Can We Extract Physics-like Energies from Generative Protein Diffusion Models?

Sudeep Sarma[†,1,*] Harrison Truscott,[1,†] Da Xu,[1] Kendall Reid,[2] Lee-Shin Chu,[1] Jacky Chen,[3] and Jeffrey J. Gray[1,‡]

[1] *Department of Chemical and Biomolecular Engineering, Johns Hopkins University, Baltimore, MD 21218, USA*

[2] *Bioinformatics Program, Center for Biotechnology Education, Johns Hopkins University, Baltimore, MD 21218, USA*

[3] *CMU-Pitt Computational Biology, Dept. of Computational & Systems Biology, University of Pittsburgh, Pittsburgh, PA 15260, USA*

(Dated: December 12, 2025)

Diffusion models have emerged as the state-of-the-art method in generative artificial intelligence (AI) and have shown great success in image synthesis, video generation, molecular design, and protein structure prediction. For biophysical problems, such as protein folding and association, a fundamental question in diffusion-based methods is how their learned functions correspond to thermodynamics. In this paper, we study diffusion models through the lens of theoretical biophysics, analyzing their underlying formulation of potentials and exploring their applications in scoring protein interactions. We develop simple theories rooted in statistical physics that relate thermodynamic potentials to the negative log of the probability of observing a system in a particular state. We include dimensional analysis of diffusion model equations and a table mapping AI and physics jargon. We then test a diffusion model's ability to capture learned energies as negative log-likelihood values, $-\log p_0(x_0)$, by integrating over the diffusion-generated path or a probability flow path. We test these integrals on a simple 1D Gaussian mixture diffusion model and a protein-docking diffusion model, DFMDock. In the 1D case, we find that integration over both diffusion and flow paths can accurately recover ground truth probabilities. When we extract the learned docking energies for cases where DFMDock succeeds, we observe energy funnels with the minimum energy near the experimental docked structure, like those we observe with Rosetta, an empirically tuned physics-based biomolecular modeling suite. The learned energy performs comparably or outperforms Rosetta interface energy in 6 out of 25 cases at ranking the correctness of docked poses. These data show that we can extract a relevant learned energy function from a diffusion model and compare it to physical energy functions.

I. INTRODUCTION

Diffusion models are a class of deep-generative models that have shown great promise in image [1–3] and video generation [4], protein design [5, 6], and protein structure prediction [7–9]. Diffusion models were originally formulated by Sohl-Dickstein *et al.* [10] in analogy with nonequilibrium thermodynamics. They work in two stages. In forward diffusion, a data distribution is transformed to a prior (typically a Gaussian) over a series of time steps to learn an effective force field (the score), which is the time-dependent gradient of the logarithmic probability density function [11]. The score is learned by a deep neural network matching the noise added in training. One can then use the learned score to sample the underlying probability distribution of the data during a backward diffusion process starting from the prior. Leading image generation tools such as DALL-E [1], Stable Diffusion [2], and Midjourney [3] and text-to-video models like Sora [4] are all based on diffusion models. In protein design, breakthroughs such as RFDiffusion [5] and Chroma [6] have led to novel biomolecules that fold and bind in the lab. RFDiffusion represents amino acid residues as rigid frames and generates protein backbones from an ideal-gas-like prior. Chroma diffuses in polymer backbone torsion space, respecting conformational statistics of polymer ensembles.

In 2024, Deepmind released AlphaFold3 (AF3) [7], replacing AlphaFold2's [12] structure module with a diffusion module capable of predicting structures of proteins, protein complexes and post-translational modifications. AF3 drew from previous pioneering studies of diffusion for biomolecules [5, 6, 13], and has since inspired several derivative models (Chai-1 [8], Boltz-2 [14], and OpenFold3 [15]). The incredible success of AF3 and other protein deep learning models is in some ways surprising because these methods outperform physics-based methods like Rosetta [16, 17], which is based on decades of research into conformational sampling and scoring that follow energy functions derived from or inspired by physical principles. It raises the question of whether these high-performing deep-learning methods have implicitly learned a potential akin to nature's free energy function.

The generative model training objective is to learn to sample from a distribution that matches known protein structures. In other words, the sampling probability should be maximized at real protein structures, simi-

* Present address: Computational Engineering Division, Lawrence Livermore National Laboratory, Livermore, CA 94550, USA; Email: sarma3@llnl.gov.

† These authors contributed equally.

‡ Also at: Program in Molecular Biophysics, Johns Hopkins University, MD 21218, USA; Data Science and AI Institute, Johns Hopkins University, MD 21218, USA; Corresponding Author: jgray@jhu.edu.

lar to Anfinsen's dogma [18], which states that protein structures are at their lowest free energy state (i.e., in equilibrium). There are many functional forms that can have minima at the same (folded) states, so an AI protein model's learned energy function does not need to match nature's energy function. Additionally, diffusion models do not typically allow direct access to the probability of a given state. Here, we seek general approaches to extract information from a diffusion model in the form of its learned potential. In this way, we can compare this potential to physics-based energy functions and consider how AI and physics-based methods might be combined or improved.

To better understand the inherent potentials that diffusion models learn, we focus on a tractable and interpretable model for the protein-protein docking problem [19]. Previously, we introduced DFMDock (Denoising Force Matching Dock), a diffusion model for rigid-body protein-protein docking [20]. It is trained by adding translational and rotational noise to experimentally determined, bound protein complexes, which the model then learns to reverse through a denoising force-matching objective. During inference, the model inputs two unbound monomers and generates the structure of the bound complex. For our model study, we use a DFMDock version trained only on the translation docking space. In this way, we can also make comparison with a 1D translational diffusion model of a Gaussian mixture.

We formulate diffusion models in the language of statistical thermodynamics and explore how protein diffusion models like DFMDock score biomolecular structures and complexes. Statistical thermodynamics provides a framework for predicting the static and dynamic properties of a many-body system from its microscopic constituents and their interactions [21]. For systems in equilibrium, the Boltzmann distribution relates the probability of finding a system in a particular configuration x to its energy $E(x)$ as:

$$p(x) = \frac{e^{-\beta E(x)}}{Z}, \tag{1}$$

where β is the system's inverse temperature and the partition function Z is integrated over the space of all possible system configurations:

$$Z = \int_{\Omega} e^{-\beta E(x)} dx. \tag{2}$$

We assess a diffusion model's ability to capture energies by introducing a framework to extract negative log-likelihood values (NLL) of a sample x_0 at diffusion time $t = 0$, $-\log p_0(x_0)$. We establish a relationship between the learned energy, $-\log p_0(x_0)$, and the thermodynamic free energy of the configuration, $E_0(x_0)$.

This relationship connects deep learning models with physics by mapping learned potentials with fundamental thermodynamic quantities. We explore multiple approaches to calculating $E_0(x_0)$ by integrating the implicitly learned energy gradient using $\nabla_x \log p$ over both noisy diffusion paths and the smooth deterministic path generated by the probability flow ordinary differential equation (ODE) as described in [11]. As a test for our approach, we train a simple 1D diffusion model to learn a Gaussian mixture and compare generated samples' $-\log p_0(x_0)$ values using our integral methods to the ground truth values. We then apply the same approach to a protein docking diffusion system and compare the learned energies with Rosetta energies (**Fig. 1**).

To our knowledge this is the first attempt at extracting potentials from a biomolecular diffusion model and comparing with a physical energy function. This is also the first exploration of using paths other than the flow-equivalent ODE for likelihood computation. We aspire to improve the interpretability of deep-learning models for protein-protein interactions and open the door to building more robust physics informed tools for biomolecular engineering.

II. RELATED WORK

Protein-protein interactions (PPIs) drive a wide range of biological and chemical processes. They are involved in most cellular functions in living organisms, such as signaling, regulation, and recognition. Understanding their three-dimensional structures provides atomic-level insight into the mechanisms of these functions [22]. Traditional protein docking methods use sampling algorithms such as local-shape matching and Monte Carlo search algorithms to generate plausible docked conformations [23, 24]. Scoring functions, such as the Rosetta interface score function, can then be used to evaluate the physical energy of these docked conformations [25]. However, these search and scoring protocols are computationally intensive. Recently, new generative deep learning methods, especially diffusion models, have shown promise in addressing the protein-docking problem.

The first diffusion model applied to molecular docking, DiffDock, learned small-molecule–protein docking with denoising score matching on the translation, rotation, and torsion spaces of small molecules [13]. The developers subsequently introduced DiffDock-PP, a diffusion model for rigid protein docking [26] that learned to translate and rotate unbound protein structures to their bound conformations. DiffMaSIF is a diffusion framework for protein docking that uses an encoder-decoder architecture to learn physical surface patch complementarity [27]. LatentDock first trains a variational autoencoder on protein sequences and structures and then diffuses in the latent space to produce the final conformations of the protein complex [28].

Like diffusion models, energy-based models (EBMs) are a class of generative models that have been applied in modeling protein-protein interactions. EBMs are trained to learn a scalar energy function over the training data distribution, where lower energy values correspond to higher probability [29]. DSMBind is an EBM whose

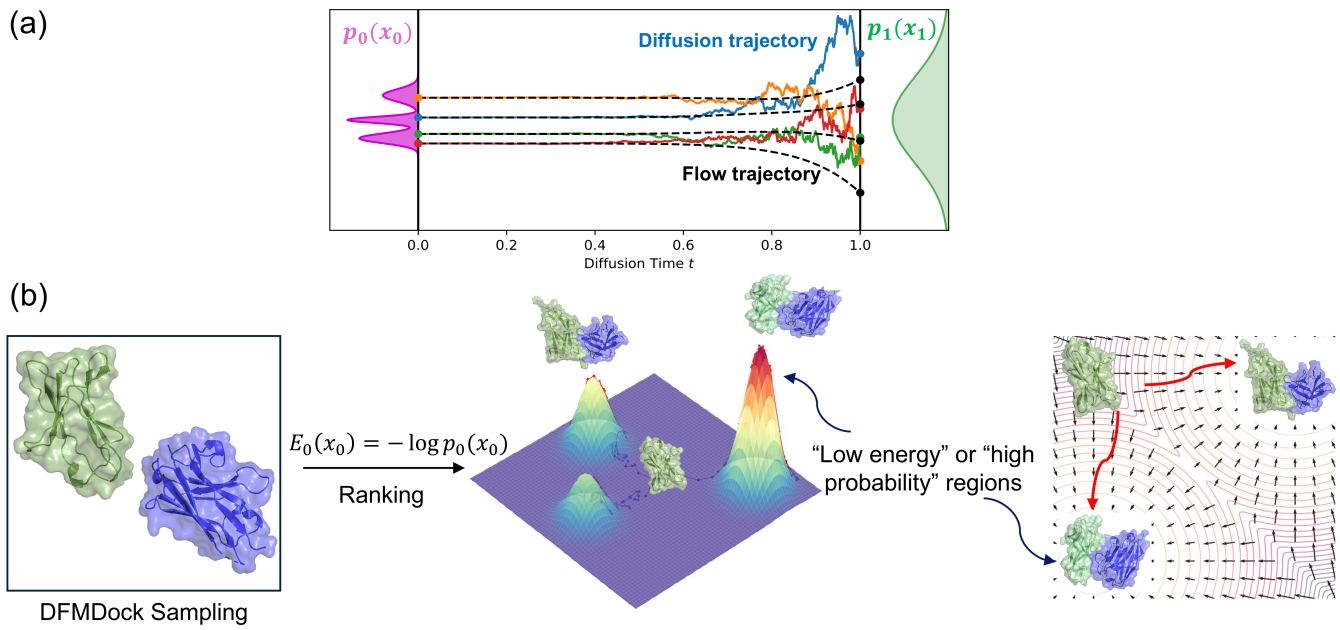

FIG. 1. (a) Diffusion (SDE paths, solid colors) and flow (ODE paths, dashed black) trajectories from a noise (prior) distribution, $p_1(x_1)$, at time $t = 1$ to the data distribution, $p_0(x_0)$, at time $t = 0$. (b) Negative log-likelihood values of docked protein complexes are a measure of learned protein-protein interaction energy.

energy function is optimized by matching its gradient to that of the forward diffusion process to predict binding energies of protein-protein interactions [30]. DockGame learned an energy function via supervision from physics-based models and self-supervision via score-matching with diffusion models for rigid multimeric protein docking [31]. EBMDock uses an energy-based learning framework and Langevin dynamics sampling for docking pose prediction [32]. Borisiak *et al.* trained an energy-based diffusion network to score mutations on peptides and CDR3 loops within TCR-pMHC interfaces [33]. Plainer *et al.* trained an energy-based model with an additional Fokker-Planck based loss on the model's temporal gradient to improve the consistency of the energy function at time $t = 0$. They showed that the gradient of this improved energy can be used as a force function for coarse-grained molecular dynamics simulations [34].

These models have shown promising empirical results. Other recent work has focused on the theoretical underpinnings of diffusion models. Diffusion models derive from non-equilibrium physics [10], but several papers have also conceptualized diffusion models using tools of equilibrium statistical mechanics. Ambrogioni *et al.* [35] showed that generative diffusion models undergo second-order phase transitions as described by mean-field theory. Sclocchi *et al.* [36] found that during the backward diffusion process, the system undergoes a phase transition such that the probability of reconstructing high-level features like the class of an image suddenly drops. Biroli *et al.* [37] showed that the backward diffusion process has three dynamical regimes, and they characterized the cross-overs between them as speciation (analogous to symmetry-breaking phenomena) and collapse (analogous to glass transition).

The principle of extracting learned likelihood from score-based generative diffusion models was first introduced by Song *et al.* (2020) [11] using their probability flow ODE. By converting the diffusion process into an equivalent process using stochasticity-free ODEs, they could use the instantaneous change-of-variables formula from [38] to integrate the probability change along a flow ODE trajectory. By contrast, we formulate likelihood recovery in terms of the Fokker-Planck equation describing the diffusion process. The instantaneous change-of-variables formula is a special case of this Fokker-Planck formulation when integrated over the flow trajectory, thanks to the equivalence between the ODE and diffusion process [11, 38], meaning our formulation generalizes the original likelihood formula to any (piecewise-differentiable) path.

Our paper focuses on devising general approaches to extract the learned likelihood function from diffusion models, and we compare the learned energy with physical energy potentials for protein docking, thus contributing to the interpretability of diffusion models in protein structure prediction and design.

TABLE I. Mathematical terms of diffusion models and their interpretations as AI or physics jargon. Example units: m, meters; J, Joules. λ is used as an arbitrary unit of diffusion "time." Dimensions: T, physical time; τ, diffusion/alchemical time; L, length; M, mass. Dimensional analysis deriving these units from the major equations is provided in Section A of the Supplemental Material.

Term	AI interpretation	Physical transformation	Unit	Dimension
t	Diffusion time	Alchemical transformation variable	λ	$[\tau]$
x_t	State	Position	m	$[L]$
$\beta = 1/kT$	Scaling constant	Inverse temperature	J^{-1}	$[T]^2[M]^{-1}[L]^{-2}$
$E_t(x_t)$	Time-dependent energy	Energy	J	$[M][L]^2[T]^{-2}$
$\nabla_x \log p(x_t, t)$	Score	-	m^{-1}	$[L]^{-1}$
$\beta^{-1}\nabla_x \log p(x_t, t)$	-	Force	Jm^{-1}	$[M][L][T]^{-2}$
$g(t)^2$	Diffusion coefficient	-	$m^2\lambda^{-1}$	$[L]^2[\tau]^{-1}$
$g(t)$	Scheduler	-	$m\lambda^{-1/2}$	$[L][\tau]^{-1/2}$
$w_t \sim \mathcal{N}(0, I)$	Perturbation kernel	Thermal fluctuations	$\lambda^{1/2}$	$[T]^{1/2}$
$f(x_t, t)$	Drift	External potential	$m\lambda^{-1}$	$[L][\tau]^{-1}$
$-\log p_0(x_0)$	Negative log-likelihood	-	1	[1]
$-\beta^{-1} \log p_0(x_0)$	-	Thermodynamic free energy	J	$[M][L]^2[T]^{-2}$

III. THEORY

A. Stochastic Differential Equations (SDEs) in Diffusion Models

The goal of generative modeling is to learn to transform a distribution of noise into samples that closely resemble a particular data distribution, such as the distribution of naturally occurring protein structures. In diffusion modeling, we first describe a transformation from noise to data—the *forward* diffusion process—and train a neural network to undo this transformation during the *reverse* diffusion process.

The forward diffusion process is obtained by incrementally noising a distribution of samples, p_0, until the samples converge to a known prior distribution, p_1. To describe the transformation of data to noise, we introduce the continuous variable $t \in [0, 1]$, which indexes a continuum of probability distributions smoothly interpolating from p_0 to p_1.

The variable t is referred to as "time" in the machine learning literature, but for a protein or other physical system, it actually represents an alchemical transformation. For example, RFDiffusion diffuses protein structures into an oriented, ordered ideal gas. We can draw parallels with alchemical free energy methods that gradually change one molecule into another along a transformation variable, allowing the calculation of the excess chemical potential through a nonphysical perturbation [39, 40]. To clarify the jargon between the AI and physical chemistry communities, we summarize the interpretation of the diffusion model terms in **Table I**, where we differentiate the alchemical time dimension $[\tau]$ from the physical time dimension $[T]$.

Mathematically, the incremental noising is formulated with a stochastic differential equation (SDE) known as the forward SDE:

$$dx_t = f(x_t, t)dt + g(t)dw_t. \quad (3)$$

In the forward diffusion process, a given point $x_0 \in \mathbb{R}^n$ sampled from the data distribution, $x_0 \sim p_0$, follows this SDE from time $t = 0$ to $t = 1$, arriving at some destination x_1 in the prior distribution, $x_1 \sim p_1$ [11]. The process is parameterized by two functions: the drift function, $f(x_t, t)\colon \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$, and the diffusion coefficient, $g(t)\colon \mathbb{R} \to \mathbb{R}$. w_t is a Wiener process describing stochastic Brownian motion, where dw_t is an infinitesimal random perturbation sampled at each time t from a Gaussian with mean 0 and variance dt.

