Contributions to the DeepTracer 3.0 Research Pipeline

Improving Documentation, Model Evaluation, and Dataset Compression for Protein Structure Prediction

Seyeon Park
Computer Science & Software Engineering, School of STEM
University of Washington Bothell
Bothell, Washington, USA
seyeon@uw.edu

ABSTRACT

This summer, I contributed to the DeepTracer 3.0 project, which applies generative AI models to predict protein structures from cryo-electron microscopy (Cryo-EM) density maps. My work focused on three areas: improving project documentation for new researchers, executing and evaluating the diffusion-based prediction pipeline, and investigating dataset compression to improve efficiency. I successfully ran the TRACE_DIFFUSION and EVALUATE steps, achieving an RMSD of 1.74 Å and ~75% coverage on the emd_0009 benchmark dataset. In addition, I tested compression methods that reduced file size by more than 20× while lowering training time. These contributions improved the pipeline's reproducibility, usability, and performance for future researchers.

KEYWORDS

Protein structure prediction, Cryo-EM, Diffusion models (DDIM, DiT), Generative AI, Atomic tracing, HDF5 dataset, Deep learning, Structural biology

1 INTRODUCTION

DeepTracer is a computational tool that applies deep learning methods to predict protein structures directly from cryo-electron microscopy (Cryo-EM) maps. Determining accurate protein structures is a central challenge in structural biology, as it enables deeper understanding of biological mechanisms and supports applications such as drug and vaccine development. Traditional methods for protein modeling can be slow and require expert intervention, but recent advances in diffusion models and transformer-based architectures offer opportunities to make the process faster and more accurate. Diffusion models operate by gradually adding noise to data in a forward process and then training a model to reverse this process to recover the original structure. In this project, we used DDIM, a more efficient variant that reconstructs structures with fewer reverse steps. Figure 1 shows this process, where data is gradually noised in the forward direction and then reconstructed in the reverse process.

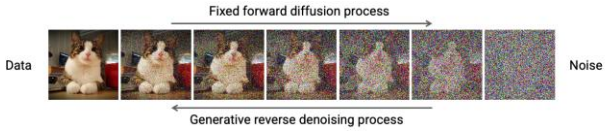

Figure 1: Forward and reverse process in a diffusion model. Data is gradually noised in the forward process and reconstructed in the reverse process. ***Image from the CVPR 2022 Tutorial on Diffusion Models (Song et al., 2022).***

During the Summer 2025 quarter, I contributed to the DeepTracer 3.0 project as a member of the DeepTracer Generative Model Team in the DAIS Lab at the University of Washington Bothell. My role focused on improving the pipeline's usability and reproducibility through documentation, running and evaluating model predictions, and testing strategies to handle large datasets more efficiently. These efforts supported both new researchers joining the project and the broader research goal of developing AI-driven approaches for structural biology.

2 WEEKLY CONTRIBUTIONS

Week 1-2: Environment Setup and Documentation

In the first week I officially started the DeepTracer research project. My focus was on setting up the environment correctly and catching up with the generative model team to understand their progress and expectations. I began working on Task 1, which involved modifying the README, though I initially made the mistake of working in the wrong directory; instead of working on my data/uw-netid directory I was working on home directory. I made progress in understanding how the DeepTracer 3.0 code functions, talked with both new and current members, and created a Google Doc to organize my process for Task 1, particularly around environment setup.

The following week I created a DeepTracer 3.0 workspace file summary to gain a clearer understanding of the code. I set up four branches in my data/uw-netid directory, including dt-integration, backbone, and personal branches for both; seyeon-readme-updates branch was a copy of backbone branch and seyeon-dt-integration

was a copy of dt-integration branch. I realized that the choice between dt-integration and backbone did not significantly affect Task 1, so I continued mainly in dt-integration with a personal branch(seyeon-dt-integration). I completed my working environment setup, documented challenges, and finalized steps such as virtual environment creation, dependency installation, WandB authentication, dataset download, GPU verification, and simulated dataset creation. I also modified the local README and attended the weekly generative model meeting.

Week 3-4: Training and Pipeline Execution

In week three I created my own dataset directory and successfully ran the pipeline to create a simulated dataset, train the diffusion models, and evaluate them. Before running, I used nvidia-smi to confirm GPU availability and trained a model across 100 epochs, reducing the loss from 0.52 to 0.214 (a 59% improvement). Training utilized six NVIDIA RTX A6000 GPUs and completed in about 40 minutes. I explored diffusion types in config.py, reviewed WandB logs, and updated the README to reflect my process. I also attended the weekly team meeting.

In the next week I addressed storage issues by cleaning up datasets and removing unnecessary files, bringing system usage down from 100% to 97%. I successfully ran the pipeline to generate four protein structure files (CA, C, N, and NaN atoms) and updated the README to explain the OUTPUT step using trained diffusion model checkpoints. I also worked to understand files such as cls_weights.py and config.py to clarify their purpose and use in the pipeline.

