Generative GNN's for Design of Ligand Molecules

Tristan Brigham*
Yale University: CPSC 583
New Haven, CT 03301
tristan.brigham@yale.edu

1 Introduction

Deep learning (DL) has rapidly advanced the frontier of automated research, enabling significant strides in molecular design. One challenge with high potential impact is generating ligands that can induce desired effects on specific proteins. Success in this area would accelerate drug discovery and address urgent biological problems. While traditional methods for screening candidate molecules consume months per molecule, approaches leveraging DL can propose and evaluate structures in microseconds — a crucial advantage given the chemical space exceeds 10^{60} possible molecules [31].

Graph neural networks (GNNs) excel at handling the variable-size inputs of molecular systems. We propose a lightweight, end-to-end pipeline for generating novel molecules conditioned on protein pocket configurations. This approach consists of three core components, two of which are detailed in this paper, while the third remains an ongoing, novel development.

1. **Pocket-Finding Model (YPFM):** Identifying potential binding sites in folded proteins is a prerequisite for targeted ligand design. We employ a U-Net-based model [29] with attention layers to pinpoint these binding pockets at high resolution, represented as voxelized data.
2. **Molecule Prediction Model (YMPM):** Once pockets are identified, we generate candidate molecules that fit these sites. We use a GAN framework with a GATConv-based generator [34, 39] and a Graph Isomorphism Network discriminator [27]. Reinforcement learning, specifically reward-weighted policy optimization, refines the generation process toward molecular candidates that meet target criteria [30].
3. **Conditioned Molecular Generation Model (YCMGM):** The final component will condition the generation process not only on pocket structure but also on desired molecular attributes—such as binding affinity, toxicity, and specific functional groups. While this facet of the system remains under active exploration, our preliminary results are promising, indicating that incorporating conditioning vectors and learned message-passing strategies may allow us to fully specify the molecular outcome. Achieving this would be transformative, enabling the direct design of tailored ligands in a manner previously unattained.

This study demonstrates the practicality of GNN-based architectures in ligand design and underscores the need for refined message-passing techniques. Combining reinforcement learning with GAN pre-training outperforms current benchmarks on molecular diversity and expressiveness with large molecular generation datasets such as MOSES. The project's development is ongoing in the Gerstein Lab [11], where the code and presented materials originate. Preliminary results indicate that our approach achieves higher accuracy in pocket prediction and generates more chemically valid molecules compared to existing methods with expanded coverage across more chemically diverse domains. These findings suggest that our pipeline can significantly enhance the efficiency and effectiveness of drug discovery processes.

2 Related Work

This project aims to create an end-to-end pipeline for generating target-specific molecular candidates conditioned on protein structures and desired attributes. Existing methods in molecular generation, protein-ligand interaction prediction, and GNN applications in chemistry inform this approach.

*Submitted to Yale University's CPSC583 as the final project. Do not distribute.

2.1 Molecular Generation and Representation

Recent advances in generative modeling have focused on translating protein structural information into candidate ligands. Lingo3DMol [8] uses a language model to produce SMILES strings conditioned on protein pockets and employs FSMILES for enhanced 2D/3D ligand representation. Similarly, transformer-based approaches [40, 41], though not fully validated [2], align with the proposed GNN-based generation by incrementally constructing molecules atom-by-atom. Although AlphaFold-based methods [17, 1] exist, they are closed-source, computationally intensive, and not tailored for target-specific molecular generation. Further, while these generative models enhance structural representation, they struggle to capture complex spatial dependencies efficiently.

2.2 Protein-Ligand Docking, Pocket Prediction, and Interaction Modeling

Efficient identification of ligand binding sites and docking configurations is crucial. RoseTTAFold [3] integrates amino acid sequences, residue-residue distance maps, and atomic coordinates to capture protein-ligand interactions. Diffusion-based docking [6] accelerates orientation predictions, though such methods may be considered later-stage enhancements. Pocket prediction models, such as SiteFerret [10] and point-cloud-based GNNs [44], leverage hierarchical clustering [5], isolation forests [38], and message-passing architectures [12, 18, 43, 32, 15] to identify promising binding sites. Models in this category are computationally expensive, but show strong accuracy and speed over conventional docking approaches.

2.3 Graph Neural Networks for Molecular Design

GNNs have shown promise in materials science and chemistry, guiding molecular search and optimization. Early work [28] laid the foundation, and subsequent scaling of datasets [22] broadened exploration spaces. GNN-based systems have predicted antibiotic efficacy [37], while Mol-CycleGAN [21], inspired by CycleGAN [45] and JT-VAE [16], optimized molecular properties and structures. While these advancements underscore GNNs' versatility in generating, refining, and evaluating candidate molecules aligned with desired target attributes, GNN's struggle with distinguishing structurally similar molecules due to limitations in expressiveness.

3 Methods

All experiments employ RDKit for molecular preprocessing and PyTorch (with PyTorch Geometric) [9, 25] for model implementation and training.

3.1 Pocket-Finding Model (YPFM)

Accurately identifying ligand-binding pockets in a protein's three-dimensional structure is a critical first step in targeted molecular design. While Graph Neural Networks (GNNs) have shown notable success due to their inherent translation and rotation invariance, we observed that a 3D U-Net architecture with integrated attention mechanisms provides higher accuracy and richer representations for this task. In particular, after initial attempts with GNN-based methods, we found that our U-Net approach yields more expressive embeddings and more precise pocket predictions. A simplified mathematical representation of our UNet architecture is in equation 9.

Our model, YPFM, leverages a 3D-UNet framework augmented with both position-wise and channel-wise attention. The model applies a series of downsampling operations to capture global context, followed by attention-based refinements at intermediate resolutions, and finally upsampling layers to return predictions to the original voxel space. The attention layers encourage the network to focus on the most salient regions and channels, ultimately highlighting potential binding pockets with greater fidelity. The output layer was a final convolutional layer with a sigmoid activation function to produce binary predictions indicating in-pocket and out-of-pocket voxels.

By training the U-Net model on permuted examples, we preserve essential invariances and ensure that the network's expressiveness remains comparable to that of GNNs. Unlike GNNs, however, the U-Net architecture naturally integrates multi-scale context and global attention. As a result, YPFM surpasses traditional GNN-based approaches not only in raw accuracy but also in providing richer semantic maps of potential binding sites. The global-focus and multi-resolution feature aggregation is encoded by the U-Net's skip connections and attention layers, which collectively enhance performance.

