GUIDEDOCK: GEOMETRY-GUIDED DIFFUSION FOR RIGID PROTEIN-PROTEIN DOCKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modeling 3D protein-protein interactions is central to structural biology and the design of drugs and therapeutics. Recent deep learning methods for rigid binder pose prediction can be more efficient than classical docking software and expensive co-folding models. However, these methods typically operate directly in $\mathrm{SE}(3)$ pose space with limited explicit priors over interface geometry. As a result, they must jointly learn which binder-receptor regions are compatible and which rigid transformations realize that compatibility, often leading to unreliable sampling and suboptimal accuracy. In this work, we introduce GUIDEDOCK, a framework that guides diffusion-based pose generation with distance maps recovered by latent flow matching. GUIDEDOCK treats recovered distance maps as soft geometric constraints that steer the diffusion process toward physically plausible rigid poses. Across DIPS-Het and Docking Benchmark 5.5, GUIDEDOCK outperforms competing deep learning-based rigid docking baselines, improving mean DockQ by 30% and reducing mean complex RMSD by up to 15% with only a single generated pose, while running up to 300 times faster than the best co-folding models.

1 INTRODUCTION

Proteins are fundamental to all life, facilitating nearly all biological processes. In particular, interactions between proteins play a crucial role in applications such as immune recognition & response (Jardetzky, 1997) and enzyme regulation (Frieden, 1971). In these interactions, a binding partner engages a receptor at specific 3D interfaces to trigger a response. Predicting how a binder docks with a target receptor is therefore highly important for designing new proteins and therapeutics.

Template- and score-based docking models (Schneidman-Duhovny et al., 2005a; Yan et al., 2017a; Schindler et al., 2017) can achieve high accuracy for protein-protein docking. More recently, deep learning methods in the AlphaFold and Boltz families (Evans et al., 2021; Passaro et al., 2025; Wohlwend et al., 2024) have become a competitive research direction by training neural networks on large structural datasets derived from the Protein Data Bank (PDB) (Wen et al., 2019; Berman et al., 2000). These co-folding models predict binder-receptor complex coordinates from scratch and can be highly accurate, but they solve a broad complex-prediction problem with substantial inference cost. They also depend on features obtained by mining terabyte-scale protein databases for multiple sequence alignments (MSAs) (Needleman & Wunsch, 1970; Feng & Doolittle, 1987), which adds overhead when evaluating large libraries of candidate pairs.

This work targets the complementary *rigid* docking regime, where candidate binder and receptor conformations are provided and the task is to infer their relative pose. This regime is well-established in the protein-docking community, serving as a foundational stage for high-throughput virtual sampling, ranking, and refinement (Janin, 2010; Vakser, 2014) where running a co-folding model for every pair can be computationally prohibitive. We do not aim to replace co-folding models in unconstrained complex prediction; instead, we focus on MSA-free rigid pose inference, where throughput and sample efficiency are central requirements. Several MSA-free deep learning approaches (Ganea et al., 2022; Ketata et al., 2023; Yu et al., 2024; Wu et al., 2024) have been proposed for this regime. However, they largely operate in raw six-dimensional pose space, with three rotational degrees of freedom for $\boldsymbol{R}$ (pitch, yaw, and roll) and three translational degrees of freedom for $\boldsymbol{t}$ (x, y, and z). The model must therefore jointly learn which binder and receptor regions are geometrically compatible and which $(\boldsymbol{R}, \boldsymbol{t})$ realizes that compatibility, which is difficult to solve directly, revealing a need for more explicit geometric priors to guide pose estimation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this work, we move beyond direct pose-space modeling by using inter-residue geometry to guide pose estimation. Rather than predicting $(\boldsymbol{R}, \boldsymbol{t})$ directly, we first learn a generative prior over pairwise distance maps between the binder and receptor. Distance maps capture continuous interface geometry while remaining invariant to global rotations and translations, providing a mid-level representation that is richer than discrete contact patterns and more structured than full coordinates. By generating plausible distance maps, we constrain the set of viable poses to refine pose optimization. We instantiate this idea in GUIDEDOCK, a novel framework that pairs a latent flow matching model over distance maps with a rigid pose diffusion model. The latent flow matching module learns to predict distances between binder and receptor residues, yielding soft constraints on the pose space that in turn steer the pose diffusion model towards more physically plausible poses, enhancing docking performance. In summary, our key contributions are the following:

- We introduce GUIDEDOCK, a geometry-guided diffusion framework that uses predicted distance maps as soft constraints for rigid pose generation.
- We show that GUIDEDOCK improves the accuracy-throughput tradeoff over MSA-free rigid docking baselines on DIPS-Het and Docking Benchmark 5.5, using only a single sampled pose while improving mean DockQ by 30% and mean complex RMSD by up to 15%.

2 PRELIMINARIES AND RELATED WORK

Notation. All proteins are composed of chains of amino acids known as *residues*. For binder and receptor proteins $B = (B_i)_{i=1}^{M}$ and $R = (R_i)_{i=1}^{N}$, we represent the 3D Cartesian coordinates of each residue by their α-carbons, denoted by $\boldsymbol{X}_B \in \mathbb{R}^{M\times 3}$ and $\boldsymbol{X}_R \in \mathbb{R}^{N\times 3}$. The α-carbons naturally define the overall structure of a protein, sufficient to identify sheets, helices, and other structural features (Webber et al., 2001). This notation naturally extends to include side-chain atoms and other backbone heavy atoms if desired. For a given complex, we define the inter-residue distance matrix between B and R as $\boldsymbol{D} \in \mathbb{R}_{+}^{M\times N}$, where $\boldsymbol{D}_{ij} = \|\boldsymbol{X}_{B_i} - \boldsymbol{X}_{R_j}\|_2$ is the Euclidean distance between residues B_i and R_j. For convenience, we may use $\boldsymbol{b}_i$ and $\boldsymbol{r}_i$ in place of $\boldsymbol{X}_{B_i}$ and $\boldsymbol{X}_{R_i}$, respectively.

Problem Definition. Our task is rigid protein-protein docking: given fixed binder and receptor conformations, we seek the binder placement that recovers the native complex. This formulation isolates the pose-inference component of docking by considering rigid-body rotations and translations. Concretely, given a binder with coordinates $\tilde{\boldsymbol{X}}_B$, we seek the rotation $\boldsymbol{R} \in \mathrm{SO}(3)$, the 3D rotation group, and translation $\boldsymbol{t} \in \mathbb{R}^3$ such that the transformed binder coordinates $\boldsymbol{R}\tilde{\boldsymbol{X}}_B + \boldsymbol{t}$ are correctly positioned relative to the receptor. We represent each candidate docking solution, or pose, as a pair $(\boldsymbol{R}, \boldsymbol{t}) \in \mathrm{SE}(3)$, the group of all 3D rigid-body transformations.

Related Work. Existing protein-protein docking models can be broadly categorized into classical score-based software and deep learning methods. Score-based methods such as PatchDock (Schneidman-Duhovny et al., 2005b) and HDock (Yan et al., 2017b) search over protein templates and candidate poses, ranking them with empirical score functions (Moal et al., 2013). These searches can be computationally expensive, motivating more efficient approaches. Co-folding models such as AlphaFold-Multimer (Evans et al., 2021) predict complete complex coordinates from sequence-derived features and are most appropriate when accuracy is prioritized over throughput. However, they solve a different and more expensive problem than rigid pose inference: they reconstruct the full complex rather than estimating the relative pose of available binding partners, and they rely on MSA features (Johansson-Åkhe & Wallner, 2022) that require substantial compute and storage. Rigid docking methods instead target the setting in which structures are assumed to not change upon docking, forming the basis of high-throughput sampling and refinement (Vakser, 2014). Regression-based models such as EquiDock (Ganea et al., 2022) and ElilDock (Yu et al., 2024) directly predict binder poses, whereas DiffDock-PP (Ketata et al., 2023), EBMDock (Wu et al., 2024), and DockGPT (McPartlon & Xu, 2023) use generative models to sample possible poses.

3 BACKGROUND

Rigid Docking as Pose Inference. We formalize the task of rigid docking as finding the highest probability rotation $\boldsymbol{R}^*$ and translation $\boldsymbol{t}^*$ given a protein complex C. Methods such as EquiDock and ElliDock solve this problem by training an $\mathrm{SE}(3)$-equivariant regression model on ground-truth $(\boldsymbol{R}, \boldsymbol{t})$ poses, matching keypoints and interfaces to predict the correct rotation and translation.

However, regression-based methods fail to capture the full landscape of potential poses. To remedy this, DiffDock-PP and EBMDock take a generative modeling approach, learning instead to sample high-probability $(\boldsymbol{R}, \boldsymbol{t})$ pairs. In this work, we focus on the diffusion approach to modeling poses. In the diffusion setting, the binder protein is randomly translated and rotated about the receptor, and a neural network is trained to predict the added noise. Formally, the forward diffusion process is described by the following stochastic differential equation (SDE) (Song et al., 2021):

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{w} \tag{1}$$

where $\boldsymbol{f}$ is a drift term, g is a scalar diffusion coefficient, and $\boldsymbol{w}_t$ is standard Brownian motion. The forward process gradually corrupts samples from the data distribution, defining a family of distributions $p_t(\boldsymbol{x}_t)$ over time that the reverse-time process aims to invert. Training a diffusion model then amounts to learning the reverse-time SDE

$$\mathrm{d}\boldsymbol{x} = [\boldsymbol{f}(\boldsymbol{x}_t, t) - g(t)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)]\mathrm{d}t + g(t)\mathrm{d}\bar{\boldsymbol{w}} \tag{2}$$

by fitting a neural network $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$ to the score function $\nabla_{\boldsymbol{x}_t} p(\boldsymbol{x}_t)$ (Song & Ermon, 2019; Song et al., 2021; Hyvärinen, 2005), where $\mathrm{d}\bar{\boldsymbol{w}}$ is a standard Wiener process in reverse time. Sampling is performed by initializing a random $(\boldsymbol{R}_T, \boldsymbol{t}_T)$ and then numerically integrating the reverse-time diffusion process. More details on diffusion models and how they are defined for rigid-body motions are in Section B and appendix C.

Generative models capture the multi-modal nature of docking better than regression-based methods, but they also introduce a selection problem. Rather than producing a single estimate of $(\boldsymbol{R}^*, \boldsymbol{t}^*)$, one must (1) draw plausible samples from $p(\boldsymbol{R}, \boldsymbol{t} \mid C)$ and (2) identify the best pose among them. Because these models are stochastic and only weakly constrained by geometry, they require several samples before producing an adequate pose.

Distance Maps for Rigid Docking. GUIDEDOCK seeks to attain the best of both worlds between generative models and geometric constraints. That is, rather than operating directly over rigid-body transformations with minimal structural constraints, we guide pose generation with pairwise distance maps that encode interface geometry. These maps act as a geometric prior by favoring diffusion trajectories whose transformations are consistent with plausible binder-receptor distances. This reduces the portion of pose space explored by the stochastic sampler and improves both sample efficiency and per-sample reliability.

In an ideal setting, an accurate distance map completely determines the binder's position relative to the receptor up to a rigid-body transformation (Vassura et al., 2008; Pietal et al., 2015). This motivates a natural but brittle baseline: *decode a pose directly from a predicted distance map*. This is straightforwardly achieved using multidimensional scaling (MDS) algorithms (Mead, 1992; Torgerson, 1952). However, direct recovery is brittle because small errors in the predicted distance map can lead to large errors in the recovered pose, and there is no guarantee that a neural network's predicted distance map $\hat{\boldsymbol{D}}$ will be a valid Euclidean distance matrix. See Appendix A for an explanation of how distance maps can be used to recover poses and an example of how invalid distance maps can lead to poor pose recovery.

4 METHOD

These shortcomings motivate the use of inter-partner distance maps in GUIDEDOCK as soft guidance rather than brittle hard constraints, as in the MDS case. The goal is to bias sampling toward plausible regions of $\mathrm{SE}(3)$ while retaining the flexibility of diffusion-based pose generation. Concretely, GUIDEDOCK decomposes rigid docking into two coupled generative processes: (1) a latent flow matching prior over binder-receptor distance maps and (2) a diffusion model over pose space whose score field is guided by the predicted distance maps through a differentiable potential. This guidance biases sampling toward poses whose induced inter-residue distances agree with the generated interface geometry, without requiring hard constraint satisfaction.

