

J Extended Technical Appendix

This Extended Technical Appendix provides a deeper dive into the experimental methodology, specific hyperparameter choices that led to the main paper’s results, additional sensitivity analyses, and information regarding the availability of code and datasets used in this work. The sections are organized as follows:

- **Code and Data Availability** (Section J.1): Links to the LD-FPG implementation and the D2R-MD dataset.
- **Estimated CO₂-equivalent emissions** (Section J.2): Discussion on the computational resources and environmental impact.
- **ChebNet Encoder Details and Ablation** (Sections J.3 and J.4): Parameters for the encoder used in main results and a sensitivity analysis on its key hyperparameters (k , K , d_z).
- **Decoder Architectures and Configurations** (Section J.5): Detailed configurations for Blind, Sequential, and Residue pooling strategies corresponding to main paper results (Tables 1 and 2), including final chosen pooling dimensions and dihedral loss settings for Blind Pooling fine-tuning. This is supplemented by illustrative hyperparameter scan examples for Blind Pooling decoders (Section J.7).
- **Latent Diffusion Model Details** (Sections J.6 and J.8): Overview of diffusion model configurations and example hyperparameter scans for the Blind Pooling strategy.
- **Conditioning Mechanism Tuning** (Section J.9): Ablation study results for the decoder conditioning mechanism.
- **Discussion on Metric Variability** (Section J.10): Contextualization of standard deviations for reported performance metrics.

While comprehensive hyperparameter sweeps were conducted for all components and strategies, due to space constraints, we primarily showcase detailed scan results for the Blind Pooling strategy to illustrate the optimization process. Similar systematic approaches were employed for other strategies. For exhaustive details beyond what is presented, readers are encouraged to consult the provided codebase or contact the authors.

J.1 Code and Data Availability

To facilitate reproducibility and further research, the LD-FPG implementation (including scripts for data preprocessing, model training, and evaluation) is made available at:

- **Code (LD-FPG)**: <https://github.com/adityasengar/LD-FPG>

The D2R-MD dataset, comprising the 2 μ s MD trajectory (12,241 frames) of the human dopamine D2 receptor used in this study, is available at:

- **Dataset (D2R-MD)**: <https://zenodo.org/records/16582853>

The dataset includes aligned heavy-atom coordinates and pre-calculated dihedral angles, along with the necessary topology files as described in Appendix B.

J.2 Estimated CO₂-equivalent emissions

To provide a transparent measure of the environmental cost of our computational pipeline, we estimate the operational (Scope-2) footprint of the $\sim 30,000$ GPU-hours consumed in this work. This total compute time was a mix of NVIDIA H100 and L40S GPUs. For our typical training workloads, H100 GPUs offered a significant speed-up, completing tasks in roughly half the time compared to L40S GPUs. For instance, a ChebNet autoencoder training of 5,000 epochs took approximately 2 seconds per epoch on an H100 GPU (totaling ~ 2.8 H100-hours for one such run).

Following [90], the *per-GPU-hour* emissions are:

$$\text{kg CO}_2\text{e} = \left(\frac{P_{\text{board}}}{1000} \right) \times u \times \text{PUE} \times \text{CI}, \quad (8)$$

where P_{board} is the vendor-specified board power, u is the average utilisation of that cap, PUE is the data-centre power usage effectiveness, and CI is the grid carbon intensity.

Assumptions. Board powers were taken from the NVIDIA L40S datasheet ($P = 350$ W) [91] and the H100 SXM-5 specification ($P = 700$ W) [92]. We adopt a conservative mean utilisation $u = 0.8$, and a modern colocation PUE of 1.30, which is below the current global fleet average of 1.56 but well within the range observed for new facilities [93]. For the electricity mix we give two reference cases: (i) the 2022 global mean ($\text{CI} = 0.436 \text{ kg CO}_2\text{e kWh}^{-1}$) derived from Ember’s Global Electricity Review [94], and (ii) the Swiss consumption-based factor of $0.112 \text{ kg CO}_2\text{e kWh}^{-1}$ for the same year, which captures the low-carbon hydro–nuclear mix and net imports [95].

Results. Table 5 summarises six scenarios. The ”all-L40S” and ”all-H100” scenarios are hypotheticals where the entire 30,000 GPU-hours were run on a single GPU type. More realistically, a mix of GPUs was used. For an illustrative scenario, we consider a split where 70% of the total operational hours (21,000 hours) were logged on L40S GPUs and 30% (9,000 hours) on H100 GPUs. The energy consumption for this specific split is calculated as the sum of energy consumed by each GPU type for their respective hours. Even the worst-case (all-H100 on the global-average grid) remains on the order of a single round-trip transatlantic flight for two passengers, while execution on a Swiss-like grid cuts emissions by $\sim 4\times$.

Table 5: Estimated operational CO_2e emissions for the $\sim 30,000$ GPU-hours used in this study. Energy is computed as $(P_{\text{board}}/1000) \times \text{Hours} \times u \times \text{PUE}$, with $u = 0.8$ and $\text{PUE} = 1.30$. For the 70/30 split, L40S = 21,000 hours; H100 = 9,000 hours.

Scenario	Energy (kWh)	Grid CI (kg/kWh)	PUE	tCO ₂ e
All L40S (30k hrs)	10,920	0.436	1.30	4.8
All H100 (30k hrs)	21,840	0.436	1.30	9.5
70% L40S / 30% H100	14,196	0.436	1.30	6.2
L40S (Swiss mix)	10,920	0.112	1.30	1.2
H100 (Swiss mix)	21,840	0.112	1.30	2.4
70% L40S / 30% H100 (Swiss)	14,196	0.112	1.30	1.6

Context and mitigation. These figures exclude embodied (Scope-3) emissions of the hardware itself, which may add $\sim 2\text{--}3 \text{ tCO}_2\text{e}$ over its lifetime. Practitioners can further reduce impact by (a) scheduling training in regions with low-carbon grids, (b) capping GPU power (the H100 supports dynamic 350–700 W limits), (c) early stopping during training and (d) prioritising model efficiency improvements that reduce GPU runtime.

