GSoC 2025 Project Proposal

Diffusion Models for Gravitational Lensing Simulation

Organization: ML4SCI

MENTORS

Michael Toomey, Pranath Reddy, Sergei Gleyzer

AUTHOR

Hamees Sayed

Table of Contents

Contents

1 Introduction and Contributor Information

- **Name:** Hamees Ul Hasan Sayed
- **Email:** hameessayed71@gmail.com, 23f1001168@ds.study.iitm.ac.in
- **Contact Number:** +91 83559 14728
- **University:** Indian Institute of Technology, Madras
- **Majors:** BS in Data Science and Application
- **GitHub:** github.com/hamees-sayed
- **Personal Site:** hamees-sayed.github.io
- **LinkedIn:** linkedin.com/in/hamees-sayed
- **CGPA:** 8.9/10

2 Abstract

Gravitational lensing occurs when a massive object bends light from a distant source, creating multiple distorted images. This effect helps scientists study dark matter, which is invisible but affects how light bends. Detecting small dark matter clumps (substructure) can reveal whether dark matter is made of WIMPs, axions, or other candidates. Traditional simulations of lensing events are slow and expensive.

This GSoC project proposes the use of diffusion models, a generative AI technique that iteratively refines images by reversing a noise distribution, to create fast and realistic lensing simulations. Diffusion models power leading text-to-image systems like Stable Diffusion and Flux. I will explore different architectures and conditioning techniques. The expected outcome is a new diffusion model optimized for astrophysical simulations, providing a powerful tool to generate data for simulation and training for further downstream tasks such as domain adaption and self-supervised learning.

3 Technical Details

3.1 Diffusion Models and Gravitational Lensing Simulation

Gravitational lensing occurs when massive celestial objects bend light from distant sources, creating distorted or multiple images. This phenomenon helps map dark matter and study gravitational structures.

Traditional gravitational lensing simulations rely on ray tracing to model how light bends around massive objects. However, these methods require detailed mass distribution modeling and are computationally intensive and time-consuming [6] [5].

To address these limitations, deep generative models can be adapted for gravitational lensing simulations. Techniques like **Flow Matching** [4] and **Latent Diffusion Models (LDMs)** [8] have proven effective in generating realistic, high-fidelity images with lower computational cost.

Flow Matching is a generative modeling technique that creates continuous transformations between noise and target data distributions. It defines a time-dependent vector field $v_t(x)$ that guides data from a source distribution $p_0(x)$ to a target distribution $p_1(x)$. The transformation follows the ordinary differential equation:

$$\frac{dx_t}{dt} = v_t(x_t), \quad x \sim p_0(x)$$

By learning the vector field $v_t(x)$, samples can be efficiently generated from the target distribution. Compared to **Denoising Diffusion Probabilistic Models (DDPM)** and **Denoising Diffusion Implicit Models (DDIM)**, Flow Matching is faster and easier to train and inference while maintaining high sample quality.

Latent Diffusion Models (LDMs) compress high-dimensional data into a lower-dimensional latent space, apply the diffusion process in that space, and then decode the results back to the original data domain. Typically, a Variational Autoencoder (VAE) is used for encoding the input data to a lower dimension latent space, significantly reducing computational overhead while preserving image quality.

To improve adherence to conditioning, I propose integrating **Classifier-Free Guidance (CFG)** [3]. CFG trades off sample diversity and prompt adherence by interpolating between conditional and unconditional predictions. The guided score function is:

$$\epsilon = \epsilon_{\text{uncond}} + w \cdot (\epsilon_{\text{cond}} - \epsilon_{\text{uncond}})$$

where ϵ is the model's output, with ϵ_{uncond} and ϵ_{cond} representing the unconditional and conditional distributions, and w is the guidance weight.

3.2 Data Representation and Backbone Integration

A robust representation of the input data is critical for high-quality generative results. My approach employs multiple strategies for data embedding: In this proposal, I'll focus on the following embedding techniques that will be used to generate robust representations of the input data:

1. **ConvNeXtV2 Embedding [10]:** This state-of-the-art convolutional architecture incorporates a fully convolutional masked autoencoder framework along with a novel Global Response Normalization (GRN) layer. This design enhances inter-channel feature competition and leads to superior feature representations.

   The embeddings generated by ConvNeXtV2 will be primarily used to feed into the Diffusion Transformer (DiT) backbone, which has shown improved performance in generating high-fidelity images and audio.

2. **Latent Encoding via VAE:** For Latent Diffusion Models, a Variational Autoencoder is employed to encode high-dimensional data into a lower-dimensional latent space. The diffusion process is then performed in this latent space, reducing computational costs while maintaining high image quality.