The forward SDE is constructed with specific choices f and g so that the resulting *marginal* probability distribution of this process, $p(x_t, t)$—the probability of observing a point x_t at time t, marginalized over all initial points x_0 and stochastic trajectories from x_0 to x_t—converges to the desired prior distribution p_1 at $t = 1$. By definition, $p(x_0, 0)$ also matches our data distribution $p_0(x_0)$. Moreover, $p(x_t, t)$ defines a continuum of distributions $p(\cdot, t)$, smoothly interpolating from p_0 to p_1.

In our 1D diffusion model and DFMDock, we set $f(x_t, t) = 0$ and $g(t) = \sigma_0 \left(\frac{\sigma_1}{\sigma_0}\right)^t$; for our 1D diffusion model, $\sigma_0 = 0.1$ and $\sigma_1 = 70$, and for DFMDock, $\sigma_0 = 0.1$ and $\sigma_1 = 30$. In the language of [11], setting $f(x_t, t) = 0$ means our diffusion process is "variance exploding." In DFMDock, dw_t represents random translations of the ligand protein.

During inference, we use the reverse SDE [41]:

$$dx_t = [f(x_t, t) - g(t)^2 \nabla_x \log p(x_t, t)]dt + g(t)d\overline{w}_t, \quad (4)$$

where time flows backwards from 1 to 0, dt is an infinitesimal negative timestep and $\overline{w}_t$ is a reverse-time Wiener process. This SDE models the reverse diffusion process: starting from $t = 1$ and a sample $x_1 \sim p_1$, the marginal distribution of the reverse SDE transforms the prior distribution p_1 into the data distribution p_0 as t *decreases* from 1 to 0. $\nabla_x \log p(x_t, t)$, known as the *score*, is the spatial gradient of the log of the marginal probability of the process at time t.

By construction, this reverse SDE has an identical marginal distribution to the forward SDE [41]. The only unknown in eq. (4) is the score, so in score-based generative modeling, the diffusion model consists of a neural network, $s_\theta(x_t, t)$, parameterized by weights θ and trained to approximate the score given a sample x_t and a time t (potentially conditioned on other aspects of the process, such as protein sequence identity). Since the marginal probability $p(x_0, 0)$ of arriving at a point x_0 via the reverse SDE should match the data distribution $p_0(x_0)$, sampling a point x_1 from the prior and following the reverse SDE using the learned score should produce samples x_0 with the same probability as sampling from the data distribution directly.

The marginal distribution is generally intractable to solve at training time, as we only have a finite dataset of samples from p_0 and thus cannot marginalize over the entire distribution. However, we can still evaluate the conditional probability $q(x_t, t|x_0)$ of reaching a point x_t at time t when starting from a single point in the data distribution, x_0, instead of the entire p_0 distribution. Vincent (2011) [42] proved that matching the learned score with q conditioned on each of the data points in turn will converge to the marginal distribution as the size of the dataset grows. This results in the so-called denoising score matching objective that is used to train score-based diffusion models:

$$\mathcal{L}(\theta, t) = \mathbb{E}_{q(x_t,t,x_0)}\left[\frac{1}{2}\left\|s_\theta(x_t, t) - \nabla_x \log q(x_t, t|x_0)\right\|_2^2\right], \tag{5}$$

where the expectation is evaluated as a weighted sum over samples from the forward diffusion process starting at the points in the training dataset. In this way, score-based diffusion models implicitly incorporate information about all possible diffusion trajectories into the score. We reason that it should therefore be possible to recover a trained model's understanding of the ground-truth probability distribution—and thus, its understanding of the ground truth *energy*—without needing to explicitly marginalize over all paths through diffusion space.

B. Forces and Energies in Diffusion Models

In the case of protein docking or folding, our data distribution p_0 is assumed to sample from the distribution of system states at thermal equilibrium, corresponding to stably folded or docked proteins. Thus, we can define an energy $E_0(x_0)$ for any equilibrium state x_0 based on p_0:

$$p_0(x_0) = \frac{e^{-\beta E_0(x_0)}}{Z_0}, \tag{6}$$

where Z_0 is the partition function associated with the system at equilibrium, calculated by integrating over the space of all system states:

$$Z_0 = \int_\Omega e^{-\beta E_0(x_0)} dx_0, \tag{7}$$

and β corresponds to the system's inverse-temperature, $\beta = \frac{1}{kT}$.

Diffusion models are trained to allow sampling from p_0, meaning they must have an implicit understanding of this energy function at equilibrium. However, they cannot output $p_0(x_0)$ directly. Instead, as discussed earlier, score-based diffusion models learn the *score* of the marginal distribution $p(x_t, t)$ of the diffusion process:

$$s_\theta(x_t, t) \approx \nabla_x \log p(x_t, t), \tag{8}$$

where $s_\theta(x_t, t)$ is a neural network parameterized by weights θ.

We can, however, analyze what this score means in terms of our energy formulation. If we also assume that the system of states x_t described by $p(x_t, t)$ at constant time t is at pseudo-equilibrium (in other words, the diffusion process is quasi-static), we can define an energy E_t for an arbitrary time t in the diffusion process by generalizing eq. (6):

$$p(x_t, t) = \frac{e^{-\beta_t E_t(x_t)}}{Z_t}, \tag{9}$$

where β_t and Z_t are the inverse temperature and partition functions respectively at time t. Since we are defining this energy based on the probability alone, however, we are free to choose β_t and Z_t, defining the scaling and reference energy of E_t respectively. For purposes of this paper we thus assume a constant $\beta_t = \beta$ and $Z_t = Z_0 = Z$:

$$p(x_t, t) = \frac{e^{-\beta E_t(x_t)}}{Z}. \tag{10}$$

Taking the log of both sides isolates the energy term (with a coefficient of $-\beta$):

$$\log p(x_t, t) = -\beta E_t(x_t) - \log Z. \tag{11}$$

And since Z is independent of x_t, taking the spatial gradient of both sides yields the score solely in terms of energy:

$$\nabla_x \log p(x_t, t) = -\beta \nabla_x E_t(x_t). \tag{12}$$

In classical physics, the negative gradient of energy with respect to position, $-\nabla_x E_t(x_t)$, is a measure of force, so we define it as $F_t(x_t)$. Thus, output of a score-based diffusion model can be understood to be a scaled dimensionless force describing a learned energy function, $E_t(x_t)$, at time t (see dimensions in **Table I**):

$$s_\theta(x_t, t) \approx \nabla_x \log p(x_t, t) = -\beta \nabla_x E_t(x_t) = \beta F_t(x_t). \tag{13}$$

For samples generated by DFMDock, the data distribution p_0 aims to capture the distribution of all protein-protein docking systems at equilibrium. x_0 denotes a docked complex and $Z_0 = \int_\Omega e^{-\beta E(x_0)} dx$ would then be the partition function of two associating proteins. For rigid docking between a receptor protein and a ligand protein, the search space Ω spans all possible translations and

rotations of the ligand, with the receptor fixed. Thus, the negative log-likelihood of a docked sample generated at $t = 0$ is, in principle, proportional to the thermodynamic free energy of a protein-protein complex.

C. Negative Log-Likelihood or "Learned Energy" Estimation

As score-based diffusion models do not produce p_0 directly, we must find some way to recover it from the neural network's output: the spatial gradient of the marginal probability distribution, $\nabla_x \log p(x_t, t)$. Following [11], we postulate that to recover $p_0(x_0)$, one may integrate the change in marginal likelihood along a path from x_1 in the known prior distribution p_1 to the sample x_0 in the unknown data distribution p_0. We formulate this integral in two ways: first, we use the physical intuition of force and energy to try to construct a work integral over a path through data space x. Second, we use the probability theory behind the marginal distribution of our SDE to formulate an integral of $\frac{\partial \log p(x_t,t)}{\partial t}$ along a path from noise to data. We find that both approaches reduce to solving the same problem: integrating the spatiotemporal gradient field in both space and time of the marginal probability $p(x_t, t)$ over a path.

D. Physics-based Energy Recovery: The Work Integral

In classical mechanics, calculation of the energy accumulated by an object acted on by an external force $F_t(x_t)$ over a path S can be achieved using the work integral:

$$\Delta E = -\int_S F_t(x_t) \cdot dx_t. \tag{14}$$

From eq. (13), $F_t(x_t, t)$ can be replaced by $\beta^{-1} s_\theta(x_t, t)$ to measure the change in learned energy:

$$\Delta E \approx -\beta^{-1} \int_S s_\theta(x_t, t) \cdot dx_t. \tag{15}$$

However, this integral is flawed, and leads to incorrect results both theoretically and numerically. To see why, we simply need to return to the derivation of the work integral:

$$\begin{aligned} \Delta E &= \int_S \nabla_x E(x_t) \cdot dx_t \quad \text{by F.T.C. for line integrals;} \\ &= \int_S -F(x_t) \cdot dx_t \qquad \text{by the definition of force.} \end{aligned} \tag{16}$$

This derivation makes a fundamental assumption that the energy function, E, is solely a function of space, x, and therefore independent of time, t. However, the connection between learned probability and learned energy gives no such guarantee, as the marginal probability $p(x_t, t)$ changes significantly with time. Thus, to compute ΔE, we need to incorporate both the spatial *and* temporal derivatives of E:

$$\begin{aligned} dE &= \nabla_x E_t(x) \cdot dx_t + \frac{\partial E_t(x)}{\partial t} dt \\ &= -F_t(x) \cdot dx_t + \frac{\partial E_t(x)}{\partial t} dt. \end{aligned} \tag{17}$$

Along a given path x_t, we can change variables by replacing dx_t with $\frac{\partial x_t}{\partial t} dt$ to formulate ΔE for a space- and time-dependent $E_t(x_t)$:

$$\begin{aligned} \Delta E &= \int_S dE \\ &= \int_0^1 \left(-F_t(x_t) \cdot \frac{\partial x_t}{\partial t} + \frac{\partial E_t(x)}{\partial t} \right) dt, \end{aligned} \tag{18}$$

where we now define the integral range from 0 to 1 to correspond to the connection between the data distribution and the noise distribution.

This integral relies on both the spatial and temporal gradients of $E_t(x)$. While the learned spatial gradient is easily accessible from the score, how to determine the learned temporal gradient is not obvious. To understand this further, we need to examine the probability theory behind the marginal distribution $p(x_t, t)$—so let's briefly reframe the problem of learned energy recovery to that of likelihood computation.

E. Negative Log-Likelihood Recovery

Our primary goal is to capture the energy of docked complexes $E_0(x_0)$. Since we can directly relate the energy at $t = 0$ with the marginal log-likelihood $\log p_0(x_0)$ via eq. (11), if we can compute this log-likelihood directly, we can subsequently recover the energy up to a constant offset. Replacing E in eq. (18) with $\log p$ (and canceling the resulting β^{-1} terms) we can consider the line integral of the gradient of this log-likelihood over some path x_t through time and space:

$$\Delta \log p = \int_0^1 \left(\nabla_x \log p(x_t, t) \cdot \frac{\partial x}{\partial t} + \frac{\partial \log p(x_t, t)}{\partial t} \right) dt. \tag{19}$$

The learned score $s_\theta(x_t, t)$ can be used to approximate $\nabla_x \log p(x_t, t)$. The time-derivative $\frac{\partial \log p}{\partial t}$ can be addressed using the Fokker-Planck equation (see Supplementary Material B), which gives us the change in the marginal probability distribution of an SDE in terms of the diffusion function g and spatial derivatives of the distribution at time t (assuming the drift $f(x_t, t) = 0$):

$$\frac{\partial \log p(x_t,t)}{\partial t} = \frac{1}{2}g(t)^2\Big(\nabla_x \cdot \nabla_x \log p(x_t,t)\Big) + \frac{1}{2}g(t)^2\|\nabla_x \log p(x_t,t)\|^2. \tag{20}$$

Now that we can write eq. (19) solely in terms of $\nabla_x \log p(x_t,t)$, we can once again substitute the learned score function s_θ from a trained diffusion model. Computing the divergence of the learned score requires taking its derivative; helpfully, modern neural network libraries (in the case of DFMDock, PyTorch) have built-in automatic differentiation capabilities used for backpropagation, so computing this derivative is tractable. Specifically, the divergence of the score is the sum of its partial derivatives along each dimension i, which can be rewritten more compactly as a trace:

$$\nabla_x s_\theta(x_t,t) = \sum_i \frac{\partial [s_\theta(x_t,t)]_i}{\partial [x_t]_i} = \mathrm{Tr}\Big(\nabla_x s_\theta(x_t,t)\Big),$$

where $\nabla_x s_\theta(x_t,t)$ is the Jacobian matrix of s_θ, whose diagonal elements are precisely the derivatives in the sum. Using this identity, we can rewrite eq. (20) in terms of s_θ:

$$\frac{\partial \log p(x_t,t)}{\partial t} \approx \frac{1}{2}g(t)^2\mathrm{Tr}\Big(\nabla_x s_\theta(x_t,t)\Big) + \frac{1}{2}g(t)^2\|s_\theta(x_t,t)\|^2. \tag{21}$$

Plugging this approximation back into eq. (19), we have an expansion for the change in log probability over some path S in terms of the neural network score. So,

$$\Delta \log p \approx \int_0^1 \Bigg(s_\theta(x_t,t)\cdot \frac{\partial x_t}{\partial t} + \frac{1}{2}g(t)^2\mathrm{Tr}\Big(\nabla_x s_\theta(x_t,t)\Big) + \frac{1}{2}g(t)^2\|s_\theta(x_t,t)\|^2\Bigg)dt. \tag{22}$$

Returning to the energy formulation, eq. (11) tells us the energy field $E_t(x_t)$ at time t is proportional to the log marginal probability, $\log p(x_t,t)$, plus the time-independent constant Z, meaning its time derivative is simply

$$\frac{\partial E_t(x_t)}{\partial t} = -\beta^{-1}\frac{\partial \log p(x_t,t)}{\partial t} \approx -\beta^{-1}\Bigg(\frac{1}{2}g(t)^2\mathrm{Tr}\Big(\nabla_x s_\theta(x_t,t)\Big) + \frac{1}{2}g(t)^2\|s_\theta(x_t,t)\|^2\Bigg). \tag{23}$$

This completes the energy integral as well:

$$\Delta E \approx -\beta^{-1}\int_0^1 \Bigg(s_\theta(x_t,t)\cdot \frac{\partial x}{\partial t} + \frac{1}{2}g(t)^2\mathrm{Tr}\Big(\nabla_x s_\theta(x_t,t)\Big) + \frac{1}{2}g(t)^2\|s_\theta(x_t,t)\|^2\Bigg)dt \approx= -\beta^{-1}\Delta \log p. \tag{24}$$

In other words: using either the physical intuition or the underlying probability produces a line integral which evaluates the change in learned energy (log likelihood) over some path x_t. Thus, if we want to compute the learned energy (likelihood) of a point x_0 in the (unknown) data distribution, it is sufficient to construct a path starting at x_0 at $t = 0$ and ending at some point x_1 at t_1 in the (known) fully noised distribution. Then, using our integral, we can find the log likelihood of x_0 as

$$\log p_0(x_0) = \log p_1(x_1) - \int_0^1 \Bigg(s_\theta(x_t,t)\cdot \frac{dx_t}{dt} + \frac{1}{2}g(t)^2\mathrm{Tr}\Big(\nabla_x s_\theta(x_t,t)\Big) + \frac{1}{2}g(t)^2\|s_\theta(x_t,t)\|^2\Bigg)dt. \tag{25}$$

The negative sign on the integral stems from the fact that we are integrating from $t = 0$ (data) to $t = 1$ (noise), while the relevant delta is from noise to data. Later, we will be integrating over an ODE trajectory, where the data endpoint is known but the noised endpoint is not, so we ensure consistency by always integrating from $t = 0$ to $t = 1$.

F. DiffLikelihood: Extracting Log-Likelihood by Integrating Over Sampled Diffusion Paths

The most natural choice of path over which to evaluate our integral is the reverse diffusion trajectory used to generate a given sample, as calculated during model inference. These trajectories are computed by evaluating the reverse SDE equation (4) and using the diffusion model's output $s_\theta(x_t,t)$ to approximate $\nabla_x \log p(x_t,t)$, resulting in a discrete trajectory of points in data-space x_{t_i} and time t_i. For numerical accuracy while only sampling points on this discrete trajectory, we use the trapezoid rule of numerical integration:

$$\int_a^b f(x)dx \approx (b-a)\frac{f(b)-f(a)}{2}. \tag{26}$$

Applying this to each time step from (x_{t_i}, t_i) to $(x_{t_{i+1}}, t_{i+1})$ produces a discrete equation for $\log p_0(x_0)$

$$\begin{aligned}\log p_0(x_0) \approx \log p_1(x_1) \\ - \sum_{i=1}^{n-1} \frac{1}{2}\Big(s_\theta(x_{t_i}, t_i) + s_\theta(x_{t_{i+1}}, t_{i+1})\Big) \cdot \Delta x_{t_i} \\ + \frac{1}{4}\Big(g(t_i)^2 \mathrm{Tr}(\nabla_x s_\theta(x_{t_i}, t_i)) \\ + g(t_i)^2 \|s_\theta(x_{t_i}, t_i)\|^2 \\ + g(t_{i+1})^2 \mathrm{Tr}(\nabla_x s_\theta(x_{t_{i+1}}, t_{i+1})) \\ + g(t_{i+1})^2 \|s_\theta(x_{t_{i+1}}, t_{i+1})\|^2\Big) \Delta t_i,\end{aligned}$$

where $\Delta x_{t_i} = x_{t_{i+1}} - x_{t_i}$ and $\Delta t_i = t_{i+1} - t_i$.

The reverse diffusion path is convenient because it allows likelihood integration concurrent with inference by reusing the score computed by the network. Likewise, the trapezoid rule is the highest order numerical method that doesn't require sampling additional intermediate points (and thus additional evaluations of the score model). By using the trapezoid rule once per pair of vertices, we effectively approximate the integral of a piecewise-differentiable path of line segments. If need be, one could integrate each segment more accurately by further interpolating to intelligently sample the score at more points along the trajectory. We find this extra precision does not significantly change the results for either of our test cases, implying that using the trapezoid rule achieves sufficient numerical accuracy along each segment (data not shown).

Despite this, we will see the noise term in eq. (4) produces paths with large displacements Δx_t (**Fig. 2a**) and noisy likelihoods (**Fig. 3a**), motivating us to explore integration over other paths.