J.3 Graph Construction and ChebNet Encoder Parameters for Main Results

For each MD frame, the graph edge index $E^{(t)}$ was constructed using a k -Nearest Neighbors search with $k = 4$ on the aligned coordinates. This value of k was chosen as it was found to maintain graph connectivity effectively while managing computational cost for graph construction at each step. The ChebNet encoder (Section 3.2) utilizes Chebyshev polynomials up to order $K = 4$. The network comprises $L_{\text{ChebNet}} = 4$ ChebConv layers. The first two layers use LeakyReLU activations, the third employs ReLU, and BatchNorm1d is applied after each ChebConv layer. The final layer’s output, the atom-wise latent embeddings $Z^{(t)}$, is L_2 normalized. The encoder latent dimension d_z was varied for different pooling strategies, with $d_z \in \{4, 8, 16\}$ being explored. The specific d_z values used for the main results with each pooling strategy are detailed in Section J.5. Key parameters for the ChebNet encoder architecture and its pre-training regimen are summarized in Table 6.

J.4 ChebNet Encoder Ablation: k-NN, Chebyshev Order (K), and Latent Dimension (d_z)

This subsection details an ablation study conducted on the ChebNet autoencoder to explore the impact of varying key graph construction and architectural parameters: the number of nearest neighbors (k) for graph construction, the Chebyshev polynomial order (K), and the latent dimension (d_z). For

Table 6: ChebNet Encoder Model parameters and training Hyperparameters used for generating main paper results.

Parameter	Value(s)
Input Features	3 (Coordinates)
Number of Atoms (N, D2R)	2191
kNN Graph k (k)	4
ChebConv Order (K)	4
Number of Layers (L_{ChebNet})	4
Hidden Dimension (d_z)	Tuned per pooling (see Sec. J.5: 4, 8, or 16)
Activation (L1, L2)	
Activation (L3)	LeakyReLU
Normalization	ReLU
Final Embedding Norm	BatchNorm1d
Optimizer	L2
Learning Rate	Adam
Batch Size	1e-4
Epochs (Pre-training)	16
Dihedral Loss during Pre-training	4000
	Disabled

each parameter combination presented in Table 7, the ChebNet autoencoder was trained for approximately 24 hours, and we report the final test loss (coordinate MSE) achieved and the average time per epoch.

It is important to note that if the performance figures here, particularly the final test losses, differ from those presented in Appendix H (Table 4), it is primarily because the encoder models in Appendix H were trained for a fixed 5000 epochs using specific parameters ($k = 4, K = 4$) which we determined yielded a sufficiently low MSE for subsequent stages of the LD-FPG pipeline. The experiments here, with a fixed 24-hour training budget, provide a different perspective on parameter sensitivity under time-constrained training.

The results in Table 7 demonstrate that the ChebNet autoencoder architecture is capable of achieving low reconstruction loss across a range of parameter settings. Generally, increasing the latent dimension d_z tends to decrease the final test loss, albeit with a slight increase in epoch time. The influence of k and K is more nuanced, showing that different combinations can yield strong performance. For instance, with $K = 4$, increasing k from 4 to 8 and then to 16, particularly for $d_z = 16$, resulted in the lowest test loss (0.000110) in this set of experiments ($K = 4, k = 8, d_z = 16$). Higher orders of K (e.g., $K = 8, 16$) also show competitive losses but come with a more significant increase in epoch computation time.

Based on these findings, the ChebNet autoencoder effectively reduces reconstruction loss under various conditions. Users and researchers are encouraged to explore these parameters to optimize performance for their specific protein systems and computational budgets. The subsequent pooling strategies (Blind, Sequential, and Residue pooling) and the main results presented in this paper were developed based on our preliminary analyses which utilized $K = 4$ and $k = 4$. This choice offered a good balance between model performance and the computational cost associated with graph construction and training, especially for the extensive MD datasets used.

Table 7: ChebNet Encoder Ablation Study Results. Each configuration was trained for approximately 24 hours.

Chebyshev Order (K)	k-NN Value (k)	Hidden Dim (d_z)	Avg Epoch Time (s)	Final Test Loss
4	4	4	4.63	0.001756
4	4	8	4.66	0.000184
4	4	16	4.74	0.000263
4	8	4	4.74	0.000371
4	8	8	4.70	0.000373
4	8	16	4.97	0.000110
4	16	4	5.45	0.001016
4	16	8	5.20	0.000387
4	16	16	5.91	0.000226
8	4	4	6.99	0.000448
8	4	8	6.91	0.000698
8	4	16	7.08	0.000649
8	8	4	7.16	0.000624
8	8	8	7.13	0.000562
8	8	16	7.34	0.000545
8	16	4	7.52	0.000256
8	16	8	7.68	0.000256
8	16	16	10.73	0.001494
16	4	4	11.52	0.000262
16	4	8	11.38	0.000484
16	4	16	11.81	0.001428
16	8	4	11.86	0.003231
16	8	8	11.55	0.001885
16	8	16	12.27	0.000741
16	16	4	12.82	0.003709
16	16	8	14.89	0.001196
16	16	16	17.49	0.003388

J.5 Decoder Architectures: General Configuration and Key Parameters for Main Results

All decoder architectures map atom-wise latent embeddings Z and a conditioner $C = Z_{\text{ref}}$ (the latent embedding of the reference structure X_{ref}) to Cartesian coordinates X_{pred} . General training parameters for decoders included the Adam optimizer, a batch size of 16, and initial training for 5000 epochs with a learning rate of 3×10^{-4} , unless specified otherwise for fine-tuning or specific pooling strategies.

Blind Pooling Strategy Details Atom-wise embeddings $Z^{(t)}$ are globally pooled using 2D adaptive average pooling to a target dimension $d_p = H \times W$. This global context, expanded per-atom, is concatenated with C and fed to an MLP (MLP_{blind}) with ReLU activations and BatchNorm1d. The configuration that yielded the main paper results for Blind Pooling ($d_z = 16$, Tables 1 & 2) is detailed in Table 8. This table also includes the parameters for the dihedral fine-tuning stage. Detailed hyperparameter scans for the Blind Pooling decoder MLP varying H , W , and N_{layers} are presented in Section J.7.