4.1 GUIDING DIFFUSION WITH DISTANCE MAPS

Given a distance map $\boldsymbol{D}$ and a sampled rigid-body pose $(\boldsymbol{R}, \boldsymbol{t})$, we define a potential energy function $U(\boldsymbol{R}, \boldsymbol{t}; \boldsymbol{D})$ that measures how well $(\boldsymbol{R}, \boldsymbol{t})$ aligns binder and receptor residues $\boldsymbol{b}_i$ and $\boldsymbol{r}_j$ to match $\boldsymbol{D}$. For simplicity, let $\boldsymbol{T} = (\boldsymbol{R}, \boldsymbol{t})$, and let $\boldsymbol{\mathcal{D}}(\boldsymbol{T}) \in \mathbb{R}^{M \times N}$ with entries $\boldsymbol{\mathcal{D}}_{ij}(\boldsymbol{T}) = \|\boldsymbol{R}\boldsymbol{b}_i + \boldsymbol{t} - \boldsymbol{r}_j\|$ be

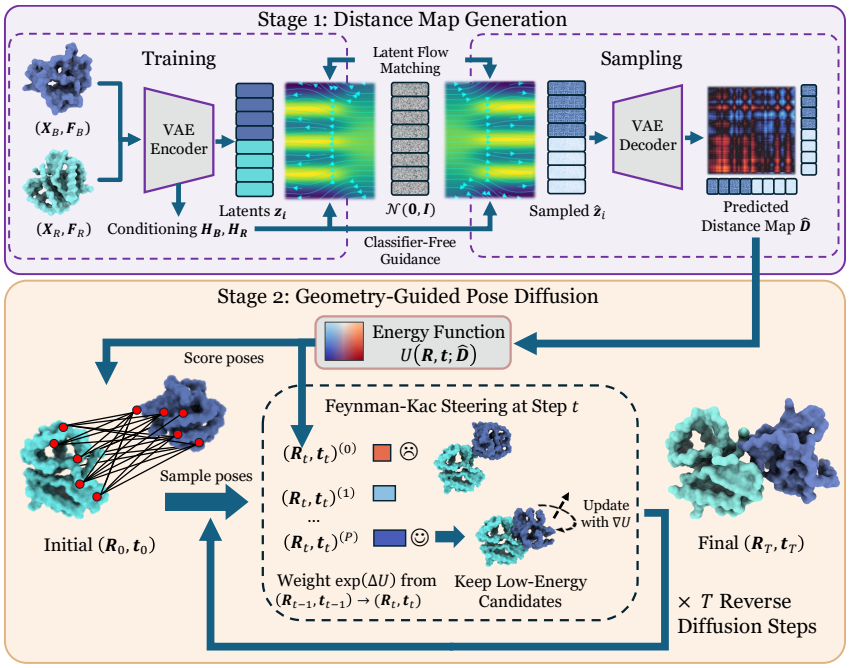


Figure 1: Pipeline for GUIDEDOCK. Binder and receptor structures are input into a latent flow matching model that generates a distance map $\boldsymbol{D}$ describing the geometry of the binding interface. This distance map is then used to guide a diffusion model over pose space via a differentiable potential $U(\boldsymbol{R}, \boldsymbol{t}; \boldsymbol{D})$ that encourages sampling of poses whose induced distance maps $\boldsymbol{\mathcal{D}}(\boldsymbol{R}, \boldsymbol{t})$ agree with $\boldsymbol{D}$.

the distance map induced by $\boldsymbol{T}$. Our potential is defined as

$$U(\boldsymbol{T}; \boldsymbol{D}) = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\boldsymbol{\mathcal{D}}_{ij}(\boldsymbol{T}) - \boldsymbol{D}_{ij}\right)^2 . \tag{3}$$

From a probabilistic perspective, this potential defines a Boltzmann likelihood over distance maps:

$$p(\boldsymbol{D} \mid \boldsymbol{T}) \propto \exp\left(-U(\boldsymbol{T}; \boldsymbol{D})\right). \tag{4}$$

Under this model, distance maps consistent with $\boldsymbol{T}$ are high probability, while inconsistent distance maps have low probability. We now combine this likelihood with the pose prior $p_\theta(\boldsymbol{T} \mid C)$ from our pose diffusion model where C stands for protein complex information (e.g., sequence and internal structural information). Following Bayes' rule, this leads to the posterior distribution over poses:

$$p(\boldsymbol{T} \mid C, \boldsymbol{D}) \propto p_\theta(\boldsymbol{T} \mid C)\, p(\boldsymbol{D} \mid \boldsymbol{T}) \tag{5}$$

$$\propto p_\theta(\boldsymbol{T} \mid C) \exp(-U(\boldsymbol{T}; \boldsymbol{D})), \tag{6}$$

which up-weights poses whose induced distance maps agree with $\boldsymbol{D}$. Directly sampling from this tilted distribution is intractable, so we approximate it with Feynman-Kac (FK) steering (Singhal et al., 2025). FK steering is a sequential Monte Carlo (SMC) procedure that samples multiple candidate poses, called *particles*, and resamples them according to a potential function. Specifically, we tilt each transition kernel of the reverse diffusion process $p_\theta\left(\boldsymbol{T}_t \mid \boldsymbol{T}_{t+1}, C\right)$ to bias the trajectory toward low-energy poses. For every particle $p \in \{1, \ldots, P\}$, we define an unnormalized tilted transition density

$$p_\theta\left(\boldsymbol{T}_t^{(p)} \mid \boldsymbol{T}_{t+1}^{(p)}, C\right) G_t^{(p)} \tag{7}$$

where $G_t^{(p)}$ is a time-dependent weight based on the induced decrease in energy from $\boldsymbol{T}_{t+1}^{(p)}$ to $\boldsymbol{T}_t^{(p)}$ defined by

$$G_t^{(p)} = \exp\left(U\left(\boldsymbol{T}_{t+1}^{(p)}; \boldsymbol{D}\right) - U\left(\boldsymbol{T}_t^{(p)}; \boldsymbol{D}\right)\right). \tag{8}$$

We then resample P particle indices with replacement from a multinomial distribution according to their normalized energy weights $w_t^{(p)} = \mathrm{Softmax}(\{G_t^{(p)}\}_{p=1}^{P})$ via

$$\left(a_t^{(1)}, \ldots, a_t^{(P)}\right) \sim \mathrm{Multinomial}\left(P;\, w_t^{(1)}, \ldots, w_t^{(P)}\right), \tag{9}$$

and set $\boldsymbol{T}_t^{(p)}$ to $\boldsymbol{T}_t^{(a_t^{(p)})}$ for each p. At the end of the reverse process when $t = 0$, we finally pick the lowest-energy $\boldsymbol{T}_0^{(p)}$.

In practice, following Wohlwend et al. (2024), we add gradient-based guidance (Bansal et al., 2024) to bias trajectories toward low-energy poses. After resampling, we rotate and translate the denoised pose $\boldsymbol{T}_t^{(p)}$ for each particle using gradients of U. See Section 4.2 for the manifold-aware update rule. To account for this guided perturbation, we adjust the importance weights for the next resampling step with the update

$$H_t^{(p)} = \frac{p_\theta\left(\boldsymbol{T}_t^{(p)} \mid \boldsymbol{T}_{t+1}^{(p)}, C\right)}{p_\theta\left(\boldsymbol{T}_t^{(p)} - \alpha \nabla_{(\boldsymbol{R}_t^{(p)}, \boldsymbol{t}_t^{(p)})} U(\boldsymbol{T}_t^{(p)}) \mid \boldsymbol{T}_{t+1}^{(p)}, C\right)} G_t^{(p)}, \tag{10}$$

where α is a scalar step size, and resample with this weight instead of only $G_t^{(p)}$ in Equation (7). This steers the reverse diffusion process toward low-energy poses while approximately preserving sampling from the tilted posterior.

4.2 From Distance Gradients to Rigid-Body Updates

Performing the rigid-body update requires computing translational and rotational gradients of the distance potential U. While automatic differentiation can be applied directly to a parameterization of $\boldsymbol{R}$, rotations lie on the nonlinear manifold $\mathrm{SO}(3)$, so naïve Euclidean gradient updates may fail to preserve orthogonality and can exhibit parameterization-dependent numerical behavior (Geist et al., 2024). To address this, we aggregate residue-level forces induced by U and convert them into a global rigid-body update in the Lie algebra of $\mathrm{SO}(3)$, following Jin et al. (2023). This yields rotational updates that remain on the rotation manifold and improve numerical stability in our experiments. To address this issue, we aggregate residue-level forces induced by the distance potential and convert them into a global rigid-body update in the Lie algebra of $\mathrm{SO}(3)$, following Jin et al. (2023). This produces a rotational update in the tangent space of $\mathrm{SO}(3)$, avoiding transitions between manifold charts and improving numerical stability in our experiments.

Specifically, let $\boldsymbol{x}_i = \boldsymbol{R}\boldsymbol{b}_i + \boldsymbol{t}$ be the transformed binder coordinate under $(\boldsymbol{R}, \boldsymbol{t})$. We can express the gradient contribution of binder residue $\boldsymbol{b}_i$ and receptor residue $\boldsymbol{r}_j$ as

$$\boldsymbol{g}_{ij} = \left(\|\boldsymbol{x}_i - \boldsymbol{r}_j\| - \hat{\boldsymbol{D}}_{ij}\right) \frac{\boldsymbol{x}_i - \boldsymbol{r}_j}{\|\boldsymbol{x}_i - \boldsymbol{r}_j\|}. \tag{11}$$

Now let $\boldsymbol{g}_i = {}^1\!/_N \sum_{j=1}^N \boldsymbol{g}_{ij}$ be the average gradient acting on binder residue i from all residues in the receptor. To convert this to a rigid-body update, we first interpret each $\boldsymbol{f}_i = -\boldsymbol{g}_i$ as a force acting on the corresponding binder residue as a rigid object. Our goal then is to compute the angular velocity (rotational gradient) of the entire binder. Let $\boldsymbol{\mu}$ be the center of mass of the binder residues, and define the centered coordinates as $\boldsymbol{\rho}_i = \boldsymbol{x}_i - \boldsymbol{\mu}$. Then, the net force and torque exerted by our guidance gradient are

$$\boldsymbol{F} = \sum_i \boldsymbol{f}_i; \quad \boldsymbol{\tau} = \sum_i \boldsymbol{\rho}_i \times \boldsymbol{f}_i \tag{12}$$

where $\times$ denotes the vector cross product. To account how the binder residues are distributed about the center of mass, we may then define the inertia tensor

$$\boldsymbol{\mathcal{I}} = \sum_i \left(\|\boldsymbol{\rho}_i\|^2 \boldsymbol{I}_3 - \boldsymbol{\rho}_i \boldsymbol{\rho}_i^\top\right). \tag{13}$$

Following Euler's rotation equation (Truesdell, 1977), the angular acceleration $\frac{d\boldsymbol{\omega}}{dt}$ can then be found using

$$\frac{d\boldsymbol{\omega}}{dt} = (\boldsymbol{\mathcal{I}} + \alpha \boldsymbol{I}_3)^{-1} \boldsymbol{\tau} \tag{14}$$

where $\alpha < 0.001$ is a small stabilizing constant. We then take an Euler step to estimate the angular velocity (rotational gradient) as

$$\nabla_{\boldsymbol{R}} U(\boldsymbol{R}, \boldsymbol{t}; \boldsymbol{D}) = (\boldsymbol{\mathcal{I}} + \alpha \boldsymbol{I}_3)^{-1} \boldsymbol{\tau} \Delta t \approx \boldsymbol{\omega}_{\Delta t} \tag{15}$$

for a small step size Δt, assuming the initial angular velocity $\boldsymbol{\omega}_0$ is 0 (Jin et al., 2023; Huang et al., 2024). On the other hand, we may straightforwardly derive the translational gradient as simply the average force applied to each of the binder's residues:

$$\nabla_{\boldsymbol{t}} U(\boldsymbol{R}, \boldsymbol{t}; \boldsymbol{D}) = \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{f}_i. \tag{16}$$

These equations make computing the guidance gradient both more stable and principled, avoiding any particular rotation parameterization. Using Equations (15) and (16), we steer the pose diffusion model's reverse trajectory after resampling and updating the importance weight Equation (10).

Because we use a predicted distance map $\hat{\boldsymbol{D}}$, some pairwise gradients may be inaccurate and move the binder toward an unrealistic pose. To mitigate this effect, we suppress each vector $\boldsymbol{x}_i - \boldsymbol{r}_j$ using a weight u_{ij} that is inversely proportional to the reconstruction error of $\hat{\boldsymbol{D}}$. Namely, let $\tilde{\boldsymbol{D}}^{\mathrm{MDS}} = \boldsymbol{V}_{:3}\boldsymbol{\Lambda}_{:3}^{1/2}$ where $\boldsymbol{V}$ and $\boldsymbol{\Lambda}$ are the eigenvalues and eigenvectors of the MDS embedding (Mead, 1992), explained in detail in Appendix A. We define

$$u_{ij} = 1 - \hat{\boldsymbol{D}}_{ij}^{-1} \left| \hat{\boldsymbol{D}}_{ij} - \tilde{\boldsymbol{D}}_{ij}^{\mathrm{MDS}} \right|. \tag{17}$$

This weight penalizes gradients of distances that are inconsistent with their MDS embedding, mitigating drift toward non-realizable poses.