G. FlowLikelihood: Extracting Log-Likelihood by Integrating over Flow-Equivalent ODEs

Since $\log p(x_t, t)$ is a state function, we are free to integrate over arbitrary paths with the same endpoints. A less noisy alternative to the diffusion trajectories is to use a flow-equivalent ODE as described by [11], where paths are generated from the following ODE:

$$\frac{dx_t}{dt} = -\frac{1}{2} g(t)^2 \nabla_x \log p(x_t, t) \approx -\frac{1}{2} g(t)^2 s_\theta(x_t, t). \quad (27)$$

The likelihood integral in eq. (25) is then evaluated over these flow trajectories from $t = 0$ to $t = 1$ with initial condition x_0, the point in the data distribution whose learned likelihood we seek.

The explicit description of the path in terms of an ODE confers two major benefits over a discretely sampled diffusion trajectory: First, both path and integrand can be computed using existing black-box ODE solvers, rather than the limited trapezoidal method used over the diffusion trajectory. Second, since we have an explicit form for dx_t, we can substitute it directly into our integrand's dx_t term. When we do so, the first and third terms cancel (see Supplementary Material C), leaving us with the much simplified:

$$\begin{aligned}\log p_0(x_0) &= \log p_1(x_1) - \int_0^1 \frac{1}{2} g(t)^2 \Big(\nabla_x \cdot \nabla_x \log p(x_t, t)\Big) dt \\ &\approx \log p_1(x_1) - \int_0^1 \frac{1}{2} g(t)^2 \mathrm{Tr}\Big(\nabla_x s_\theta(x_t, t)\Big) dt.\end{aligned} \quad (28)$$

Interestingly, this result also follows from interpreting the process of 1) sampling from the prior and 2) following the probability-flow ODE as a continuous normalizing flow model as introduced by Chen *et al.* in [38]; Song *et al.* [11] used this derivation path when first introducing likelihood calculation for diffusion models.

Integrating over the flow trajectory simplifies the integral and provides a smoother, easier to integrate path than the noisy, discrete diffusion trajectory. However, it comes at the computational cost of having to generate the path for each sample. Unlike diffusion trajectory integration, since a flow trajectory can be computed for any point in the data space, we are no longer limited to assessing the likelihood of points generated during inference.

Pseudocode algorithms for integration over diffusion (**Supp. Algorithm 1**) and flow (**Supp. Algorithm 2**) trajectories can be found in Supplementary Material D.

IV. RESULTS

A. Temporal Analysis of Diffusion Trajectories from 1D Diffusion Model

To validate our theoretical analysis, we trained a 1D diffusion model to learn a simple Gaussian mixture model (see Section VI for training procedure), generated samples from our learned model, and evaluated the recovery of learned energy over both reverse diffusion and flow trajectories.

Fig. 2a shows four sampled reverse diffusion trajectories. The diffusion trajectories from the model are noisy at early times ($t = 1$) and become progressively smoother as the model gradually transforms the points to the data distribution ($t \to 0$) and the amount of added noise per diffusion step decreases. By solving the probability flow ODE (eq. 27), we also compute and show the flow paths that result in the same sample point at $t = 0$ (dashed lines). **Fig. 2b** shows the learned score function $s_\theta(x_t, t) = \nabla_x \log p(x_t, t)$ along those four diffusion trajectories. The score has low magnitude near $t = 1$ which indicates low confidence of steering towards the high-likelihood regions of the data. As time decreases, the magnitude of the score increases, and the model shows stronger guidance for x_t towards high-likelihood regions of the data distribution.

In **Fig. 2c**, we visualize the scaled score-term used in the reverse SDE, $\tilde{s}_\theta(x_t, t) = -\frac{1}{2}g(t)^2 \nabla_x \log p(x_t, t)$, where $g(t)$ is the noise-schedule (diffusion coefficient). As we use an exponential noise schedule $g(t) = \sigma_0 \left(\frac{\sigma_1}{\sigma_0}\right)^t$ which exhibits strong weighting, $\tilde{s}_\theta(x_t, t)$ shows large fluctuations near $t = 1$, and its magnitude decays as $t \to 0$. To better interpret how the system evolves from noise to data, we use the ground truth distribution of the trimodal Gaussian to plot $-\log p_0^{GT}(x_t)$ evaluated over the 1000 reverse diffusion steps (**Fig. 2d**). The ground truth energies are high and noisy at early times, but the fluctuations gradually decay and converge to a stable value as x_t approaches the high-likelihood regions of the data distribution $p_0(x_0)$.

B. Integration Over Diffusion and Flow Trajectories Recover Probabilities that Approximate the True Data Distribution

In **Fig. 3a-b**, we plot the 1D system ground truth (training) distribution $p_0(x_0)$ (dashed black curve) and the prior distribution $p_1(x_1)$ (dashed gray curve). To check the 1D diffusion model's ability to generate samples from the data distribution, we plot the kernel density estimate (KDE) of 10,000 sample data points generated from the model (blue curve), showing that the model has learned the ground truth distribution well (Kolmogorov-Smirnov distance of 0.012).

Next, we examined the energies recovered from our integral formulations. We computed negative log-likelihood values (NLL) of samples generated by our Gaussian mixture diffusion model by integrating over both diffusion and flow trajectories. To evaluate the quality of the NLL values, we plot the recovered probabilities $p_0(x_0) = e^{-E_0(x_0)}$ (red dots in **Fig. 3a-b**). The distribution recovered by integrating over diffusion trajectories shows noisy deviation from the training data distribution, but the binned average probability (purple line) matches the ground-truth training distribution almost exactly. In contrast, integrating over flow trajectories yields a noise-free smooth curve that closely follows the ground-truth distribution. Flow-integrated likelihoods demonstrate consistent skew error near the central peak, something not seen when integrating over diffusion trajectories.

To quantify the consistency between the integrals, **Fig. 3** shows scatter plots of the recovered probability versus ground truth probability for the three integral methods (right panels). The Pearson correlation coefficients ($r_p = 0.970$ for diffusion and $r_p = 0.979$ for flow) indicate that learned likelihoods from both diffusion and flow trajectories approximate the underlying data distribution well.

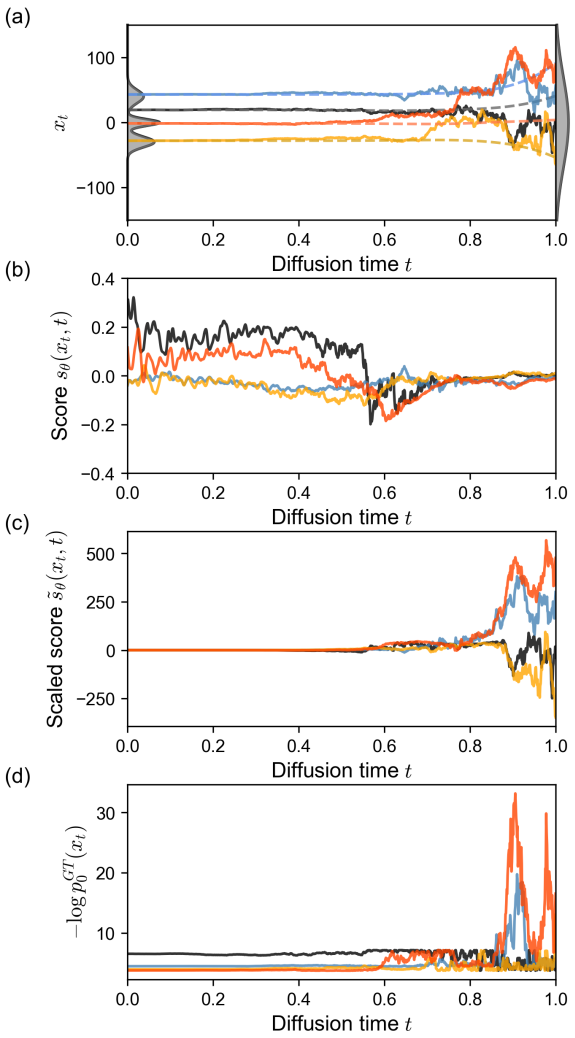

FIG. 2. Trajectories for four individual 1D samples. (a) Diffusion trajectories x_t from the Gaussian mixture diffusion model. Flow paths that result in the same ending points as the diffusion paths are shown as dashed lines. Ground-truth distribution (left) and prior distribution (right) shown in gray. (b) The learned score function $s_\theta(x_t, t) = \nabla_x \log p(x_t, t)$. (c) The scaled score function $\tilde{s}_\theta(x_t, t) = -\frac{1}{2}g(t)^2 \nabla_x \log p(x_t, t)$ with the exponential noise schedule $g(t) = \sigma_0 \left(\frac{\sigma_1}{\sigma_0}\right)^t$. (d) Ground truth energy $-\log p_0^{GT}(x_t)$ evaluated over 1000 diffusion steps from $t = 1$ to $t = 0$.

C. Temporal Analysis of Protein Docking Diffusion Trajectories from DFMDock

Following the 1D diffusion model analysis, we now apply the same approaches to a 3D, rigid-body, translational diffusion model for protein docking, where the diffusion process occurs over $x_t = (x, y, z)$ in 3D Euclidean space. We performed our analysis on the DFMDock model [20], which learns to dock protein complexes given two unbound protein monomers. We limited the docking space to translation only (see Section VI), as the Fokker-Planck

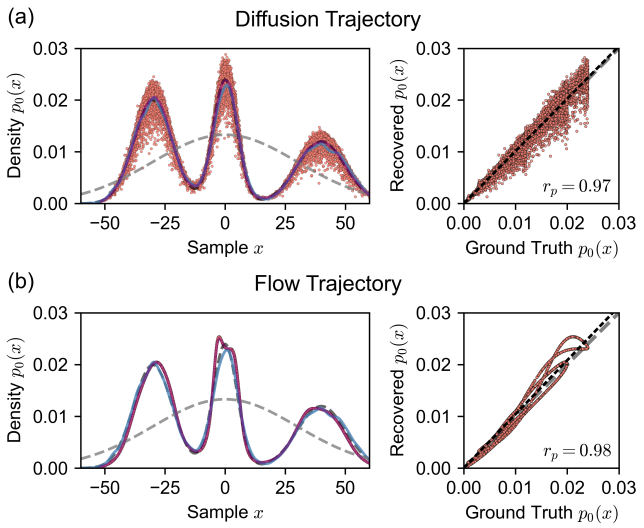

FIG. 3. Recovered probabilities $p_0(x_0) = e^{-\beta E_0(x_0)}$ for samples generated from a 1D Gaussian mixture diffusion model integrated over (a) diffusion trajectories and (b) flow trajectories (red dots, left panels). For the diffusion trajectories in (a), the binned mean of recovered probabilities is shown as a purple curve. The dashed black curve represents the data distribution $p_0(x_0)$, the dashed gray curve shows the unimodal Gaussian distribution (the prior) $p_1(x_1)$, and the blue curve is a kernel density estimate of samples ($N = 10,000$) generated from the diffusion model. The right panels show correlation plots of recovered probability values from both trajectories versus the ground truth probability values. A $y = x$ line is shown in gray dashes, and a dotted linear model is shown in black with Pearson coefficients r_p.

equation used assumes $\mathbb{R}^n$ Euclidean space and is incompatible with general Riemannian manifolds like the SE(3) space used in rotational diffusion. We performed DFMDock inference on 25 targets from Docking Benchmark 5.5 (DB5.5) [43, 44] with 120 samples for each target.

We first examined the score and the learned energies over the 40-step reverse diffusion trajectories starting from the unbound monomers at $t = 1$ to the bound complex at $t = 0$ for two example targets. As an example of a complex for which DFMDock generated accurate samples, we selected the complex of subtilisin BPN' with an inhibitor (PDB: 2SIC). DFMDock-generated samples of docked 2SIC structures exhibited DockQ scores of up to 0.96 (high quality). As an example of a complex for which DFMDock generated diverse but low-accuracy structures, we chose PKR kinase domain-eIF2α (PDB: 2A1A); the best generated docked 2A1A structure only reached a DockQ score of 0.71, a medium quality ranking. Two denoising trajectories from $t = 1$ to $t = 0$ representing the most successful docking attempts for each target are shown in **Fig. 4a,e**. Similar to 1D cases, we observe increasing score $s_\theta(x_t, t)$ (**Fig. 4b,f**) and decreasing scaled score $\tilde{s}_\theta(x_t, t)$ (**Fig. 4c,g**) moving towards $t = 0$.

As flow trajectories can be used to retrieve the energy of a pose without needing its diffusion trajectory, we used them to compute the learned energy, $-\log p_0(x_t)$, of intermediate poses x_t under the learned data distribution p_0 (**Fig. 4d,f**). For the 2SIC trajectories, the learned energy shows a general downward trend over the course of 40 diffusion steps, suggesting that the reverse diffusion process increasingly favored more energetically plausible protein-protein interactions learned by the model. This decrease in energy can be correlated with improved docking quality as the model samples more refined conformations. The 2A1A trajectories, on the other hand, display deep and sharp oscillations and few general trends as the diffusion time approaches zero, notably *not* decreasing as time goes to zero.

D. Learned Energies Compare to Rosetta Energy for Scoring Protein-Protein Interactions

We then computed the learned energy $-\log p_0(x_0)$ of 120 DFMDock generated structures for all 25 protein-protein complexes. As in the 1D case, we extracted potentials for each sample by integrating eqs. (25) and (28) over diffusion and flow trajectories respectively (**Supp. Fig. S1-S4**). To analyze whether these potentials are low for near-native structures, we used two measures of similarity to the ground truth structures, interface-residue RMSD (**Supp. Fig. S1, S3**) and DockQ (a composite measure for docking quality, **Supp. Fig. S2, S4**).

Given that the 1D case showed the learned energy recovered from diffusion trajectories were noisy, we focus on the learned docking energy recovered from flow trajectories. A comparison of the recovery of DFMDock learned energies over flow and diffusion trajectories can be found in **Supp. Fig. S9**. Additionally, we calculated the interface energies using Rosetta, a leading physics-inspired biomolecular modeling package, for all 25 targets (**Supp. Figs. S5, S6**). Here, we focus on our analysis of the learned energies for the same two example targets analyzed in Section IV.C, 2SIC and 2A1A (**Fig. 5**).

For both 2SIC and 2A1A, the Rosetta energy reveals a funnel-like curve with the global minimum near the ground truth (native) structure. The learned energy for 2SIC shares a near-native global minimum (**Fig. 5a**). DFMDock's 2SIC energy landscape seems to have a sharp funnel centered near the native structure, with a relatively flat distribution outside of it. That is, many samples lie at approximately the same non-minimal energy ($\beta E_0 \approx 4$), including several samples ranked well by DockQ or interface RMSD. The inability to distinguish between medium- and low-quality structures might indicate that DFMDock has learned little about physical long-range interactions that might guide a protein towards its docked conformation, instead learning that structures without proper short-range interactions are unlikely to be native.

In the case of 2A1A, the DFMDock learned energies fail to capture the correct binding energetics, with the minimum-energy samples largely being incorrect (iRMSD ≈ 20, DockQ ≈ 0), while Rosetta energy correctly ranks

near-native poses best. The contrast between DFMDock's ability to sample high-quality poses and its inability to properly rank poses with its energy for certain targets indicates that the DFMDock learned energy does not reflect the physical principles of protein-protein interactions in all cases. When the learned energy of the ground truth poses (**Fig. 5**, **Supp. Fig. S3, S4**, yellow stars) are compared to those of the generated poses, they are often higher than the best docking pose and fall within two standard deviations of lower quality ones. The high learned energies of ground truth poses suggests that DFMDock has not sufficiently learned to capture protein-protein interactions.

We next explored whether the funnel-like behavior and the orthogonality of the DFMDock learned energy is useful for ranking docking poses. **Fig. 6a** compares the top ranked model quality (DockQ) for the 25 protein-protein complexes with the learned energy, Rosetta energy, and an oracle setting (where we rank poses based on their DockQ score relative to the ground truth pose). The learned energy performs comparably or outperforms Rosetta interface energy in 6 out of 25 cases in identifying correct docking poses among the sampled poses. **Fig. 6b,c** shows a structural comparison between the learned energy and Rosetta interface energy's top-ranked predictions for two targets. For 2SIC, both the learned energy and Rosetta interface energy identify (distinct) high quality poses, with DockQ = 0.91 in each case. However, for 2A1A, the learned energy fails to identify even an acceptable quality structure (DockQ = 0.01) while Rosetta energy identifies a medium quality pose (DockQ = 0.71).