Sequential Pooling Strategy Details ($d_z = 8$) This strategy involves a BackboneDecoder followed by a SidechainDecoder.

- **Backbone Decoder Stage Configuration (for Tables 1 & 2):**
 - Pooling: Backbone-specific atom embeddings ($d_z = 8$) were pooled to $d_{p,bb} = H_{bb} \times W_{bb} = 45 \times 3 = 135$.
- **Sidechain Decoder Stage Configuration (for Tables 1 & 2):**

Table 8: Blind Pooling Decoder Hyperparameters for Main Paper Results ($d_z = 16$).

Parameter	Value
Encoder Latent Dim (d_z)	16
Conditioner (C)	Z_{ref}
Pooling Type	Blind
Pooling Height (H)	50
Pooling Width (W)	2
Pooled Dimension (d_p)	100
MLP Hidden Dimension	128
MLP Total Layers (N_{layers})	12
MLP Activation	ReLU
MLP Normalization	BatchNorm1d
Optimizer	Adam
Learning Rate (Initial)	3e-4
Batch Size	16
Epochs (Initial Training)	5000
<i>Fine-tuning with Dihedral Loss (for Table 1)</i>	
Base Coord Weight (w_{base})	1.0
Dihedral Loss Enabled	Yes
Stochastic Fraction (f_{dih})	0.1
Dihedral Divergence Type	KL
Dihedral MSE Weight (λ_{mse})	0.00
Dihedral Div Weight (λ_{div})	2.00
Epochs (Fine-tuning)	2000
Torsion Info File	condensed_residues.json

- Architecture: `arch_type=1` (see Appendix D.2.2).
- Pooling: Sidechain-specific atom embeddings ($d_z = 8$) were pooled to $d_{p,sc} = H_{sc} \times W_{sc} = 54 \times 2 = 108$.

Residue Pooling Strategy Details ($d_z = 4$) Per-residue atom embeddings $Z_{R_j}^{(t)}$ are pooled to local contexts $z_{\text{res},j}$.

• **Decoder Configuration (for Tables 1 & 2):**

- Pooling: For each of $N_{\text{res}} = 273$ residues, embeddings ($d_z = 4$) were pooled to $d'_p = 1 \times W = 1 \times 4 = 4$.

J.6 Latent Diffusion Models: Configuration and Training Overview

The Denoising Diffusion Probabilistic Models (DDPMs) [47], as outlined in Appendix E, were implemented to operate on the pooled latent embeddings \mathbf{h}_0 derived from each specific pooling strategy (with dimensions $d_p = 100$ for Blind, $d_p = 135$ for Sequential Backbone, $d_p = 108$ for Sequential Sidechain, and $d_p = 1092$ for Residue, as detailed in Section J.5). Prior to input, these \mathbf{h}_0 embeddings were normalized (zero mean, unit variance) based on statistics from the training set.

Denoising Network Architectures (ϵ_θ) The primary denoising network used for generating the main paper results was a multi-layer perceptron, `DiffusionMLP_v2`, consisting of four linear layers with ReLU activations. This network takes the flattened, noisy latent vector \mathbf{h}_t concatenated with the normalized timestep t/T as input.

- For **Blind Pooling** ($d_p = 100$) and **Sequential Pooling** (for both backbone $d_{p,bb} = 135$ and sidechain $d_{p,sc} = 108$ latents, typically handled by separate diffusion models or a model adapted for the concatenated dimension if combined), the MLP hidden dimension was set to 1024.
- For **Residue Pooling** (operating on the effective concatenated input of $d_p = 1092$), the MLP hidden dimension was increased to 4096 to accommodate the larger input dimensionality.

Alternative MLP architectures (e.g., DiffusionMLP, DiffusionMLP_v3 with varying layer counts and hidden sizes) and a DiffusionConv2D model (treating the pooled latent as an image-like tensor) were also explored, particularly during the hyperparameter scans for the Blind Pooling strategy (see Section J.8).

Diffusion Process and Training Details The DDPMs were trained by minimizing the MSE loss between the true and predicted noise (Eq. 2).

- **Variance Schedule:** A linear variance schedule was used for β_t , progressing from β_{start} to β_{end} over T timesteps.
- **Key Hyperparameters (Typical Values for Main Results):**
 - Total diffusion timesteps (T): Commonly set around 500 (e.g., for Blind/Sequential) or higher (e.g., 1500 for Residue, reflecting findings from grid searches).
 - β_{start} : Typically 5×10^{-6} or 0.005.
 - β_{end} : Typically 0.02 or 0.1.

Specific optimal values were determined through grid searches, examples of which are shown in Section J.8.

- **Optimization:** The Adam optimizer was used with a learning rate typically set to 1×10^{-5} .
- **Training Epochs:** Blind and Sequential pooling diffusion models were trained for approximately 3,000-10,000 epochs. The Residue pooling diffusion model, with its larger latent space, often required more extensive training, up to 150,000 epochs or more, to converge.
- **Data Handling:** Pooled latent embeddings (\mathbf{h}_0) were generated from the entire training set using the best-performing frozen encoder and the respective trained decoder’s pooling mechanism, then saved to HDF5 files for efficient loading during diffusion model training. For Residue pooling, this involved concatenating the N_{res} per-residue pooled vectors for each frame.
- **Checkpointing:** Model checkpoints were saved periodically (e.g., every 50 or 100 epochs) to select the best performing model based on validation metrics (if a validation set of latents was used) or training loss convergence. For Residue pooling, which benefited from multi-epoch sampling for visualization (Appendix I), more frequent checkpointing was sometimes employed.

Sampling New latent embeddings $\mathbf{h}_0^{\text{gen}}$ were generated by the trained ϵ_θ model using the ancestral sampling procedure described in Algorithm 2. These sampled latents were then un-normalized (using the saved training set statistics) before being passed to the corresponding frozen decoder to generate all-atom coordinates $X_{\text{pred}}^{\text{gen}}$.