Our methods align with the principles used in advanced protein structure prediction tools like RoseTTAFold [3], but we focus specifically on pocket prediction. By incorporating attention into

the 3D-UNet, we emphasize relevant spatial and channel-wise features that lead to more stable and interpretable representations. The integration of voxel-based 3D-UNet + attention leads to better semantic richness than GNN-based point-cloud methods.

3.2 Molecule Prediction Model (YMPM)

YMPM integrates tertiary targets into a GAN to produce candidate molecules. The generator, conditioned on random noise, employs Residual Graph Attention message-passing layers [35] to construct molecular graphs. It predicts critical atomic properties—such as atom type, formal charge, hybridization, aromaticity—and determines bond existence and type. The model is trained on a mixture of MOSES and PDBbind data, providing a broad molecular distribution and facilitating generalization.

The discriminator, built using Residual Graph Isomorphism Network (GIN) layers [39], evaluates generated molecules by aggregating node-level features into a fixed-size representation. After global pooling, the network applies fully connected layers with LeakyReLU activations to produce a probability that the input molecule is real or generated. Training the discriminator uses Binary Cross-Entropy Loss [13], and gradient clipping [24] stabilizes updates.

YMPM's training procedure alternates between adversarial and RL objectives. For the first half of each epoch, the generator attempts to fool the discriminator, improving authenticity. For the second half, RL rewards are introduced to encourage chemically meaningful outcomes. Here, policy gradient methods [33] update the generator's parameters, rewarding correctness of valence states and bond structures, as well as diversity and novelty. Penalties are applied for invalid or redundant molecules. To prevent mode collapse, we employ temperature-based sampling strategies reminiscent of parallel tempering [7], allowing exploration of a broader chemical space. We also pause discriminator training when its loss becomes too small, ensuring balanced adversarial dynamics. Additional stabilizing strategies include gradient clipping [42], mixed precision [23], and early stopping [19]. All generated molecules undergo post-processing to fix valence.

Direct evaluation of protein-binding efficacy is intractable. In fact, determining actual protein-ligand binding affinity is an intricate subfield of machine learning. We rely on the MOSES framework and physicochemical metrics such as Quantitative Estimation of Drug-likeness (QED), molecular weight, valence, and internal diversity to evaluate performance. Internal diversity, as quantified in 1, correlates with how well the model's output generalizes to unseen chemical domains.

This approach improves upon traditional techniques in two primary ways. First, the model is lightweight and computationally efficient, generating molecules faster than many larger architectures. Second, the attention-driven graph modules and GIN-based discriminator, possessing the same expressiveness as the Weisfeiler–Lehman isomorphism test [39], provide a nuanced understanding of molecular structure. Coupled with RL to guide the generator toward chemically rich and diverse solutions, YMPM represents a flexible and powerful method for molecular generation.

3.3 Conditioned Molecular Generation Model

Building upon YMPM, we introduce a contrastive learning scheme with negative sampling to incorporate protein pocket information directly into the molecule generation process. By conditioning the generator on learned pocket embeddings, we aim to produce molecules more likely to bind at the specified sites. Although this approach conceptually merges the pocket-finding and molecular generation models into a unified system, the initial training runs did not yield satisfactory results. As such, we omit detailed experiments for this component in the sections that follow.

4 Experiments

This paper uses two datasets. The first, PDBBind [36], includes approximately 24,000 complexes with chemical codes for ligands and proteins. To address data noise, we preprocess the dataset by realigning atomic coordinates using custom Python scripts. Additionally, we remove hydrogen atoms due to their high positional variance, enabling our model to predict only the molecular backbones.

To generate 7, we use the VMD visualization software [14]. After cleaning and removing duplicates, we obtain 14,482 unique ligand-protein pairs for training. Figures 8a and 8b show that protein atom and residue counts are right-skewed, with means of approximately 5,000 atoms and 500 residues, respectively. Some polymers have significantly higher counts in both categories. We analyze ligand

descriptive statistics in 9. These molecular-level statistics, combined with atom-wise information, serve as features in our prediction pipeline. Compared to proteins, ligands exhibit more uniform distributions in atom and residue counts, with means below 100. Key features are defined in 3.

The second dataset is MOSES [26], a benchmarking platform for molecular generative models that provides a curated dataset, reference models, and standardized evaluation metrics to advance machine learning research in drug discovery. As shown in 10, the distributions of key molecular variables are highly similar between PDBBind and MOSES. However, MOSES is substantially larger than PDBbind, reducing the original ZINC Clean Leads collection from 4,591,276 to 1,936,962 molecular structures. PDBBind focuses on experimentally measured binding affinities for biomolecular complexes from the Protein Data Bank (PDB). It emphasizes protein-ligand complexes, linking binding affinity data with detailed structural information. Therefore, although PDBBind is smaller than MOSES, it provides more refined and experimentally validated protein-ligand complexes.

4.1 Pocket Prediction Model (YPFM)

We trained the pocket prediction model using a learning rate of 0.001. A comprehensive list of hyperparameters is provided in Table 7. Prior to training, proteins were voxelized into $32 \times 32 \times 32$ grids, ensuring model invariance to protein size variations. Each voxel was assigned eight features, computed using an intensive algorithm detailed in Table 3, resulting in a tensor of dimensions $32 \times 32 \times 32 \times 8$ for each protein.

Training involved seven iterations, during which loss and error metrics were recorded. Figure 5a illustrates the Binary Cross-Entropy (BCE) loss (10) and Mean Absolute Error (MAE) (11) between predicted in-pocket and out-of-pocket voxels. To mitigate overfitting, we incorporated dropout layers with a dropout rate of 0.2 and employed batch normalization after each convolutional layer.