3. **Patch based Embedding:** In models like the **Vision Transformer (ViT)**, images are divided into fixed-size patches, which are then flattened and projected into a vector space using a linear embedding layer. Positional encodings are added to retain spatial relationships before feeding them into the transformer. Unlike CNNs, which process local features hierarchically, ViT treats patches as independent tokens, relying on self-attention mechanisms to model global dependencies.

The robust embeddings generated by ConvNeXtV2, along with the VAE-encoded latent representations, are fed into the chosen backbone architecture. In the given evaluation task, the transformer-based **ViT** backbone has demonstrated superior performance compared to the convolutional **UNet** approach, achieving lower Fréchet Inception Distance (FID) scores and reduced inference latency. As shown in Table 2, the ViT-based model attains an FID of **27.70** with an inference latency of just **109 ms** for 50 steps—nearly as fast as the blink of an eye—whereas the U-Net++ (ResNet-34) model records a higher FID of **34.84** and a latency of **123 ms**.

Further latency optimizations can be applied post-training, such as using **float16/bfloat16** precision, capturing CUDA graph operations via `torch.compile` and implementing faster attention mechanism such as **Flash Attention 2** [1].

3.3 Evaluation Task Results and Approach

The Jupyter Notebooks and results of the evaluation tasks are summarised in the GitHub repository: hamees-sayed/deeplense_assignment.

3.3.1 Common Test I: Multi-Class Classification

For this experiment, I performed multi-class classification on a dataset of gravitational lensing images, categorized into three classes: No Substructure, Subhalo/Sphere Substructure, and Vortex Substructure.

The dataset was organized into two main folders: **train** and **validation**. The training set was further split into **90% for training** and **10% for validation**, while the final evaluation was conducted on the validation folder.

Preprocessing Pipeline After experimenting with various configurations, the best results were achieved using the following preprocessing steps:

- Images were resized to **256×256**, center-cropped to **224×224**, and normalized with a mean and standard deviation of **0.5**. Data augmentation was applied only during training, including **random rotation (10°)** and **horizontal flipping (50% probability)** for generalization.

Model Selection & Training Two deep learning models were trained for comparison: ResNet-34, a widely used residual network known for efficient feature extraction, and EfficientNet-B3, an architecture optimized for parameter efficiency.

While EfficientNet-B3 **converged significantly faster** at **15 epochs**, while ResNet-34 required **45 epochs** to reach its optimal performance. However, despite the difference in convergence speed, ResNet-34 **outperformed EfficientNet-B3 in final classification accuracy by a small margin of 0.51%**. The test accuracy for ResNet-34 was **96.39%**, whereas EfficientNet-B3 achieved **95.88%**.

The models were further evaluated using the Area Under the Curve (AUC) metric for each class, as shown in Table 1.

Table 1: AUC scores for each class

Model	No Substructure	Subhalo/Sphere	Vortex
ResNet-34	0.99	0.99	1.00
EfficientNet-B3	0.99	0.99	1.00

These results highlight that both models effectively distinguish gravitational lensing structures with high confidence, though ResNet-34 demonstrated slightly better overall performance at the cost of longer convergence time.

3.3.2 Specific Test IV: Diffusion Models

In this experiment, I trained a Flow Matching model for gravitational lensing image generation using two different backbone architectures: **U-Net++** and **Vision Transformer (ViT)**.

Backbone Architectures

U-Net++: This model used **ResNet-34** as the encoder with **ImageNet-pretrained weights** to improve feature extraction. Its design helps retain spatial details, which is important for generating high-quality images.

Vision Transformer (ViT): The transformer-based model used **ViT-Base-Patch16-224**, which represents images as **patch embeddings**. Like U-Net++, it was also initialized with **ImageNet-pretrained weights**.

Evaluation Metrics The models were compared based on:

- **Fréchet Inception Distance (FID)** – Measures how close the generated images are to real images.
- **Inference Latency** – Measures the time taken for a single forward pass with **50 timesteps**.

Inference was run in `float16` precision with `torch.compile` to speed up CUDA graph operations. The **Euler** solver was used for ordinary differential equation, and latency was tested on an **NVIDIA L4 GPU** with a batch size of **1**.

Table 2: Performance Comparison of Flow Matching Models

Backbone	**FID ↓**	**Latency (ms, 50 steps) ↓**
U-Net++ (ResNet-34)	34.84	123
ViT-Base-Patch16-224	**27.70**	**109**

Results and Analysis

The results show that both models can generate gravitational lensing images, but the **ViT-based model** produced **better image quality (lower FID)** and was **faster (lower latency)**. This makes it a strong choice for improving generative models for lensing image synthesis. Given its performance, further exploration of transformer-based architectures, particularly **Diffusion Transformers (DiT)**, could lead to even better results.