J.7 Blind Pooling Decoder Hyperparameter Scan Results

Full test set Mean Squared Error (MSE) results (MSE_{bb} for backbone, MSE_{sc} for sidechain) for the 50 Blind Pooling decoder configurations tested for each encoder latent dimension ($d_z \in \{4, 8, 16\}$) are presented in Tables 9-11. These experiments systematically varied the pooling height (H), pooling width (W), and the number of layers in the prediction MLP ($D \equiv N_{\text{layers}}$).

Table 9: Extended Technical Appendix: Full blind pooling Decoder Test MSE Results ($d_z = 4$).

H	W	N_{layers}	MSE_{bb}	MSE_{sc}
10	1	12	0.2520	0.7303
10	2	4	0.1834	0.5825
10	2	8	0.1673	0.5212
15	1	12	0.1965	0.5815
15	1	8	0.2092	0.5998
15	2	4	0.1405	0.4809
15	2	8	0.1820	0.5614
20	1	8	0.1845	0.5507
20	2	4	0.1257	0.4462
20	3	12	0.1081	0.4075
25	1	12	0.1737	0.5146
30	2	4	0.1038	0.3974
30	2	8	0.1064	0.3979
30	3	12	0.0910	0.3712
30	3	4	0.0931	0.3793
35	3	12	0.0888	0.3620
35	3	4	0.0905	0.3614
40	1	12	0.1504	0.4688
40	2	12	0.1006	0.3791
45	1	8	0.1513	0.4630
50	1	12	0.1477	0.4535
50	1	8	0.1442	0.4501
50	2	12	0.0978	0.3705
50	3	8	0.0783	0.3341
55	1	12	0.1492	0.4465
55	1	4	0.1466	0.4461
55	1	8	0.1488	0.4488
55	3	8	0.0820	0.3320
5	1	8	0.4069	1.1194
5	3	8	0.2970	0.8042
60	2	12	0.0884	0.3439
60	3	4	0.0777	0.3159
65	2	4	0.0989	0.3617
65	3	12	0.0791	0.3221
70	1	8	0.1417	0.4279
70	2	12	0.0940	0.3475
75	2	8	0.0919	0.3454
75	3	8	0.0837	0.3211
80	2	4	0.0994	0.3441
85	1	12	0.1427	0.4171
85	1	4	0.1344	0.4082
85	3	4	0.0812	0.3143
85	3	8	0.0815	0.3100
90	1	4	0.1396	0.4112
90	3	4	0.0809	0.3081
90	3	8	0.0794	0.3013
95	1	8	0.1378	0.4088
95	2	12	0.0819	0.3109
95	2	4	0.0973	0.3317
95	3	12	0.0740	0.2926

Table 10: Extended Technical Appendix: Full Blind Pooling Decoder Test MSE Results ($d_z = 8$).

H	W	N_{layers}	MSE_{bb}	MSE_{sc}
100	1	8	0.1393	0.4227
100	2	12	0.1022	0.3531
100	3	4	0.1640	0.4863
100	5	8	0.0744	0.2935
100	7	12	0.0840	0.3013
10	2	12	0.1704	0.5434
15	1	4	0.2284	0.6786
15	1	8	0.2323	0.7050
15	4	8	0.1288	0.4483
15	5	4	0.1173	0.4283
15	6	4	0.1252	0.4360
20	5	8	0.0968	0.3849
25	1	12	0.1878	0.5622
25	4	8	0.1044	0.3991
25	7	4	0.1005	0.3880
35	1	4	0.1677	0.5122
35	2	4	0.1065	0.3973
35	2	8	0.1064	0.3959
35	6	8	0.0874	0.3569
40	2	4	0.1008	0.3852
40	5	8	0.0806	0.3443
45	1	12	0.1610	0.4846
50	1	4	0.1598	0.4785
50	3	12	0.1274	0.4183
50	5	12	0.0767	0.3290
50	6	12	0.0842	0.3432
55	2	8	0.1003	0.3711
55	5	4	0.0729	0.3135
55	6	8	0.0879	0.3459
5	1	12	0.3740	1.0205
5	2	12	0.2496	0.7218
5	4	8	0.2107	0.6336
5	7	8	0.2631	0.7695
60	1	12	0.1494	0.4528
60	3	4	0.1375	0.4204
60	5	4	0.0735	0.3114
65	5	8	0.0724	0.3072
70	2	4	0.0924	0.3475
70	2	8	0.0952	0.3547
70	3	12	0.1671	0.4741
75	1	12	0.1461	0.4409
75	4	12	0.0831	0.3242
80	1	4	0.1422	0.4347
80	2	8	0.0909	0.3364
85	1	12	0.1485	0.4374
90	5	8	0.0747	0.2947
95	1	12	0.1404	0.4253
95	4	8	0.0891	0.3212
95	5	4	0.0730	0.2876
95	6	8	0.0796	0.3036

Table 11: Extended Technical Appendix: Full Blind Pooling Decoder Test MSE Results ($d_z = 16$).

H	W	N_{layers}	MSE_{bb}	MSE_{sc}
100	15	4	0.0858	0.3033
100	2	12	0.0933	0.3349
100	5	4	0.0792	0.2984
100	7	4	0.0734	0.2837
10	10	12	0.1243	0.4333
10	3	8	0.1725	0.5446
10	9	12	0.1329	0.4545
15	14	8	0.1205	0.4164
15	1	12	0.2228	0.6437
15	3	12	0.1506	0.4957
15	7	4	0.1129	0.4104
20	12	4	0.1012	0.3890
20	13	8	0.1190	0.4311
30	12	12	0.0879	0.3680
30	5	8	0.0846	0.3538
35	11	12	0.0934	0.3649
40	15	12	0.0912	0.3521
40	6	12	0.0746	0.3294
40	9	12	0.0783	0.3328
45	11	12	0.0835	0.3447
45	3	8	0.1040	0.3805
45	6	8	0.0759	0.3289
50	2	12	0.1102	0.3934
50	6	12	0.0718	0.3170
50	7	12	0.0759	0.3265
50	9	12	0.0751	0.3236
55	10	4	0.0808	0.3271
55	11	4	0.0786	0.3163
55	12	8	0.0791	0.3243
55	15	4	0.0795	0.3259
55	4	8	0.0890	0.3470
55	7	8	0.0728	0.3107
5	13	12	0.1901	0.5524
60	10	4	0.0734	0.3047
60	4	4	0.0833	0.3386
60	6	8	0.0699	0.3002
60	9	12	0.0803	0.3196
65	4	4	0.0827	0.3329
70	8	4	0.0846	0.3238
70	9	12	0.0817	0.3218
75	5	8	0.0714	0.2992
80	11	12	0.0790	0.3069
80	4	12	0.0815	0.3219
80	5	8	0.0717	0.2981
85	11	4	0.0797	0.2999
90	10	8	0.0724	0.2820
90	12	4	0.0758	0.2919
90	4	4	0.0780	0.3140
90	6	4	0.0690	0.2854
95	9	4	0.0802	0.2987