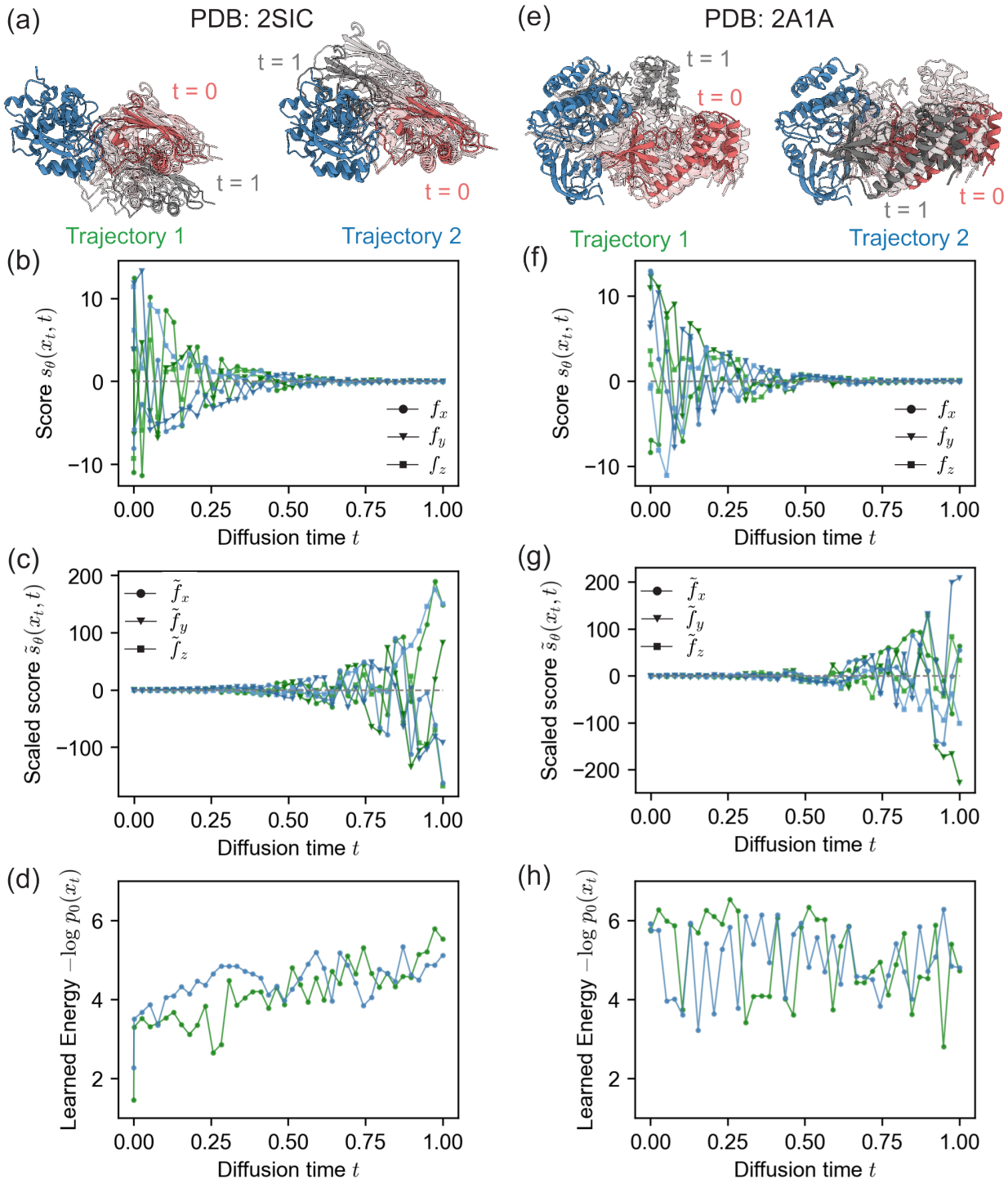

FIG. 4. DFMDock reverse diffusion trajectories for two successful docking samples of two protein complexes, (a-d) 2SIC and (e-h) 2A1A. (a,e) Top two DockQ structures. Receptor protein, blue; ligand protein, gray ($t = 1$) to red ($t = 0$). (b,f) Score function learned by DFMDock $s_\theta(x_t, t) = \nabla_x \log p(x_t, t)$ along x, y, and z directions. (c,g) Scaled score function $\tilde{s}_\theta(x_t, t) = -\frac{1}{2} g(t)^2 \nabla_x \log p(x_t, t)$ with $g(t) = \sigma_0 \left(\frac{\sigma_1}{\sigma_0}\right)^t$ along x, y, and z directions. (d,h) Learned energy, $-\log p_0(x_t)$, recovered from integrating flow trajectories terminating at each point along the reverse diffusion trajectory.

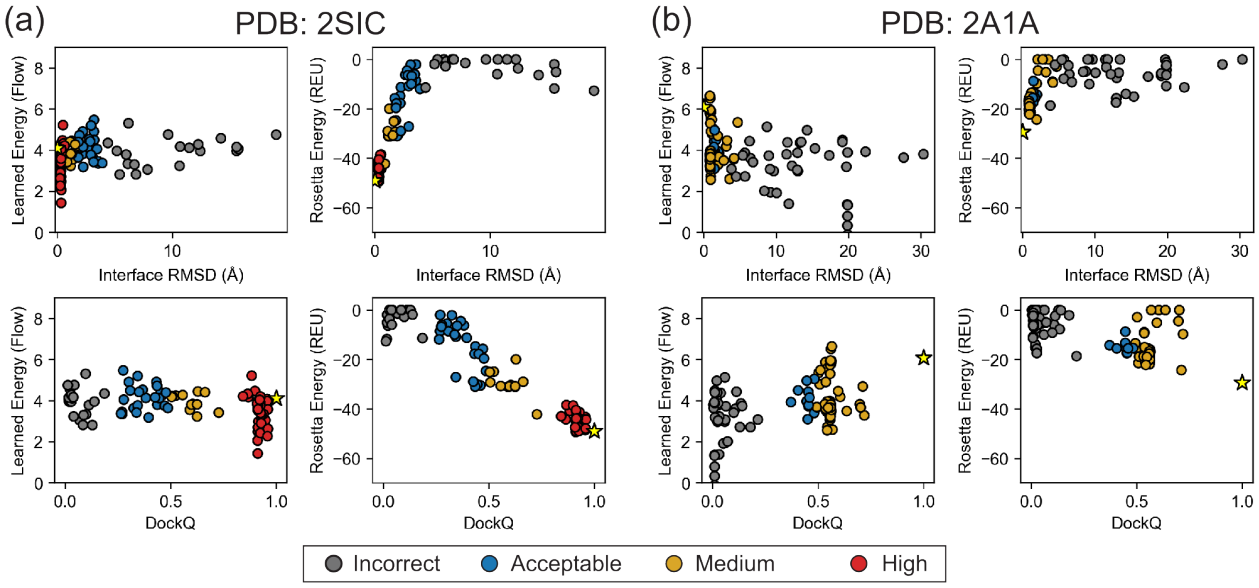

FIG. 5. Learned energy (recovered from integrating over flow trajectories) and Rosetta energy of 120 generated docking poses plotted against interface RMSD (top) or DockQ (bottom) for (a) PDB ID: 2SIC and (b) PDB ID: 2A1A. Individual points are colored by the quality of the corresponding docking pose based on the CAPRI classification [45]: incorrect, gray; acceptable, blue; medium quality, gold; high quality, red. Energies computed for the ground truth structures are shown as yellow stars.

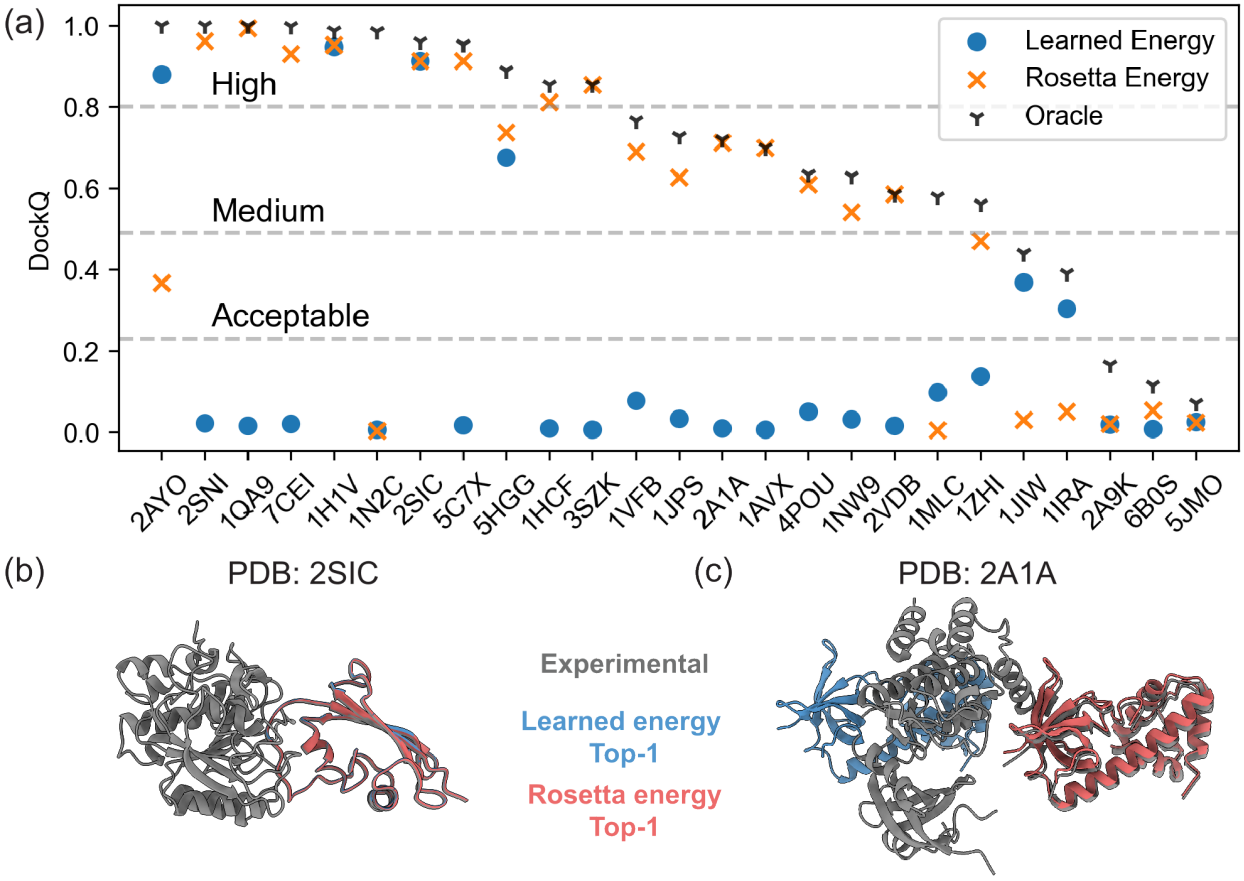

FIG. 6. (a) Comparison of top-ranked model quality (DockQ) for 25 targets from the DB5.5 test set with learned energy over flow trajectories (blue circles), Rosetta energy (orange crosses), or in the oracle setting (black triangle). Top predictions ranked by the learned energy (blue) and Rosetta energy (red) for (b) 2SIC and (c) 2A1A.

V. DISCUSSION AND CONCLUSION

The goal of this work was to interpret diffusion models through the lens of statistical thermodynamics by analyzing the underlying learned potential function (at $t = 0$) and exploring its applications in scoring protein complexes. We developed theory rooted in statistical thermodynamics to relate the probability of observing a system in a particular state to the energy of that state, the inverse temperature β. We defined the energy function that a diffusion model implicitly learns as the *learned energy* and showed that it is equal to the negative log-likelihood, $-\log p_0(x_0)$, up to a constant reference energy $\log Z$.

Since diffusion models do not explicitly learn the data distribution $p_0(x_0)$, but rather the score function, $\nabla_x \log p(x_t, t)$, we developed methods to evaluate $p_0(x_0)$. We postulated that to recover $p_0(x_0)$, we can integrate along a path from the known prior distribution $p_1(x_1)$ to the target data distribution, $p_0(x_0)$. We constructed the path integral over x_t to calculate the likelihood, and evaluated it over two paths: (1) discretized diffusion trajectories from the reverse SDE, and (2) flow trajectories, which follow a smooth, deterministic path defined by an ODE equivalent in marginal probability to the reverse SDE.

We initially tested our approach on a simple 1D diffusion model of a Gaussian mixture and evaluated $-\log p_0(x_0)$ and $p_0(x_0)$ values of samples generated by the model using our integral methods and compared them to the known analytical solution. While diffusion trajectory integrals are convenient since they can be calculated from model inference points, their integration results in noisy likelihood recovery, even as smoothed likelihoods well approximate the ground truth distribution. Fortunately, flow trajectories are both smooth and integrable using black-box ODE solvers. Flow trajectories effectively and continuously approximate the probabilities of the true data distribution at the cost of an additional path calculation.

Building on these results, we applied the same methodology to a protein-protein diffusion docking model, DFMDock, which generates protein complex structures given two unbound protein monomer structures. We extracted the learned potentials from DFMDock to score protein complexes and compared them to Rosetta energies. The learned energies from DFMDock-generated docking poses reveal binding energy funnels that sometimes match Rosetta's interface energy funnels in that the near-native structures have lower energies. But in other cases, the learned energy funnels have minima at non-native structures, and the native structures have higher energies. This work shows that we can examine ensembles of diffusion-generated protein structures in a similar manner as in physics-based energy approaches.

An open question in the field of protein structure prediction and design is whether and how AI models might learn a thermodynamic function of protein folding and association. Better interpretability of AI models in biology is also important from a biosecurity perspective. There are other investigations of these questions in the literature. Ahdritz *et al.* [46] released OpenFold, an open-source version of AF2 and carefully investigated model's learning process and the model's capacity to generalize to unseen regions of fold space. Roney *et al.* [47] hypothesized that AF2 has learned an implicit energy function through its confidence module, and demonstrated that AF2 can be used to rank the quality of candidate protein structures without needing coevolution data.

To our knowledge, this work is the first attempt at extracting potentials from a biomolecular diffusion model and comparing them with physical energy functions. An exciting future direction is to apply our methodology developed here on state-of-the-art diffusion-based structure prediction models such as AlphaFold3 [7] or Boltz-2 [14] to interrogate their learned energy functions associated with protein folding stability. Another important direction is to generalize likelihood calculation to be compatible with diffusion models over arbitrary Riemannian manifolds such as the SE(3) space used in the original DFMDock with rotations [20] and frame-based models like DiffDock-PP [26]. As we are not limited only to computing the likelihood of generated samples, exploring the learned energy landscape around generated structures might reveal important details about the shape of the learned energy funnel. For example, whether the structure is at a local minimum could be used as a ranking metric, and learned energy surfaces could be used for local refinement.

While we explored here how physical energies can be accessed from learned diffusion models, another approach is to inject biophysical knowledge into AI models. Kulyte *et al.* [48] used molecular dynamics (MD) force-fields to guide diffusion models for antibody design. Similarly, Wang *et al.* [49] used MD force-fields to guide diffusion models for protein conformation generation. Lewis *et al.* [50] trained a diffusion model on MD conformational ensembles to improve realistic diversity of conformational sampling. Analyses like ours could help probe the effects of adding physical information into deep-learning based biomolecular design and structure prediction tools.

Our method for likelihood calculation integrates the spatiotemporal gradient of the marginal probability $p(x_t, t)$ over two paths: the probability flow ODE, and a discrete sampling of the reverse-time SDE. Interestingly, this gradient is (by definition) a conservative vector field over the combined diffusion space $\mathbb{R}^{n+1}$ (combining data-space $x_t \in \mathbb{R}^n$ and time $t \in \mathbb{R}$), which means that the integral of this gradient, $\Delta \log p$, should be dependent solely on the endpoints of the path, not on the trajectory we use for integration. Since we have an explicit formula for $\log p_1$, this means the calculated value of $\log p_0 = \log p_1 - \Delta \log p$ should be the same for any path we use during integration.

In practice, however, we find this is not the case: flow trajectories do not yield the same value as diffusion trajectories, and different diffusion paths terminating at near-identical samples can have significantly different likelihoods. There is evidence that diffusion models do

not generally learn spatially-conservative [51] or Fokker-Planck consistent [52] scores, and so the accuracy of the score may vary significantly depending on the similarity of a sample to data seen during training. There may be further insight to be gained by exploring the conservativity in the score learned by networks like DFMDock and AlphaFold and the impacts inaccuracies have on different trajectories and methods of likelihood computation.

VI. DATA

A. Gaussian Mixture Model Training

To train a 1D diffusion model to learn a simple Gaussian mixture distribution, we generated 60,000 training data points from a trimodal Gaussian, defined as a weighted sum of three individual Gaussian components. The probability density function is

$$P_0(x) = w_1 \cdot \mathcal{N}(x; \mu_1, b_1^2) + w_2 \cdot \mathcal{N}(x; \mu_2, b_2^2) + w_3 \cdot \mathcal{N}(x; \mu_3, b_3^2), \tag{29}$$

where the means are $\mu_1 = -30$, $\mu_2 = 0$, and $\mu_3 = 40$; the standard deviations are $b_1 = 8.0$, $b_2 = 5.0$, and $b_3 = 10.0$; and the mixture weights are $w_1 = 0.4$, $w_2 = 0.3$, and $w_3 = 0.3$. For testing, we sampled 10,000 synthetic points with 1,000 diffusion steps from the learned model and compared the probability density function of the analytical formulation with the kernel density estimate of the generated samples (**Fig. 3**).

B. DFMDock Inference

We performed inference using the DFMDock model [20] which is trained on DIPS-hetero [53], a subset of DIPS with approximately 11,000 heterodimers. Although DFMDock was originally designed to reverse both translational and rotational noise, here we restricted the added rigid-body noise during both training and inference to be translation-only, such that all processes occur in the 3D Euclidean space $\mathbb{R}^3$. Additionally, the original formulation of DFMDock uses a random sampling method to construct the interaction graph as input to the neural network, whereas here the graph is generated deterministically using k-nearest neighbors. This ensures the stochasticity during inference comes solely from added perturbations. Using the same training parameters as [20], we retrained the model with these modifications and observed slightly degraded performance. As in [20], we sampled 120 poses with 40 diffusion steps on 25 targets from the Docking Benchmark 5.5 (DB5.5) [43, 44], a widely used dataset for assessing docking performance. For Rosetta energies computed for the sampled poses, we first performed the `local_docking_refine` protocol before computing the I_{sc} score with the REF15 energy function.

ACKNOWLEDGEMENTS

This work was supported by National Institutes of Health grant R35-GM141881 and by Moderna. Computational resources were provided by the Advanced Research Computing at Hopkins (ARCH). The authors thank Jeremias Sulam and Jacopo Teneggi for suggesting that we explore an analytic, 1D model to gain better insight into our DockQ energies.

CODE AVAILABILITY

The training, inference, and likelihood evaluation code for the 1D diffusion model and the likelihood evaluation code for DFMDock are available at `https://github.com/Graylab/DiffEnergy`.