4.3 Architecture

We now provide an overview of the GuideDock architecture. Binder and receptor features are passed to a latent flow matching model (Rombach et al., 2022; Dao et al., 2023) that predicts a distance map $\boldsymbol{D}$ used for steering through the equations in Sections 4.1 and 4.2. Candidate poses are generated by a pose diffusion model trained with a denoising score matching objective. We opt for a generative prior over $\boldsymbol{D}$ to enable sampling multiple distance maps.

Input Representation. In all components of GuideDock, we initialize residue features with 1280-dimensional embeddings from E1-600M (Jain et al., 2025) before projecting them to the model dimension d. We use E1 rather than models such as ESM-2 (Lin et al., 2022) because E1 uses retrieved homologous sequences during pre-training, which strengthens evolutionary signals relevant to distance prediction. This provides some benefits of MSAs without their inference-time cost. See Appendix H.2 for studies comparing E1-600M with ESM-2-650M.

Pose Diffusion Model. The backbone pose diffusion model contains geometry-aware multi-head cross-attention layers followed by successive custom FrameFlow layers (Yim et al., 2023a; 2024) from FlashIPA (Liu et al., 2025), chosen for their $\mathrm{SE}(3)$-equivariance and speed compared to tensor product-based networks (Thomas et al., 2018). Let $\boldsymbol{F}_B \in \mathbb{R}^{M \times d}$ and $\boldsymbol{F}_R \in \mathbb{R}^{N \times d}$ be the initial binder and receptor sequence embeddings. We first use a cross-attention mechanism to mix their features using their inter-residue distances in a learnable attention bias mechanism. See Appendix F.3 for details. The resulting embeddings $\boldsymbol{F}'_B$ and $\boldsymbol{F}'_R$ are then concatenated along the sequence dimension to form the complex residue embeddings $\boldsymbol{F}_C \in \mathbb{R}^{(M+N) \times d}$, which are then processed with

$$\boldsymbol{H}_C = \mathrm{GeomLayer}\left(\boldsymbol{F}'_C, \boldsymbol{X}_C, \boldsymbol{P}_t\right) + \boldsymbol{F}'_C \tag{18}$$

where $\boldsymbol{P}_t$ stands for the noise embedding of time step t, and $\mathrm{GeomLayer}$ stands for FrameFlow-style geometric layers detailed in Appendix F.1. $\boldsymbol{H}_C$ is then fed to a shallow transformer (Vaswani et al., 2017) and then passed through separate gated residual MLP heads to predict the translational and rotational scores. Additional architectural and training details are provided in Appendix F.3.

Distance Map VAE. The distance map generative model relies on a variational autoencoder (VAE) (Kingma & Welling, 2014) to decode denoised embeddings into distance maps. Similar to the pose diffusion model, the VAE encoder consists of FrameFlow-style layers followed by a transformer with intra-chain distance embeddings as attention biases. Let $\boldsymbol{F}_C \in \mathbb{R}^{(M+N) \times d}$ denote the concatenated binder and receptor features. The encoder outputs embeddings

$$\boldsymbol{Z}_C^{\mathrm{enc}} = \mathrm{Transformer}(\mathrm{GeomLayer}(\boldsymbol{F}_C)). \tag{19}$$

Then, the mean and variance of the Gaussian distribution over the residue latents are computed with

$$\boldsymbol{\mu}_C, \log \boldsymbol{\sigma}_C = \text{Linear}(\text{LayerNorm}(\boldsymbol{Z}_C^{\text{enc}})) \tag{20}$$

from which we sample latent tokens $\boldsymbol{Z}_C = \boldsymbol{\mu}_C + \sigma_C \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$. During decoding, each latent token is processed by a stack of transformer layers in the model dimension d. Pairwise distances are then predicted using a two-layer MLP from the up-projected latent embeddings. The VAE is trained with MSE loss on ground-truth distance with an added cross-entropy loss for predicting amino acid sequence identities to preserve chemical information. See Appendix F.4 for details.

Distance Map Latent Flow Matching Model. For the latent flow matching model, sampled latents $\boldsymbol{Z}$ are projected up to the model dimension d with a linear layer and passed to a transformer using intra-chain distances as attention biases, similar to the VAE, resulting in embeddings $\boldsymbol{Z}'_C$. Treating $\boldsymbol{Z}'_C$ as a "clean" sample $\boldsymbol{Z}_1$, we sample a random "noisy" latent $\boldsymbol{Z}_0 \sim \mathcal{N}(0, \boldsymbol{I})$ and a time step $t \sim \mathcal{U}(0, 1)$ and then construct an interpolated latent via

$$\boldsymbol{Z}_t = t\boldsymbol{Z}_0 + (1-t)\boldsymbol{Z}_1. \tag{21}$$

During training, we predict the final clean latent with a DiT-based (Peebles & Xie, 2022) denoiser network g_ϕ via $\hat{\boldsymbol{Z}}_1 = \text{Linear}(g_\phi(\boldsymbol{Z}_t, t, \boldsymbol{F}_C))$ and draw samples by integrating the corresponding vector field $u_t(\boldsymbol{Z}_t \mid \hat{\boldsymbol{Z}}_1)$. We opt for a flow matching model because it offers better sampling efficiency than DDPM and score-matching alternatives (Song & Ermon, 2019; Ho et al., 2020). To recover distance maps, we use classifier-free guidance (Ho & Salimans, 2021) to condition the flow matching model with the single-chain protein features from the VAE encoder trunk. See Appendix F.5 for additional details.

5 EXPERIMENTS

Datasets. We evaluate GUIDEDOCK on two datasets: the Database of Interacting Protein Structures (DIPS) (Townshend et al., 2019) and Docking Benchmark 5.5 (DB5.5) (Vreven et al., 2015). DIPS is a large dataset of protein complexes sourced from the PDB, whereas DB5.5 is an expert-curated benchmark with known high-difficulty examples. We use the DIPS-Het (Wang et al., 2023) subset of DIPS to evaluate heterogeneous, non-homomeric complexes, including antibody-antigen and enzyme-inhibitor interactions. DIPS-Het contains 9,093 training examples, 1,146 validation examples, and 1,153 test examples. For DB5.5, we use the 25-example test split from Ganea et al. (2022).

Training and Inference Details. All components of GUIDEDOCK are trained and validated exclusively on the DIPS-Het training and validation sets, respectively, to prevent leakage. We validate the backbone pose diffusion model on a fixed set of 30 randomly selected proteins from the DIPS-Het validation set and use the highest-DockQ checkpoint in our experiments. The VAE and latent flow matching models are trained separately and validated on the same data, using distance map MSE as the monitoring metric. During inference, we use 50 denoising steps for the pose diffusion model and 100 steps for the latent flow matching model, sampling one distance map per complex and using 4 particles for FK steering. See Appendix H for performance when averaging multiple distance maps. All training was performed on a single 80GB NVIDIA A100 GPU. For a fair comparison with the inference-time study from Wu et al. (2024), we report mean inference time using an RTX 3090 GPU and 12 CPU cores from an AMD EPYC CPU on a SLURM cluster.

Metrics. We evaluate docking poses using the complex and interface root-mean-square deviation (RMSD), which are continuous coordinate-based metrics measuring faithfulness to the native complex/interface. We also report DockQ (Basu & Wallner, 2016), which is an overall quality metric for evaluating poses based on both RMSD and recovered contacts. See Appendix E for details. We report the mean, median, and standard deviation of each metric.

Baselines. We group baselines by evaluation regime. PatchDock (Schneidman-Duhovny et al., 2005b) and HDock (Yan et al., 2017b) are classical score-based references, AlphaFold-Multimer (Evans et al., 2021), Boltz-1x (Wohlwend et al., 2024), and Chai-1 (Chai Discovery, 2024) are included as high-cost co-folding references, and EquiDock (Ganea et al., 2022), DiffDock-PP (Ketata et al., 2023), ElliDock (Yu et al., 2024), and EBMDock (Wu et al., 2024) are the primary MSA-free rigid docking comparisons. This grouping reflects that co-folding models reconstruct

Table 1: Rigid protein-protein docking results on 100 samples from the DIPS-Het test set. Arrows indicate whether higher/lower is better.

		Complex RMSD ↓			Interface RMSD ↓			DockQ ↑			Time (s) ↓
Category	**Method**	Mean	Median	Std	Mean	Median	Std	Mean	Median	Std	Mean
Score-based	PatchDock	19.34	17.95	10.30	17.16	16.17	10.35	0.04	0.02	0.20	2232
	HDock	2.04	0.23	5.98	2.67	0.25	8.51	0.88	0.98	0.28	782
Co-folding	AlphaFold-Multimer	6.45	4.66	8.17	6.14	2.12	8.43	0.48	0.52	0.31	1940
	Boltz-1x	9.62	4.96	10.26	6.47	2.63	8.01	0.35	0.33	0.22	120.5
	Chai-1	7.79	3.96	8.51	5.64	1.77	7.45	0.38	0.39	0.22	168.6
MSA-free rigid	EQUIDOCK	11.73	10.94	7.19	11.43	10.82	6.60	0.12	0.05	0.18	3.9
	DIFFDOCK-PP (5)	12.09	11.87	9.88	13.79	12.50	11.72	0.24	0.04	0.35	30.9
	ELLIDOCK	13.64	13.33	7.03	15.50	13.80	7.05	0.06	0.03	0.08	1.9
	EBMDOCK (5)	9.09	7.95	6.83	8.98	6.93	6.20	0.22	0.12	0.22	8.3
	GUIDEDOCK (1)	8.17	5.94	7.29	8.95	6.68	7.71	0.27	0.16	0.28	7.5

Table 2: Rigid protein-protein docking results on DB5.5.

		Complex RMSD ↓			Interface RMSD ↓			DockQ ↑			Time (s) ↓
Category	**Method**	Mean	Median	Std	Mean	Median	Std	Mean	Median	Std	Mean
Score-based	PatchDock	16.46	16.52	7.56	15.60	15.29	6.03	0.05	0.02	0.01	951
	HDock	5.55	0.42	9.42	5.19	0.31	8.99	0.72	0.97	0.42	703
Co-folding	AlphaFold-Multimer	7.65	4.86	8.03	6.41	1.69	7.66	0.48	0.55	0.10	2540
	Boltz-1x	6.23	2.77	5.56	4.50	1.10	5.52	0.38	0.52	0.25	128.9
	Chai-1	6.01	2.88	6.66	4.48	1.14	6.19	0.39	0.49	0.25	203.4
MSA-free rigid	EQUIDOCK	14.73	14.14	5.31	13.23	11.97	4.93	0.04	0.04	0.03	3.8
	DIFFDOCK-PP (5)	15.42	14.11	8.16	16.88	15.06	11.40	0.04	0.03	0.03	37.3
	ELLIDOCK	14.41	13.00	6.78	12.48	11.13	4.97	0.06	0.04	0.06	1.5
	EBMDOCK (5)	14.77	15.31	5.11	14.27	12.93	5.94	0.06	0.03	0.06	7.8
	GUIDEDOCK (1)	11.46	11.52	4.98	11.40	10.74	5.15	0.09	0.05	0.10	7.4

complexes from sequence-derived features rather than solving the same fixed-conformation pose-inference problem. Whenever possible, we use the performance of each method as reported in its original paper or in papers where the method was benchmarked by other authors. We note that AlphaFold-Multimer was trained on proteins submitted to the PDB up to September 30, 2021 (Jumper et al., 2021), so its performance may be optimistic given that DIPS and DB5.5 contain proteins released prior to this date, resulting in potential leakage. Parentheses $(\cdot)$ in tables indicate the number of samples drawn, from which a single pose is chosen according to each method's selection strategy. For EBMDock, this refers to the lowest-energy pose; for DiffDock-PP, this refers to the pose with the highest score from the authors' pre-trained confidence model.

5.1 DIPS-HET AND DB5.5 RESULTS

Following Wu et al. (2024), we evaluate GUIDEDOCK on 100 samples from the DIPS-Het test set and on DB5.5. Results are shown in Table 1 and Table 2. Our primary comparison class is MSA-free rigid docking models that infer poses under low per-complex inference costs. Within this class, GUIDEDOCK improves mean DockQ by 30%, reduces mean CRMSD by 15%, and achieves nearly twice the mean DockQ of DiffDock-PP on DB5.5 using only a single sampled pose.

We include co-folding models as high-cost references rather than direct competitors, since they solve the broader full-complex prediction problem using large-scale training (e.g. over 270 000 examples for Boltz-1x (Wohlwend et al., 2024) vs 9093 for GUIDEDOCK)) and MSA-derived features. GUIDEDOCK targets the complementary high-throughput setting where candidate structures are already available and fast MSA-free pose inference is required.

DB5.5 remains a particularly difficult for MSA-free deep docking methods, with low absolute DockQ across the class. Nevertheless, GUIDEDOCK consistently improves CRMSD and IRMSD, showing that geometry guidance moves predictions closer to the native pose even when high-DockQ complexes remain challenging. This pattern is further shown in Figure 3a, where GUIDEDOCK has noticeably lower RMSD variability than DiffDock-PP, EquiDock, ElliDock, and EBMDock on DB5.5. Qualitative examples are provided in Appendix I.