J.8 Diffusion Model Hyperparameter Scans: Blind Pooling

Extensive hyperparameter optimization was performed for the latent diffusion models across the different pooling strategies and denoiser architectures detailed above. For brevity, and to illustrate the typical scope of these optimizations, the detailed tables that follow (Tables 12-17) focus on presenting results for the **Blind Pooling** strategy. These tables showcase performance across varying encoder latent dimensions ($d_z \in \{4, 8, 16\}$), for both MLP (specifically DiffusionMLP_v2) and Conv2D denoiser architectures, against a grid of diffusion hyperparameters (total timesteps T , β_{start} , and β_{end}). While not all combinations are exhaustively tabulated for every pooling method in this appendix, similar systematic grid search approaches were applied to optimize the Sequential and Residue pooling diffusion models, tailoring parameter ranges and model configurations (e.g., MLP hidden dimensions, training epochs) according to their specific input characteristics and computational demands.

Table 12: Extended Technical Appendix: Full Diffusion Grid Results (blind pooling, $d_z = 4$, $H = 30$, $W = 2$, MLP Denoiser). "(Rec.)" denotes Decoder Reconstruction from ground-truth latents; "(Diff.)" denotes full Diffusion pipeline results.

Exp ID	IDDT _{All} (Rec.)	IDDT _{BB} (Rec.)	TM _{All} (Rec.)	TM _{BB} (Rec.)	IDDT _{All} (Diff.)	IDDT _{BB} (Diff.)	TM _{All} (Diff.)	TM _{BB} (Diff.)
1	0.716	0.792	0.961	0.957	0.651	0.699	0.949	0.936
2	0.717	0.793	0.962	0.957	0.611	0.660	0.938	0.923
3	0.716	0.792	0.961	0.957	0.615	0.661	0.942	0.926
4	0.717	0.793	0.961	0.957	0.621	0.667	0.945	0.932
5	0.714	0.792	0.961	0.957	0.610	0.655	0.938	0.920
6	0.716	0.793	0.961	0.957	0.629	0.679	0.949	0.937
7	0.717	0.792	0.961	0.957	0.531	0.574	0.909	0.880
8	0.716	0.793	0.961	0.957	0.376	0.392	0.773	0.710
9	0.715	0.794	0.961	0.957	0.543	0.583	0.910	0.880
10	0.715	0.793	0.961	0.957	0.495	0.528	0.880	0.842
11	0.717	0.794	0.961	0.957	0.573	0.608	0.922	0.899
12	0.717	0.793	0.961	0.957	0.571	0.607	0.927	0.905
13	0.716	0.793	0.961	0.957	0.594	0.641	0.939	0.922
14	0.714	0.793	0.961	0.957	0.605	0.645	0.936	0.918
15	0.716	0.792	0.961	0.957	0.515	0.542	0.888	0.867

Table 13: Extended Technical Appendix: Full Diffusion Grid Results (blind pooling, $d_z = 4$, $H = 30$, $W = 2$, Conv2D Denoiser). "(Rec.)" denotes Decoder Reconstruction from ground-truth latents; "(Diff.)" denotes full Diffusion pipeline results.

Exp ID	IDDT _{All} (Rec.)	IDDT _{BB} (Rec.)	TM _{All} (Rec.)	TM _{BB} (Rec.)	IDDT _{All} (Diff.)	IDDT _{BB} (Diff.)	TM _{All} (Diff.)	TM _{BB} (Diff.)
1	0.716	0.794	0.961	0.957	0.240	0.244	0.375	0.287
2	0.717	0.792	0.961	0.957	0.239	0.246	0.376	0.286
3	0.714	0.792	0.961	0.957	0.239	0.247	0.375	0.287
4	0.716	0.793	0.961	0.957	0.237	0.244	0.377	0.283
5	0.717	0.793	0.961	0.957	0.237	0.243	0.371	0.282
6	0.717	0.794	0.961	0.957	0.240	0.245	0.375	0.288
7	0.716	0.795	0.961	0.957	0.239	0.244	0.376	0.289
8	0.716	0.793	0.961	0.957	0.236	0.243	0.375	0.284
9	0.715	0.792	0.961	0.957	0.237	0.246	0.379	0.285
10	0.716	0.792	0.961	0.957	0.235	0.243	0.375	0.284
11	0.715	0.794	0.961	0.957	0.236	0.243	0.378	0.287
12	0.716	0.795	0.961	0.957	0.236	0.243	0.379	0.288
13	0.716	0.793	0.961	0.957	0.239	0.245	0.372	0.283
14	0.715	0.794	0.961	0.957	0.235	0.243	0.373	0.282
15	0.718	0.793	0.961	0.957	0.239	0.245	0.377	0.286

Table 14: Extended Technical Appendix: Full Diffusion Grid Results (blind pooling, $d_z = 8$, $H = 35$, $W = 2$, MLP Denoiser). "(Rec.)" denotes Decoder Reconstruction from ground-truth latents; "(Diff.)" denotes full Diffusion pipeline results.

