiation in High Energy Physics experiments.
- Enables more effective latent structure representation with Diffusion models
- Enhancing pattern recognition in noisy collision data.

1.3 Background Research

Quark-Gluon Classification in High-Energy Physics

The **Large Hadron Collider (LHC)** at CERN is the world's most powerful particle accelerator, designed to study the fundamental forces of nature. By colliding hadrons (protons or heavy ions) at near-light speeds, it recreates the high-energy conditions that existed just after the **Big Bang**.

These collisions produce **particle jets**—sprays of particles formed when quarks and gluons interact. Simulating these events at high fidelity is essential for understanding rare physical processes and searching for new physics beyond the **Standard Model**. However, traditional

Monte Carlo-based simulations are computationally expensive, making generative models a more efficient alternative.

Generative models have gained significant traction in High Energy Physics (HEP) as an efficient alternative for event simulation. Unlike traditional Monte Carlo-based methods, generative models learn the underlying data distribution and can rapidly generate realistic samples.

Techniques such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and Diffusion Models have been applied to model the latent structure of jet events.

Diffusion Models in HEP:

Diffusion models have recently emerged as a powerful class of generative models, offering superior sample quality and diversity compared to VAEs and GANs. Inspired by non-equilibrium thermodynamics, they transform data into a Gaussian noise distribution through a forward process and gradually recover it using a learned denoising process.
The advantage of diffusion models in HEP lies in their ability to accurately model complex, high-dimensional, and continuous distributions, making them well-suited for generating realistic point cloud data representing particle hits in detectors.

Diffusion Model Architecture:

Forward and Reverse Diffusion Processes:

- **Forward Process:** The model iteratively adds Gaussian noise to the input data over several timesteps, eventually transforming it into pure noise.

- **Reverse Process:** During generation, the model gradually removes noise, reconstructing a sample from the latent space.
  Mathematically, the forward process is modeled by a Markov chain with a noise schedule:

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

x_t = Noisy sample at timestep t

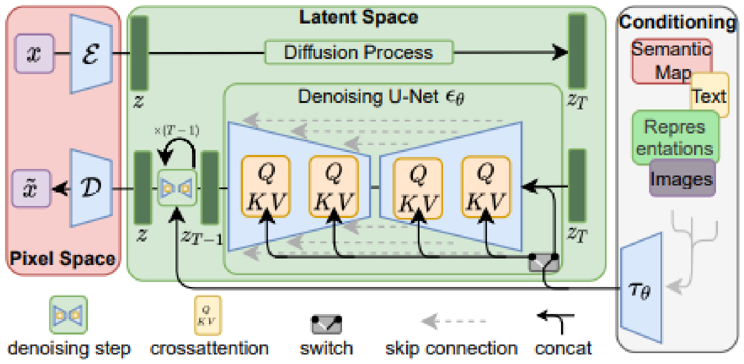
β_t = Noise schedule

The reverse process uses a parameterized neural network to iteratively denoise the sample:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta^2(t)I)$$

μ_θ = Predicted mean

σ_θ = Predicted variance



The **Latent Diffusion Model (LDM)** architecture begins by encoding input images into a **latent space** representation through an encoder. The model applies **iterative noise diffusion** over multiple steps, progressively degrading the latent representation. During denoising, a **U-Net with cross-attention blocks** uses learnable parameters Q,K,VQ, K, VQ,K,V to remove the noise while incorporating external conditioning signals like **text, semantic maps, or images**. Finally, the decoder reconstructs the denoised latent representation back into the pixel space, producing the final **reconstructed image**.

Key Operations in the Denoising U-Net

The U-Net performs the following operations during the reverse diffusion:

Cross-Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V$$

Where:

- Q,K,V = learnable query, key, and value matrices
- d = dimension of the latent space

Skip Connections:

Output=Concat(U-Net layers,Skip Connections)

The skip connections preserve low-level details, improving reconstruction quality.

2. Goals and Deliverables

2.1 Deliverables

- Train an auto-encoder model on the three input image channels(ECAL, HCAL, and Tracks) to learn latent representations, and then provide side-by-side comparisons of the original and reconstructed images.
- Develop a Graph based classification model, based on DGCNN for tagging jet events, after transforming the input images into point clouds, and then converting them into graphs, using kNN algorithm.
- Applied **LDMs in latent space** for jet event reconstruction, using UNet as the core architecture.

