Energy-Based Flow Matching for Molecular Docking

Wenyin Zhou[1] **Christopher Iliffe Sprague**[1,2] **Hossein Azizpour**[1]

[1]KTH Stockholm [2]SciLifeLab

`{wenyinz, sprague, azizpour}@kth.se`

Abstract

Molecular docking, which predicts the bound structure of protein-ligand conformations, is essential for structure-based drug design. Recent advances in generative modeling, such as diffusion models and flow matching, have achieved great success in this task by modeling docking conformations as a distribution. With these models, a confidence model/head can be trained to rank the generated samples for downstream study. In this work, we focus on flow matching and adopt an energy-based perspective to understand the confidence model. This results in a mapping function, represented by a deep network, that is directly learned to iteratively map random configurations, i.e., samples from the source distribution, to bound structures, i.e., points in the target data manifold. This yields a conceptually simple and empirically effective flow matching setup with interesting connections to fundamental properties such as idempotency and stability, as well as empirical structure prediction techniques such as refinement. Experiments on PDBBind and Binding MOAD for both single and multi-ligand docking consistently demonstrate the method's effectiveness where it outperforms recent baselines of standard flow matching and task-associated diffusion model, using similar computational budget.

1 Introduction

Structure prediction tasks, such as protein folding and molecular docking, play a critical role in understanding biological mechanisms. Regression-based structure prediction models [17, 20, 3, 34, 44, 26, 24] have demonstrated impressive results on these tasks by directly predicting the structure via a deep neural network, but these approaches fail to capture aleatoric uncertainty due to the possibility of multiple molecular conformations. More specifically, since regression models do not explicitly model the underlying distribution, the generated structures can be limited in diversity and not align with the physiochemical properties of the molecules.

With the recent breakthroughs in generative modeling based on diffusion models [12, 32, 30] and flow matching [21, 22, 2], sampling-based methods have attracted significant interest in molecular docking [10, 8, 27, 14, 1, 9, 25]. These methods aim to learn a time-dependent vector field (or score function) that guides randomly sampled structures from a source distribution to a target distribution of true structures. In contrast to regression-based models, several possible configurations can be generated. Together with a confidence model, the generated samples can be scored and selected for prediction.

In energy-based models (EBMs) [19], the energy function assigns low energy to positive examples and high energy to negative ones, shaping the landscape by maximizing the discrepancy between contrastive samples. Interestingly, this approach aligns with confidence models in molecular docking, where the energy function can serve as a scoring function, selecting the best sample with minimal energy.

In this work, we develop an energy-based formulation for flow matching, offering a unified perspective for pose generation and scoring in sampling-based docking. To improve flow matching training, we shape the loss landscape using contrastive predictions from the flow model. Although the exact energy function is intractable, we leverage the reconstruction error, a commonly adopted energy function for approximation. In the context of conditional flow matching, this energy function also aligns with the conditional log-likelihood of the training sample. Through the data parameterization of the flow matching, we present a simple instance of energy-based flow matching, IDFlow, where the neural network iteratively predicts and refines the generated sample.

Importantly, our learned *idempotent* function (IDFlow) simultaneously serves as both a neural sampler and a refiner, leading to the proposal of a predictor-refiner sampler, analogous to the predictor-corrector sampler in diffusion models [32]. This approach iteratively refines the sampling trajectory toward points where the energy function approximates zero, effectively enhancing flow matching with minimal increase in training cost. The key contributions are as follows:

- We adopted an energy-based learning perspective for flow matching and sampling-based molecular docking.
- We provide a simple instance of this perspective, IDFlow, which iteratively predicts and refines the sample and draws interesting connections to related approaches.
- Empirical performance on both single and multi-ligand docking consistently shows the proposed method outperforms the recent flow matching and task-associated diffusion models.