In prior work, point-cloud-driven GNN models have been employed for related tasks. For instance, PGPocket [44] uses a point-cloud representation of the protein's surface and a GNN to identify binding sites. While such GNN approaches demonstrate a roughly 60% success rate when considering a predicted pocket correct if it lies within 4 Å of the true binding site, they can struggle to match the semantic richness that emerges from our voxel-based U-Net. In contrast, our 3D-UNet method achieves a voxel-wise mean average error (MAE) of just 2.5%, representing a substantial improvement. Although direct comparisons are not perfect due to differences in input representation (voxel grids versus point clouds) and output granularity, these results strongly suggest that the U-Net's global attention and rich contextual embeddings lead to more robust and accurate pocket prediction.

Feature importance was assessed using SHAP values, as shown in Table 2. These values indicate the relative contribution of each feature to accurate pocket prediction, with feature descriptions available in Table 3 [20].

4.2 Molecule Generation Model (YMPM)

We evaluated the molecular generation model on various subdomains and found it to perform strongly in several areas of interest. As shown in Figure 5, the reinforcement learning procedure stabilized the training process.

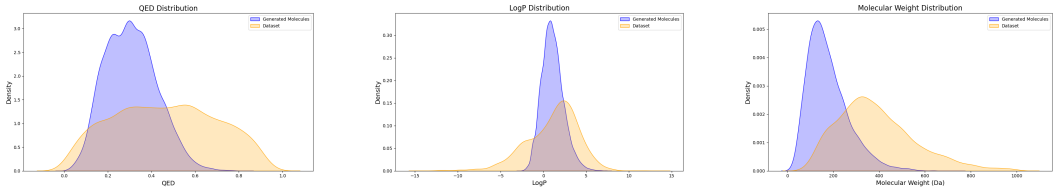
To evaluate the effectiveness of our direct generation approach, we compared the YMPM model against benchmark models provided by MOSES [26], which includes iterative and autoregressive deep network baselines such as CharRNN, VAE, AAE, ORGAN, and JTN-VAE. Although MOSES models generally outperform many state-of-the-art (SOTA) methods like GCPN in terms of speed, most remain iterative processes rather than direct generation methods. Utilizing a reinforcement learning framework, we employed hyperparameters detailed in Table 5 and additional configurations in Table 4, including a learning rate of 0.0001, an input noise dimension of 64, and at least eight graph layers in both the generator and discriminator. We one-hot encode atom types from ['C', 'O', 'N', 'H', 'S', 'Br', 'P', 'Cl', 'F', 'I'] and bond types as single, double, triple, or aromatic. Our results, presented in Table 1, demonstrate that the YMPM model achieves a 100% validity rate and the highest novelty score among all baselines, indicating its superior ability to explore new chemical spaces. Additionally, YMPM surpasses other models in internal diversity (IntDiv1) and exhibits high Unique@1k and Unique@10k values, showcasing its robust capacity to generate diverse and chemically novel molecular structures. Metrics like Valid Rate (1) and Novelty Rate (5) are critical for practical applications in drug discovery, while metrics like IntDiv1 (6) emphasize diversity within the generated set.

Table 1: Comparison of Baseline Models from MOSES [26] and YMPM Model

Model	Valid (↑)	Unique@1k (↑)	Unique@10k (↑)	Novelty (↑)	IntDiv1 (↑)	Runs (n)
Train	**1.000**	**1.000**	**1.000**	0.000	0.857	3
HMM	0.076 ± 0.0322	0.623 ± 0.1224	0.567 ± 0.1424	0.999 ± 0.001	0.847 ± 0.0403	3
NGram	0.238 ± 0.0025	0.974 ± 0.0108	0.922 ± 0.0019	0.969 ± 0.001	0.874 ± 0.0002	3
Combinatorial	**1.000 ± 0.0000**	0.998 ± 0.0015	0.991 ± 0.0009	0.988 ± 0.001	0.873 ± 0.0002	3
CharRNN	0.975 ± 0.0260	**1.000 ± 0.0000**	**0.999 ± 0.0000**	0.842 ± 0.051	0.856 ± 0.0000	3
VAE	0.977 ± 0.0010	**1.000 ± 0.0000**	0.998 ± 0.0010	0.695 ± 0.007	0.856 ± 0.0000	3
AAE	0.937 ± 0.0340	**1.000 ± 0.0000**	0.997 ± 0.0020	0.793 ± 0.028	0.856 ± 0.0030	3
JTN-VAE	**1.000 ± 0.0000**	**1.000 ± 0.0000**	0.999 ± 0.0003	0.914 ± 0.006	0.855 ± 0.0034	3
LatentGAN	0.897 ± 0.0020	**1.000 ± 0.0000**	0.997 ± 0.0050	0.949 ± 0.001	0.857 ± 0.0000	3
YMPM	**1.000 ± 0.0000**	0.932 ± 0.0150	0.788 ± 0.0200	**1.000 ± 0.0000**	**0.897 ± 0.0050**	8

Prior to post-processing, the model tends to overestimate valence, generating excessive bonds (Figure 4). Figure 1 shows that YMPM closely matches the QED and LogP distributions of the training set with KL-Divergence values of 0.48 and 0.40, respectively. Like other MOSES baselines, the generated distributions are somewhat narrower than the true distributions. Post-processing removes some disconnected atoms, shifting the molecular weight distribution away from the training data.

Reinforcement prevents mode collapse by encouraging atomic diversity. Without it, the model frequently collapses to repetitive structures (Figure 2). Examples generated with reinforcement learning (Figures 3a, 3b, and 3c) illustrate the model's improved capacity to produce unique and chemically meaningful compounds. Still, the model struggles to represent aromatic rings and overproduces edges. While reinforcement learning mitigates these issues, it cannot eliminate it entirely. Increasing the atom count also induces outsized numbers of errors. 6 shows the distribution of generated atoms is valid, and the model grasps small molecules better.



(a) QED Distribution (b) LogP Distribution (c) Molecular Weight Distribution

Figure 1: QED, LogP, and Molecular Weight Distribution Comparison

A key factor in reinforcement learning is maintaining a stable rewards distribution. We addressed this by computing percentile rank values for each generated molecule's discriminator-likelihood scores, rewarding those that effectively fooled the discriminator. Although we experimented with reward baselines to center new rewards around historical values, this did not yield notable improvements.

While the proposed direct generation and reinforcement learning strategy enhances molecular diversity, validity, and novelty, further refinements are required to handle complex molecular structures.