4 Project Deliverables

The project will focus on applying **Diffusion Models for Gravitational Lensing Simulation**. I will start with a thorough **literature review** to understand current simulation techniques and identify potential improvements. Next, relevant datasets will be curated and preprocessed for training. Different diffusion model architectures will be experimented with, evaluated using metrics like FID, and compared to traditional techniques. Comprehensive documentation will ensure reproducibility and guide future work.

4.1 Objective and Implementation Details

The objective of this project is to develop a diffusion-based generative model for gravitational lensing simulation, offering an efficient and accurate alternative to traditional simulation methods. The steps for achieving this include:

- **Literature Review:**
  - A comprehensive review of existing techniques for gravitational lensing simulations, such as PyAutoLens and GalSim.
  - Analyzing state-of-the-art diffusion models like Flow Matching, and Latent Diffusion Models.
  - Identifying gaps in current methods and understanding how generative models can address these gaps.
- **Dataset Preparation:**
  - Curating and preprocessing relevant gravitational lensing datasets.
  - Applying standard data processing techniques such as resizing, normalization, and augmentation for consistency and quality.
- **Experimenting with Different Diffusion Model Architectures:**
  - Exploring various model architectures including but not limited to:
    - **Diffusion Transformers** [7]
    - **U-Net** [9]
    - **Vision Transformer** [2]
  - Evaluating both unconditional and conditional setups to capture complex lensing phenomena.

- **Evaluation Metrics:**
  - **Fréchet Inception Distance (FID)** for evaluating the quality of generated images.
  - **Perceptual Similarity (LPIPS)** to assess how perceptually similar the generated image is to a real image.
  - **Inference Latency** and **Memory Consumption** for measuring the computational efficiency of the models.
- **Documentation:**
  - Maintaining extensive documentation throughout the project to ensure reproducibility and transparency.
  - Compiling the final results into a comprehensive research report detailing methodologies, evaluation, and conclusions.

4.2 Timeline for Deliverables

- **Community Bonding Period:** During this period, I will engage with my mentors and the ML4SCI community, interact with them, and review past contributors' work, focusing on methods and results. I will also familiarize myself with key physics concepts and related papers on gravitational lensing.
- **Week 1:** Conduct a literature review on diffusion models, focusing on their application to physical simulations. Study existing methods in gravitational lensing simulations and related generative models to identify potential improvements.
- **Week 2:** Curate and preprocess gravitational lensing datasets, ensuring the data is properly cleaned and formatted. Implement standard preprocessing steps like normalization, and data augmentation and set up the data pipeline for model training.
- **Week 3:** Implement the first diffusion model architecture using Flow Matching with a UNet-based architecture as the baseline. Begin implementing Vision Transformers (ViT) for unconditional diffusion tasks. Train and evaluate Flow Matching with UNet and the initial ViT implementation. Compare early results from UNet vs. ViT.
- **Week 4:** Expand experiments with more complex architectures, including Diffusion Transformers and Latent Diffusion Models (LDM) with a VAE-based encoder-decoder. Compare their performance and assess their suitability for gravitational

lensing data generation. Refine model architectures based on initial results from Week 3.

- **Week 5:** Implement conditional diffusion models, incorporating Conditional Flow Matching (CFM) and Classifier-Free Guidance (CFG). Train and evaluate these models, comparing them against the best unconditional architectures from Week 3 & 4 using both qualitative analysis and quantitative metrics like Fréchet Inception Distance (FID).

- **Week 6:** Conduct a comprehensive evaluation of all models developed until Phase 1. Prepare a report summarizing progress, results, challenges, and insights. Receive feedback from mentors. Publish the first blog post detailing the methodology and key takeaways from Phase 1.

- **Week 7:** Build a classification model to evaluate the quality of conditional models from Week 5. This model will help assess how well the generated data aligns with expected gravitational lensing properties.

- **Week 8:** Based on the classification model results, refine the conditional diffusion model further. Address any performance gaps and incorporate necessary adjustments to improve the quality of generated samples. Catch up on any pending tasks from previous weeks.

- **Week 9:** Conduct a comprehensive training session on the best-performing model architecture, incorporating both unconditional and conditional setups. Optimize training techniques and ensure stability.

- **Week 10:** Focus on optimizing the model for inference speed and memory consumption. Implement techniques such as:

  - Half precision (FP16/BF16) training and inference.
  - Capturing CUDA operation graph using`torch.compile`
  - Batch processing for efficient inference
  - Faster attention mechanisms such as Flash Attention.

  Finalize the second blog post, detailing model optimization strategies and their impact.