Exp ID	IDDT _{All} (Rec.)	IDDT _{BB} (Rec.)	TM _{All} (Rec.)	TM _{BB} (Rec.)	IDDT _{All} (Diff.)	IDDT _{BB} (Diff.)	TM _{All} (Diff.)	TM _{BB} (Diff.)
1	0.714	0.793	0.961	0.957	0.671	0.728	0.957	0.947
2	0.714	0.792	0.961	0.957	0.634	0.678	0.951	0.935
3	0.715	0.791	0.961	0.957	0.601	0.648	0.944	0.930
4	0.715	0.792	0.961	0.958	0.663	0.722	0.955	0.945
5	0.715	0.793	0.961	0.957	0.631	0.680	0.951	0.938
6	0.714	0.792	0.961	0.957	0.677	0.733	0.960	0.950
7	0.713	0.793	0.961	0.958	0.669	0.722	0.959	0.949
8	0.715	0.794	0.961	0.957	0.638	0.689	0.953	0.939
9	0.714	0.792	0.961	0.957	0.567	0.603	0.924	0.901
10	0.714	0.792	0.961	0.957	0.613	0.663	0.945	0.931
11	0.715	0.793	0.961	0.957	0.631	0.682	0.951	0.939
12	0.715	0.793	0.961	0.957	0.628	0.676	0.954	0.941
13	0.715	0.793	0.961	0.957	0.652	0.709	0.954	0.943
14	0.713	0.793	0.961	0.957	0.567	0.605	0.937	0.917
15	0.712	0.793	0.961	0.957	0.425	0.446	0.865	0.817

Table 15: Extended Technical Appendix: Full Diffusion Grid Results (blind pooling, $d_z = 8$, $H = 35$, $W = 2$, Conv2D Denoiser). "(Rec.)" denotes Decoder Reconstruction from ground-truth latents; "(Diff.)" denotes full Diffusion pipeline results.

Exp ID	IDDT _{All} (Rec.)	IDDT _{BB} (Rec.)	TM _{All} (Rec.)	TM _{BB} (Rec.)	IDDT _{All} (Diff.)	IDDT _{BB} (Diff.)	TM _{All} (Diff.)	TM _{BB} (Diff.)
1	0.713	0.792	0.961	0.957	0.243	0.252	0.376	0.285
2	0.715	0.792	0.961	0.957	0.240	0.248	0.373	0.283
3	0.715	0.792	0.961	0.958	0.241	0.248	0.373	0.281
4	0.715	0.794	0.961	0.958	0.241	0.248	0.372	0.286
5	0.713	0.790	0.961	0.958	0.242	0.250	0.375	0.287
6	0.715	0.793	0.961	0.957	0.245	0.251	0.378	0.294
7	0.714	0.792	0.961	0.957	0.239	0.248	0.366	0.276
8	0.714	0.794	0.961	0.957	0.239	0.245	0.366	0.276
9	0.715	0.792	0.961	0.957	0.239	0.247	0.367	0.282
10	0.715	0.793	0.961	0.957	0.236	0.245	0.356	0.268
11	0.714	0.793	0.961	0.957	0.240	0.248	0.372	0.282
12	0.715	0.793	0.961	0.957	0.240	0.250	0.373	0.284
13	0.714	0.794	0.961	0.957	0.239	0.249	0.371	0.279
14	0.714	0.792	0.961	0.957	0.240	0.247	0.366	0.278
15	0.714	0.794	0.961	0.958	0.242	0.251	0.368	0.283

Table 16: Extended Technical Appendix: Full Diffusion Grid Results (blind pooling, $d_z = 16$, $H = 50$, $W = 2$, MLP Denoiser). "(Rec.)" denotes Decoder Reconstruction from ground-truth latents; "(Diff.)" denotes full Diffusion pipeline results.

Exp ID	IDDT _{All} (Rec.)	IDDT _{BB} (Rec.)	TM _{All} (Rec.)	TM _{BB} (Rec.)	IDDT _{All} (Diff.)	IDDT _{BB} (Diff.)	TM _{All} (Diff.)	TM _{BB} (Diff.)
1	0.714	0.792	0.961	0.957	0.709	0.780	0.962	0.957
2	0.715	0.790	0.961	0.957	0.707	0.781	0.962	0.958
3	0.716	0.791	0.961	0.956	0.706	0.779	0.961	0.954
4	0.714	0.792	0.961	0.957	0.719	0.792	0.964	0.960
5	0.713	0.791	0.961	0.957	0.703	0.775	0.962	0.957
6	0.717	0.791	0.961	0.957	0.706	0.780	0.961	0.955
7	0.715	0.791	0.961	0.957	0.702	0.769	0.962	0.956
8	0.716	0.791	0.961	0.957	0.697	0.758	0.960	0.952
9	0.717	0.791	0.961	0.957	0.698	0.763	0.964	0.958
10	0.715	0.791	0.961	0.957	0.688	0.755	0.960	0.952
11	0.715	0.792	0.961	0.957	0.689	0.752	0.959	0.950
12	0.714	0.792	0.961	0.956	0.704	0.772	0.960	0.953
13	0.715	0.794	0.961	0.957	0.699	0.766	0.962	0.954
14	0.715	0.793	0.961	0.956	0.690	0.750	0.960	0.953
15	0.715	0.791	0.961	0.957	0.677	0.740	0.957	0.948

Table 17: Extended Technical Appendix: Full Diffusion Grid Results (blind pooling, $d_z = 16$, $H = 50$, $W = 2$, Conv2D Denoiser). "(Rec.)" denotes Decoder Reconstruction from ground-truth latents; "(Diff.)" denotes full Diffusion pipeline results.