5 Conclusion

Our work introduces a novel RL-GAN framework for direct molecular graph generation, achieving 100% validity, improved novelty scores, and surpassing baselines in internal diversity on large datasets like MOSES, while significantly reducing per-molecule generation time. Although we have only briefly demonstrated the potential of pocket-conditioned generation, these results lay the groundwork for more advanced pocket-specific methods. By refining this approach and fully integrating pocket-level conditioning, we can advance cheminformatics toward generating highly targeted, chemically valid molecules for diverse biological applications.

6 Reproducibility

All code for reproduction of results can be found in [4].

Cited References

[1] Josh Abramson et al. "Accurate structure prediction of biomolecular interactions with AlphaFold 3". In: *Nature* 630.8016 (2024), pp. 493–498. DOI: 10.1038/s41586-024-07487-w.

[2] Anonymous. *Molecular De Novo Design through Transformer-based Reinforcement Learning*. OpenReview. Rejected by Transactions on Machine Learning Research (TMLR). Feb. 2024. URL: https://openreview.net/forum?id=3s8v6A14VN.

[3] Minkyung Baek et al. "Accurate prediction of protein structures and interactions using a three-track neural network". In: *Science* 373.6557 (2021), pp. 871–876. DOI: 10.1126/science.abj8754. eprint: https://www.science.org/doi/pdf/10.1126/science.abj8754. URL: https://www.science.org/doi/abs/10.1126/science.abj8754.

[4] Tristan Brigham. *Gerstein Lab Pocket Prediction*. Collaboration with Alan Ianeselli at Yale University focusing on machine learning for protein binding pocket prediction. 2024. URL: https://github.com/TristanB22/Gernstein_Lab_Pocket_Prediction.

[5] Vincent Cohen-Addad et al. *Hierarchical Clustering: Objective Functions and Algorithms*. 2017. arXiv: 1704.02147 [cs.DS]. URL: https://arxiv.org/abs/1704.02147.

[6] Gabriele Corso et al. *DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking*. 2023. arXiv: 2210.01776 [q-bio.BM]. URL: https://arxiv.org/abs/2210.01776.

[7] David J. Earl and Michael W. Deem. "Parallel tempering: Theory, applications, and new perspectives". In: *Physical Chemistry Chemical Physics* 7 (2005), pp. 3910–3916.

[8] Wei Feng et al. "Generation of 3D molecules in pockets via a language model". In: *Nature Machine Intelligence* 6.1 (2024), p. 62. DOI: 10.1038/s42256-023-00775-6.

[9] Matthias Fey and Jan Pieter Lenssen. "Fast graph representation learning with PyTorch Geometric". In: *NeurIPS Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[10] Luca Gagliardi and Walter Rocchia. "SiteFerret: Beyond Simple Pocket Identification in Proteins". In: *Journal of Chemical Theory and Computation* 19.15 (2023). PMID: 37470784, pp. 5242–5259. DOI: 10.1021/acs.jctc.2c01306. eprint: https://doi.org/10.1021/acs.jctc.2c01306. URL: https://doi.org/10.1021/acs.jctc.2c01306.

[11] Gerstein Lab. *Gerstein Lab Bioinformatics*. https://www.gersteinlab.org/. Accessed: 2024-12-08. 2013.

[12] Justin Gilmer et al. *Neural Message Passing for Quantum Chemistry*. 2017. arXiv: 1704.01212 [cs.LG]. URL: https://arxiv.org/abs/1704.01212.

[13] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[14] William Humphrey, Andrew Dalke, and Klaus Schulten. "VMD: Visual molecular dynamics". In: *Journal of Molecular Graphics* 14.1 (1996), pp. 33–38. DOI: 10.1016/0263-7855(96)00018-5.

[15] Ryuichiro Ishitani, Mizuki Takemoto, and Kentaro Tomii. "Protein Ligand Binding Site Prediction Using Graph Transformer Neural Network". In: *PLoS ONE* 19.4 (2024), e0308425.

[16] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. "Junction Tree Variational Autoencoder for Molecular Graph Generation". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. 2018, pp. 2323–2332.

[17] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589. DOI: 10.1038/s41586-021-03819-2.

[18] Shuya Li et al. "PocketAnchor: Learning structure-based pocket representations for protein-ligand interaction prediction". In: *Cell Systems* 14 (8 2023), 692–705.e6. DOI: 10.1016/j.cels.2023.05.005.

[19] Yuanzhi Li, Yingyu Liang, and Alexander Rakhlin. "Gradient Descent with Early Stopping is Provably Robust to Label Noise". In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. 2020, pp. 982–990. URL: http://proceedings.mlr.press/v108/li20j.html.

[20] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. 2017. URL: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[21] Łukasz Maziarka et al. "Mol-CycleGAN: a generative model for molecular optimization". In: *Journal of Cheminformatics* 12.1 (2020), p. 2.

[22] Amil Merchant et al. "Scaling deep learning for materials discovery". In: *Nature* 624 (2023), pp. 80–85. DOI: 10.1038/s41586-023-06735-9.

[23] Paulius Micikevicius et al. "Mixed Precision Training". In: *International Conference on Learning Representations*. 2018. URL: https://arxiv.org/abs/1710.03740.

[24] Paulius Micikevicius et al. "Mixed precision training". In: *International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2017, pp. 448–460.

[25] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[26] Daniil Polykovskiy et al. "Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models". In: *Frontiers in Pharmacology* (2020).

[27] PyG Team. *torch_geometric.nn.conv.GATConv*. Version 2.5.3. 2024. URL: https://pytorch-geometric.readthedocs.io/en/2.5.3/generated/torch_geometric.nn.conv.GATConv.html.

[28] Patrick Reiser et al. "Graph neural networks for materials science and chemistry". In: *Communications Materials* 3.1 (2022), p. 93. DOI: 10.1038/s43246-022-00315-6.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*. 2015, pp. 234–241.

[30] Muhammad Sarmad, Hyunjoo Jenny Lee, and Young Min Kim. *RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion*. 2019. arXiv: 1904.12304 [cs.CV]. URL: https://arxiv.org/abs/1904.12304.

[31] Schrödinger. *What is Chemical Space?* https://www.schrodinger.com/extrapolations/what-is-chemical-space. Accessed: 2024-09-16.