Week 5-6: Evaluation and Debugging

During the fifth week I resolved a cuDNN compatibility error caused by conflicting CUDA versions and outdated dependencies. After reviewing my setup, I realized that I had originally installed the requirements.txt outside of the virtual environment. Many of these packages, such as TensorFlow, were related to the original DeepTracer code, so the mistake did not cause issues until I reached the TRACE_DIFFUSION step. At that point, mismatched dependencies triggered errors, including the well-known protobuf version conflict. To fix this, I set up the virtual environment properly, reinstalled the dependencies inside it, and upgraded TensorFlow and cuDNN. Since this protobuf issue has already been resolved in the backbone branch (which includes an updated, conflict-free requirements.txt), I plan to use that version going forward.

With the environment corrected, I was able to run the TRACE_DIFFUSION pipeline and predict 1,234 protein residues for the EMD-0009 dataset. I updated the README with TRACE_DIFFUSION commands and completed the EVALUATE pipeline, achieving an RMSD of 1.74Å, 51.8% residue matching, 67% secondary structure matching, 75% coverage, and 9.2% sequence matching (see Figure 2). I then pushed my updated README to my GitHub branch. I also used UCSF Chimera to visualize the AI-predicted structures against the reference PDB, which helped me assess structural alignment beyond numerical metrics (see Figure 3).

	A	B	C	D	E	F	G	H	I	J	K	L
1	EMDB	Predicted Residues	Date	PDB	Deposited	RMSD	Backbone	% Matching	% SSE Matching	% Sequence Matching	% False Positives	% False Connections
2	emd_0009	1234	2025-07-29 04:42:16	6gj1	1638	1.74	#NUM!	51.83	67.02	9.19	31.20	42.95

Figure 2: Evaluation metrics for the EMD-0009 dataset. The AI-predicted structure contained 1,234 residues compared to 1,638 in the deposited reference (PDB: 6gj1). Coverage (~75%) was calculated as predicted ÷ deposited residues. Additional metrics include RMSD, residue matching, secondary structure matching, sequence matching, false positives, and false connections.

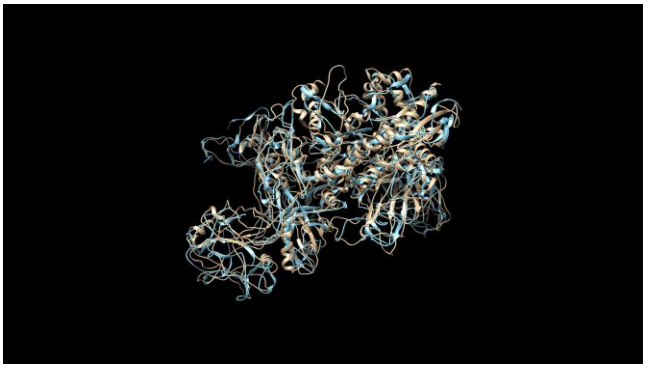

Figure 3: Structural alignment of AI-predicted model (emd_0009.pdb, blue) and deposited reference structure (6gj1.pdb, gold) visualized in UCSF Chimera. Overlaying both models provides a visual assessment of structural similarity beyond numerical metrics such as RMSD.

In week six I received feedback on my first README draft on the seyeon-dt-integration branch and revised it accordingly before updating my GitHub branch. I also spent time reading related literature to better understand DeepTracer and diffusion models, including Pfab et al. [2021] for the original DeepTracer approach, Ho et al. [2020] for DDPM, and Peebles & Xie [2023] for DiT. In addition, I began preparing for future tasks by comparing the dt-integration and backbone branches on GitHub in terms of code execution. This revealed that the backbone branch shifted prediction targets from atom-level to backbone-level classes, but the general workflow of the pipeline remained consistent across both.

Week 7-8: Dataset Compression & Report Writing

Starting in week seven, I switched from the dt-integration branch to backbone to work with the most recent code and documented the differences in how the pipeline runs. I then updated the README (building on Manish's edits) and submitted a pull request to Nathaniel to merge my 'seyeon-readme-updates' branch into 'backbone'. I also began my next task, which was exploring lossless compression methods for HDF5 datasets, and started drafting my 3-minute flash talk and reflection letter.

During week eight I focused on two major areas: practicing and successfully delivering a three-minute flash talk to my research group, and experimenting with HDF5 dataset compression. For the compression study, I compared gzip and lzf methods using my dataset directory (EMD-0009 and EMD-9380) with a maximum of 100 training epochs. The results showed that gzip reduced the **file size from 1.7 GB to just 84 MB (~20× smaller) and**

shortened training time by ~14%. The lzf method reduced the file size to 112 MB and achieved nearly identical training performance to gzip (Table 1). Overall, gzip provided the best balance between disk savings and runtime efficiency, as compression not only minimized storage requirements but also made training faster by reducing I/O overhead.

Table 1: Effect of HDF5 compression methods on file size and training time (100 epochs).