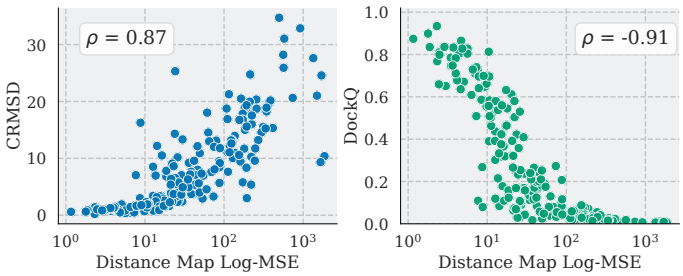
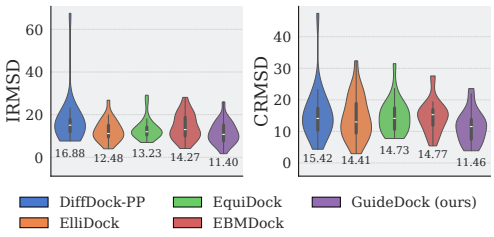
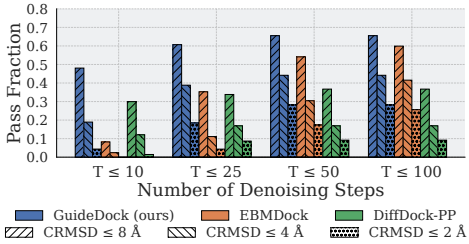


Figure 2: Metrics versus the predicted distance map's log-MSE.



(a) CRMSD and IRMSD on DB5.5. Means of the RMSD distributions are shown concurrently.



(b) Fraction of samples satisfying CRMSD thresholds over denoising step cutoffs T on DIPS-Het.

Runtime clarifies our specific high-throughput regime further. GUIDEDOCK runs up to 300x faster than AlphaFold-Multimer, up to 20x faster than Boltz-1x and Chai-1, and over 100x faster than HDock. While regression-based methods such as EquiDock and ElliDock remain expectedly faster, GUIDEDOCK outperforms them across all metrics while maintaining the same order of magnitude of speed. GUIDEDOCK also improves time-to-performance among generative rigid docking methods by requiring only one sampled pose instead of 5 or more to obtain superior results. Compared with EBMDock, it also requires fewer denoising steps (50 versus 100) and does not compute gradients of a neural potential during sampling. As shown in Figure 3b, GUIDEDOCK reaches higher pass rates across CRMSD thresholds and denoising-step cutoffs than the other generative models. These results indicate that distance-map guidance helps the backbone diffusion model search a smaller and more accurate region of pose space.

5.2 SENSITIVITY TO DISTANCE MAP ACCURACY

In this section, we analyze how the performance of GUIDEDOCK depends on the accuracy of predicted distance maps. Effective geometric guidance should perform best when the predicted distance map is accurate. To test this, we compute correlations between distance map log-MSE and CRMSD, IRMSD, and DockQ on the DIPS-Het test set. Figure 2 shows a strong relationship between each docking metric and distance map log-MSE, measured by Spearman's ρ. As MSE decreases, each metric improves substantially, indicating that the predicted distance maps provide useful steering information for pose generation. See Appendix H for additional sensitivity analyses on the number of sampled distance maps and the number of particles for FK steering. See also Appendix G.1 for ablations on the effectiveness of the generative guidance mechanism.

6 CONCLUSION

In this work, we introduced GUIDEDOCK, a geometry-guided diffusion model for MSA-free rigid protein-protein pose inference. By using predicted distance maps as soft constraints, GUIDEDOCK improves RMSD and DockQ metrics over MSA-free deep learning baselines while requiring substantially less computation than classical docking software and large co-folding models. Within the MSA-free rigid-docking regime, GUIDEDOCK exhibits new state-of-the-art performance, paving the way for more effective high-throughput tools for engineering protein-protein interactions. Future work includes extending this approach to flexible docking, exploring biomolecular interactions beyond proteins, and integrating evolutionary information beyond residue distances without explicit MSAs.

AI USE STATEMENT

LLMs (GPT-5.5 Pro) were used to review and refine the phrasing and presentation of this manuscript. LLM-based coding agents (GPT-5.5) were also used to audit and implement basic features of the method and all figure generation code, all of which was verified for correctness by the primary author. All other components of the research process, e.g. ideation, formulation, arguments, and experimental setups are human-made. We take responsibility for the final content of this work, including all text and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

See Appendix F for full details regarding the model implementation including the Flash Attention-based IPA modules used, training algorithms, and hyperparameters. Regarding data, we derive our preprocessing and splitting pipelines from EBMDock (Wu et al., 2024), whose code can be found at `https://github.com/wuhuaijin/EBMDock`.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Roni Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *International Conference on Learning Representations*, 2024.

Sankar Basu and Björn Wallner. DockQ: A quality measure for protein-protein docking models. *PLoS One*, 11(8):e0161879, August 2016.

H M Berman, J Westbrook, Z Feng, G Gilliland, T N Bhat, H Weissig, I N Shindyalov, and P E Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, January 2000.

Valentin De Bortoli, Emile Mathieu, Michael John Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016*, pp. 10–21. Association for Computational Linguistics (ACL), 2016.

Chai Discovery. Chai-1: Decoding the molecular interactions of life. *bioRxiv*, 2024. doi: 10.1101/2024.10.10.615955. URL `https://www.biorxiv.org/content/early/2024/10/11/2024.10.10.615955`.

Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. In *International Conference on Learning Representations*, 2023.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

Richard Evans, Michael O'Neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, Olaf Ronneberger, Sebastian Bodenstein, Michal Zielinski, Alex Bridgland, Anna Potapenko, Andrew Cowie, Kathryn Tunyasuvunakool, Rishub Jain, Ellen Clancy, Pushmeet Kohli, John Jumper, and Demis Hassabis. Protein complex prediction with AlphaFold-Multimer. October 2021.

Da-Fei Feng and Russell F Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *J. Mol. Evol.*, 25(4):351–360, August 1987.

C Frieden. Protein-protein interaction and enzymatic activity. *Annu. Rev. Biochem.*, 40(1):653–696, 1971.

Octavian-Eugen Ganea, Xinyuan Huang, Charlotte Bunne, Yatao Bian, Regina Barzilay, Tommi S. Jaakkola, and Andreas Krause. Independent SE(3)-equivariant models for end-to-end rigid protein docking. In *International Conference on Learning Representations*, 2022.

Andreas René Geist, Jonas Frey, Mikel Zhobro, Anna Levina, and Georg Martius. Learning with 3D rotations, a hitchhiker's guide to SO(3). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15331–15350. PMLR, 21–27 Jul 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Yufei Huang, Odin Zhang, Lirong Wu, Cheng Tan, Haitao Lin, Zhangyang Gao, Siyuan Li, and Stan Z. Li. Re-dock: Towards flexible and realistic molecular docking with diffusion bridge. In *International Conference on Machine Learning*, 2024.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

Sarthak Jain, Joel Beazer, Jeffrey A. Ruffolo, Aadyot Bhatnagar, and Ali Madani. E1: Retrieval-augmented protein encoder models. *bioRxiv*, 2025.

Joël Janin. Protein–protein docking tested in blind predictions: the capri experiment. *Mol. BioSyst.*, 6:2351–2362, 2010. doi: 10.1039/C005060C.

Joël Janin, Kim Henrick, John Moult, Lynn Ten Eyck, Michael J E Sternberg, Sandor Vajda, Ilya Vakser, Shoshana J Wodak, and Critical Assessment of PRedicted Interactions. CAPRI: A critical assessment of PRedicted interactions. *Proteins*, 52(1):2–9, July 2003.

T Jardetzky. Not just another fab: the crystal structure of a TcR-MHC-peptide complex. *Structure*, 5 (2):159–163, February 1997.

Wengong Jin, Siranush Sarkizova, Xun Chen, Nir Hacohen, and Caroline Uhler. Unsupervised protein-ligand binding energy prediction via neural euler's rotation equation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Isak Johansson-Åkhe and Björn Wallner. Improving peptide-protein docking with AlphaFold-Multimer using forced sampling. *Front. Bioinform.*, 2(959160):959160, September 2022.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596 (7873):583–589, 2021.

W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, 1976. doi: https://doi.org/10.1107/S0567739476001873.

Mohamed Amine Ketata, Cedrik Laue, Ruslan Mammadov, Hannes Stark, Menghua Wu, Gabriele Corso, Céline Marquet, Regina Barzilay, and Tommi S. Jaakkola. Diffdock-PP: Rigid protein-protein docking with diffusion models. In *ICLR 2023 - Machine Learning for Drug Discovery workshop*, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *ICLR*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *International Conference on Learning Representations*, 2014.

Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*. Stochastic Modelling and Applied Probability. Springer, Berlin, Germany, December 2010.

Adam Leach, Sebastian M Schmon, Matteo T. Degiacomi, and Chris G. Willcocks. Denoising diffusion probabilistic models on SO(3) for rotational alignment. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

Marc F Lensink, Raúl Méndez, and Shoshana J Wodak. Docking and scoring protein complexes: CAPRI 3rd edition. *Proteins*, 69(4):704–718, December 2007.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023. doi: 10.1126/science.ade2574.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Andrew Liu, Axel Elaldi, Nicholas T Franklin, Nathan Russell, Gurinder S. Atwal, Yih-En Andrew Ban, and Olivia Viessmann. Flash invariant point attention. In *Advances in Neural Information Processing Systems*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Matthew McPartlon and Jinbo Xu. Deep learning for flexible and site-specific protein docking and design. *bioRxiv*, 2023.

A Mead. Review of the development of multidimensional scaling methods. *Statistician*, 41(1):27, 1992.

Iain H Moal, Mieczyslaw Torchala, Paul A Bates, and Juan Fernández-Recio. The scoring of poses in protein-protein docking: current capabilities and future directions. *BMC Bioinformatics*, 14(1):286, October 2013.

S B Needleman and C D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48(3):443–453, March 1970.

Dmitry I Nikolayev and Tatjana I Savyolov. Normal distribution on the rotation group so(3). *Texture Stress Microstruct.*, 29(3-4):201–233, January 1997.

Saro Passaro, Gabriele Corso, Jeremy Wohlwend, Mateo Reveiz, Stephan Thaler, Vignesh Ram Somnath, Noah Getz, Tally Portnoi, Julien Roy, Hannes Stark, David Kwabi-Addo, Dominique Beaini, Tommi Jaakkola, and Regina Barzilay. Boltz-2: Towards accurate and efficient binding affinity prediction. *bioRxiv*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

Michal J Pietal, Janusz M Bujnicki, and Lukasz P Kozlowski. GDFuzz3D: a method for protein 3D structure reconstruction from contact maps, based on a non-euclidean distance function. *Bioinformatics*, 31(21):3499–3505, November 2015.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Christina E M Schindler, Isaure Chauvot de Beauchêne, Sjoerd J de Vries, and Martin Zacharias. Protein-protein and peptide-protein docking and refinement using ATTRACT in CAPRI. *Proteins*, 85(3):391–398, March 2017.

Dina Schneidman-Duhovny, Yuval Inbar, Ruth Nussinov, and Haim J Wolfson. PatchDock and SymmDock: servers for rigid and symmetric docking. *Nucleic Acids Research*, 33(Web Server issue):W363–7, July 2005a.

Dina Schneidman-Duhovny, Yuval Inbar, Ruth Nussinov, and Haim J Wolfson. PatchDock and SymmDock: servers for rigid and symmetric docking. *Nucleic Acids Research*, 33(Web Server issue):W363–7, July 2005b.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1548–1554. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *International Conference on Machine Learning*, 2025.

Yang Song and Stefano Ermon. *Generative modeling by estimating gradients of the data distribution*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds, 2018.

Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, December 1952.

Raphael Townshend, Rishi Bedi, Patricia Suriana, and Ron Dror. End-to-end learning on 3d protein structure for interface prediction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

C. Truesdell. *A First Course in Rational Continuum Mechanics. Vol. 1: General Concepts*, volume 71 of *Pure and Applied Mathematics*. Academic Press, New York, first edition, 1977.

Ilya A Vakser. Protein-protein docking: from interaction to interactome. *Biophys. J.*, 107(8): 1785–1793, October 2014.

Marco Vassura, Luciano Margara, Pietro Di Lena, Filippo Medri, Piero Fariselli, and Rita Casadio. Reconstruction of 3d structures from protein contact maps. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 5(3):357–367, July 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

Thom Vreven, Iain H Moal, Anna Vangone, Brian G Pierce, Panagiotis L Kastritis, Mieczyslaw Torchala, Raphael Chaleil, Brian Jiménez-García, Paul A Bates, Juan Fernandez-Recio, Alexandre M J J Bonvin, and Zhiping Weng. Updates to the integrated protein-protein interaction benchmarks: Docking benchmark version 5 and affinity benchmark version 2. *Journal of Molecular Biology*, 427(19):3031–3041, September 2015.