[32] Florian Sestak et al. "VN-EGNN: E(3)-Equivariant Graph Neural Networks with Virtual Nodes Enhance Protein Binding Site Identification". In: *arXiv preprint arXiv:2404.07194* (2024).

[33] Richard S Sutton and Andrew G Barto. "Reinforcement learning: An introduction". In: 2018.

[34] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML]. URL: https://arxiv.org/abs/1710.10903.

[35] Petar Veličković et al. "Graph Attention Networks". In: *arXiv preprint arXiv:1710.10903* (2018).

[36] Renxiao Wang et al. "The PDBbind Database: Collection of Binding Affinities for Protein–Ligand Complexes with Known Three-Dimensional Structures". In: *Journal of Medicinal Chemistry* 47.12 (2004), pp. 2977–2980. DOI: 10.1021/jm0305801.

[37] Felix Wong et al. "Discovery of a structural class of antibiotics with explainable deep learning". In: *Nature* 626.7997 (2024), p. 177. DOI: 10.1038/s41586-023-06887-8.

[38] Hongzuo Xu et al. "Deep Isolation Forest for Anomaly Detection". In: *IEEE Transactions on Knowledge and Data Engineering* 35.12 (Dec. 2023), pp. 12591–12604. ISSN: 2326-3865. DOI: 10.1109/tkde.2023.3270293. URL: http://dx.doi.org/10.1109/TKDE.2023.3270293.

[39] Keyulu Xu et al. *How Powerful are Graph Neural Networks?* 2019. arXiv: 1810.00826 [cs.LG]. URL: https://arxiv.org/abs/1810.00826.

[40] Pengcheng Xu et al. "REINVENT-Transformer: Molecular De Novo Design through Transformer-based Reinforcement Learning". In: *arXiv preprint arXiv:2310.05365* (2024).

[41] Jiaxuan You et al. *Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation*. 2019. arXiv: 1806.02473 [cs.LG]. URL: https://arxiv.org/abs/1806.02473.

[42] Jingzhao Zhang et al. "Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity". In: *International Conference on Learning Representations*. 2020. URL: https://arxiv.org/abs/1905.11881.

[43] Yang Zhang et al. "EquiPocket: an E(3)-Equivariant Geometric Graph Neural Network for Ligand Binding Site Prediction". In: *arXiv preprint arXiv:2302.12177* (2023).

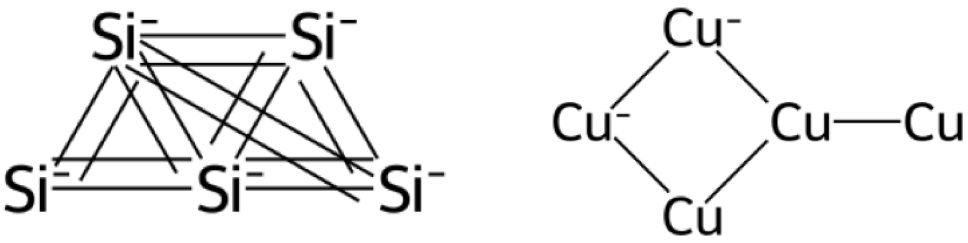
[44] Yanpeng Zhao et al. “A Point Cloud Graph Neural Network for Protein–Ligand Binding Site Prediction”. In: *International Journal of Molecular Sciences* 25.17 (2024), p. 9280. DOI: 10.3390/ijms25179280.

[45] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

A Additional Material

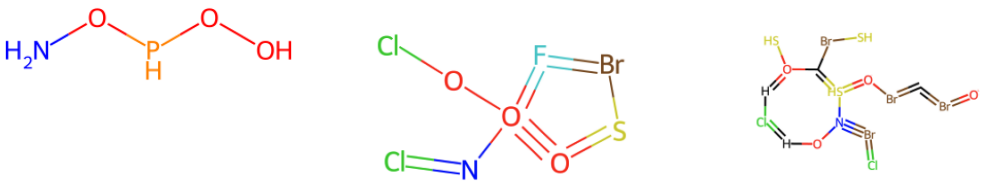
Below, the reader can find additional material that is helpful to the understanding of the project. Figures are included with brief explanations of their relevance to the project.

A.1 Visualizations



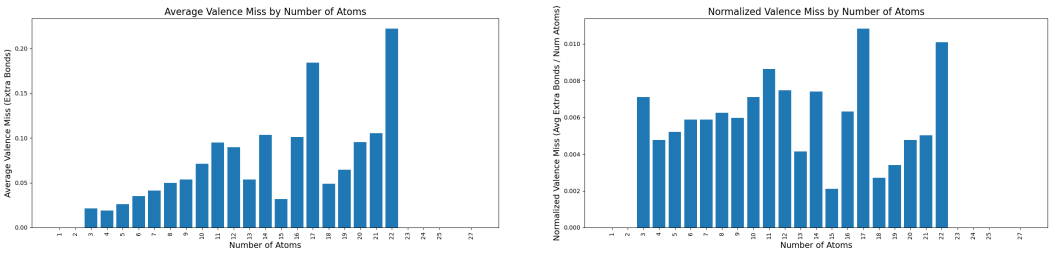
(a) Impossible Molecule (b) Uniform Molecule

Figure 2: Early Generated Molecules (Pre-RL)



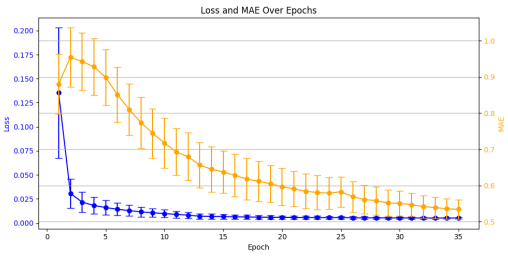
(a) Good Small Molecule (b) Moderate Medium Molecule (c) Poor Large Molecule

Figure 3: Examples of generated molecules: small, medium, and large molecules, respectively.