2 Background

2.1 Energy-Based Models

Energy-based models (EBMs) consider an energy function $E_\theta(x) \in \mathbb{R}_+$ and assign a scalar value to each data point. Learning within this paradigm requires devising an energy landscape and a loss function to shape that energy. For instance, the L_2 regression loss could be seen as directly using the energy function for the loss where the energy architecture is the L_2 norm between the network output and the label:

$$\textbf{Energy}: \quad E_\theta(x) = ||f_\theta(x) - y||^2, \qquad \textbf{Loss}: \quad L_\theta(x) = E_\theta(x) \tag{1}$$

where (x, y) is a training pair of input and label. Training with this strategy primarily reduces the energy of the training examples. In general, energy-based learning maximizes the energy margin between positive and negative training examples while avoiding the collapse of the overall energy landscape. Specifically, the collapse happens when the network produces identical outputs regardless of the input. For a detailed review, see [19].

2.2 Flow Matching

Flow matching [21, 22, 2] is a simulation-free training method for continuous-normalizing flows [6] that connects two arbitrary distributions with flexible construction of the probability paths. This allows for building straight paths between any source and target distribution, which holds great potential for accelerated sampling. Specifically, flow matching aims to learn a time-dependent vector field v_t evolving the samples from the prior $p_0(x)$ to the target distribution $p_1(x)$ via an ordinary differential equation (ODE):

$$\frac{dx}{dt} = v_{\theta,t}(x) \tag{2}$$

for $t \in [0, 1]$ and θ are the model parameters, where x and $v_{\theta,t}(x)$ could be either defined in the Euclidean space or on a Riemannian manifold [5]. To achieve this, the flow matching objective [21] regresses the vector field to the true vector field $u_t(x)$:

$$L_{\text{FM}}(\theta) := \mathbb{E}_{t\sim U(0,1), x\sim p_t(x)} \|v_{\theta,t}(x) - u_t(x)\|_2^2. \tag{3}$$

However, this objective function cannot be used for training as the true vector field is intractable in practice. Previous work [21, 37] shows that with the construction of conditional probability paths, the conditional flow matching objective shares the same gradient as the flow matching objective $\nabla_\theta L_{\text{FM}} = \nabla_\theta L_{\text{CFM}}$, such that learning from the conditional vector amount to learning from the marginal vector field:

$$L_{\text{CFM}}(\theta) := \mathbb{E}_{t\sim U(0,1), x\sim p_t(x|x_0,x_1)} \|v_{\theta,t}(x) - u_t(x|x_0, x_1)\|_2^2 \tag{4}$$

where $u_t(x|x_0, x_1)$ is the conditional vector field that generates the conditional probability path $p_t(x|x_0, x_1)$.

$\mathbf{x}_1$ parameterization. Similar to diffusion models [38, 43], one can opt to parameterize the network output as the vector field $u_t(x|x_0, x_1)$ or the data x_1 [35, 15, 41, 4, 42]. The latter results in an objective function that directly pushes down the energy of the training data:

$$L_{\text{CFM}}(\theta) := \mathbb{E}_{t\sim U(0,1), x\sim p_t(x|x_0,x_1)} \|f_{\theta,t}(x) - x_1\|_2^2 \tag{5}$$

where x_1 and x_0 are sampled from the training set and the prior distribution, respectively. The x_1 parameterization yields an Euler-like step sampling algorithm:

$$\frac{dx}{dt} = \frac{f_{\theta,t}(x) - x}{1 - t} \tag{6}$$

where f_θ is the mapping between a point from the sampling trajectory to the corresponding target data point, and is, hereafter, referred to as the *flow map*. A general format of the vector field with data parameterization can be found in A.1. In this work, we focus on the x_1 parameterization for its direct connections to the regression model and generative adversarial nets [11].