Yiqun Wang, Yuning Shen, Shi Chen, Lihao Wang, Fei YE, and Hao Zhou. Learning harmonic molecular representations on riemannian manifold. In *International Conference on Learning Representations*, 2023.

C L Webber, Jr, A Giuliani, J P Zbilut, and A Colosimo. Elucidating protein secondary structures using alpha-carbon recurrence quantifications. *Proteins*, 44(3):292–303, August 2001.

Zeyu Wen, Jiahua He, Huanyu Tao, and Sheng-You Huang. PepBDB: a comprehensive structural database of biological peptide-protein interactions. *Bioinformatics*, 35(1):175–177, January 2019.

Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Noah Getz, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Liam Atkinson, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, and Regina Barzilay. Boltz-1: Democratizing biomolecular interaction modeling. *bioRxiv*, 2024.

Huaijin Wu, Wei Liu, Yatao Bian, Jiaxiang Wu, Nianzu Yang, and Junchi Yan. EBMDock: Neural probabilistic protein-protein docking via a differentiable energy model. In *International Conference on Learning Representations*, 2024.

Yumeng Yan, Di Zhang, Pei Zhou, Botong Li, and Sheng-You Huang. HDOCK: a web server for protein–protein and protein–DNA/RNA docking based on a hybrid strategy. *Nucleic Acids Reseach*, 45(W1):W365–W373, July 2017a.

Yumeng Yan, Di Zhang, Pei Zhou, Botong Li, and Sheng-You Huang. HDOCK: a web server for protein–protein and protein–DNA/RNA docking based on a hybrid strategy. *Nucleic Acids Reseach*, 45(W1):W365–W373, July 2017b.

Jason Yim, Andrew Campbell, Andrew YK Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S Veeling, Regina Barzilay, Tommi Jaakkola, et al. Fast protein backbone generation with se (3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023a.

Jason Yim, Brian L. Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se(3) diffusion model with application to protein backbone generation. In *International Conference on Machine Learning*. JMLR.org, 2023b.

Jason Yim, Andrew Campbell, Emile Mathieu, Andrew Y. K. Foong, Michael Gastegger, Jose Jimenez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Frank Noe, Regina Barzilay, and Tommi Jaakkola. Improved motif-scaffolding with SE(3) flow matching. *Transactions on Machine Learning Research*, 2024.

Ziyang Yu, Wenbing Huang, and Yang Liu. Rigid protein-protein docking via equivariant elliptic-paraboloid interface prediction. In *International Conference on Learning Representations*, 2024.

Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems 32*, Vancouver, Canada, 2019.

A MULTIDIMENSIONAL SCALING

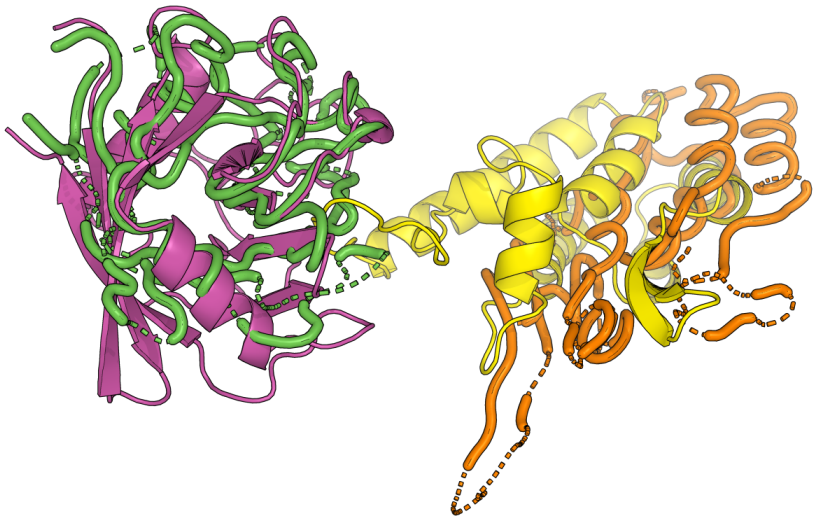


Figure 4: Example of collapse of MDS when $\hat{\boldsymbol{D}}$ is ill-conditioned. Magenta and yellow structures are ground-truth. Small perturbations in distances can result in incorrect internal geometries (green) and structures nearly collapsing into a plane (orange).

Finding the coordinates that satisfy a distance map is known as the multidimensional scaling (MDS) problem (Mead, 1992; Torgerson, 1952). The first step in classical MDS is to double-center the squared Euclidean distances $\boldsymbol{D}^2 \in \mathbb{R}^{n \times n}$:

$$\boldsymbol{B} = -\frac{1}{2}\boldsymbol{J}\boldsymbol{D}^2\boldsymbol{J}; \quad \boldsymbol{J} = \boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top. \tag{22}$$

$\boldsymbol{B}$ can then be rewritten as the gram matrix $\boldsymbol{B} = \boldsymbol{X}^\top\boldsymbol{X}$ where $\boldsymbol{X}$ contains the original 3D coordinates of the complex up to a rigid transformation. $\boldsymbol{X}$ is then recovered by

$$\boldsymbol{X} = \boldsymbol{V}_{:3}\boldsymbol{\Lambda}_{:3}^{\frac{1}{2}} \tag{23}$$

where $\boldsymbol{V}_{:3}$ contains the top-3 eigenvectors of $\boldsymbol{B}$ sorted by its top-3 eigenvalues $\boldsymbol{\Lambda}_{:3}$. Finally, the Kabsch algorithm (Kabsch, 1976) can be used to recover the $(\boldsymbol{R}, \boldsymbol{t})$ needed to align a non-docked structure to $\boldsymbol{X}$, recovering the native complex. However, in the rigid docking case, while the intra-binder and intra-receptor residue distances are known, the binder-receptor distances are not, and directly using a neural network's predicted distances $\hat{\boldsymbol{D}}$ can only approximate a true Euclidean embedding, introducing small drifts that may lead to unusable structures. See Appendix A for an example. Thus, it is usually infeasible to directly use $\hat{\boldsymbol{D}}$ in an MDS pipeline for rigid docking.

B DENOISING SCORE MATCHING MODELS

Denoising score matching models (DSMs) (Vincent, 2011; Song & Ermon, 2019; Song et al., 2021) are diffusion-based generative models that are trained to predict the gradient of the log-density of the data, known as the *score function*, at progressing noise levels. In the DSM framework, clean samples $\boldsymbol{x}_0 \sim p_{\text{data}}$ are corrupted through a forward process described by the stochastic differential equation (SDE)

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{w} \tag{24}$$

where $\boldsymbol{f}$ is a drift term, g is a scalar diffusion coefficient, and $\boldsymbol{w}_t$ is standard Brownian motion. Let p_t denote the marginal density of $\boldsymbol{x}_t$ at time t. In DSMs, a time-conditioned neural network $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$ is trained to approximate the score function $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ by minimizing the loss

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{\boldsymbol{x}_0}\mathbb{E}_{\boldsymbol{x}_t|\boldsymbol{x}_0}\left[||\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t \mid \boldsymbol{x}_0)||^2\right] \tag{25}$$

where $t \sim \text{Uniform}(0, 1)$. Samples are then drawn by solving the reverse SDE

$$\mathrm{d}\boldsymbol{x} = [\boldsymbol{f}(\boldsymbol{x}_t, t) - g(t)\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)]\mathrm{d}t + g(t)\mathrm{d}\bar{\boldsymbol{w}} \tag{26}$$

where $\mathrm{d}\bar{\boldsymbol{w}}$ is a standard Wiener process in reverse time. In practice, these models are trained by discretizing time from 0 to 1 into T steps and using closed-form perturbation kernels to approximate the forward process. For example, the forward process may be defined by $\boldsymbol{x}_t = \boldsymbol{x}_0 + \sigma_t\boldsymbol{\epsilon}$ where σ_t is the noise level at time t, and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$. This permits a closed form for the score function,

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t \mid \boldsymbol{x}_0) = -\frac{1}{\sigma_t^2}(\boldsymbol{x}_t - \boldsymbol{x}_0), \tag{27}$$

which we train $\boldsymbol{s}_\theta$ to predict. Sampling is then approximated numerically by starting from a random $\boldsymbol{x}_T \sim \mathcal{N}(0, \sigma_T^2\boldsymbol{I})$ and iteratively taking Euler-Maruyama steps from $t = T$ to 0 via the update rule

$$\boldsymbol{x}_{t-1} = \boldsymbol{x}_t + g(t)^2\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\Delta t + g(t)\sqrt{\Delta t}\,\boldsymbol{\epsilon} \tag{28}$$

where Δt is a fixed step size (Kloeden & Platen, 2010). We use this type of model as the backbone pose generator for GUIDEDOCK.

C DIFFUSION OVER TRANSLATIONS AND ROTATIONS

Formally, a diffusion process over translations can be defined by

$$\boldsymbol{t}_t = \boldsymbol{t}_0 + \sigma_t\boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}) \tag{29}$$

for steps $1 \leq t \leq T$ where $\boldsymbol{t}_0 \in \mathbb{R}^3$ is the center of mass of the clean binder protein, and σ_t is the variance of the noise added at step t.

While defining a diffusion process over translations $\boldsymbol{t}$ is straightforward with a Gaussian prior in $\mathbb{R}^3$, sampling rotations $\boldsymbol{R} \in \text{SO}(3)$ requires additional care (Bortoli et al., 2022). In Euclidean diffusion, noise is sampled from a standard Gaussian with variance σ_t^2; for rotations, noise is sampled from IGSO(3) (Nikolayev & Savyolov, 1997; Leach et al., 2022), whose density is defined by

$$p(\omega, \sigma_t) = \frac{1 - \cos\omega}{\pi} f_{\sigma_t}(\omega) \tag{30}$$

where

$$f_{\sigma_t}(\omega) = \sum_{l=0}^{\infty}(2l+1)\exp\left(-l(l+1)\sigma_t^2/2\right)\frac{\sin((l+1/2)\omega)}{\sin(\omega/2)} \tag{31}$$

for $\omega \in [0, \pi)$. By sampling a unit axis-angle rotation vector $\hat{\boldsymbol{\omega}}$ and a random angle ω, the score function we train a neural network to learn is now

$$\nabla \log p_t(\boldsymbol{R}_t \mid \boldsymbol{R}) = \left(\frac{d}{d\omega}\log f(\omega)\right)\hat{\boldsymbol{\omega}} \tag{32}$$

where $\boldsymbol{R}_t = (\omega\hat{\boldsymbol{\omega}})\,\boldsymbol{R}$ (Yim et al., 2023b; Corso et al., 2023; Ketata et al., 2023). In practice, following Ketata et al. (2023) and Leach et al. (2022), we compute a truncated expansion of Equation (31) and then use inverse CDF sampling to sample a random rotation angle ω. The axis-angle vector $\hat{\boldsymbol{\omega}}$ is computed by $\hat{\boldsymbol{\omega}} = \frac{\boldsymbol{x}}{||\boldsymbol{x}||}$ where $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I})$.

D DRAWBACKS OF DIRECT POSE DIFFUSION

To support the discussion in Section 3, we train a pose diffusion model on the same objective as DiffDock-PP on the DIPS-Het dataset (Wang et al., 2023) and take increasing numbers of samples k from 1 to 30. We then report the fraction of best-of-k complexes determined by lowest complex RMSD (CRMSD) that satisfy various RMSD thresholds following previous work (Ketata et al., 2023; Lensink et al., 2007) on 30 random samples from the DIPS-Het test set. See Appendix E for details on these metrics. In Figure 5, the fraction of complexes with interface RMSD (IRMSD) < 4Å reaches 50% only after 10 samples even with perfect selection. The authors of DiffDock-PP report similar behavior in their work, requiring up to 10 samples before the fraction of oracle-chosen poses with CRMSD < 4Å approaches 50% (Ketata et al., 2023). This suggests that unguided pose diffusion has low sample reliability. Without strong geometric priors, these models need to explore a large and weakly constrained space of potential rigid transforms before arriving at a reasonable pose.

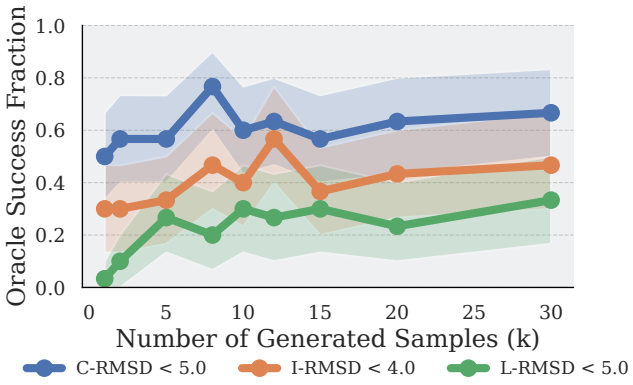


Figure 5: Best-of-k fraction of samples from a guidance-free diffusion model that satisfy quality thresholds with 95% confidence intervals.