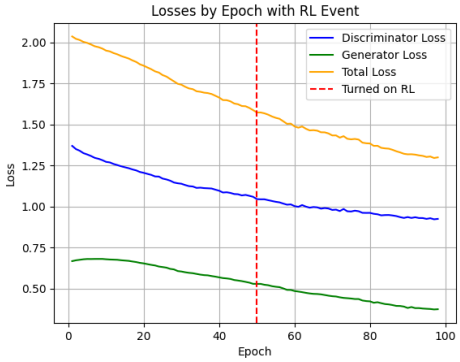


(a) Average valence miss by number of atoms. (b) Normalized valence miss by number of atoms.

Figure 4: Comparison of valence miss metrics.

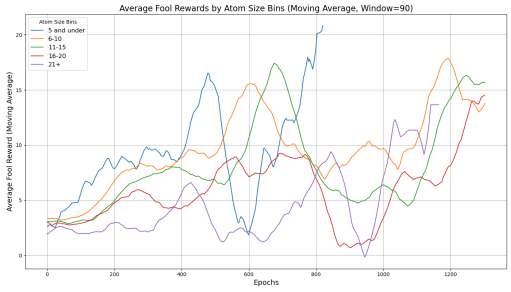


(a) MAE and BCE Loss for Pocket Prediction with UNet on 7 Runs.

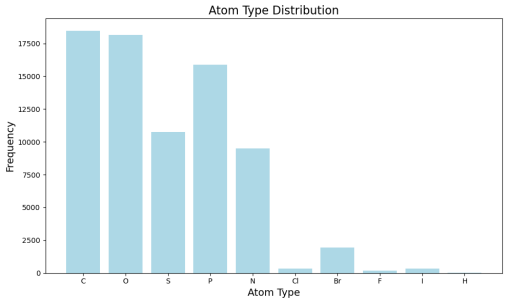


(b) Reinforcement Learning GAN Loss Plot.

Figure 5: Loss metrics for UNet and Reinforcement Learning GAN models.

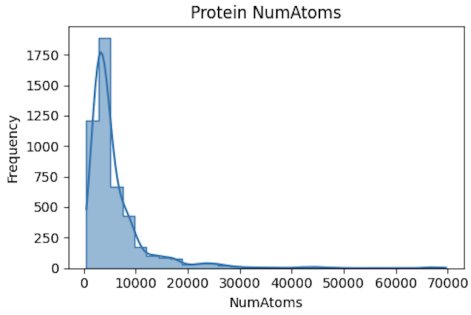


(a) Average Fool Loss by Molecule Size

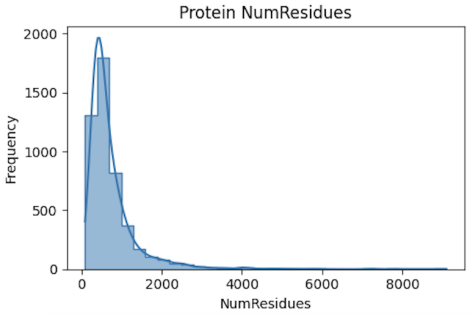


(b) Atom Type Distribution in Generated Dataset

Figure 6: Analysis of Generated Molecules: (a) Average Fool Loss by Molecule Size, and (b) Atom Type Distribution in the Generated Dataset.

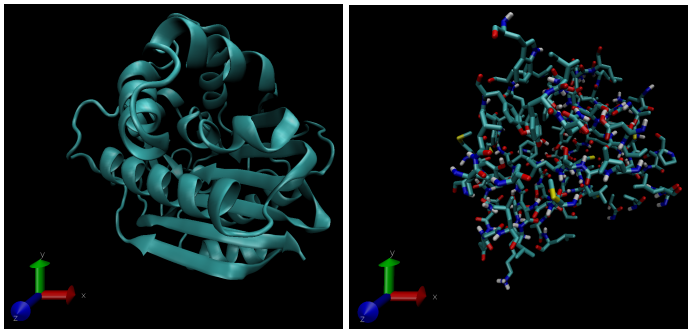


(a) Protein # Atoms Distribution

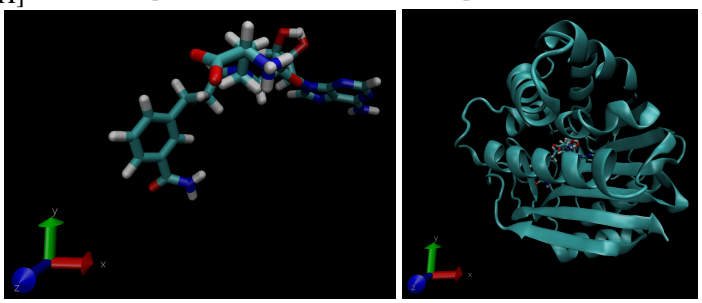


(b) Protein # Residues Distribution

Figure 8: [PDBbind] Protein Statistics

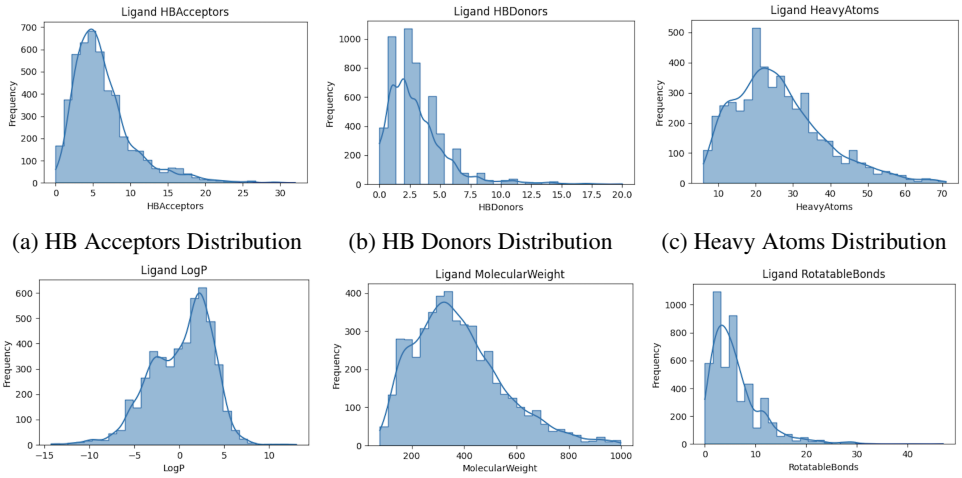


[H] (a) Example Protein (6PVE) (b) Example Protein Pocket (6PVE)