Exp ID	IDDT _{All} (Rec.)	IDDT _{BB} (Rec.)	TM _{All} (Rec.)	TM _{BB} (Rec.)	IDDT _{All} (Diff.)	IDDT _{BB} (Diff.)	TM _{All} (Diff.)	TM _{BB} (Diff.)
1	0.714	0.791	0.961	0.957	0.261	0.267	0.397	0.305
2	0.715	0.790	0.961	0.957	0.263	0.270	0.399	0.308
3	0.714	0.791	0.961	0.957	0.263	0.270	0.404	0.312
4	0.715	0.792	0.961	0.957	0.264	0.270	0.404	0.309
5	0.717	0.792	0.961	0.957	0.261	0.266	0.401	0.308
6	0.713	0.792	0.961	0.957	0.262	0.270	0.399	0.306
7	0.716	0.793	0.961	0.957	0.262	0.269	0.397	0.309
8	0.715	0.791	0.961	0.957	0.264	0.269	0.405	0.314
9	0.716	0.792	0.961	0.957	0.266	0.272	0.406	0.314
10	0.715	0.791	0.961	0.957	0.261	0.269	0.402	0.307
11	0.714	0.792	0.961	0.957	0.265	0.270	0.406	0.311
12	0.715	0.793	0.961	0.957	0.262	0.271	0.404	0.312
13	0.716	0.792	0.961	0.957	0.266	0.274	0.403	0.312
14	0.713	0.792	0.961	0.957	0.260	0.268	0.395	0.307
15	0.716	0.792	0.961	0.957	0.263	0.270	0.406	0.306

J.9 Conditioning Mechanism Tuning

As detailed in Section 3.2 (Conditioning Mechanism paragraph), the decoder component of LD-FPG is conditioned on information derived from a reference structure, X_{ref} . Our primary conditioning strategy utilizes the latent representation of this reference, $C = Z_{\text{ref}} = \Theta^*(X_{\text{ref}}, E_{\text{ref}})$, obtained from the pre-trained and frozen encoder Θ^* . This choice was motivated by preliminary findings suggesting its superiority over direct conditioning on raw Cartesian coordinates, $C = X_{\text{ref}}$.

To rigorously validate this and quantify the performance difference, we conducted comprehensive ablation studies. These studies compared models conditioned on Z_{ref} (indicated by the `_zref` suffix in their experimental set identifiers) against baseline models conditioned on X_{ref} . The comparisons spanned different architectural configurations of the decoder, including:

- **Pooling Strategy:** Experiments were run using both Blind Pooling and Residue Pooling decoders, as described in Section 3.3. The Residue Pooling strategy itself had variants explored: a standard Residue (1level) application and a hierarchical Residue (2level)

approach. In the Residue (2level) configuration, the processing involved a two-stage pooling scheme: atom-wise latent embeddings $Z^{(t)}$ first underwent a global blind pooling operation. The output context vector from this initial global pooling was then subsequently used as an input or integrated within a residue-specific pooling or processing mechanism before the final MLP prediction for each residue’s atoms.

- **Cross-Attention (CA):** For some configurations, a Cross-Attention mechanism (denoted `_CA_on`) was employed. This module, defined as:

```
class CrossAttentionBlock(nn.Module):
    def __init__(self, q_dim, kv_dim, att_dim=64):
        super().__init__(); self.s = att_dim**-0.5
        self.q=nn.Linear(q_dim,att_dim,bias=False)
        self.k=nn.Linear(kv_dim,att_dim,bias=False)
        self.v=nn.Linear(kv_dim,att_dim,bias=False)
    def forward(self, q, k, v):
        Q=self.q(q); K=self.k(k); V=self.v(v)
        sc = torch.matmul(Q, K.transpose(-1, -2)) * self.s
        return torch.matmul(F.softmax(sc, dim=-1), V)
```

allows the pooled latent embeddings derived from the input dynamic conformation (\mathbf{h}_0) to interact with and selectively integrate information from the reference structure representation (e.g., Z_{ref}). Configurations without this mechanism (`_CA_off`) typically use simpler concatenation or direct feeding of \mathbf{h}_0 and the conditioner to the subsequent MLP.

It is important to note that these ablation studies comparing conditioning mechanisms were conducted with other architectural and training hyperparameters (beyond those swept for $H, W, H2, W2$ within each group, such as encoder specifics or base learning rate schedules) fixed to a representative set; this foundational parameter set is not discussed in detail here.

The optimizer was generally Adam, except for one set of experiments explicitly testing AdamW (e.g., `blind_res_CA_off_zref_lr.Adamw`). The primary metric for comparison was the minimum test loss achieved across an extensive hyperparameter search (varying decoder-specific pooling dimensions H, W and MLP architecture parameters $H2, W2$) within each experimental group.

Table 18: Ablation Study Results for Decoder Conditioning Mechanism. Performance is compared based on minimum test loss. ‘ X_{ref} Cond.’ refers to conditioning on raw Cartesian coordinates, while ‘ Z_{ref} Cond.’ refers to conditioning on latent reference embeddings. “1level” and “2level” refer to specific variants of the Residue Pooling architecture, with “2level” involving an initial global blind pooling stage prior to residue-specific processing.

Pooling	Cross-Attention	X_{ref} Cond.	Z_{ref} Cond.	Improvement (%)
Blind	Off	0.7763 (15)	0.5327 (32)	31.4
Blind	On	1.6787 (16)	0.5835 (24)	65.2
Residue	On	1.2417 (6)	0.2453 (14)	80.2
Residue (1level)	Off	0.2290 (19)	0.1058 (10)	53.8
Residue (2level)	Off	0.7599 (20)	0.5749 (49)	24.3

The results presented in Table 18 consistently show that conditioning on the latent reference embeddings (Z_{ref}) leads to substantially lower minimum test losses compared to conditioning on raw Cartesian coordinates (X_{ref}). This performance advantage holds across both Blind and Residue pooling strategies (including its “1level” and “2level” variants) and is evident whether Cross-Attention is used or not. For instance, with Blind Pooling and CA off, Z_{ref} conditioning achieved a $\sim 31.4\%$ lower loss. This effect was even more pronounced for Residue Pooling (CA On), where an $\sim 80.2\%$ improvement was observed. Even for the more complex Residue (2level) architecture, Z_{ref} conditioning provided a notable $\sim 24.3\%$ reduction in loss.

The single experimental group `blind_res_CA_off_zref_lr.Adamw` (16 runs, Min Test Loss 0.7592), which used Z_{ref} conditioning with the AdamW optimizer, yielded a higher loss than its Adam-optimized counterpart (`blind_res_CA_off_zref`, Min Test Loss 0.5327). While this suggests that Adam may have been more suitable or better tuned for this particular setup, it does not

detract from the central finding regarding the conditioning variable itself, as no corresponding X_{ref} -conditioned AdamW experiments were available for direct comparison.