E Docking Evaluation Metrics

In this work, we follow the Critical Assessment of Prediction of Interactions (CAPRI) (Janin et al., 2003) and evaluate generated complexes using complex RMSD (C-RMSD), interface RMSD (I-RMSD), and DockQ. Let $\boldsymbol{X} \in \mathbb{R}^{(M+N)\times 3}$ be the set of ground-truth complex coordinates, and let $\hat{\boldsymbol{X}} \in \mathbb{R}^{(M+N)\times 3}$ be the set of predicted complex coordinates aligned with $\boldsymbol{X}$ using the Kabsch algorithm. Then, the C-RMSD is defined as

$$\text{C-RMSD}(\boldsymbol{X}, \hat{\boldsymbol{X}}) = \sqrt{\frac{1}{M+N}|\boldsymbol{X} - \hat{\boldsymbol{X}}|_F^2}, \tag{33}$$

where $|\cdot|_F$ denotes the Frobenius norm. A lower C-RMSD indicates a higher-quality pose. The I-RMSD is defined similarly, except the coordinates considered are restricted to residues that are in contact at the interface of the complex. Here, contact is defined as being within 8Å.

DockQ is a joint metric combining I-RMSD as well as ligand RMSD (L-RMSD) and F_{nat}. L-RMSD is an RMSD metric based on the coordinates of the binder rather than the complex or the interface, and F_{nat} is the recall score of the recovered contacts in $\hat{\boldsymbol{X}}$. Formally, the DockQ score of a predicted complex $\hat{\boldsymbol{X}}$ is defined as

$$\text{DockQ}(\boldsymbol{X}, \hat{\boldsymbol{X}}) = \frac{1}{3}\left(F_{\text{nat}} + \frac{1}{1+\left(\frac{\text{I-RMSD}(\boldsymbol{X},\hat{\boldsymbol{X}})}{1.5}\right)^2} + \frac{1}{1+\left(\frac{\text{L-RMSD}(\boldsymbol{X},\hat{\boldsymbol{X}})}{8.5}\right)^2}\right). \tag{34}$$

DockQ scores are smooth and range from 0 to 1, with a higher score indicating a more accurate pose.

F Model Details

F.1 GeomLayer Details

The GeomLayer layers are based on layers from FrameFlow (Yim et al., 2023a; 2024) with minor modifications. They consist of layers of invariant point attention (IPA) (Jumper et al., 2021) from the official FlashIPA (Liu et al., 2025) repository interwoven with edge updates. We make the following modifications to the FrameFlow layers:

- We refrain from using explicit pair embeddings for efficiency and instead compute them as outer products of the 1D sequence and frame features. This amounts to setting `mode="flash_1d_bias"` in FlashIPA.

- We replace all layer normalization (Ba et al., 2016) with RMSNorm (Zhang & Sennrich, 2019).

Node and Edge Updates. Formally, let $\boldsymbol{H}_\ell \in \mathbb{R}^{(M+N)\times d}$ be the residue embedding matrix at layer ℓ, and let $\boldsymbol{T}$ denote the frames of those residues. A full $\mathrm{GeomLayer}$ is defined by a sequence of node and edge updates. We compute node updates via the following equations.

$$\boldsymbol{H}_{\text{ipa}} = \mathrm{RMSNorm}(\mathrm{FlashIPA}(\boldsymbol{H}_\ell, \boldsymbol{Z}_\ell, \boldsymbol{T}) + \boldsymbol{H}_\ell) \tag{35}$$
$$\boldsymbol{H}_{\text{skip}} = \mathrm{Linear}(\boldsymbol{H}_0) \tag{36}$$
$$\boldsymbol{H}_{\text{in}} = \boldsymbol{H}_{\text{ipa}} \,||\, \boldsymbol{H}_{\text{skip}} \tag{37}$$
$$\boldsymbol{H}_{\text{trans}} = \mathrm{Transformer}(\boldsymbol{H}_{\text{in}}) \tag{38}$$
$$\boldsymbol{H}_{\text{out}} = \mathrm{Linear}(\boldsymbol{H}_{\text{trans}}) + \boldsymbol{H}_\ell \tag{39}$$
$$\boldsymbol{H}_{\ell+1} = \mathrm{MLP}(\boldsymbol{H}_{\text{out}}) \tag{40}$$

Edge updates $\boldsymbol{Z}_\ell$ are then computed via the following equations.

$$\boldsymbol{H}_{\text{down}} = \mathrm{Linear}(\boldsymbol{H}_{\ell+1}) \tag{41}$$
$$\boldsymbol{Z}_{\text{in}}^{ij} = \boldsymbol{H}_{\text{down}}^{i} \,||\, \boldsymbol{H}_{\text{down}}^{j} \,||\, \boldsymbol{Z}_\ell^{ij} \tag{42}$$
$$\boldsymbol{Z}_{\ell+1} = \mathrm{RMSNorm}(\mathrm{MLP}(\boldsymbol{Z}_{\text{in}})) \tag{43}$$

Here, $\boldsymbol{Z}_\ell^{ij}$ is a virtual pairwise edge feature formed as the rank-r outer product

$$\boldsymbol{Z}_\ell^{ij} = \sum_{k=1}^{r} \boldsymbol{Z}_\ell^{1,i,k} \odot \boldsymbol{Z}_\ell^{2,j,k}. \tag{44}$$

FlashIPA consumes the factorized representations $\boldsymbol{Z}_\ell^1$ and $\boldsymbol{Z}_\ell^2$ directly without materializing a full pair embedding matrix.

Pairwise Features. In `mode="flash_1d_bias"`, the factor streams are computed by the FlashIPA edge embedder as functions of (1) node embeddings, (2) absolute sequence position, and (3) local geometric neighborhoods. First, we compute node-conditioned factors by a linear projection and reshape into two streams:

$$\boldsymbol{p}_i = \mathrm{Linear}(\boldsymbol{h}_\ell^i) \in \mathbb{R}^{2\times r\times d}. \tag{45}$$

Next, we compute an absolute position embedding $\boldsymbol{e}_i$ (sinusoidal index embedding) and project it into two factor streams:

$$\boldsymbol{e}_i = \mathrm{Linear}(\mathrm{IndexEmbed}(i)) \in \mathbb{R}^{2\times r\times d}. \tag{46}$$

We then incorporate local geometry using k-nearest-neighbor distograms. Specifically, for each residue i we compute KNN indices and binned distances

$$(\boldsymbol{d}_i, \boldsymbol{n}_i) = \mathrm{DistogramKNN}(\boldsymbol{t}_\ell, i; k), \tag{47}$$

embed the neighbor indices, concatenate with the binned distances, and linearly project to two factor streams:

$$\boldsymbol{g}_i = \mathrm{Linear}\Big([\boldsymbol{d}_i,\ \mathrm{IndexEmbed}(\boldsymbol{n}_i)]\Big) \in \mathbb{R}^{2\times r\times d}. \tag{48}$$

Finally, we concatenate these per-residue features and pass them through a shared MLP with RMSNorm to obtain the two edge factor streams:

$$\Big[\boldsymbol{z}_\ell^{1,i}, \boldsymbol{z}_\ell^{2,i}\Big] = \mathrm{RMSNorm}\Big(\,\mathrm{MLP}_{\text{edge}}\,\big([\boldsymbol{p}_i,\ \boldsymbol{e}_i,\ \boldsymbol{g}_i]\big)\Big), \tag{49}$$

where $\boldsymbol{z}_\ell^{1,i}, \boldsymbol{z}_\ell^{2,i} \in \mathbb{R}^{r\times d}$. In GUIDEDOCK, we use $r = 2$. More details on how these embeddings are used for a virtual pair embedding are below in Appendix F.2.

Algorithm 1 FlashIPA with factorized pair representations

Given input sequence $\boldsymbol{s} \in \mathbb{R}^{L\times d_{in}}$, factorized pair representation $\boldsymbol{z}_{ij} = \boldsymbol{z}_i^{1\intercal}\boldsymbol{z}_j^2$ (where $\boldsymbol{z}^1, \boldsymbol{z}^2 \in \mathbb{R}^{L\times r\times d_z}$ and $\boldsymbol{z} \in \mathbb{R}^{L\times d_z}$), frames $T_i = (R_i, \boldsymbol{t}_i), \forall i, j \in \{1, ..., L\}$

$\boldsymbol{q}_i^h, \vec{\boldsymbol{q}}_i^{hp}, \boldsymbol{k}_i^h, \vec{\boldsymbol{k}}_i^{hp}, \boldsymbol{v}_i^h, \vec{\boldsymbol{v}}_i^{hp} \leftarrow \text{Linear}(\boldsymbol{s}_i)$

$\boldsymbol{b}_i^{h1} \leftarrow \text{Linear}\left(\boldsymbol{z}_i^1\right)$

$\boldsymbol{b}_i^{h2} \leftarrow \text{Linear}\left(\boldsymbol{z}_i^2\right)$

$\hat{\boldsymbol{q}}_i^h \leftarrow \text{concat}\left(\boldsymbol{q}_i^h, \left\{T_i \circ \vec{\boldsymbol{q}}_i^{hp}\right\}_{p=1}^{N_{\text{Query}}}, \left\{\left\|T_i \circ \vec{\boldsymbol{q}}_i^{hp}\right\|_2^2\right\}_{p=1}^{N_{\text{Query}}}, \mathbf{1}_{N_{\text{Query}}}, \boldsymbol{b}_i^{h1}\right)$

$\hat{\boldsymbol{k}}_i^h \leftarrow \text{concat}\left(\frac{w_L}{\sqrt{c}}\boldsymbol{k}_i^h, \left\{\gamma^h w_L w_C T_i \circ \vec{\boldsymbol{k}}_i^{hp}\right\}_{p=1}^{N_{\text{Query}}}, \frac{-\gamma^h w_L w_C}{2}\mathbf{1}_{N_{\text{Query}}}, \left\{\frac{-\gamma^h w_L w_C}{2}\left\|T_i \circ \vec{\boldsymbol{k}}_i^{hp}\right\|_2^2\right\}_{p=1}^{N_{\text{Query}}}, \boldsymbol{b}_i^{h2}\right)$

$\hat{\boldsymbol{v}}_i^h \leftarrow \text{concat}\left(\boldsymbol{v}_i^h, \left\{T_i \circ \vec{\boldsymbol{v}}_i^{hp}\right\}_{p=1}^{N_{\text{Values}}}, \boldsymbol{z}_i^2\right)$

$\hat{\boldsymbol{o}}_i^h \leftarrow \text{FlashAttention}(\hat{\boldsymbol{Q}}^h, \hat{\boldsymbol{K}}^h, \hat{\boldsymbol{V}}^h)_i$

$\boldsymbol{o}_i^h, \vec{\boldsymbol{o}}_i^{hp}, \tilde{\boldsymbol{o}}_i^h \leftarrow \text{split}(\hat{\boldsymbol{o}}_i^h)$

$\tilde{\boldsymbol{o}}_i^h \leftarrow \boldsymbol{z}_i^{1\intercal}\tilde{\boldsymbol{o}}_i^h$

$\tilde{\boldsymbol{s}}_i \leftarrow \text{Linear}\left(\text{concat}_{h,p}\left(\tilde{\boldsymbol{o}}_i^h, \boldsymbol{o}_i^h, \vec{\boldsymbol{o}}_i^{hp}, \left\|\vec{\boldsymbol{o}}_i^{hp}\right\|_2\right)\right)$

Return $\tilde{\boldsymbol{s}}$

F.2 FLASHIPA

IPA generates residue embeddings from coordinate frames $\boldsymbol{T}_i$ and sequence features $\boldsymbol{s}_i$. For each attention head h, the algorithm combines scalar queries, keys, and values $(\boldsymbol{q}_i^h, \boldsymbol{k}_i^h, \boldsymbol{v}_i^h)$ with Euclidean query, key, and value vectors $(\vec{\boldsymbol{q}}_i^{hp}, \vec{\boldsymbol{k}}_i^{hp}, \vec{\boldsymbol{v}}_i^{hp})$, where p denotes a point index. FlashIPA accelerates IPA by computing attention weights block-wise with an online algorithm similar to Flash Attention (Dao et al., 2022). Its key modification is a low-rank factorization of pair features that enables compatibility with Flash Attention primitives. We show the full algorithm from Liu et al. (2025) in Appendix 1, which we adopt without modification.

In all components of GUIDEDOCK, we use the hyperparameters detailed in Table 3b.