- **Week 11:** Complete documentation, ensuring reproducibility of results and models. Finalize the research report, summarizing key findings, optimizations, and learnings from the project.

- **Week 12:** Finalize all deliverables, including cleaning up code, writing the final blog post, and preparing for the final evaluation. Ensure all results, models, and documentation are in place for submission. Discuss potential post GSoC collaborations with mentors.

5 Other Information

5.1 Why ML4SCI?

I like pure science, deep learning, research, and open source. ML4SCI combines these, making it a great fit for me. I want to work on problems where machine learning helps scientific discovery, and ML4SCI focuses on that. Since it is part of CERN, it offers access to real-world scientific challenges and datasets. I have worked on deep learning models, including diffusion models and transformers, and explored ways to optimize inference using tools like TensorRT and CUDA. I want to apply these skills to scientific problems. My goal is to do a PhD, and working in ML4SCI would help me build the right experience. By contributing to ML4SCI, I can work on open research and collaborate with others who share my interest in science and deep learning.

5.2 Academic Background

I am pursuing a Bachelor of Science in Data Science from IIT Madras with a CGPA of 8.9. As part of my coursework, I studied Machine Learning Foundation and Machine Learning Techniques under Prof. Harish Guruprasad and Prof. Arun Rajkumar, scoring 9 SGPA in both. I also completed a Deep Learning course taught by Prof. Mitesh Khapra.

Beyond my university courses, I completed Stanford's CS229 Machine Learning MOOC, which introduced me to Deep Learning and Reinforcement Learning. I also took hands-on courses from Hugging Face, further strengthening my practical understanding of deep learning.

5.3 Past Experience

I worked at SPRING Lab, IIT Madras under Prof. Srinivasan Umesh, focusing on machine translation for low-resource Indic languages. I developed translation models using BERT, IndicTrans, NLLB, and SeamlessM4T, optimizing them for resource-scarce settings. My work was recognized as the best overall submission in the WMT shared task, hosted alongside EMNLP 2024, and led to a publication.

Following this, I joined smallest.ai as a Data Science intern, where I worked on Diffusion Transformers for real-time, zero-shot voice cloning and TTS. I also built the TTS SDK from scratch, developed custom plugins, and designed data pipelines for preprocessing and streaming large scale speech data. Additionally, I optimized inference latency from 1.5 seconds to 300 milliseconds using CUDA Graphs, fused operations, and TensorRT. My internship eventually transitioned into a full-time role.

Additionally, I was a Section Leader for Stanford's Code in Place program, where I taught Python to a group of 15 students worldwide.

5.4 Past Interaction with Mentors

I reached out to Pranath Reddy on LinkedIn soon after ML4SCI was announced as a participating organization. Being particularly interested in the "Diffusion Models for Gravitational Lensing Simulation" project under Deeplense, I wanted to engage early and understand the expectations. Given my prior experience with Diffusion Models, I was eager to contribute and sought clarity on the application process. Pranath has been incredibly helpful in guiding me through the evaluation tasks, answering my questions, and providing insightful feedback on my draft proposal. His responses have made the process much smoother, and I appreciate the support from the ML4SCI community.

5.5 Commitments

My degree program is flexible, allowing me to adjust my schedule as needed. If selected for GSoC, I will be taking a term break during the summer (three months) to fully dedicate my time to the project. Previously, I balanced my Data Scientist role with coursework later in the day, and during GSoC, my focus will shift from studies to the project while maintaining my professional responsibilities.

In case of unavoidable circumstances leading to a temporary reduction in working hours, I will ensure to compensate for the time in the following weeks. I will communicate any such situations with my mentors well in advance to ensure smooth project progress. For regular updates and discussions, I will be available via Teams, Zoom, and Google Meet in the Indian Standard Time Zone (GMT +5:30).

6 References

References

[1] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[3] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[4] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023.

[5] Nicole J Moore, Yang Zhao, Greg Schmidt, and Duncan T Moore. Ray-tracing simulation of gravitational lensing using a gradient-index model. *Optical Engineering*, 60(1):015104–015104, 2021.

[6] James. W. Nightingale, Richard G. Hayes, Ashley J. Kelly, Aristeidis Amvrosiadis, Amy Etherington, Qiuhan He, Nan Li, Xiaoyue Cao, Jonathan Frawley, Shaun Cole, Andrea Enia, Carlos S. Frenk, David R. Harvey, Ran Li, Richard J. Massey, Mattia Negrello, and Andrew Robertson. Pyautolens: Open-source strong gravitational lensing. *J. Open Source Softw.*, 6:2825, 2021.

[7] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023.

[8] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[10] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders, 2023.