[1] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, arXiv preprint arXiv:2102.12092 (2021).
[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, arXiv preprint arXiv:2112.10752 (2022).
[3] A. Borji, arXiv preprint arXiv:2210.00586 (2022).
[4] Y. Liu, K. Zhang, Y. Li, Z. Yan, C. Gao, R. Chen, Z. Yuan, Y. Huang, H. Sun, J. Gao, L. He, and L. Sun, arXiv preprint arXiv:2402.17177 (2024).
[5] J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, N. Hanikel, S. J. Pellock, A. Courbet, W. Sheffler, J. Wang, P. Venkatesh, I. Sappington, S. V. Torres, A. Lauko, V. De Bortoli, E. Mathieu, S. Ovchinnikov, R. Barzilay, T. S. Jaakkola, F. DiMaio, M. Baek, and D. Baker, Nature **620**, 10.1038/s41586-023-06415-8 (2023).
[6] J. B. Ingraham, M. Baranov, Z. Costello, K. W. Barber, W. Wang, A. Ismail, V. Frappier, D. M. Lord, C. Ng-Thow-Hing, E. R. Van Vlack, S. Tie, V. Xue, S. C. Cowles, A. Leung, J. V. Rodrigues, C. L. Morales-Perez, A. M. Ayoub, R. Green, K. Puentes, F. Oplinger, N. V. Panwar, F. Obermeyer, A. R. Root, A. L. Beam, F. J. Poelwijk, and G. Grigoryan, Nature **623**, 1070–1078 (2023).
[7] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, S. W. Bodenstein, D. A. Evans, C.-C. Hung, M. O'Neill, D. Reiman, K. Tunyasuvunakool, Z. Wu, A. Žemgulytė, E. Arvaniti, C. Beattie, O. Bertolli, A. Bridgland, A. Cherepanov, M. Congreve, A. I. Cowen-Rivers, A. Cowie, M. Figurnov, F. B. Fuchs, H. Gladman, R. Jain, Y. A. Khan, C. M. R. Low, K. Perlin, A. Potapenko, P. Savy, S. Singh, A. Stecula, A. Thillaisundaram, C. Tong, S. Yakneen, E. D. Zhong, M. Zielinski, A. Žídek, V. Bapst, P. Kohli, M. Jaderberg, D. Hassabis, and J. M. Jumper, Nature **630**, 10.1038/s41586-024-07487-w (2024).
[8] J. Boitreaud, J. Dent, M. McPartlon, J. Meier, V. Reis, A. Rogozhnikov, and K. Wu, bioRxiv preprint 10.1101/2024.10.10.615955 (2024).
[9] J. Wohlwend, G. Corso, S. Passaro, M. Reveiz, K. Leidal, W. Swiderski, T. Portnoi, I. Chinn, J. Silterra, T. Jaakkola, and R. Barzilay, bioRxiv preprint 10.1101/2024.11.19.624167 (2024).
[10] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, arXiv preprint arXiv:1503.03585 (2015).
[11] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, arXiv preprint arXiv:2011.13456 (2020).
[12] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, Nature **596**, 583 (2021).
[13] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. Jaakkola, arXiv preprint arXiv:2210.01776 (2022).
[14] S. Passaro, G. Corso, J. Wohlwend, M. Reveiz, S. Thaler, V. R. Somnath, N. Getz, T. Portnoi, J. Roy, H. Stark, D. Kwabi-Addo, D. Beaini, T. Jaakkola, and R. Barzilay, bioRxiv 10.1101/2025.06.14.659707 (2025), https://www.biorxiv.org/.
[15] The OpenFold3 Team, Openfold3-preview (2025).
[16] A. Leaver-Fay, M. Tyka, S. M. Lewis, O. F. Lange, J. Thompson, R. Jacak, K. W. Kaufman, P. D. Renfrew, C. A. Smith, W. Sheffler, I. W. Davis, S. Cooper, A. Treuille, D. J. Mandell, F. Richter, Y.-E. A. Ban, S. J. Fleishman, J. E. Corn, D. E. Kim, S. Lyskov, M. Berrondo, S. Mentzer, Z. Popović, J. J. Havranek, J. Karanicolas, R. Das, J. Meiler, T. Kortemme, J. J. Gray, B. Kuhlman, D. Baker, and P. Bradley, Rosetta3, in *Computer Methods, Part C* (Elsevier, 2011) p. 545–574.
[17] R. F. Alford, A. Leaver-Fay, J. R. Jeliazkov, M. J. O'Meara, F. P. DiMaio, H. Park, M. V. Shapovalov, P. D. Renfrew, V. K. Mulligan, K. Kappel, J. W. Labonte, M. S. Pacella, R. Bonneau, P. Bradley, R. L. Dunbrack, R. Das, D. Baker, B. Kuhlman, T. Kortemme, and J. J. Gray, Journal of Chemical Theory and Computation **13**, 3031–3048 (2017).
[18] C. B. Anfinsen, **181**, 223, https://www.science.org/doi/pdf/10.1126/science.181.4096.223.
[19] I. A. Vakser, Biophysical Journal **107**, 1785–1793 (2014).
[20] L.-S. Chu, S. Sarma, and J. J. Gray, bioRxiv 10.1101/2024.09.27.615401 (2024).
[21] D. J. Wales, Annual Review of Physical Chemistry **69**, 401–425 (2018).
[22] S. Sledzieski, R. Singh, L. Cowen, and B. Berger, Cell Systems **12**, 969 (2021).
[23] S.-Y. Huang, Drug Discovery Today **19**, 1081–1096 (2014).
[24] N. A. Marze, S. S. Roy Burman, W. Sheffler, and J. J. Gray, Bioinformatics **34**, 3461–3469 (2018).
[25] G. Lemmon and J. Meiler, Rosetta ligand docking with flexible xml protocols, in *Computational Drug Discovery and Design* (Springer New York, 2011) p. 143–155.
[26] M. A. Ketata, C. Laue, R. Mammadov, H. Stärk, M. Wu, G. Corso, C. Marquet, R. Barzilay, and T. S. Jaakkola, arXiv preprint arXiv:2304.03889 (2023).
[27] F. Sverrisson, M. Akdel, D. Abramson, J. Feydy, A. Goncearenco, Y. Adeshina, D. Kovtun, C. Marquet, X. Zhang, D. Baugher, *et al.*, in *Machine Learning in Structural Biology workshop at NeurIPS 2023* (2023).
[28] M. McPartlon, C. Marquet, T. Geffner, D. Kovtun, A. Goncearenco, Z. Carpenter, L. Naef, M. Bronstein, and J. Xu, MLSB (2023).
[29] Y. Song and D. P. Kingma, arXiv preprint arXiv:2101.03288 (2023).
[30] W. Jin, X. Chen, A. Vetticaden, S. Sarzikova, R. Raychowdhury, C. Uhler, and N. Hacohen, bioRxiv preprint , 2023 (2023).
[31] V. R. Somnath, P. G. Sessa, M. R. Martinez, and A. Krause, arXiv preprint arXiv:2310.06177 (2023).
[32] H. Wu, W. Liu, Y. Bian, J. Wu, N. Yang, and J. Yan, in *The Twelfth International Conference on Learning Representations* (2024).
[33] K. Borisiak, G. M. Visani, and A. Nourmohammad, arXiv preprint arXiv:2409.18201 (2024).
[34] M. Plainer, H. Wu, L. Klein, S. Günnemann, and F. Noé, Consistent sampling and simulation: Molecular dynamics with energy-based diffusion models (2025), arXiv:2506.17139 [cs.LG].
[35] L. Ambrogioni, Entropy **27**, 291 (2025).
[36] A. Sclocchi, A. Favero, and M. Wyart, Proceedings of the National Academy of Sciences **122**, 10.1073/pnas.2408799121 (2025).

[37] G. Biroli, T. Bonnaire, V. de Bortoli, and M. Mézard, Nature Communications **15**, 10.1038/s41467-024-54281-3 (2024).

[38] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud 10.48550/ARXIV.1806.07366 (2018).

[39] D. L. Beveridge and F. M. DiCapua, Annual Review of Biophysics and Biophysical Chemistry **18**, 431–492 (1989).

[40] R. Abel, L. Wang, E. D. Harder, B. J. Berne, and R. A. Friesner, Accounts of Chemical Research **50**, 1625–1632 (2017).

[41] B. D. Anderson, Stochastic Processes and their Applications **12**, 313 (1982).

[42] P. Vincent, Neural Computation **23**, 1661 (2011).

[43] T. Vreven, I. H. Moal, A. Vangone, B. G. Pierce, P. L. Kastritis, M. Torchala, R. Chaleil, B. Jiménez-García, P. A. Bates, J. Fernandez-Recio, A. M. Bonvin, and Z. Weng, Journal of Molecular Biology **427**, 3031 (2015).

[44] J. D. Guest, T. Vreven, J. Zhou, I. Moal, J. R. Jeliazkov, J. J. Gray, Z. Weng, and B. G. Pierce, Structure **29**, 606 (2021).

[45] J. Janin, K. Henrick, J. Moult, L. T. Eyck, M. J. E. Sternberg, S. Vajda, I. Vakser, and S. J. Wodak, Proteins: Structure, Function, and Bioinformatics **52**, 2 (2003).

[46] G. Ahdritz, N. Bouatta, C. Floristean, S. Kadyan, Q. Xia, W. Gerecke, T. J. O'Donnell, D. Berenberg, I. Fisk, N. Zanichelli, B. Zhang, A. Nowaczynski, B. Wang, M. M. Stepniewska-Dziubinska, S. Zhang, A. Ojewole, M. E. Guney, S. Biderman, A. M. Watkins, S. Ra, P. R. Lorenzo, L. Nivon, B. Weitzner, Y.-E. A. Ban, S. Chen, M. Zhang, C. Li, S. L. Song, Y. He, P. K. Sorger, E. Mostaque, Z. Zhang, R. Bonneau, and M. AlQuraishi, Nature Methods **21**, 1514 (2024).

[47] J. P. Roney and S. Ovchinnikov, Physical Review Letters **129**, 10.1103/physrevlett.129.238101 (2022).

[48] P. Kulytė, F. Vargas, S. V. Mathis, Y. G. Wang, J. M. Hernández-Lobato, and P. Liò, Improving antibody design with force-guided sampling in diffusion models, arXiv preprint arXiv:2406.05832v2 (2024), version 2, last revised 9 Sep 2024.

[49] Y. Wang, L. Wang, Y. Shen, Y. Wang, H. Yuan, Y. Wu, and Q. Gu, in *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 235, edited by R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp (PMLR, 2024) pp. 56835–56859.

[50] S. Lewis, T. Hempel, J. Jiménez-Luna, M. Gastegger, Y. Xie, A. Y. Foong, V. G. Satorras, O. Abdin, B. S. Veeling, I. Zaporozhets, Y. Chen, S. Yang, A. E. Foster, A. Schneuing, J. Nigam, F. Barbero, S. Vincent, A. Campbell, J. Yim, M. Lienen, Y. Shi, S. Zheng, H. Schulz, U. Munir, R. Sordillo, R. Tomioka, C. Clementi, and F. Noé, Science , eadv9817 (2025).

[51] T. Salimans and J. Ho, in *Energy Based Models Workshop - ICLR 2021* (2021).

[52] C.-H. Lai, Y. Takida, N. Murata, T. Uesaka, Y. Mitsufuji, and S. Ermon, Fp-diffusion: Improving score-based diffusion models by enforcing the underlying score fokker-planck equation (2023), arXiv:2210.04296 [cs.LG].

[53] A. Morehead, C. Chen, A. Sedova, and J. Cheng, **10**, 509.

[54] B. Øksendal, in *Stochastic Differential Equations: An Introduction with Applications* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2003) pp. 65–84.

SUPPORTING INFORMATION

A. Dimensional Analysis of Key Equations

We perform dimensional analysis on the fundamental equations underlying diffusion (and their physics counterparts) to derive the units for **Table I**; $[M]$ represents units of mass, $[L]$ units of length, $[T]$ units of *physical* time, and $[\tau]$ units of *diffusion/alchemical* time. We begin with the forward diffusion equation, eq. (3):

$$dx_t = f(x_t, t)dt + g(t)dw_t.$$

where x_t represents a point in Euclidean space $\mathbb{R}^n$ and thus has units of length $[L]$. More generally, x_t can be a point on an arbitrary Riemannian manifold; every such manifold has a Riemannian metric, a measure of distance between two points, so units of length still broadly apply. The Wiener term dw_t is sampled from a Gaussian with variance dt, defined in terms of alchemical/diffusion time, meaning its units must be $[\tau]^{1/2}$, the square root of the unit of the variance. The units of the f and g functions can thus be inferred:

$$dx_t \sim [L], \qquad dt \sim [\tau], \qquad dw_t \sim [\tau]^{1/2},$$

$$f(x_t, t) \sim \frac{[L]}{[\tau]}, \qquad \text{“Drift”}$$

$$g(t) \sim \frac{[L]}{[\tau]^{1/2}}. \qquad \text{“Scheduler”}$$

The reverse diffusion equation, eq. (4),

$$dx_t = \left[f(x_t, t) - g(t)^2 \nabla_x \log p(x_t, t)\right] dt + g(t)dw_t$$

introduces the marginal probability $p(x_t, t)$, which is unitless, and reveals the units of $g(t)^2$ and $\nabla_x \log p(x_t, t)$:

$$p(x_t, t) \sim [1],$$

$$g(t)^2 \sim \frac{[L]^2}{[\tau]}, \qquad \text{“Diffusion Coefficient”}$$

$$\nabla_x \log p(x_t, t) \sim \frac{1}{[L]}. \qquad \text{“Score”}$$

Equations (10) and (11) connect the marginal probability to energy:

$$p(x_t, t) = \frac{e^{-\beta E_t(x_t)}}{Z};$$
$$\log p(x_t, t) = -\beta E_t(x_t) - \log Z.$$

These equations tell us $\beta E_t(x_t)$ and Z must be unitless, and the dimensions of energy are defined by physics:

$$Z \sim [1], \quad E_t(x_t) \sim \frac{[M][L]^2}{[T]^2}, \quad \beta \sim \frac{[T]^2}{[M][L]^2},$$

$$-\beta^{-1} \log p(x_t, t) \sim \frac{[M][L]^2}{[T]^2}. \qquad \text{“Free Energy”}$$

Finally, equation (13) connects energy and physics:

$$s_\theta(x_t, t) \approx \nabla_x \log p(x_t, t) = -\beta \nabla_x E_t(x_t),$$

introducing learned score $s_\theta(x_t, t)$ and force $-\nabla_x E_t(x_t)$:

$$s_\theta(x_t, t) \sim \frac{1}{[L]}, \qquad \text{“Learned Score”}$$

$$\nabla_x E_t(x_t) = -\beta^{-1} \nabla_x \log p(x_t, t) \sim \frac{[M][L]}{[T]^2}. \qquad \text{“Force”}$$

For completeness, we also include dimensional analysis of our integral formulations. For eq. (19):

$$\Delta \log p = \int_0^1 \left(\nabla_x \log p(x_t, t) \cdot \frac{\partial x_t}{\partial t} + \frac{\partial \log p(x_t, t)}{\partial t} \right) dt,$$

$$\frac{\partial x_t}{\partial t} \sim \frac{[L]}{[\tau]}, \qquad \frac{\partial \log p(x_t, t)}{\partial t} \sim \frac{[1]}{[\tau]}.$$

And eq. (20):

$$\frac{\partial \log p(x_t, t)}{\partial t} = \frac{1}{2} g(t)^2 \Big(\nabla_x \cdot \nabla_x \log p(x_t, t) \Big) + \frac{1}{2} g(t)^2 \|\nabla_x \log p(x_t, t)\|^2,$$

$$\frac{1}{2} g(t)^2 \Big(\nabla_x \cdot \nabla_x \log p(x_t, t) \Big) \sim \frac{[L]^2}{[\tau]} \cdot \frac{[1]}{[L]^2} = \frac{[1]}{[\tau]},$$

$$\frac{1}{2} g(t)^2 \|\nabla_x \log p(x_t, t)\|^2 \sim \frac{[L]^2}{[\tau]} \cdot \frac{[1]}{[L]^2} = \frac{[1]}{[\tau]}.$$

B. Derivation of Logarithmic Fokker Planck Equation

The Fokker-Planck equation is a partial differential equation (PDE) that describes the time-evolution of the marginal probability distribution $p(x_t, t)$ associated with a given stochastic differential equation (SDE). It has many variants depending on the form of the SDE; for $dx_t = f(x_t, t)dt + g(t)dw_t$ the Fokker-Planck equation is [54]

$$\begin{aligned}\frac{\partial p(x_t,t)}{\partial t} &= -\sum_{i=1}^{n} \frac{\partial}{\partial [x_t]_i}\Big([f(x_t,t)]_i \; p(x_t,t)\Big) \\ &\quad + \frac{1}{2} g(t)^2 \sum_{i=1}^{n} \frac{\partial^2}{\partial [x_t]_i^2} p(x_t,t),\end{aligned} \tag{S-1}$$

where n is the dimensionality of x_t, $[x_t]_i$ is the i-th component of x_t, and $[f(x_t,t)]_i$ is the i-th component of the drift term $f(x_t,t)$.