F.3 POSE DIFFUSION MODEL

Cross Attention Mechanism. Given initial binder and receptor embeddings $\boldsymbol{F}_B \in \mathbb{R}^{M\times d}$ and $\boldsymbol{F}_R \in \mathbb{R}^{N\times d}$, we first compute the receptor-informed binder features as

$$\boldsymbol{F}_B = \text{Softmax}\left(\frac{(\boldsymbol{F}_R\boldsymbol{W}_K)(\boldsymbol{F}_B\boldsymbol{W}_Q)^\top}{\sqrt{d}} + \boldsymbol{\phi}\right)\boldsymbol{F}_R\boldsymbol{W}_V \tag{50}$$

and similarly for the binder-informed receptor embeddings $\boldsymbol{F}_R$. Here, $\boldsymbol{W}_{\{Q,K,V\}}$ are query, key, and value projections, and $\boldsymbol{\phi} \in \mathbb{R}^{M\times N}$ is a learnable attention bias matrix where $\boldsymbol{\phi}_{ij} = \text{Linear}(\text{GELU}(\boldsymbol{e}_{ij}))$, and $\boldsymbol{e}_{ij} \in \mathbb{R}^b$ is a vector of radial basis expansions of the inter-residue distance d_{ij}:

$$[\boldsymbol{e}_{ij}]_k = \exp\left(-\gamma\|d_{ij} - \mu_k\|^2\right), \quad k \in \{1, \cdots, b\}. \tag{51}$$

This enables geometry-aware cross-chain feature interactions.

Noise Conditioning. To inject timestep information into our score network, we compute noise embeddings $\boldsymbol{P}_t \in \mathbb{R}^{(M+N)\times d}$ using a 2-layer MLP on the same sinusoidal embeddings as Vaswani et al. (2017) broadcasted over the complex's sequence dimension. We condition each GeomLayer by inserting $\boldsymbol{P}_t$ into the node update in Equations (39) and (40):

$$\boldsymbol{h}_{\text{out}} = \text{AddGate}(\text{Linear}(\boldsymbol{h}_{\text{trans}}) + \boldsymbol{h}_\ell, \boldsymbol{P}_t) \tag{39*}$$

$$\boldsymbol{h}_{\ell+1} = \text{AddGate}(\text{MLP}(\boldsymbol{h}_{\text{out}}), \boldsymbol{P}_t) \tag{40*}$$

Here, $\mathrm{AddGate}(\boldsymbol{X}, \boldsymbol{Y})$ is defined by the following additive residual gate from Shi et al. (2021):

$$\mathrm{AddGate}(\boldsymbol{X}, \boldsymbol{Y}) = \boldsymbol{X} \odot \boldsymbol{G} + \boldsymbol{Y} \odot (\boldsymbol{1} - \boldsymbol{G}), \tag{52}$$

$$\boldsymbol{G} = \sigma\big(\mathrm{Linear}\big([\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{X} - \boldsymbol{Y}]\big)\big)\,. \tag{53}$$

Score Heads. To compute the rotational and translational scores, we first post-process the residue embeddings using

$$\boldsymbol{H}'_C = \mathrm{Transformer}(\mathrm{LayerNorm}(\boldsymbol{H}_C + \boldsymbol{F}_C)). \tag{54}$$

Then, we conduct mean pooling along the sequence dimension to acquire binder and receptor embeddings via

$$\boldsymbol{H}''_B = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{H}'_B; \quad \boldsymbol{H}''_R = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{H}'_R \tag{55}$$

where $\boldsymbol{H}'_B, \boldsymbol{H}'_R$ are the binder and residue subsets of $\boldsymbol{H}'_C$. For score computation, we then obtain a complex embedding with $\boldsymbol{H}''_C = \mathrm{MLP}(\boldsymbol{H}''_B \,||\, \boldsymbol{H}''_R) \in \mathbb{R}^d$ and compute vectors in $\mathbb{R}^3$ via noise-conditioned vector heads,

$$\boldsymbol{f}_{\mathrm{tr}} = \mathrm{VectorHead}_{\mathrm{tr}}(\boldsymbol{H}''_C,\ \boldsymbol{P}_t)\,; \quad \boldsymbol{f}_{\mathrm{rot}} = \mathrm{VectorHead}_{\mathrm{rot}}(\boldsymbol{H}''_C,\ \boldsymbol{P}_t)\,, \tag{56}$$

where each $\mathrm{VectorHead}$ is a separate gated residual MLP with a Swish activation (Ramachandran et al., 2017). We decompose each vector into a direction and magnitude by normalizing,

$$\hat{\boldsymbol{f}}_{\mathrm{tr}} = \frac{\boldsymbol{f}_{\mathrm{tr}}}{\|\boldsymbol{f}_{\mathrm{tr}}\|}; \quad \hat{\boldsymbol{f}}_{\mathrm{rot}} = \frac{\boldsymbol{f}_{\mathrm{rot}}}{\|\boldsymbol{f}_{\mathrm{rot}}\|}, \tag{57}$$

and pass the magnitudes to noise-conditioned scalar heads to obtain scale factors,

$$s_{\mathrm{tr}} = \mathrm{ScalarHead}_{\mathrm{tr}}(\|\boldsymbol{f}_{\mathrm{tr}}\|,\ \boldsymbol{P}_t)\,; \quad s_{\mathrm{rot}} = \mathrm{ScalarHead}_{\mathrm{rot}}(\|\boldsymbol{f}_{\mathrm{rot}}\|,\ \boldsymbol{P}_t)\,, \tag{58}$$

where each $\mathrm{ScalarHead}$ is a separate bias-free version of each $\mathrm{VectorHead}$. Finally, the predicted translational and rotational scores are given by

$$\hat{\boldsymbol{s}}_{\mathrm{tr}} = \frac{s_{\mathrm{tr}}}{\sigma_{\mathrm{tr}}}\,\hat{\boldsymbol{f}}_{\mathrm{tr}}; \quad \hat{\boldsymbol{s}}_{\mathrm{rot}} = s_{\mathrm{rot}}\,\mathcal{S}(\sigma_{\mathrm{rot}})\,\hat{\boldsymbol{f}}_{\mathrm{rot}}, \tag{59}$$

where σ_{tr} and σ_{rot} denote the translational and rotational noise scales, respectively, and $\mathcal{S}(\cdot)$ denotes the root-mean-square norm of $\mathrm{IGSO}(3)$ at the corresponding noise level, which accounts for the geometry of the rotation manifold, following Corso et al. (2023); Ketata et al. (2023).

Training. As discussed in Section 4.3, we train our pose diffusion model on a denoising score matching loss. For N binders and T sampled timesteps from $\mathcal{U}(0,1)$, we minimize the translational score matching loss $\mathcal{L}_{\mathrm{tr}}$ and rotational score matching loss $\mathcal{L}_{\mathrm{rot}}$ defined by:

$$\mathcal{L}_{\mathrm{tr}} = \frac{1}{N}\frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T}\left\|\hat{\boldsymbol{s}}_{\mathrm{tr}} - \nabla_{\boldsymbol{x}_i^{(t)}} \log p_t\left(\boldsymbol{x}_i^{(t)} \mid \boldsymbol{x}_i^{(0)}\right)\right\|^2, \tag{60}$$

$$\mathcal{L}_{\mathrm{rot}} = \frac{1}{N}\frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T}\left\|\hat{\boldsymbol{s}}_{\mathrm{rot}} - \nabla_{\boldsymbol{R}_i^{(t)}} \log p_t\left(\boldsymbol{R}_i^{(t)} \mid \boldsymbol{R}_i^{(0)}\right)\right\|^2. \tag{61}$$

We witnessed severe instability with just the sum of these losses. To counteract this, we make the following modifications:

- We multiply $\mathcal{L}_{\mathrm{tr}}$ with $\lambda_t = (1 - e^{-t})/e^{-t/2}$, which is a time-weighted decay to enhance stability near low noise levels (Yim et al., 2023b).
- We add a backbone coordinate MSE loss $\mathcal{L}_{\mathrm{bb}}$ only for time steps $t < 0.2$ to enforce stable predictions. Following Yim et al. (2023a), this loss is defined as

  $$\mathcal{L}_{\mathrm{bb}} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{X}_i - \hat{\boldsymbol{X}}_i)^2, \tag{62}$$

  where $\hat{\boldsymbol{X}}_i$ is predicted by a linear head using the binder features $\boldsymbol{H}'_B$ from Equation (54).
- Instead of forming each batch from N different complexes, we form each batch from T different timesteps sampled from $\mathcal{U}(0,1)$ for one complex at a time.

Finally, the final loss function for the pose diffusion model is

$$\mathcal{L}_{\mathrm{pose}} = \mathcal{L}_{\mathrm{tr}} + \mathcal{L}_{\mathrm{rot}} + \lambda_{\mathrm{bb}}\mathcal{L}_{\mathrm{bb}} \tag{63}$$

where $\lambda_{\mathrm{bb}} = 0.2$ is a balancing hyperparameter. Table 3a details the other relevant training hyperparameters as well as the gradient step sizes from Equation (10) α_{tr} and α_{rot}.

Table 3: Pose Diffusion Model Hyperparameters.

(a) Core hyperparameters.

Hyperparameter	Value
Gradient clip norm	1
Hidden dimension	160
Initialization	Kaiming uniform (He et al., 2015)
Optimizer	Adam (Kingma & Ba, 2015)
Weight decay	1×10^{-5}
Learning rate	5×10^{-5}
Adam β_1, β_2	$0.9,\ 0.999$
Batch size	8
Number of $\mathrm{GeomLayer}$ layers	9
Translational noise scale σ_{tr}	20
Rotational noise scale σ_{rot}	1.5
Translational step size α_{tr}	2.0
Rotational step size α_{rot}	5.0

(b) Invariant Point Attention (IPA).

Hyperparameter	Value
Attention heads	8
Query points	8
Point values	12
Distogram bins	22
Nearest neighbors	20

F.4 DISTANCE VAE

Architecture. In the distance VAE, we reuse the same $\mathrm{GeomLayer}$ architecture for encoding geometric features of each binder and receptor chain independently. To prevent the leakage of any ground-truth interface information and distances, we randomly rotate and translate the binder with variance σ_{rot} and σ_{tr} before passing them into any frame-aware layers.

Distance Prediction. We encountered instabilities and poor performance when predicting raw ground-truth inter-chain residue distances. To remedy this, we whiten the target distances according to the DIPS-Het training set's mean and standard deviation via $(d_{ij} - \mu_d)/\sigma_d$. We then predict this normalized target during training and denormalize the output of our neural network during distance prediction. Let $\boldsymbol{H}_i$ and $\boldsymbol{H}_j$ denote the final decoder embeddings for residues i and j. Pairwise distances are predicted with a two-layer MLP:

$$\hat{\boldsymbol{D}}_{ij} = \mathrm{MLP}(\boldsymbol{H}_i \,||\, \boldsymbol{H}_j \,||\, \boldsymbol{w}_{ij}(\boldsymbol{H}_i \otimes \boldsymbol{H}_j)), \tag{64}$$

where $||$ denotes concatenation, $\otimes$ is the outer product, and $\boldsymbol{w}_{ij} = \mathrm{Linear}(\boldsymbol{Z}_i \otimes \boldsymbol{Z}_j)$ is a residual weight from latent embeddings.

Training Details. We train our VAE with the core distance prediction MSE loss $\mathcal{L}_{\mathrm{MSE}}$:

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{M \times N} \sum_{i \in B, j \in R} (\boldsymbol{D}_{ij} - \hat{\boldsymbol{D}}_{ij})^2. \tag{65}$$

To encourage the preservation of chemical information of each residue, we add a cross-entropy loss $\mathcal{L}_{\mathrm{id}}$ for predicting amino acid residue identities defined by

$$\mathcal{L}_{\mathrm{id}} = \frac{1}{M+N} \sum_{i \in C} \mathrm{CE}(a_i, \hat{a}_i); \quad \hat{a}_i = \mathrm{Linear}(\mathrm{H_i}) \tag{66}$$

where a_i stands for the amino acid identity of residue i in the complex. Finally, we add a per-channel KL divergence penalty $\mathcal{L}_{\mathrm{KL}}$ following Rombach et al. (2022):

$$\mathcal{L}_{\mathrm{KL}} = \frac{1}{M+N} \sum_{i \in C} D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{Z}_i; \mu_{\boldsymbol{Z}_i}, \sigma_{\boldsymbol{Z}_i}) \,||\, \mathcal{N}(0,1)^d). \tag{67}$$

The final combined loss for the VAE is then

$$\mathcal{L}_{\mathrm{vae}} = \mathcal{L}_{\mathrm{MSE}} + \lambda_{\mathrm{id}} \mathcal{L}_{\mathrm{id}} + \lambda_{\mathrm{KL}} \mathcal{L}_{\mathrm{KL}} \tag{68}$$

where $\lambda_{\mathrm{id}} = 0.005$, and $\lambda_{\mathrm{KL}} = 0.001$. To improve performance, we chose to only introduce the KL loss with a sigmoid schedule (Bowman et al., 2016) starting at 8000 steps. See Table 4a for other hyperparameter settings.

Table 4: VAE and Flow Matching Model Hyperparameters.

(a) VAE.

Hyperparameter	Value
Gradient clip norm	1
Hidden dimension	96
Initialization	Uniform
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Weight decay	1×10^{-2}
Learning rate	1×10^{-4}
Adam β_1, β_2	0.9, 0.999
Batch size	8
Number of GeomLayer layers	5
Number of transformer blocks	8
Number of transformer attention heads	8
Latent dimension	8

(b) Latent Flow Matching.