In conclusion, these ablation studies provide robust quantitative evidence supporting the design choice of using the encoder-derived latent representation Z_{ref} as the primary conditioning mechanism for the decoder in LD-FPG. This strategy enables the model to more effectively learn and represent conformational deformations relative to a well-characterized reference state, leading to improved generative performance.

J.10 Discussion on Metric Variability and Standard Deviations

Throughout this paper, our evaluation process involves training each model component (encoder, decoder, diffusion model) and then assessing its performance by generating an ensemble of protein conformations. These generated ensembles are then compared against the reference Molecular Dynamics (MD) ensemble, with key metrics such as IDDT, TM-score, and Jensen-Shannon Divergence (JSD) for dihedral angles reported. For clarity and conciseness, the main summary tables (e.g., Table 1 and Table 2) present the mean values for per-structure metrics like IDDT and TM-score. This section provides further context on the variability observed in these metrics, typically characterized by their standard deviations (SD).

While standard deviations for metrics like IDDT and TM-score offer insight into the spread of structural quality within an ensemble, we argue that the $\sum \text{JSD}$ values for backbone and side-chain dihedral angles (reported directly in Tables 1 and 2) serve as a more direct and potent indicator of how well the *distributional variability* and local conformational diversity of the reference MD ensemble are captured. A low $\sum \text{JSD}$ signifies that the variety and relative populations of local backbone (ϕ, ψ) and side-chain (χ) conformations in the generated ensemble closely mirror those of the MD data. Furthermore, qualitative assessments of landscape coverage, such as the Principal Component Analysis (PCA) of latent embeddings (Figure 4, top row) and the A100 activation index distributions (Figure 4, bottom row), provide richer, more nuanced views of overall conformational diversity and the exploration of functionally relevant states than a single SD value of a global structural similarity score might offer.

Nevertheless, to provide a comprehensive picture, we report the observed standard deviations for IDDT_{All} and $\text{TM-score}_{\text{All}}$ across different stages of our pipeline, calculated over the entire test set ensemble for each respective model configuration:

- **IDDT_{All} Standard Deviations:**

- Ground Truth MD Ensemble (vs. X_{ref}): $\text{SD} \approx 0.028$
- ChebNet Encoder Reconstruction Head (Table 4): $\text{SD} \approx 0.030$
- Decoder-Generated Ensembles (Table 1, from ground-truth latents):
 - * Blind Pooling ($d_z = 16$): $\text{SD} \approx 0.031$
 - * Sequential Pooling ($d_z = 8$): $\text{SD} \approx 0.032$
 - * Residue Pooling ($d_z = 4$): $\text{SD} \approx 0.040$
- Diffusion-Generated Ensembles (Table 2):
 - * Blind Pooling: $\text{SD} \approx 0.042$
 - * Sequential Pooling: $\text{SD} \approx 0.058$
 - * Residue Pooling: $\text{SD} \approx 0.091$

- **TM-score_{All} Standard Deviations:**

- Ground Truth MD Ensemble (vs. X_{ref}): $\text{SD} \approx 0.0059$
- ChebNet Encoder Reconstruction Head (Table 4): $\text{SD} \approx 0.0060$
- Decoder-Generated Ensembles (Table 1):
 - * Blind Pooling ($d_z = 16$): $\text{SD} \approx 0.0062$
 - * Sequential Pooling ($d_z = 8$): $\text{SD} \approx 0.0065$
 - * Residue Pooling ($d_z = 4$): $\text{SD} \approx 0.020$
- Diffusion-Generated Ensembles (Table 2):
 - * Blind Pooling: $\text{SD} \approx 0.015$
 - * Sequential Pooling: $\text{SD} \approx 0.025$

* Residue Pooling: $SD \approx 0.048$

These standard deviations were computed by evaluating the respective metric for each structure in the generated test ensemble against the reference structure (X_{ref}) and then calculating the standard deviation of these scores. The Python script provided (calc_iddt.py) illustrates the methodology for calculating such mean and standard deviation values for IDDT scores over an ensemble, including options for subsampling if required for very large datasets.

Interpreting these SD values: The SDs for IDDT and TM-score for the ChebNet encoder and the decoder-generated ensembles (from ground-truth latents) are generally low and quite comparable to the inherent variability observed within the ground truth MD ensemble itself when compared against the static X_{ref} . This indicates that the encoder and decoders maintain a consistent level of structural fidelity. For the diffusion-generated ensembles, the SD values, particularly for IDDT, show a modest increase. For instance, the SD for IDDT_{All} ranges from approximately 0.04 to 0.09. This suggests a slightly greater degree of structural heterogeneity in the ensembles produced by the full generative pipeline compared to the direct decoder outputs or the original MD data’s deviation from X_{ref} . This increased spread can be acceptable, or even indicative of the model exploring the learned conformational manifold, as long as the mean fidelity remains high (as reported in Table 2) and, crucially, the critical distributional features of the ensemble (captured by $\sum \text{JSD}$ for dihedrals and A100 landscape analysis) are well-reproduced. The very low SDs for TM-score across all stages consistently affirm that the global fold of the protein is well-preserved.

In summary, while mean values are prioritized in the main tables for direct comparison of central tendencies, the analysis of $\sum \text{JSD}$ for dihedral distributions, PCA of latent spaces, A100 activation profiles, and the contextualized standard deviations discussed here collectively provide a comprehensive assessment of both the fidelity and the diversity of the conformational ensembles generated by LD-FPG.

References for Extended Technical Appendix

- [90] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43, 2020.
- [91] <https://www.nvidia.com/en-us/data-center/140s/>.
- [92] <https://www.hyperstack.cloud/technical-resources/performance-benchmarks/comparing-nvidia-h100-pcie-vs-sxm-performance-use-cases-and-more>.
- [93] <https://journal.uptimeinstitute.com/large-data-centers-are-mostly-more-efficient-analysis-confirms/>.
- [94] <https://ember-energy.org/chapter/electricity-transition-in-2022-ger2023/>.
- [95] <https://aliunid.com/wp-content/uploads/2023/04/2023-media-information-aliunid-CO2-content-in-Swiss-electricity.pdf>.