The Fokker-Planck equation is useful, but it is dependent on the value and spatial gradients of probability $p(x_t,t)$, neither of which we have; rather, our score model is trained to predict the spatial gradients of the log probability. We can convert between the two using the chain rule:

$$\begin{aligned}\frac{\partial \log f(x)}{\partial x_i} &= \frac{d \log f(x)}{df(x)} \frac{\partial f(x)}{\partial x_i} \\ &= \frac{1}{|f(x)|} \frac{\partial f(x)}{\partial x_i}.\end{aligned}$$

Since $p(x_t,t) > 0$,

$$\frac{\partial \log p(x_t,t)}{\partial t} = \frac{1}{p(x_t,t)} \frac{\partial p(x_t,t)}{\partial t}. \tag{S-2}$$

Thus, we can rewrite the Fokker-Planck equation (S-1) in a logarithmic form like so:

$$\begin{aligned}\frac{\partial \log p(x_t,t)}{\partial t} &= \frac{1}{p(x_t,t)} \frac{\partial p(x_t,t)}{\partial t} \\ &= -\frac{1}{p(x_t,t)} \sum_{i=1}^{n} \frac{\partial}{\partial [x_t]_i}\Big([f(x_t,t)]_i \; p(x_t,t)\Big) \\ &\quad + \frac{1}{p(x_t,t)} \frac{1}{2} g(t)^2 \sum_{i=1}^{n} \frac{\partial^2}{\partial [x_t]_i^2} p(x_t,t).\end{aligned} \tag{S-3}$$

This equation is still dependent on derivatives of $p(x_t,t)$, but we can convert them into logarithmic derivatives as well by multiplying by $\frac{1}{p(x_t,t)}$. The first-order term is relatively straightforward; first, we expand the derivative using the product rule:

$$\begin{aligned}\frac{\partial \log p(x_t,t)}{\partial t} &= -\frac{1}{p(x_t,t)} \sum_{i=1}^{n} p(x_t,t) \; \left(\frac{\partial}{\partial [x_t]_i} [f(x_t,t)]_i \right) \\ &\quad - \frac{1}{p(x_t,t)} \sum_{i=1}^{n} [f(x_t,t)]_i \; \left(\frac{\partial}{\partial [x_t]_i} p(x_t,t) \right) \\ &\quad + \frac{1}{p(x_t,t)} \frac{1}{2} g(t)^2 \sum_{i=1}^{n} \frac{\partial^2}{\partial [x_t]_i^2} \left(p(x_t,t) \right).\end{aligned} \tag{S-4}$$

In the first term, $p(x_t,t)$ cancels, and in the second term, the $p(x_t,t)$ derivative becomes logarithmic by eq. (S-2):

$$\begin{aligned}\frac{\partial \log p(x_t,t)}{\partial t} &= -\sum_{i=1}^{n} \frac{\partial}{\partial [x_t]_i} [f(x_t,t)]_i \\ &\quad - \sum_{i=1}^{n} [f(x_t,t)]_i \; \left(\frac{\partial}{\partial [x_t]_i} \log p(x_t,t) \right) \\ &\quad + \frac{1}{p(x_t,t)} \frac{1}{2} g(t)^2 \sum_{i=1}^{n} \frac{\partial^2}{\partial [x_t]_i^2} p(x_t,t).\end{aligned} \tag{S-5}$$

The second-order derivative in the third term requires more work. We can again use eq. (S-2) to derive an identity for logarithmic second derivatives:

$$\begin{aligned}
\frac{\partial^2}{\partial [x_t]_i^2} p(x_t, t) &= \frac{\partial}{\partial [x_t]_i} \left(\frac{\partial}{\partial [x_t]_i} p(x_t, t) \right) \\
&= \frac{\partial}{\partial [x_t]_i} \left(\left(\frac{\partial}{\partial [x_t]_i} \log p(x_t, t) \right) \; p(x_t, t) \right) \\
&= \left(\frac{\partial^2}{\partial [x_t]_i^2} \log p(x_t, t) \right) p(x_t, t) + \left(\frac{\partial}{\partial [x_t]_i} \log p(x_t, t) \right) \left(\frac{\partial}{\partial [x_t]_i} p(x_t, t) \right) \\
&= \left(\frac{\partial^2}{\partial [x_t]_i^2} \log p(x_t, t) \right) p(x_t, t) + \left(\frac{\partial}{\partial [x_t]_i} \log p(x_t, t) \right) \left(\frac{\partial}{\partial [x_t]_i} \log p(x_t, t) \right) p(x_t, t) \\
&= \left(\frac{\partial^2}{\partial [x_t]_i^2} \log p(x_t, t) \right) p(x_t, t) + \left(\frac{\partial}{\partial [x_t]_i} \log p(x_t, t) \right)^2 p(x_t, t). \qquad \text{(S-6)}
\end{aligned}$$

Substituting this back into equation (S-5), the problematic term splits into a second-order logarithmic term and a squared first-order term, and the final remaining $p(x_t, t)$ coefficients cancel:

$$\begin{aligned}
\frac{\partial \log p(x_t, t)}{\partial t} &= - \sum_{i=1}^{n} \frac{\partial}{\partial [x_t]_i} [f(x_t, t)]_i \\
&\quad - \sum_{i=1}^{n} [f(x_t, t)]_i \left(\frac{\partial}{\partial [x_t]_i} \log p(x_t, t) \right) \\
&\quad + \frac{1}{2} g(t)^2 \sum_{i=1}^{n} \frac{\partial^2}{\partial [x_t]_i^2} \log p(x_t, t) \\
&\quad + \frac{1}{2} g(t)^2 \sum_{i=1}^{n} \left(\frac{\partial}{\partial [x_t]_i} \log p(x_t, t) \right)^2, \qquad \text{(S-7)}
\end{aligned}$$

giving us a form of the logarithmic Fokker-Planck equation that we can evaluate using only a trained score model. These sums can also be written using vector notation:

$$\begin{aligned}
\frac{\partial \log p(x_t, t)}{\partial t} &= -\nabla_x \cdot f(x_t, t) \\
&\quad - f(x_t, t) \cdot \nabla_x \log p(x_t, t) \\
&\quad + \frac{1}{2} g(t)^2 \nabla_x \cdot \nabla_x \log p(x_t, t) \\
&\quad + \frac{1}{2} g(t)^2 \|\nabla_x \log p(x_t, t)\|^2. \qquad \text{(S-8)}
\end{aligned}$$

Finally, since our models are variance-exploding, their drift term $f(x_t, t) = 0$, simplifying to eq. (20) in the main text:

$$\begin{aligned}
\frac{\partial \log p(x_t, t)}{\partial t} &= \frac{1}{2} g(t)^2 \Big(\nabla_x \cdot \nabla_x \log p(x_t, t) \Big) \\
&\quad + \frac{1}{2} g(t)^2 \|\nabla_x \log p(x_t, t)\|^2. \qquad (20)
\end{aligned}$$

B.1 Reverse-Time Fokker-Planck

Technically, the process whose marginal distribution we seek is the *reverse* diffusion process, eq. (4):

$$dx_t = [f(x_t, t) - g(t)^2 \nabla_x \log p(x_t, t)]dt + g(t) d\overline{w}_t. \qquad (4)$$

As mentioned earlier, the reverse diffusion process is carefully chosen such that its marginal distribution matches that of the forward process, and therefore the Fokker-Planck equations should be identical for both (in fact, Anderson (1982) [41] derived the reverse SDE precisely by matching the Fokker-Planck for the forward and backward equations). For thoroughness, we will also derive the marginal distribution for the reverse process.

The most important thing to keep track of is the direction of time and the sign of its change. The Fokker-Planck equation assumes a positive dt and a forward-time Wiener process dw_t, but the reverse SDE involves a negative dt and a reverse-time Wiener process $d\overline{w}_t$. To adjust, we perform change-of-variables on t to produce an equivalent forward-time process in the variable s; by replacing t in eq. (4) with $1-s$, changing the sign of the first term to account for replacing $dt < 0$ with $ds > 0$, and using the forward-time Wiener term:

$$dx_{1-s} = [-f(x_{1-s}, 1-s) + g(1-s)^2 \nabla_x \log p(x_{1-s}, 1-s)]ds + g(1-s)dw_s. \tag{S-9}$$

The logarithmic Fokker-Planck equation for this SDE then describes the change in the reverse marginal probability, $\overline{p}$, with respect to s:

$$\begin{aligned}
\frac{\partial \log \overline{p}(x_{1-s}, 1-s)}{\partial s} &= -\nabla_x \cdot [-f(x_{1-s}, 1-s) + g(1-s)^2 \nabla_x \log p(x_{1-s}, 1-s)] \\
&- [-f(x_{1-s}, 1-s) + g(1-s)^2 \nabla_x \log p(x_{1-s}, 1-s)] \cdot \nabla_x \log \overline{p}(x_{1-s}, 1-s) \\
&+ \frac{1}{2} g(1-s)^2 \nabla_x \cdot \nabla_x \log \overline{p}(x_{1-s}, 1-s) \\
&+ \frac{1}{2} g(1-s)^2 \|\nabla_x \log \overline{p}(x_{1-s}, 1-s)\|^2 \\
&= \nabla_x \cdot f(x_{1-s}, 1-s) - g(1-s)^2 \nabla_x \cdot \nabla_x \log p(x_{1-s}, 1-s) \\
&+ f(x_{1-s}, 1-s) \cdot \nabla_x \log \overline{p}(x_{1-s}, 1-s) - g(1-s)^2 \nabla_x \log p(x_{1-s}, 1-s) \cdot \nabla_x \log \overline{p}(x_{1-s}, 1-s) \\
&+ \frac{1}{2} g(1-s)^2 \nabla_x \cdot \nabla_x \log \overline{p}(x_{1-s}, 1-s) \\
&+ \frac{1}{2} g(1-s)^2 \|\nabla_x \log \overline{p}(x_{1-s}, 1-s)\|^2 \\
&= \nabla_x \cdot f(x_{1-s}, 1-s) \\
&+ f(x_{1-s}, 1-s) \cdot \nabla_x \log \overline{p}(x_{1-s}, 1-s) \\
&+ g(1-s)^2 \nabla_x \cdot \nabla_x \left(\frac{1}{2} \log p(x_{1-s}, 1-s) - \log \overline{p}(x_{1-s}, 1-s) \right) \\
&+ g(1-s)^2 \nabla_x \log \overline{p}(x_{1-s}, 1-s) \cdot \nabla_x \left(\frac{1}{2} \log p(x_{1-s}, 1-s) - \log \overline{p}(x_{1-s}, 1-s) \right).
\end{aligned} \tag{S-10}$$

To compare with the Fokker-Planck for the forward process, eq. (S-8), we'd like the derivative with respect to t, not s. Re-substituting $1-s$ with t and using the chain rule:

$$\begin{aligned}
\frac{\partial \log \overline{p}(x_{1-s}, 1-s)}{\partial s} &= \frac{\partial \log \overline{p}(x_t, t)}{\partial (1-t)} \\
&= -\frac{\partial \log \overline{p}(x_t, t)}{\partial t},
\end{aligned} \tag{S-11}$$

which means the temporal gradient of the reverse process in terms of t is:

$$\begin{aligned}
\frac{\partial \log \overline{p}(x_t, t)}{\partial t} &= -\frac{\partial \log \overline{p}(x_{1-s}, 1-s)}{\partial s} \\
&= -\nabla_x \cdot f(x_t, t) \\
&- f(x_t, t) \cdot \nabla_x \log \overline{p}(x_t, t) \\
&+ g(t)^2 \nabla_x \cdot \nabla_x \left(\log \overline{p}(x_t, t) - \frac{1}{2} \log p(x_t, t) \right) \\
&+ g(t)^2 \nabla_x \log \overline{p}(x_t, t) \cdot \nabla_x \left(\log \overline{p}(x_t, t) - \frac{1}{2} \log p(x_t, t) \right).
\end{aligned} \tag{S-12}$$

To show that $p(x_t, t) = \overline{p}(x_t, t)$, then, it is sufficient to show that $p(x_t, t)$ is a solution to eq. (S-12) and that both functions satisfy the same initial value problems. That $p(x_t, t)$ satisfies eq. (S-12) is easy to show by substituting $p(x_t, t)$ for $\overline{p}(x_t, t)$ and observing that it simplifies to the forward logarithmic Fokker-Planck equation, eq. (S-8):

$$\begin{aligned}
\frac{\partial \log \overline{p}(x_t, t)}{\partial t} &= -\nabla_x \cdot f(x_t, t) \\
&- f(x_t, t) \cdot \nabla_x \log p(x_t, t) \\
&+ g(t)^2 \nabla_x \cdot \nabla_x \left(\log p(x_t, t) - \frac{1}{2} \log p(x_t, t) \right) \\
&+ g(t)^2 \nabla_x \log p(x_t, t) \cdot \nabla_x \left(\log p(x_t, t) - \frac{1}{2} \log p(x_t, t) \right) \qquad \text{(S-13)}
\end{aligned}$$

$$\begin{aligned}
&= -\nabla_x \cdot f(x_t, t) \\
&- f(x_t, t) \cdot \nabla_x \log p(x_t, t) \\
&+ \frac{1}{2} g(t)^2 \nabla_x \cdot \nabla_x \log p(x_t, t) \\
&+ \frac{1}{2} g(t)^2 \|\nabla_x \log p(x_t, t)\|^2, \qquad \text{(S-8)}
\end{aligned}$$

and the initial condition, $\overline{p}(x_1, 1) = p_1(x_1)$, is satisfied by our assumption that we add sufficient noise for $p(x_1, 1) = p_1(x_1)$. Since $p(x_t, t)$ satisfies the differential equation for $\overline{p}(x_t, t)$ with the same initial conditions, we conclude that $p(x_t, t) = \overline{p}(x_t, t)$ as expected. To save on this algebraic manipulation in the main body of the paper, we always use the Fokker-Planck of the forward equation and assume $dt > 0$.

C. Derivation of Flow Trajectory Integral

Applying eq. (25) to the flow ODE path, we can simplify the equation analytically by subbing in the path's $\frac{dx_t}{dt}$ as defined by the ODE, eq. (27):

$$\begin{aligned}
\frac{dx_t}{dt} &= -\frac{1}{2} g(t)^2 \nabla_x \log p(x_t, t) \qquad \text{(27)} \\
&\approx -\frac{1}{2} g(t)^2 s_\theta(x_t, t).
\end{aligned}$$

Substituting this into eq. (25):

$$\begin{aligned}
\log p_0(x_0) &= \log p_1(x_1) - \int_0^1 \left(s_\theta(x_t, t) \cdot \frac{dx_t}{dt} + \frac{1}{2} g(t)^2 \mathrm{Tr}\Big(\nabla_x s_\theta(x_t, t)\Big) + \frac{1}{2} g(t)^2 \|s_\theta(x_t, t)\|^2 \right) dt \qquad \text{(25)} \\
&= \log p_1(x_1) - \int_0^1 \left(s_\theta(x_t, t) \cdot \left(-\frac{1}{2} g(t)^2 s_\theta(x_t, t) \right) + \frac{1}{2} g(t)^2 \mathrm{Tr}\Big(\nabla_x s_\theta(x_t, t)\Big) + \frac{1}{2} g(t)^2 \|s_\theta(x_t, t)\|^2 \right) dt \\
&= \log p_1(x_1) - \int_0^1 \left(\cancel{-\frac{1}{2} g(t)^2 \|s_\theta(x_t, t)\|^2} + \frac{1}{2} g(t)^2 \mathrm{Tr}\Big(\nabla_x s_\theta(x_t, t)\Big) + \cancel{\frac{1}{2} g(t)^2 \|s_\theta(x_t, t)\|^2} \right) dt \\
&= \log p_1(x_1) - \int_0^1 \frac{1}{2} g(t)^2 \mathrm{Tr}\Big(\nabla_x s_\theta(x_t, t)\Big) dt. \qquad \text{(28)}
\end{aligned}$$

D. Pseudocodes for Log Likelihood Integrals

These are pseudocode algorithms for integrating over discrete diffusion trajectories using the trapezoid rule (Algorithm 1) and flow trajectories using black-box ODE solvers (Algorithm 2).

Algorithm 1 Trapezoidal Integration over Diffusion Trajectories

Input: Diffusion trajectory samples $\{(x_{t_0}, t_0), (x_{t_1}, t_1), \ldots, (x_{t_n}, t_n)\}$ where $t_i \in [0,1], t_0 = 0, t_1 = 1$; score model $s_\theta(x_t, t)$; noise schedule $g(t)$
Output: Log-likelihood estimate $\log p_0(x_0) = \log p_1(x_1) - \Delta \log p$
function INTEGRAND(x_t, t,Δx, Δt)
    return $s_\theta(x_t, t) \cdot \Delta x + \frac{1}{2} g(t)^2 \mathrm{Tr}\Big(\nabla_x s_\theta(x_t, t)\Big) + \frac{1}{2} g(t)^2 \|s_\theta(x_t, t)\|^2$
end function
Initialize $\Delta \log p \leftarrow 0$
for $i = 0$ to $n - 1$ **do**
    $\Delta x_i \leftarrow x_{t_{i+1}} - x_{t_i}$
    $\Delta t_i \leftarrow t_{i+1} - t_i$
    $f_a \leftarrow \text{Integrand}(x_{t_i}, t_i, \Delta x_i, \Delta t_i)$
    $f_b \leftarrow \text{Integrand}(x_{t_{i+1}}, t_{i+1}, \Delta x_i, \Delta t_i)$
    $\Delta \log p \leftarrow \Delta \log p + \frac{1}{2}(f_a + f_b)$
end for
Compute $\log p_1(x_1) \sim \mathcal{N}(0, I)$ ▷ Gaussian prior
$\log p_0(x_0) \leftarrow \log p_1(x_1) - \Delta \log p$
return $\log p_0(x_0)$

Algorithm 2 ODE Integration over Flow Paths

Input: Diffusion sample x_0; score model $s_\theta(x_t, t)$; noise schedule $g(t)$
Output: Log-likelihood $\log p_0(x_0)$
function DRIFT(x_t, t)
    return $-g(t)^2 \cdot s_\theta(x_t, t)$
end function
function TRACEJACOBIAN(x_t, t)
    return $\frac{1}{2} g(t)^2 \mathrm{Tr}\Big(\nabla_x s_\theta(x_t, t)\Big)$
end function
Solve the following ODEs from $t = 0$ to $t = 1$ using an ODE solver:
- $\frac{dx_t}{dt} = \text{Drift}(x_t, t)$ with initial condition x_0
- $\frac{d(\texttt{logp})}{dt} = \text{TraceJacobian}(x_t, t)$ with initial value 0
- **return** $\{x_t|_{t=0}^{1}\}, \Delta \log p$

Compute $\log p_1(x_1) \sim \mathcal{N}(0, I)$ ▷ Gaussian prior
$\log p_0(x_0) \leftarrow \log p_1(x_1) - \Delta \log p$
return $\log p_0(x_0)$

E. Additional Supplementary Figures

Supp. Fig.	y-axis	x-axis
Supp. Fig. S1	Diffusion Trajectory Learned Energy	Interface RMSD
Supp. Fig. S2	Diffusion Trajectory Learned Energy	DockQ
Supp. Fig. S3	Flow Trajectory Learned Energy	Interface RMSD
Supp. Fig. S4	Flow Trajectory Learned Energy	DockQ
Supp. Fig. S5	Rosetta Energy	Interface RMSD
Supp. Fig. S6	Rosetta Energy	DockQ
Supp. Fig. S7	Flow Trajectory Learned Energy	Rosetta Energy
Supp. Fig. S8	Diffusion Trajectory Learned Energy	Rosetta Energy
Supp. Fig. S9	Diffusion Trajectory Learned Energy	Flow Trajectory Learned Energy