Hyperparameter	Value
Gradient clip norm	10
Hidden dimension	128
Initialization	Uniform
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Weight decay	1×10^{-2}
Learning rate	1×10^{-4}
Adam β_1, β_2	0.9, 0.999
Batch size	8
Number of DiT blocks	7
Number of DiT attention heads	16
Number of transformer blocks	5
Number of transformer attention heads	8
Conditioning dropout $p_\varnothing$	0.2
Guidance scale γ	10

F.5 Distance Map Latent Flow Matching Model

Architecture. Unlike the VAE and pose diffusion model, we do not use IPA-based layers in the latent flow matching model since it works exclusively in latent space. The DiT architecture is largely unmodified, introducing the noise embedding via AdaLNZero layers and adding protein sequence information through residual connections in each DiT block.

Training Details. We train our latent flow matching model with the standard flow matching loss (Lipman et al., 2023)

$$\mathcal{L}_{\text{fm}} = \frac{1}{(1-t)^2} \frac{1}{M+N} \sum_{r \in C} \|\boldsymbol{z}_1^{(r)} - \hat{\boldsymbol{z}}_1^{(r)}\|^2, \tag{69}$$

where $\boldsymbol{z}_1^{(r)} \in \mathbb{R}^d$ denotes the latent vector of residue r. During training, with probability $p_\varnothing$, we replace the protein conditioning features with a learnable null context embedding $\boldsymbol{c}_\varnothing$ to balance unconditional and conditional generation. During sampling, the strength of the conditioning is controlled by a guidance scale hyperparameter γ via

$$\hat{\boldsymbol{Z}}_1 = (1-\gamma)\hat{\boldsymbol{Z}}_{\text{uncond}} + \gamma \hat{\boldsymbol{Z}}_{\text{cond}} \tag{70}$$

where $\hat{\boldsymbol{Z}}_{\text{uncond}} = g_\phi(\boldsymbol{Z}_t, t, \boldsymbol{c}_\varnothing)$ and $\hat{\boldsymbol{Z}}_{\text{cond}} = g_\phi(\boldsymbol{Z}_t, t, \boldsymbol{F}_C)$. Our flow matching model achieves the best performance with strong guidance hyperparameters $p_\varnothing = 0.2$ and guidance scale $\gamma = 10$. See Table 4b for other hyperparameter settings.

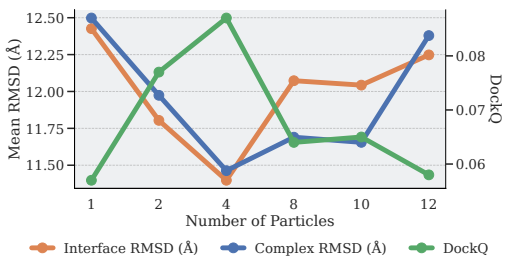
G Ablation Studies

G.1 Distance Potential Guidance Ablation

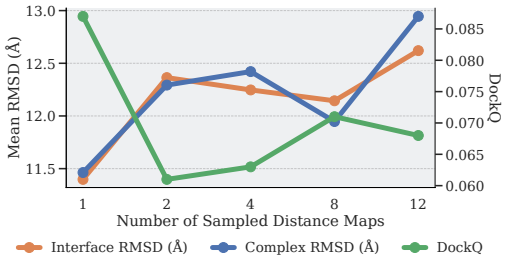
In this section, we study the effect of distance potential guidance in GuideDock. Across datasets, we compare performance with guidance, without guidance using only the backbone pose diffusion model, and with guidance from ground-truth distance maps instead of the latent flow matching model. Table 5 shows that potential guidance improves performance across all metrics. The ground-truth distance-map condition is an oracle setting, but it provides useful evidence that more accurate interface geometry could translate into substantially better rigid pose recovery.

Table 5: Ablation study on the effect of distance guidance.

		CRMSD ↓			IRMSD ↓			DockQ ↑		
Dataset	**Method**	Mean	Median	Std	Mean	Median	Std	Mean	Median	Std
DIPS-Het	With guidance	8.17	5.94	7.29	8.95	6.68	7.71	0.27	0.16	0.28
	No guidance	8.73	6.38	7.55	9.70	7.90	7.76	0.20	0.12	0.20
	Ground truth $\mathbf{D}$	1.98	0.92	2.90	2.18	1.06	2.77	0.65	0.76	0.31
DB5.5	With guidance	11.46	11.52	4.98	11.40	10.74	5.15	0.09	0.05	0.10
	No guidance	14.86	14.09	5.96	14.08	13.27	5.72	0.05	0.03	0.07
	Ground truth $\mathbf{D}$	3.20	2.49	2.07	3.32	2.85	2.03	0.38	0.41	0.23



(a) Mean performance on DB5.5 versus number of particles used in FK steering.



(b) Mean performance on DB5.5 versus number of distance maps sampled and averaged.

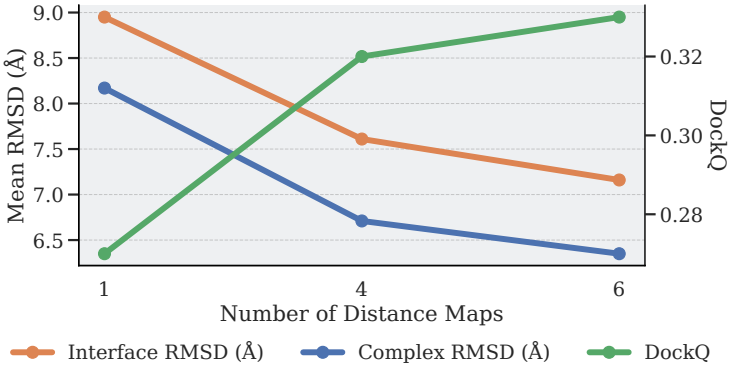


Figure 7: Mean top-1 performance on DIPS-Het versus number of distance maps sampled.

G.2 Generative Distance Model Ablation

In this section, we compare our generative distance model with a simple MLP regressor that directly predicts the inter-chain residue distance map from the same input features. As shown in Table 6, the regressor underperforms the generative model across all metrics and datasets in our experiments. This result does not rule out stronger deterministic distance predictors, but it supports the practical value of the current generative design. A further benefit of the generative model is that sampled distance maps can be used to diversify candidate poses, as shown in Appendix H.

Table 6: Ablation study comparing our generative distance model with a simple MLP regressor.

		CRMSD ↓			**IRMSD ↓**			**DockQ ↑**		
Dataset	**Method**	Mean	Median	Std	Mean	Median	Std	Mean	Median	Std
DIPS-Het	MLP regressor	8.62	6.73	7.22	9.32	7.66	7.41	0.24	0.12	0.24
	Generative (ours)	8.17	5.94	7.29	8.95	6.68	7.71	0.27	0.16	0.28
DB5.5	MLP regressor	12.84	11.92	5.56	12.82	12.27	5.58	0.06	0.04	0.04
	Generative (ours)	11.46	11.52	4.98	11.40	10.74	5.15	0.09	0.05	0.10

H Sensitivity Studies

H.1 Particle and Distance Map Sensitivity Studies

Here, we study the effects of the number of particles used in FK steering and the number of sampled distance maps used for guidance. Figure 6a shows that the best performance occurs around 4 particles, which is the value used in our experiments. Beyond 4 particles, performance quickly plateaus and does not recover. This suggests that a small set of high-quality pose candidates may be sufficient for docking, whereas increasing particle diversity alone is not necessarily beneficial. Efficiency also

decreases as the number of particles increases, with runtime reaching 25 seconds per complex when sampling 12 particles.

Figure 6b shows a similar pattern: performance decreases as more distance maps are sampled and averaged for guidance. In this sensitivity study, sampled distance maps are averaged before being passed to the guidance equations in Section 4. This averaging likely causes the performance drop because it can create smeared constraints that blur contacts and non-contacts. In contrast, using sampled distance maps as separate seeds to diversify generated poses improves performance across metrics. In Figure 7, we plot RMSD and DockQ against the number of generated distance maps, each corresponding to one pose. The metrics generally improve as the number of distance map-conditioned samples increases, consistent with the multi-modal nature of docking: high-quality poses tend to lie in sharp, well-separated modes defined by residue-residue constraints. Averaging distance maps blurs these constraints, whereas sampling multiple maps allows each run to explore a distinct mode.

H.2 Protein Language Model Embedding Sensitivity Study

Table 7: Performance Difference between ESM-2-650M and E1-600M embeddings.

		CRMSD ↓			**IRMSD ↓**			**DockQ ↑**		
Dataset	**Method**	Mean	Median	Std	Mean	Median	Std	Mean	Median	Std
DIPS-Het	ESM-2-650M	8.34	6.92	6.60	9.08	6.78	7.30	0.24	0.13	0.25
	E1-600M (ours)	8.17	5.94	7.29	8.95	6.68	7.71	0.27	0.16	0.28
DB5.5	ESM-2-650M	12.77	12.72	5.55	12.12	12.00	4.37	0.08	0.04	0.11
	E1-600M (ours)	11.46	11.52	4.98	11.40	10.74	5.15	0.09	0.05	0.10

In Table 7, we compare the performance of GuideDock when using ESM-2-650M (Lin et al., 2023) versus E1-600M (Jain et al., 2025) embeddings for protein sequence representation. In these experiments, switching from E1-600M to ESM-2-650M yields only modest changes across metrics, suggesting that the gains of GuideDock are not solely attributable to the choice of protein language model. The performance difference may be more pronounced with smaller PLMs, which we leave to future work.

I Example Complexes

This section provides additional visualizations of generated complexes compared with other MSA-free deep learning baselines. DIPS-Het complex IDs follow the convention {`PDB code`}-{`binder chain`}-{`receptor chains`}.

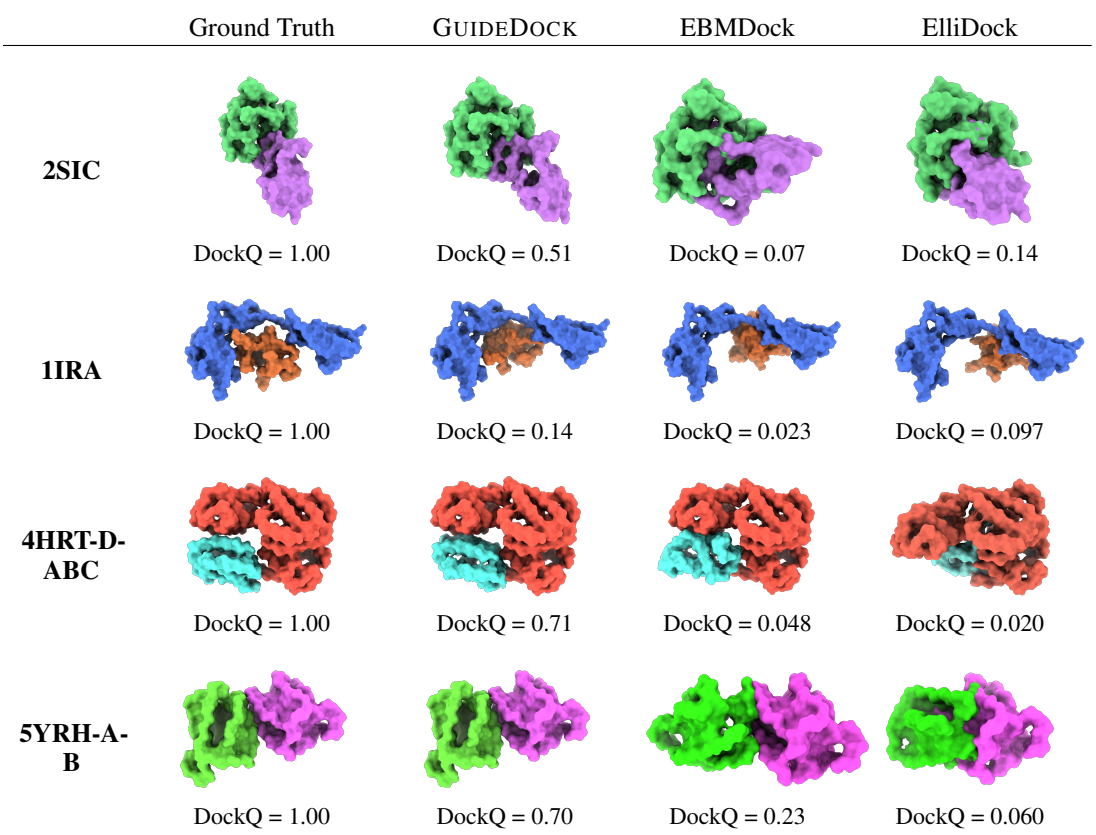


Figure 8: Example complexes generated by GUIDEDOCK compared to common MSA-free baselines. Each row corresponds to a complex, and each column corresponds to a method.