(c) Example Ligand (6PVE) (d) Example Ligand in Protein

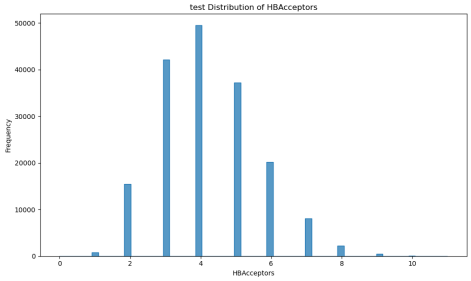
Figure 7: [PDBbind] An Example Ligand, Protein and Their Likely Binding Configuration



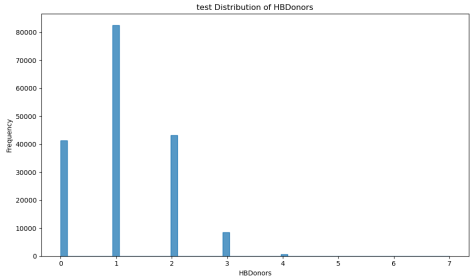
(a) HB Acceptors Distribution (b) HB Donors Distribution (c) Heavy Atoms Distribution

(d) LogP Distribution (e) Molecular Weight Distribution (f) Rotatable Bonds Distribution

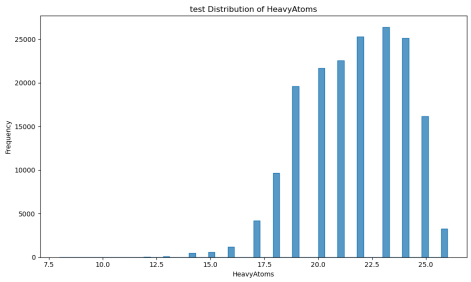
Figure 9: [PDBbind] Distributions of Descriptive Statistics for PDBBind Dataset



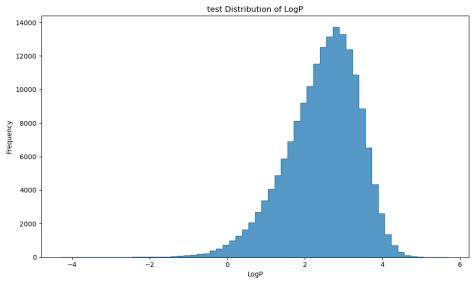
(a) HBAcceptors Distribution



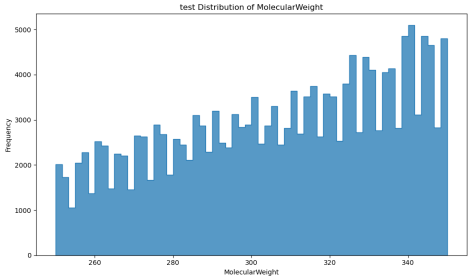
(b) HBDonors Distribution



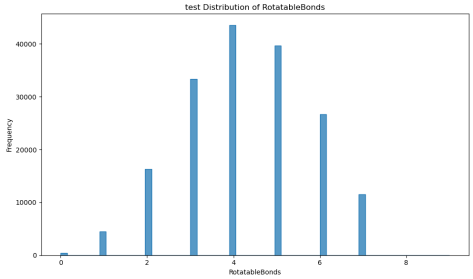
(c) Heavy Atoms Distribution



(d) LogP Distribution



(e) Molecular Weight Distribution



(f) Rotatable Bonds Distribution

Figure 10: Distributions of Descriptive Statistics for MOSES Dataset [26]

A.2 Tables

Table 2: SHAP Explanation Values (Mean and Std. Deviation) (n = 7)

Feature	Mean SHAP Value	Standard Deviation
N	0.000829	0.000035
bfactors	0.000043	0.000002
buriedness	0.000201	0.000010
charge	0.000077	0.000004
radius	0.000125	0.000006
hbdon	0.000102	0.000005
hbacc	0.000117	0.000007
sasa	0.000042	0.000002

Table 3: Descriptions of Voxelized Protein Features and their Significance in the Model.

Feature	Description and Significance
N	Represents the normalized number of atoms that can be found in a given pocket of the voxelized protein that we are considering.
bfactors	Indicates flexibility or thermal motion of residues in the protein voxel that is being considered.
buriedness	Measures the minimum number of atoms that are between this atom and an open space of a minimum volume in the molecule.
charge	Refers to electrostatic charge in the voxel.
radius	Represents atomic or residue radii in the voxel. This information is critical to assessing the spatial discrimination of molecular features, such as steric effects or binding pockets.
hbdon	The number of hydrogen donor groups that exist in this voxel – helpful for predicting interaction sites.
hbacc	The number of hydrogen bond acceptors in the voxel – helpful for predicting interaction sites.
sasa	Reflects solvent-accessible surface area in the voxel.

Table 4: Descriptions of additional configurations used in the molecule generation process.

Configuration	Description	Value
`Checkpointing`	Saves model weights every 5 epochs during training to prevent data loss and allow for resuming training.	Every 5 epochs
`Visualization`	Generates and saves images of molecular structures at each epoch for inspection and monitoring.	Enabled
`Validation Flags`	Configurable flags that enable or disable specific structural penalties and rewards during training.	Enabled
`Baseline Samples`	Number of molecules used to create a baseline z-score distribution for reward normalization.	1000
`Logging Level`	Controls the granularity of logs during training, set to DEBUG for detailed tracking.	DEBUG
`Large Size Threshold`	The size threshold beyond which penalties for molecule size apply.	60

Table 5: Descriptions of reinforcement learning penalties and rewards used in the molecule generation process.