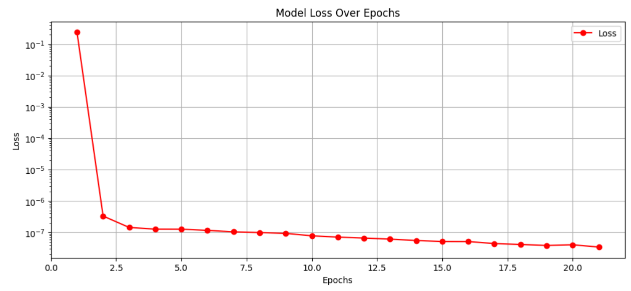
2.2 Prerequisite Tasks

The solutions to common and specific tasks I have completed can be found here:

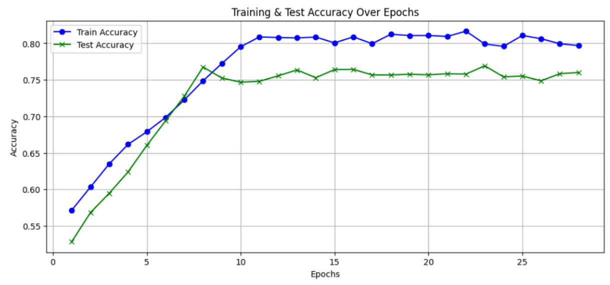
https://github.com/SDeyGitHubber/GENIE_Tasks_Shouvik_Dey

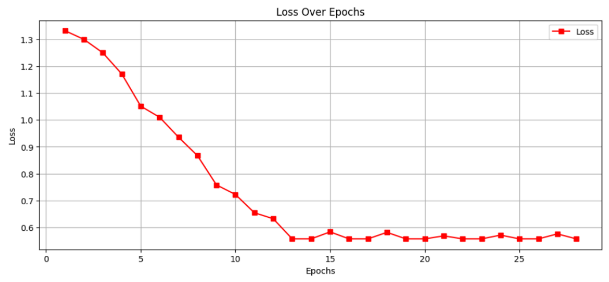
2.3 Implemented Work

As part of the **Common Task 1**, I started by loading the **HDF5 dataset** containing quark/gluon jet events with three channels (ECAL, HCAL, and Tracks). Since the dataset was large, I implemented a **batch-wise loader** with lazy loading to avoid memory issues, storing each batch in **Google Drive** for future use. Next, I built an **auto-encoder model** with convolutional layers to learn compressed latent representations of the jet events. After training the model, I visualized the **original and reconstructed events** side by side to evaluate the model's performance.

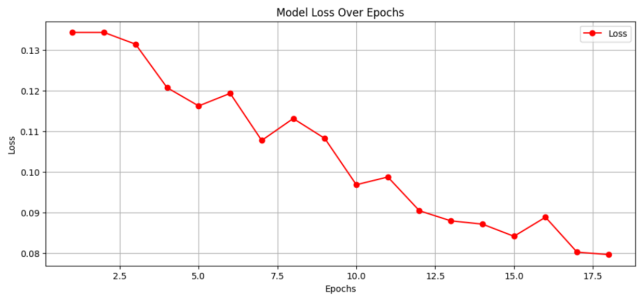


As part of **Common Task 2: Jets as Graphs**, I implemented a **graph-based GNN model** using **DGCNN** for **quark/gluon jet classification**. According to the task requirements, I converted the HDF5-stored jet images into **point cloud datasets** by extracting only the non-zero pixels. I then cast the point cloud data into **kNN graph representations** using **cKDTree**, defining nodes by their **spatial and energy features** and connecting them to their **5 nearest neighbors** as edges. For the model, I used **DGCNN** with **GraphConv layers** and mean pooling for feature aggregation, alongside **gradient accumulation**, **mixed-precision training**, and **early stopping** for efficiency.





As part of **Specific Task 3**, I implemented a **Latent Diffusion Model (LDM)** to reconstruct **quark/gluon jet events**. I applied **gradient checkpointing, AMP, and dynamic batch resizing** for memory efficiency during training. For evaluation, I used **SSIM, LPIPS, and FID** revealing **high visual similarity but low generative diversity**, indicating the need for **further fine-tuning**.



3. Schedule of Deliverables

3.1. Application Review Period:

(8th April - 8th May):

During this assessment period, I will further review my programming skills in Python using PyTorch and Keras, and the necessary libraries for the Latent Diffusion project like Diffusers and PyTorch Geometric (PyG). I shall also enhance and expand my knowledge by strengthening my theoretical foundations required for this project. For ensuring this, I shall be Exploring **recent research papers** on diffusion models for high-energy physics (HEP), and also gain insights into latent space learning techniques and benchmarking techniques.I shall also experiment with small scale diffusion model implementations, and test them on datasets

3.2 Community Bonding Period

(8th May - 1st June):

During this period, I will focus on getting to know the mentors, thoroughly reading the project documentation, and familiarizing myself with the details, especially since this has been introduced for the first time in ML4Sci. Afterward, I will also discuss ideas with the mentors, gather feedback, finalize the datasets,set performance benchmarks, iterate on improvements, and ultimately begin working once all conditions are deemed satisfactory.