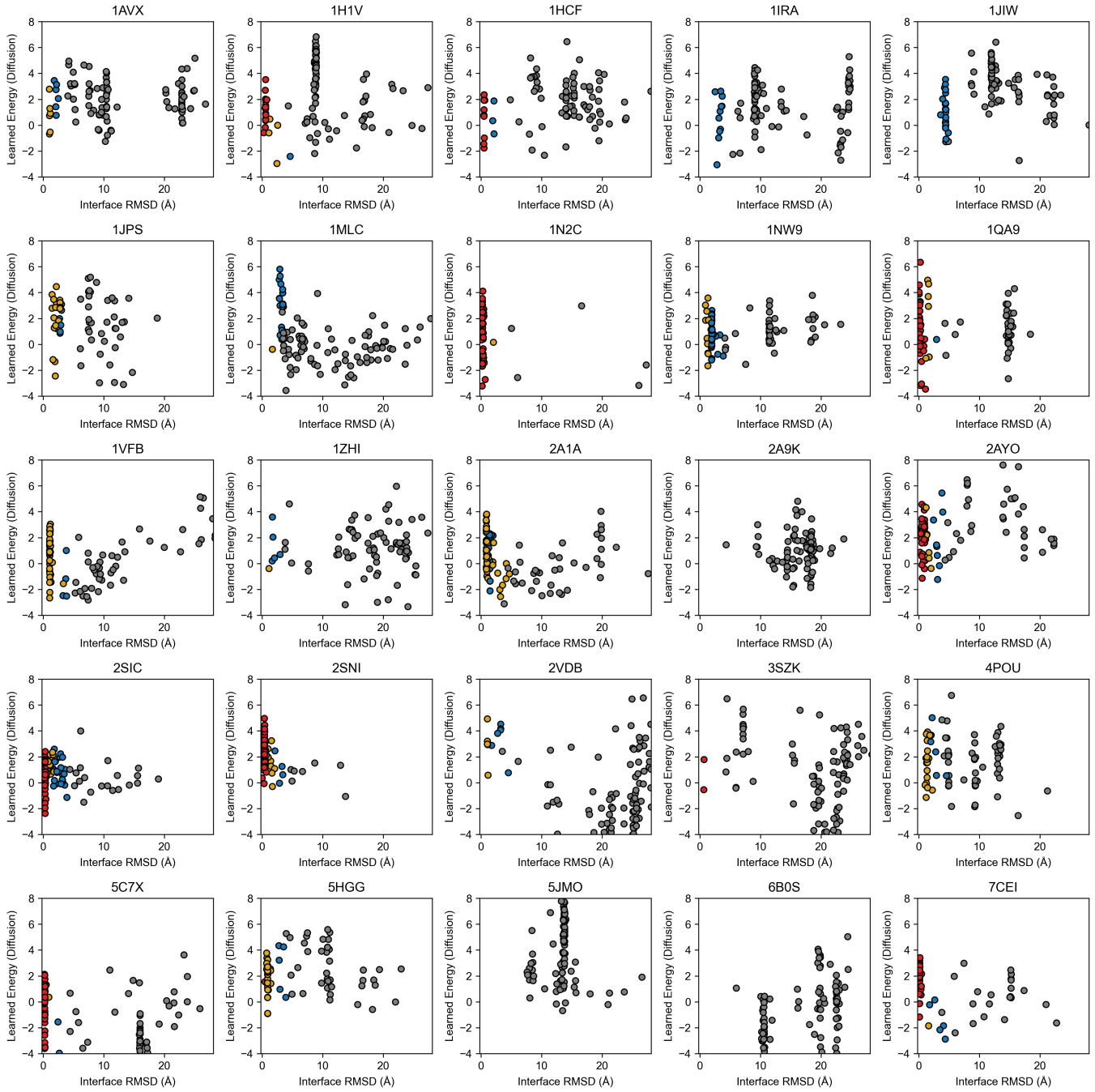

FIG. S1. Learned energy computed from integrating over diffusion trajectories, plotted against interface RMSD of 120 docking poses generated from DFMDock for 25 targets in the DB5.5 dataset. Individual points are colored by their docking quality based on the CAPRI classification (incorrect: gray, acceptable: blue, medium: gold, high: red).

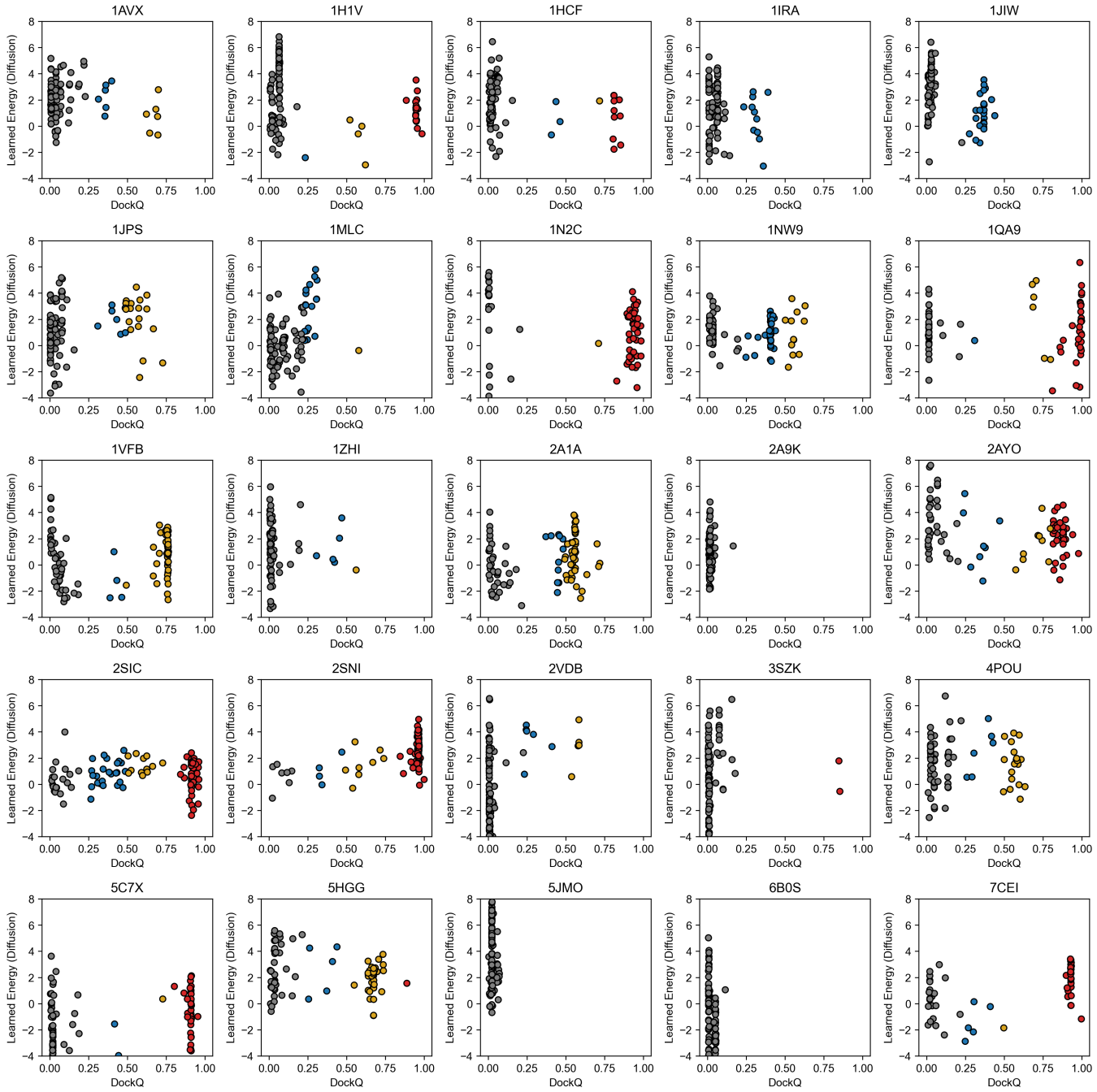

FIG. S2. Learned energy computed from integrating over diffusion trajectories, plotted against DockQ of 120 docking poses generated from DFMDock for 25 targets in the DB5.5 dataset. Individual points are colored by their docking quality based on the CAPRI classification (incorrect: gray, acceptable: blue, medium: gold, high: red).

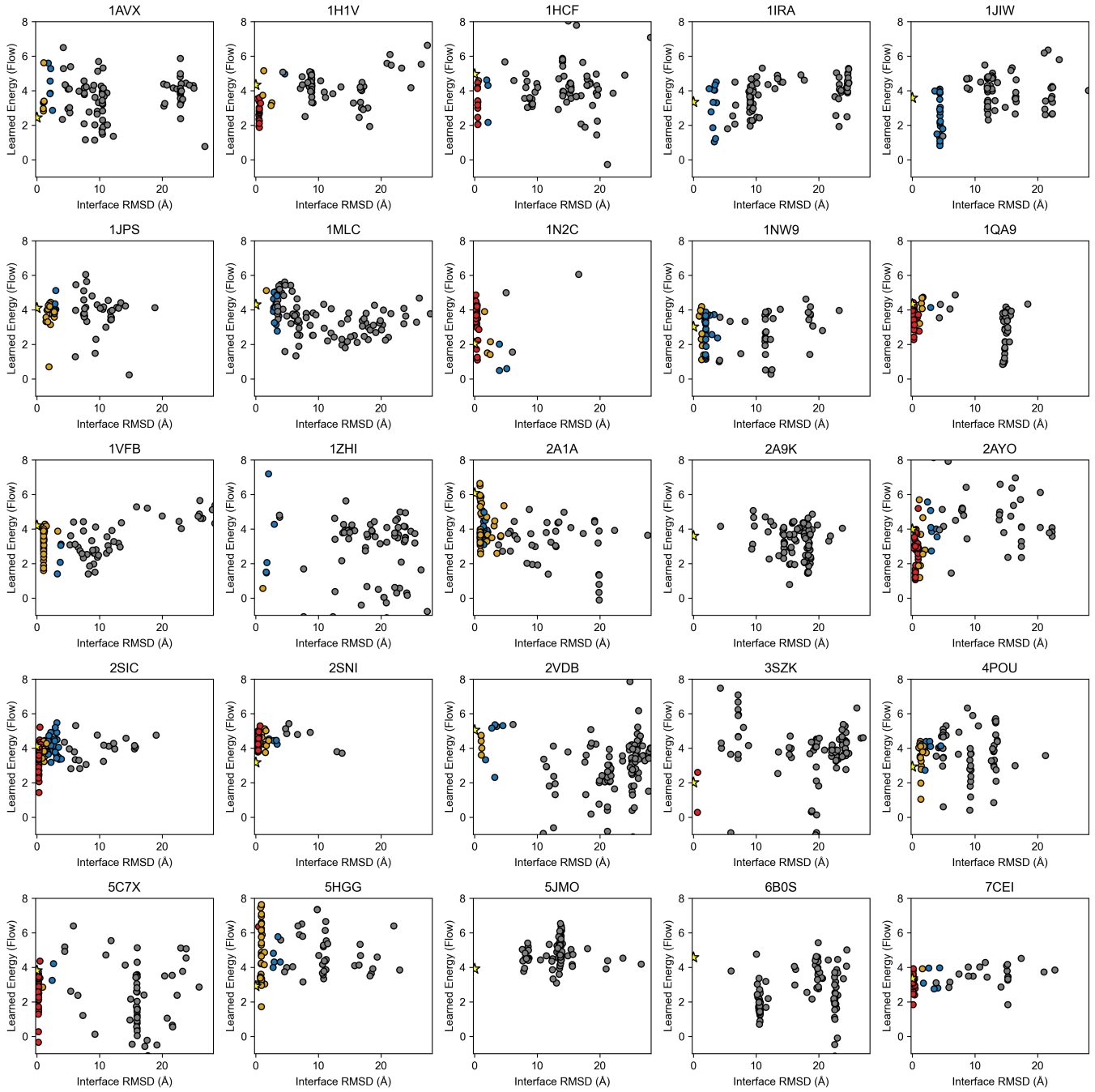

FIG. S3. Learned energy computed from integrating over flow trajectories, plotted against interface RMSD of 120 docking poses generated from DFMDock for 25 targets in the DB5.5 dataset. Individual points are colored by their docking quality based on the CAPRI classification (incorrect: gray, acceptable: blue, medium: gold, high: red). Energy of the ground truth structure is shown as a yellow star.

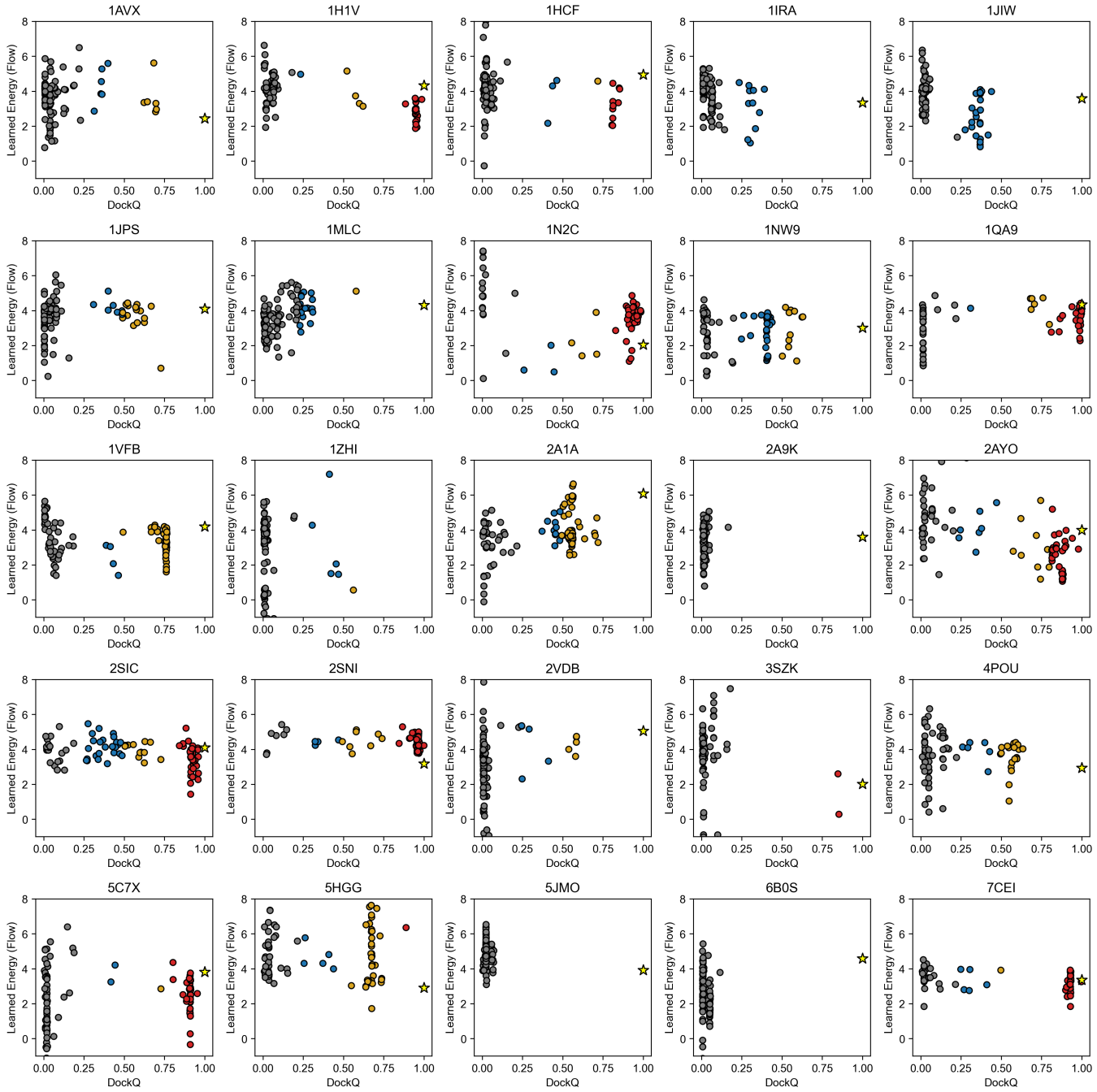

FIG. S4. Learned energy computed from integrating over flow trajectories, plotted against DockQ of 120 docking poses generated from DFMDock for 25 targets in the DB5.5 dataset. Individual points are colored by their docking quality based on the CAPRI classification (incorrect: gray, acceptable: blue, medium: gold, high: red). Energy of the ground truth structure is shown as a yellow star.

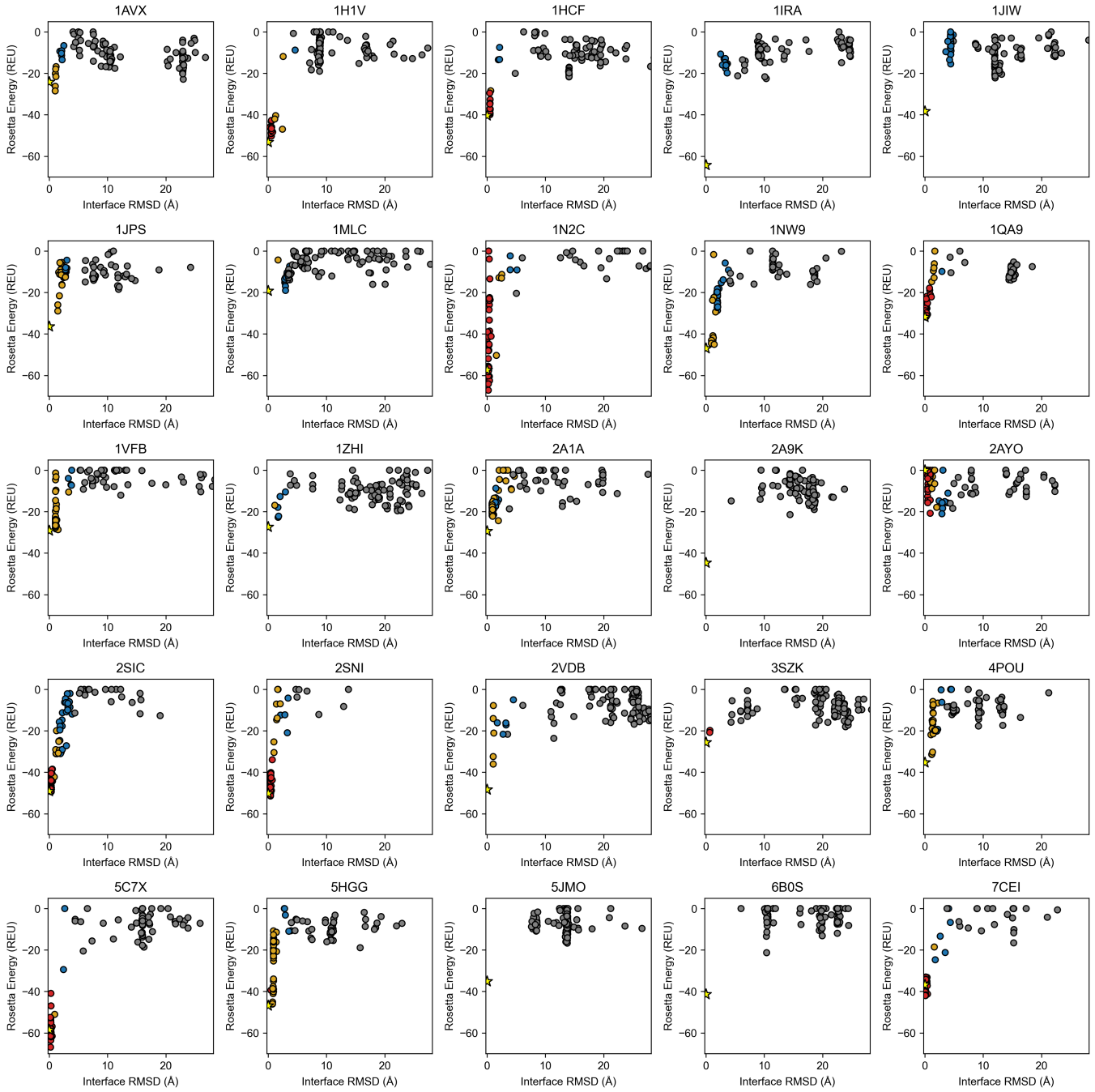

FIG. S5. Rosetta energy plotted against interface RMSD of 120 docking poses generated from DFMDock for 25 targets in the DB5.5 dataset. Individual points are colored by their docking quality based on the CAPRI classification (incorrect: gray, acceptable: blue, medium: gold, high: red). Energy of the ground truth structure is shown as a yellow star.