Setting	File size	Total Training Time (100 epochs)
Uncompressed	1.7 GB	3 hr 05 min 01 sec
gzip	84 MB	2 hr 39 min 11 sec
lzf	112 MB	2 hr 40 min 13 sec

3 NEXT STEPS

Building on these lessons and the feedback I received, I will expand my HDF5 experiments beyond GB-scale tests. In particular, I will parameterize the HDF5 writer to expose chunk size, compression type (gzip vs. lzf), compression_opts (levels 1, 4, 6, 9), shuffle, and fletcher32 so they can be systematically compared. Since performance characteristics may differ at different dataset scales, I will benchmark small, medium, and simulated TB-scale datasets rather than relying only on GB-scale runs. For read-heavy and mixed workloads, I will test whether lzf outperforms gzip at larger scales, while also measuring the impact of chunk shapes and filter combinations on efficiency. Metrics will include wall-clock epoch time, I/O throughput, CPU/GPU utilization, and on-disk size under cold and warm caches. The goal is to produce both a recommended default configuration (per workload and data size) and a reproducible benchmark suite integrated into the repository.

4 DISCUSSION

Over the course of the quarter, I learned how to work at the intersection of AI and structural biology by running end-to-end protein prediction pipelines. I gained hands-on experience with diffusion-based models, model evaluation metrics such as RMSD and secondary structure alignment, and tools such as WandB for experiment tracking and UCSF Chimera for structural visualization. I also developed practical skills in debugging environment issues, documenting workflows for new researchers, and optimizing dataset storage using HDF5 compression. Importantly, I learned that clear documentation and reproducibility are as critical as model accuracy for long-term research progress.

One major challenge was the steep learning curve in understanding both new biological terms (Cryo-EM, residues, backbones) and new technical tools (HDF5, diffusion models). Another challenge was keeping track of evolving methods, as at first, I expected we would use U-Nets, but in practice the team moved to transformer-based architectures (DiT); similarly, I had to learn the distinction between DDPM and DDIM. These shifts were at times confusing, but ultimately helped me appreciate that research methods are not fixed and often improve as projects evolve. I also faced challenges with the scale of the data, as Cryo-EM maps and generated datasets quickly filled available storage, requiring me to investigate compression and cleanup strategies. Finally, balancing documentation updates with ongoing pipeline changes was difficult, since the backbone and dt-integration branches occasionally diverged in workflows. These experiences strengthened my problem-solving skills and taught me to adapt to changing methods, approach problems incrementally, and document every step for reproducibility.

5 CONCLUSION

This project gave me the opportunity to contribute to the DeepTracer 3.0 pipeline by focusing on documentation, evaluation, and dataset optimization. My contributions improved the usability of the workflow for new researchers and introduced efficiency gains for handling large datasets. Through this process, I strengthened my technical skills in diffusion models, evaluation metrics, and data management, while also learning how to navigate challenges in interdisciplinary research. Overall, this experience has prepared me to take on future work at the intersection of computer science and biology, where clear documentation, reproducibility, and adaptability are just as important as technical accuracy.

ACKNOWLEDGMENTS

I would like to thank the senior members of the DeepTracer Generative Model Team, including Nathaniel Jewel, Haowen Guan, and Haydn Tamura, for their guidance. I especially want to thank Dr. Dong Si for welcoming me into the DAIS research group and giving me the opportunity to participate in the Summer Research Fellowship. I also appreciate my fellow team members who joined the project around the same time as I did, as their collaboration and enthusiasm made the research experience more engaging and enjoyable.

REFERENCES

[1] Yang Song, Stefano Ermon, Jonathan Ho, and Prafulla Dhariwal. 2022. Tutorial on Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2022, Tutorial). CVF/IEEE, New Orleans, LA.* https://cvpr2022-tutorial-diffusion-models.github.io

[2] Jonas Pfab, Nhut Minh Phan, and Dong Si. 2020. DeepTracer for fast de novo cryo-EM protein structure modeling and special studies on CoV-related complexes. In *Proceedings of the National Academy of Sciences (PNAS),* 118, 2 Dec. 2020, e2017525118. https://doi.org/10.1073/pnas.2017525118

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems (NeurIPS 2020).* Curran Associates, Inc., 6840–6851. https://arxiv.org/abs/2006.11239

[4] William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on*

Computer Vision (ICCV '23). IEEE, 4175–4185. DOI:https://doi.org/10.1109/ICCV51070.2023.00389

[5] Weilun Feng, Chuanguang Yang, Zhulin An, Libo Huang, Boyu Diao, Fei Wang, and Yongjun Xu. 2024. Relational Diffusion Distillation for Efficient Image Generation. In *Proceedings of the 32nd ACM International Conference on Multimedia (MM '24)*, October 28–November 1, 2024, Melbourne, VIC, Australia. *ACM, New York, NY, USA, 90 pages.* https://doi.org/10.1145/3664647.3680768