Validity Penalty (VALIDITY_PENALTY)	Penalizes molecules with invalid structures, such as atoms with valences exceeding their allowed maximum.
Disconnect Penalty (DISCONNECT_PENALTY)	Penalizes generated molecules that contain disconnected components, rewarding those that form a single connected graph.
Distribution Penalty (DISTRIBUTION_PENALTY)	Penalizes generated molecules whose atom type distribution deviates from the target dataset distribution.
Same Atom Penalty (SAME_ATOM_PENALTY)	Penalizes molecules with excessive repetition of a single atom type beyond a defined threshold.
Duplicate Molecule Penalty (DUPLICATE_PENALTY)	Penalizes generating identical molecules within a batch to encourage diversity.
Edge Density Penalty (EDGE_DENSITY_PENALTY)	Penalizes molecules with edge density significantly different from the dataset average.
Invalid Edge Penalty (INVALID_EDGES_SCALING)	Penalizes molecules with invalid edges, such as bonds between non-existing atoms or self-loops.
Edge Count Penalty (EDGE_COUNT_PENALTY_SCALING)	Penalizes molecules with an excessive number of edges, discouraging overly connected structures.
Size Penalty (SIZE_SCALER)	Penalizes molecules that exceed a predefined maximum size by a proportional amount.
Valence Penalty (VALENCE_PENALTY_SCALE)	Penalizes atoms in a molecule whose explicit and implicit valences exceed their allowed maximum.
Validity Reward (VALIDITY_REWARD)	Rewards molecules with valid structures, encouraging the generation of chemically feasible compounds.
Fool Reward (FOOL_REWARD)	Rewards molecules that successfully fool the discriminator, promoting the generator's ability to create realistic structures.
Motif Reward (MOTIF_REWARD)	Rewards molecules containing specific chemical motifs, fostering the generation of meaningful and functional structures.
Similarity Reward (SIMILARITY_REWARD)	Rewards molecules that are similar to those in the training dataset, maintaining consistency with learned distributions.

Table 6: Descriptions of hyperparameters used in the molecule generation process.

Hyperparameter	**Description**	**Value**
`Batch Size`	The number of samples processed in one training iteration. Affects gradient stability and memory usage.	128
`Epochs`	The number of complete passes through the dataset during training.	50
`Learning Rate`	The step size for updating model parameters during optimization.	0.0001
`Noise Dimension`	The size of the latent vector input to the generator for creating molecules.	64
`Hidden Dimension`	The size of the hidden layers in both generator and discriminator models.	256
`Number of Layers`	The depth of the neural networks in the generator and discriminator.	8
`Maximum Nodes per Molecule`	The upper limit on the number of nodes (atoms) in a generated molecular graph.	50
`Reinforcement Learning Factor`	Determines how many RL updates are performed per epoch.	10
`Temperature`	Adjusts the sharpness of probability distributions during sampling, influencing exploration vs. exploitation.	1.0
`Scaling Factor`	Amplifies the impact of RL updates on generator parameters.	0.5
`RL Batch Size`	The number of samples in each batch during reinforcement learning updates.	64

Table 7: Descriptions of hyperparameters used in the UNet model training process.

Hyperparameter	**Description**	**Value**
`Batch Size`	The number of samples processed in one training iteration. Affects gradient stability and memory usage.	8
`Epochs`	The number of complete passes through the dataset during training.	40
`Learning Rate`	The step size for updating model parameters during optimization.	0.001
`Optimizer`	The optimization algorithm used to update model weights.	`Adam`
`Early Stopping`	Whether to halt training if no improvement in loss is observed.	True
`Early Stopping Epochs`	The number of consecutive epochs without improvement to trigger early stopping.	5
`Number of Workers`	The number of subprocesses used for data loading.	6
`Use Attention Layers`	Whether to include attention mechanisms in the UNet architecture.	True
`Save Directory`	The directory where trained models are saved.	`Trained_Models`

A.3 Mathematical Equations

This section describes the evaluation metrics used to assess the performance of the molecular generator in the MOSES paper.

A.3.1 Total Generated

Definition: The total number of molecules generated by the model. This provides a baseline for understanding the scale of the generation task.

A.3.2 Valid Rate

Definition: The fraction of generated molecules that are chemically valid. A molecule is valid if it adheres to chemical rules, such as valence constraints, and can be sanitized by RDKit.

$$\text{Valid Rate} = \frac{\text{Number of Valid Molecules}}{\text{Total Generated Molecules}} \tag{1}$$

A.3.3 Unique@1k and Unique@10k

Definition: The fraction of unique molecules in the first $1,000$ and $10,000$ valid molecules, respectively.

$$\text{Unique@1k} = \frac{\text{Number of Unique Molecules in First 1,000 Valid}}{1,000} \tag{2}$$

$$\text{Unique@10k} = \frac{\text{Number of Unique Molecules in First 10,000 Valid}}{\text{Total Valid in First 10,000}} \tag{3}$$

A.3.4 Filters

Definition: The fraction of valid molecules that pass chemical filters, such as BRENK, which eliminate toxic or unstable substructures.

$$\text{Filters Rate} = \frac{\text{Number of Molecules Passing Filters}}{\text{Total Valid Molecules}} \tag{4}$$

A.3.5 Novelty

Definition: The fraction of valid molecules that do not appear in the training dataset.

$$\text{Novelty Rate} = \frac{\text{Number of Novel Molecules}}{\text{Total Valid Molecules}} \tag{5}$$

A.3.6 Internal Diversity (IntDiv1)

Definition: A measure of diversity among generated molecules based on the average Tanimoto similarity of their molecular fingerprints.

$$\text{IntDiv1} = 1 - \text{Average Tanimoto Similarity} \tag{6}$$

A.3.7 Average Bonds Removed for Valence

Definition: The average number of bonds removed to fix valence violations in generated molecules.

$$\text{Avg Bonds Removed} = \frac{\text{Total Bonds Removed}}{\text{Total Molecules}} \tag{7}$$

A.3.8 Average Invalid Valences Before Removal

Definition: The average count of invalid valences detected before bond removal.

$$\text{Avg Invalid Valences} = \frac{\text{Total Invalid Valences Detected}}{\text{Total Molecules}} \tag{8}$$

A.3.9 UNet Model Structure

Definition: The mathematical representation of the UNet model.

$$\hat{y} = f_{\text{decoder}}(f_{\text{bottleneck}}(f_{\text{encoder}}(x))) \tag{9}$$

A.3.10 Binary Cross Entropy Loss ($\mathcal{L}_{\text{BCE}}$)

Definition: Finds the difference between the predicted probabilities and the actual binary labels.

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{10}$$

A.3.11 Mean Absolute Error (MAE)

Definition: Mean Absolute Error (MAE) gets the average absolute differences between predicted values and the real target values.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{11}$$