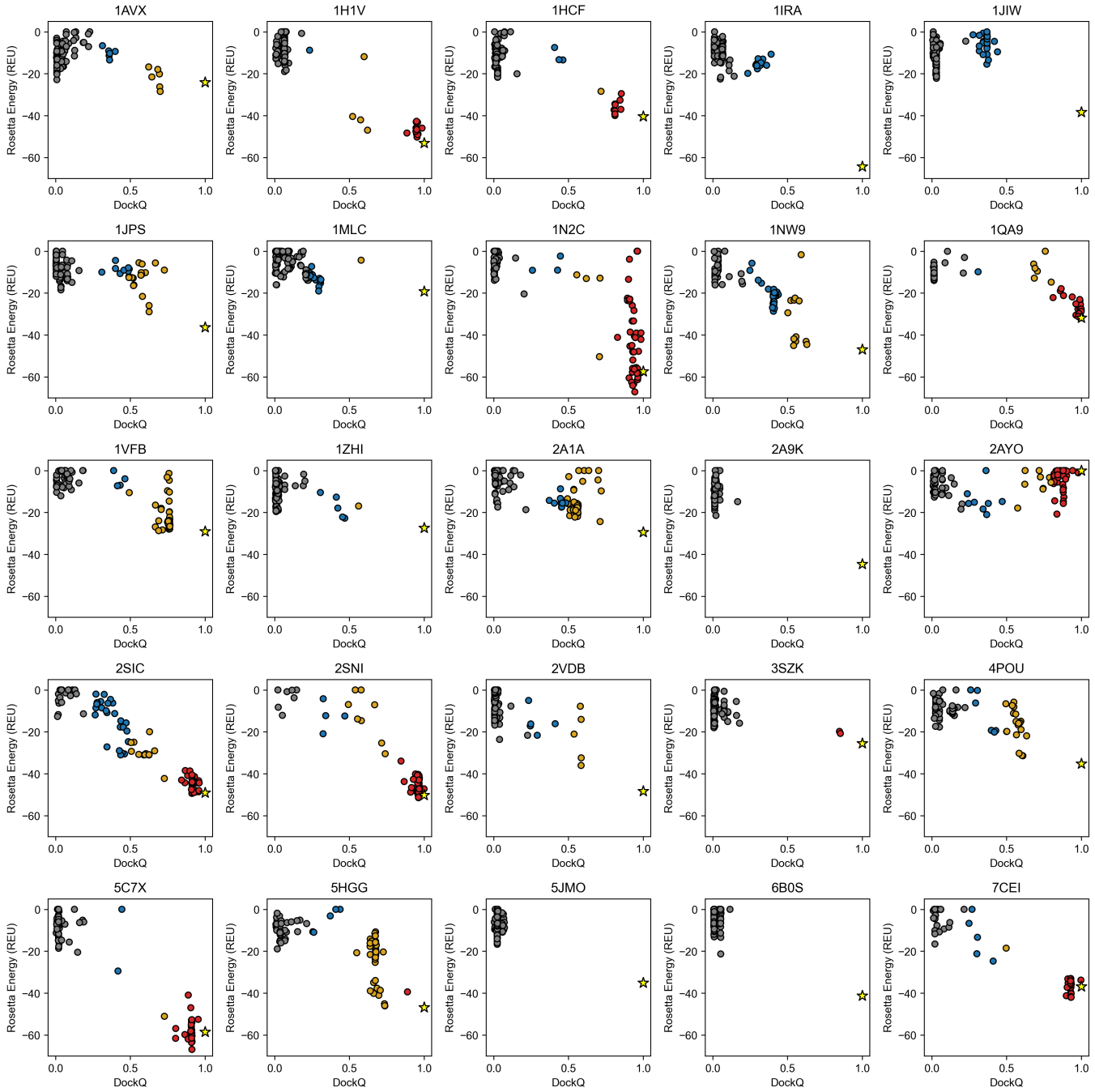

FIG. S6. Rosetta energy plotted against DockQ of 120 docking poses generated from DFMDock for 25 targets in the DB5.5 dataset. Individual points are colored by their docking quality based on the CAPRI classification (incorrect: gray, acceptable: blue, medium: gold, high: red). Energy of the ground truth structure is shown as a yellow star.

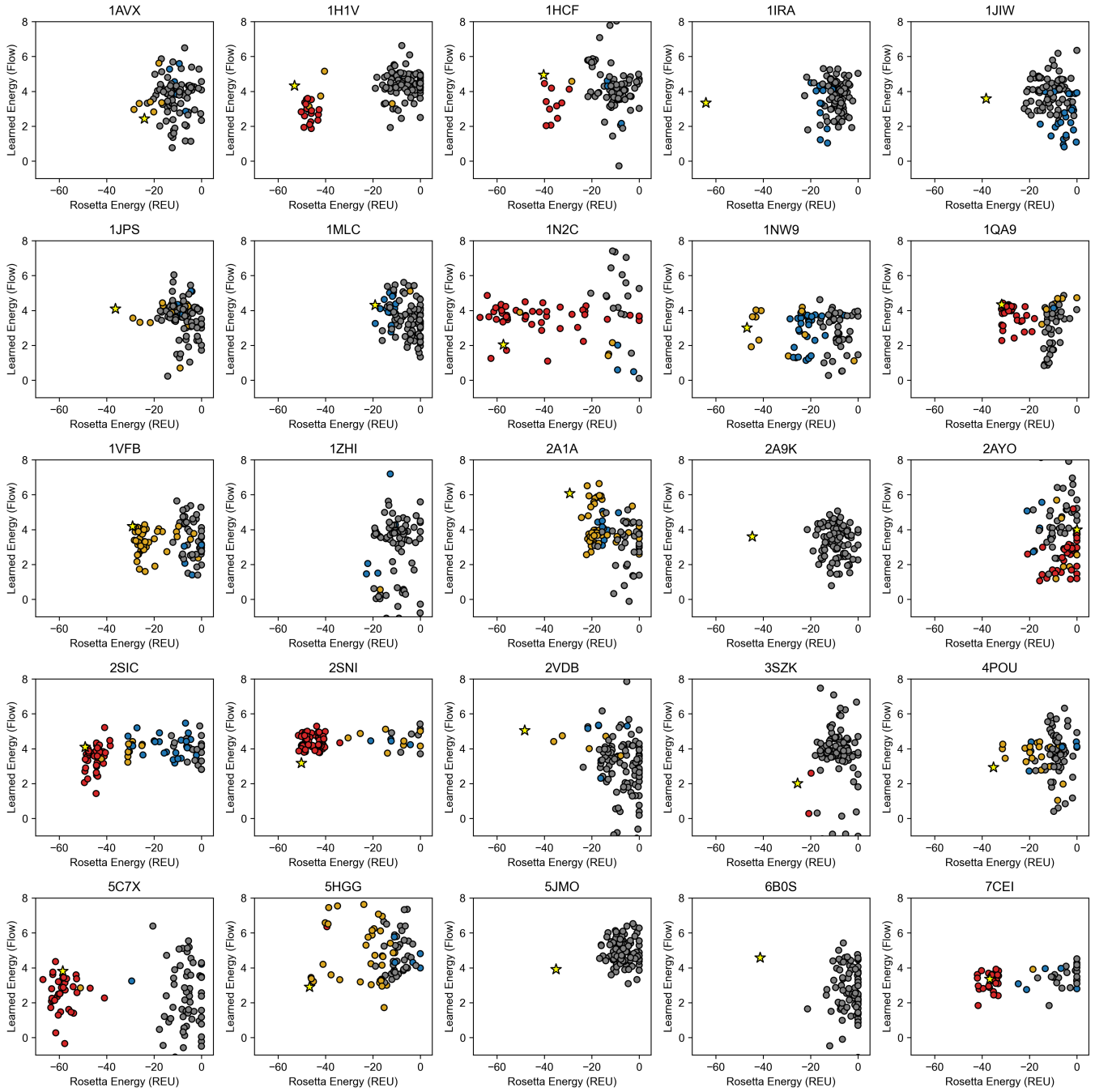

FIG. S7. Learned energy computed from integrating over flow trajectories, plotted against Rosetta energy of 120 docking poses generated from DFMDock for 25 targets in the DB5.5 dataset. Individual points are colored by their docking quality based on the CAPRI classification (incorrect: gray, acceptable: blue, medium: gold, high: red). Energy of the ground truth structure is shown as a yellow star.

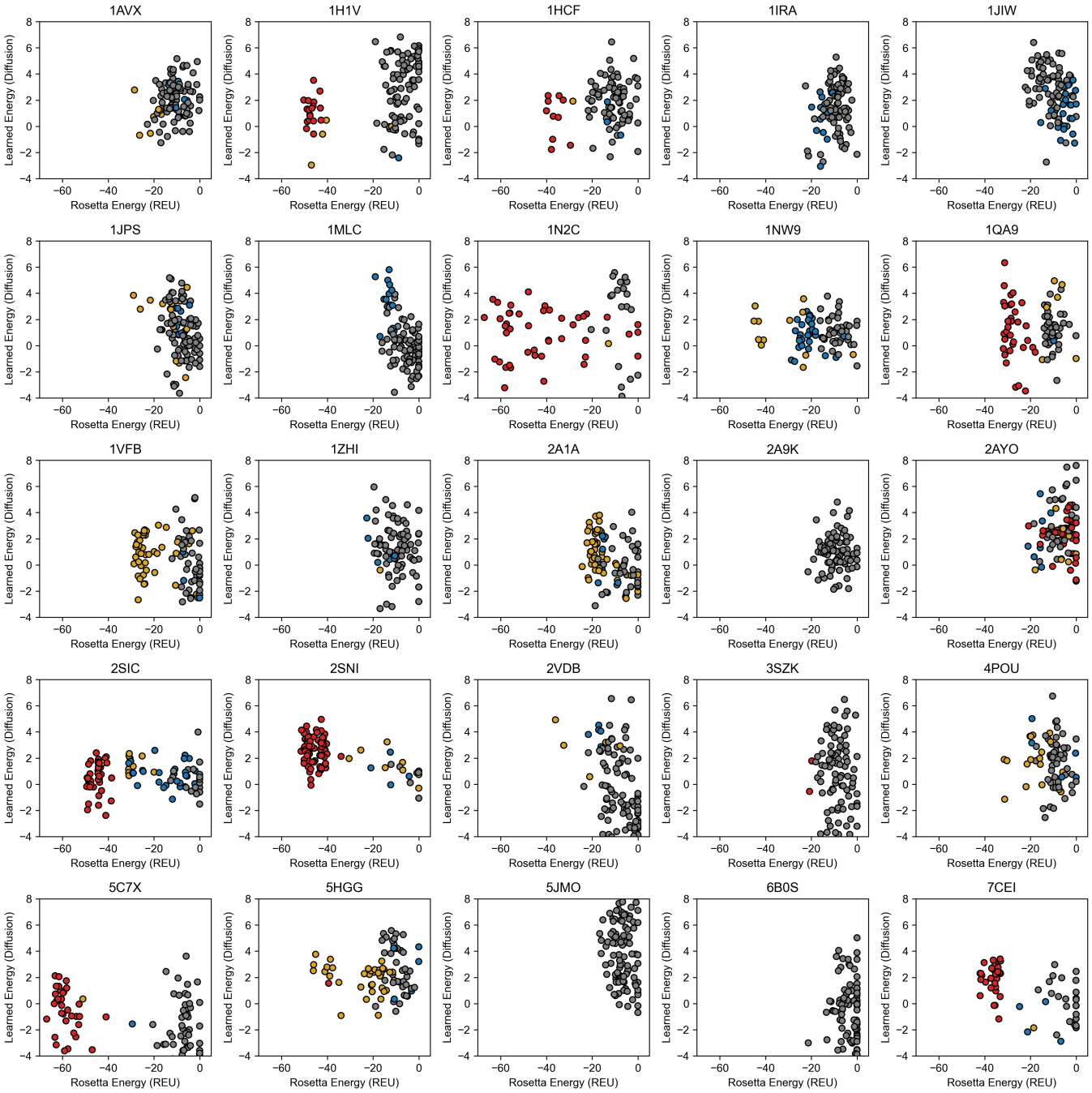

FIG. S8. Learned energy computed from integrating over flow trajectories, plotted against Rosetta energy of 120 docking poses generated from DFMDock for 25 targets in the DB5.5 dataset. Individual points are colored by their docking quality based on the CAPRI classification (incorrect: gray, acceptable: blue, medium: gold, high: red).

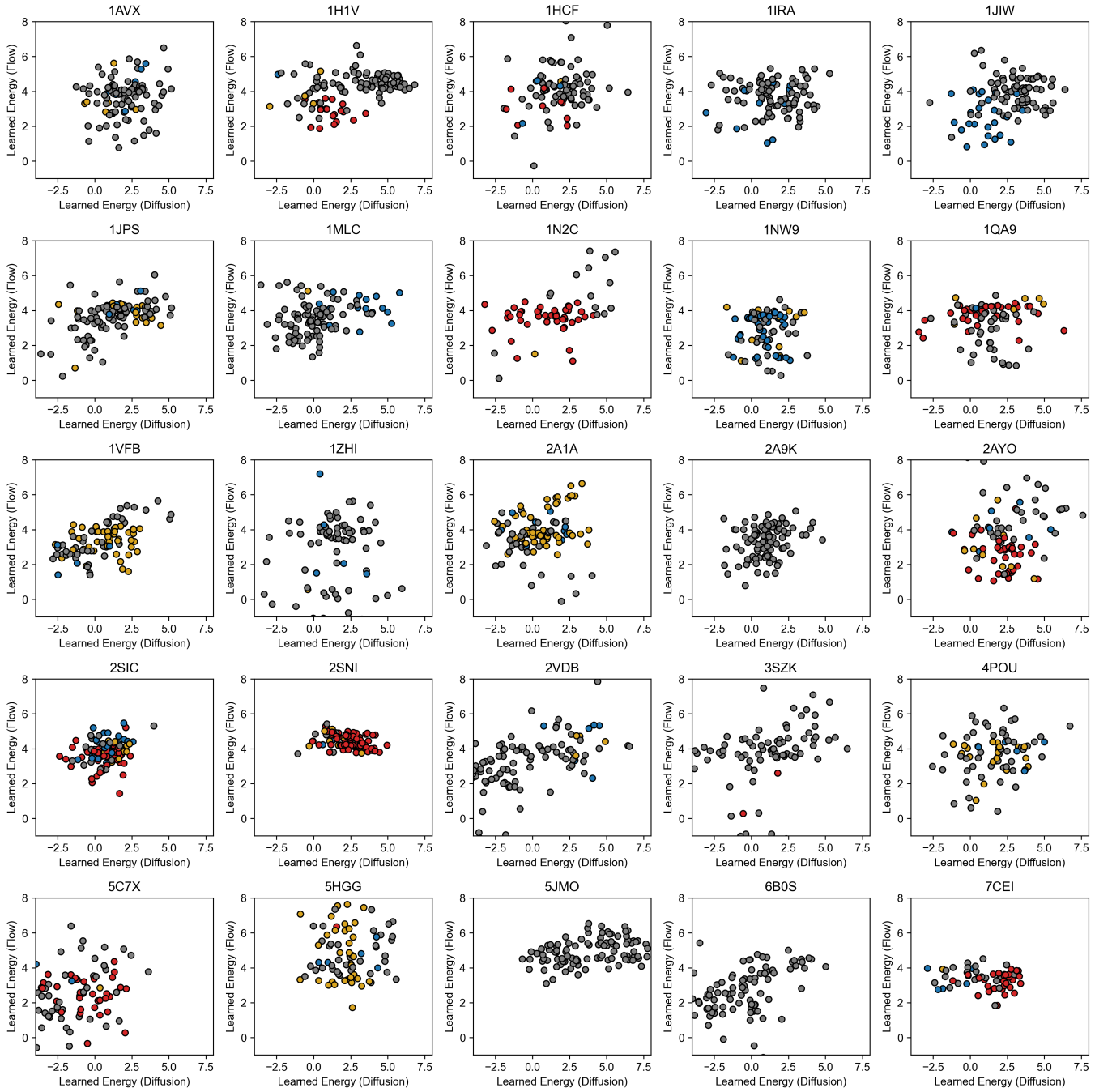

FIG. S9. Learned energy computed from integrating over flow trajectories plotted against integrating over diffusion trajectories, generated from DFMDock for 25 targets in the DB5.5 dataset. Individual points are colored by their docking quality based on the CAPRI classification (incorrect: gray, acceptable: blue, medium: gold, high: red).