3.3. Programming Period:

Week 1–2 (2nd June – 15th June): Initial Setup and Research

- **Objective:** Laying the groundwork by reviewing existing literature and diffusion models.
- **Tasks:**
    - Research and review **existing diffusion models** (e.g., LDM, DDPM) applied to point cloud and HEP data.
    - Identify relevant architectures and **study their latent space learning capabilities**.
    - Set up the **local development environment**, including necessary libraries (PyTorch, PyG, and diffusion model frameworks).
    - **Preparation of the dataset:** Extract point cloud data from jet images and normalize the features for consistency.
- **Testing & Documentation:**
    - Perform initial data visualization and sanity checks.
    - Document the preprocessing steps and observations.

Week 3–4 (16th June – 29th June): Preprocessing and Baseline Model

- **Objective:** Build a robust data processing pipeline and implement a baseline diffusion model.
- **Tasks:**
    - Implement the **data preprocessing pipeline.**
    - Develop the **baseline diffusion model** with a UNet denoising architecture, or s.
    - Set up **initial training and testing frameworks** with basic configurations.
- **Testing & Documentation:**
    - Run initial model tests on small batches to verify the pipeline.
    - Document the architecture, training configuration, and preprocessing steps.

Week 5–6 (30th June – 13th July): Model Refinement and Initial Evaluation

- **Objective:** Train the baseline diffusion model and establish baseline performance.
- **Tasks:**
    - Train the **baseline diffusion model** on the HEP point cloud dataset with moderate hyperparameters.
    - Apply **mixed-precision training**, gradient accumulation, and other methods for efficiency.
    - **Evaluation of baseline performance** using LPIPS and other relevant metrics.
    - Identify early signs of overfitting or underfitting.

- **Testing & Documentation:**
  - Perform continuous testing on validation sets.
  - Document model performance, including loss trends and initial metric results.

Week 7–8 (16th July – 29th July): Model Expansion and Architecture Experiments

- **Objective:** Improve the model architecture and optimize training efficiency.
- **Tasks:**
  - Experiment with **different denoising backbones** (e.g., Transformer-based models) to capture complex latent structures.
  - Optimize the model by applying **gradient checkpointing, dynamic batch sizing, mixed-precision, and other techniques** for memory efficiency.
  - Introduce **regularization techniques** to prevent overfitting.
- **Testing & Documentation:**
  - Run experiments with varied architectures and compare their reconstruction quality.
  - Document architecture variations, results, and observations.

Week 9–10 (30th July – 12th August): Benchmarking and Fine-tuning

- **Objective:** Compare the diffusion model's performance against other generative models.
- **Tasks:**
  - **Benchmark the diffusion model** against VAEs and GANs on the HEP dataset.
  - Compare generative fidelity, diversity, and computational efficiency across models.
  - Perform fine-tuning of the diffusion model to further improve accuracy.

- **Testing & Documentation:**
  - Regularly test the model on unseen data and document comparative performance results.
  - Summarize insights from the benchmarking experiments.

Week 11–12 (13th August – 26th August): Final Refinement and Results Compilation

- **Objective:** Consolidate the final model and compile performance results.
- **Tasks:**
  - Perform final fine-tuning and hyperparameter optimization.

    - **Generate visualizations**: Side-by-side comparison of original and reconstructed events.
    - Summarize final metrics (SSIM, FID, LPIPS) and performance insights.
- **Testing & Documentation:**
    - Validate the final model on unseen test data.
    - Compile results, graphs, and performance metrics for the final report.

Final Week (27th August – 1st September): Wrap-up and Documentation

- **Objective:** Finalize the project and prepare for submission.
- **Tasks:**
    - **Polish the codebase**, ensuring it is clean and well-documented.
    - Finalize and organize **GitHub documentation**, including:
        - Model architecture details.
        - Preprocessing steps.
        - Benchmarking results.
    - Prepare **visualizations and final reports**, clearly presenting key results.
- **Deliverables:**
    - Complete, tested, and documented diffusion model pipeline.
    - Comprehensive project report with visualizations and benchmark results.
    - GitHub repository with organized code, documentation, and instructions.

Lastly, all the work completed during Google Summer of Code (GSoC '25) will be thoroughly documented in GitHub README files, Medium blog posts, and presentations for the end-term evaluations. The documentation will encompass the entire journey of the project, including prior research and initial contributions before the cohort began, discussions and brainstorming sessions with mentors during the bonding period, and the final implementation. Additionally, it will include visualizations, key findings, and major highlights of the project. This comprehensive documentation will ensure that the project's progress, methodologies, and outcomes are well-documented and accessible for future reference and further development.

4. Biographical Information

4.1. Academic Details:

I am Shouvik Dey, a fourth-year undergraduate pursuing an Integrated B.Tech + M.Tech degree in Information Technology at the Atal Bihari Vajpayee Indian Institute of Information Technology and Management, Gwalior, India. Currently, I am in my eighth semester.

I prepared for the Joint Entrance Examination (JEE) Mains and Advanced from 2019 to 2021. Both are highly competitive entrance exams for admission to India's prestigious research and engineering institutes. I secured a rank in the top 1 percentile in JEE Mains and top 2 percentile in JEE Advanced, which are taken by approximately 12–13 lakh students each year, leading to my admission to this institute.

Over the past two years, I have been actively practicing Machine Learning and Artificial Intelligence. I consider myself proficient in Python and C++. Alongside my passion for computer science, I have always been interested in the natural sciences, especially Physics and Mathematics. My fascination with particle physics began in school after reading the globally renowned book A Brief History of Time by Stephen Hawking. More recently, reading about **CERN's Large Hadron Collider (LHC)** and its role in discovering the **Higgs boson** made me excited about the scale and complexity of high-energy physics experiments.

Beyond my academic pursuits, I enjoy participating in hackathons and other ML/AI and development based challenges. I was a runner-up of the Tathya Hackathon '24, a nation-wide hackathon to provide ML-based solutions to real-world challenges. In this event, I developed a Deep learning based time series model to predict AQI, based on diverse inputs. Additionally, I have participated in several other contests and even achieved good ranks like in KaggleX '24, Amazon ML Challenge '24, and more.

4.2 My Motivation for High Energy Physics and Deep Learning 12

I truly value the opportunity provided by **Google Summer of Code (GSoC)** and **ML4SCI** to contribute to **high-energy physics (HEP)** research using **generative models**. My fascination with particle physics stems from its ability to **unveil the universe's fundamental building blocks**—from quarks and gluons to complex jet formations. During my academic journey, I became increasingly drawn to **deep learning's role in HEP**, particularly how generative models like **diffusion networks** can efficiently simulate particle collision events, reducing the reliance on computationally expensive simulations. This project offers the perfect platform to **apply my skills in deep learning** to solve real-world physics problems while collaborating with experts in the field, making meaningful contributions to scientific discovery.

5. Availability Schedule

5.1. Working Hours

I can commit the required time for the project to achieve the timely deliverables as outlined below:

- Working Timings for Weekdays (3-4 hours daily)

Preferred Timings: 8 p.m. to 1 a.m. IST

- Working Timings for Weekends (4-6 hours daily)

Preferred Timings: 10 a.m.to 1 p.m. IST and/or 4 p.m. to 8 p.m.

I have summer holidays from May 6 to July 25, which allows me to work flexibly according to the needs of the project, as directed by the mentors. After the summer break, I have college classes scheduled from 9 a.m. to 5 p.m. (in Indian Standard Time with some free slots), which will allow me to work comfortably in the late evenings or nights, from 8 p.m. to 1 a.m(Indian Standard Time)., depending on the project requirements.

5.2. Mentor-Mentee Meetings and Updates

I can assure you that I will work in the following manner:

- Weekly or biweekly meetings with mentors (preferably on Google Meet/Zoom, but any platform will do) to discuss new progress, including both theoretical ideas and code implementations.
- Pushing the code to GitHub repositories after seeking feedback, and correcting the code if negative feedback is received, based on the suggestions..
- Finalizing the current steps and creating a list of the next achievable tasks.

5.3. Post Google Summer of Code '25

I believe that open-source initiatives are not limited to the timespan of the program; contributions beyond that period also matter. I would love to continue working on these projects after GSoC and certainly apply as a mentee next year, or perhaps even as a mentor!

6. References

[1] Information on LHC and HL-LHC. Available at: https://hilumilhc.web.cern.ch/ and https://home.cern/science/accelerators/ large-hadron-collider.

[2] Diffusers Documentation : https://huggingface.co/docs/diffusers/en/index

[3] ‘Dynamic Graph CNN(DGCNN) for Learning on Point Clouds’ published on 2018: https://arxiv.org/abs/1801.07829

[4] **cKDTree - SciPy Documentation**. Available at: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.cKDTree.html

[5] **Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B.** "High-Resolution Image Synthesis with Latent Diffusion Models." *Computer Vision and Pattern Recognition (CVPR)*, 2022. Available at: https://arxiv.org/abs/2112.10752

[6] **ML4Sci - Diffusion Models for Particle Physics**. Available at: https://ml4science.github.io

[7] **SSIM and FID Evaluation Metrics**. Available at: https://github.com/mseitzer/pytorch-fid

[8] PyTorch Geometric Documentation: https://pytorch-geometric.readthedocs.io/en/latest/