3 Method

3.1 Energy-based Flow Matching

Confidence Model as the Energy Function. In sampling-based molecular docking, a confidence model ranks the samples by outputting their corresponding confidence scores. To draw a connection to confidence models, we can write down the energy as:

$$E(\hat{x}_1) = D_\theta(\hat{x}_1), \qquad \hat{x}_1 = f_{\theta,t}(x) \tag{7}$$

where D_θ is the architecture, the internal structure of $E(\hat{x}_1)$, and $\hat{x}_1$ is the sample produced by the flow map f_θ. In practice, the energy architecture D could be a separate neural network trained by generating samples for every training example and using the training annotations to produce binary labels to distinguish the positive (high accuracy samples) and negative (low accuracy samples) examples [10]. Then the logits of such a network can be used as the confidence score (energy function) for an unseen example.

From objective Eq. 5, it becomes evident that only the energy associated with the trajectory sample x_t is pushed down, while the contrastive samples $\hat{x}_1$ generated by the flow model, in contrast to x_1, remain unaffected by this process. The idea of energy-based flow matching is to better shape the loss landscape, improve the training with contrastive samples $\hat{x}_1$, and encourage reaching the minimum of the energy function.

Shaping the Loss Landscape with $\hat{x}_1$. We first need to define an energy function $E(\hat{x}_1) : \mathbb{R}^d \to \mathbb{R}_+$, such that high-probability data lies around its minima. Following [45], a simple energy architecture could be the reconstruction error for some function G:

$$E(\hat{x}_1) = ||G(\hat{x}_1) - \hat{x}_1||^2. \tag{8}$$

Since the conversion from energy to probability could be achieved through a Boltzmann distribution [31], this energy function aligns with the conditional likelihood function in the context of flow matching A.2. Assuming G is a function that perfectly maps any $\hat{x}_1$ to x_1 on the data manifold, Eq. 8 will induce high energy (larger reconstruction error) for "bad" $\hat{x}_1$ and low energy for "good" $\hat{x}_1$. Ideally, G could be any neural network that keeps the dimensionality of the input data. Crucially, following this, during training, the flow map $f_\theta(x, t)$ receives gradients not only from the conditional flow matching loss but also from the energy loss Eq. 8 of the contrastive samples.

Sample from the Energy Function. With the energy function devised, we can define an energy-based density with the Boltzmann distribution:

$$p(x) = \frac{\exp(-E(x))}{Z} \tag{9}$$

where Z is some unknown normalizing constant. Taking the derivative of the log-likelihood with respect to x yields:

$$\nabla_x \log(p(x)) = -\nabla_x E(x). \tag{10}$$

Now, assume that we define a gradient flow vector field:

$$v(x) = -\nabla_x E(x). \tag{11}$$

Interestingly, as the energy Eq. 8 is lower bounded by 0, the stability of the flow map is guaranteed as:

$$f_\infty(x) \in \{x \in \mathbb{R}^n \mid \nabla_x E(x) = 0\} = \mathcal{M} \tag{12}$$

where $\mathcal{M}$ is the subset of the equilibrium configuration. This implies starting from any $x \in \mathbb{R}^d$, the gradient flow will converge to some local minima of the energy function E, where the likelihood of the data is locally maximized.

3.2 Idempotent Flow Map

In general, the energy function defined in Eq. 8 builds G to be a "neural refiner". To learn the refiner, we could simply define the objective for G as:

$$L_G = ||G(\hat{x}_1) - x_1||^2. \tag{13}$$

Intuitively, the refiner G is easier to learn than the flow map f, and thus the flow map could have the capacity to *also refine* the mapped sample. This, interestingly, results in learning an idempotent flow map to its prediction:

$$f_{\theta,t}(x) = f_{\theta,t}(f_{\theta,t}(x)). \tag{14}$$

Hence, the energy landscape is shaped by mapping the points off the manifold $\hat{x}_1$ to points on the data manifold x_1 [18]. Relating to sampling the energy minimum, the idempotence of the flow map is clear from Eq. 12:

$$f_\infty(f_\infty(x)) = f_\infty(x) \tag{15}$$

where $f_\infty(x) = f(x, \infty)$ denotes the stationary solution, i.e., iterating the flow map infinitely many times yields the same result. Under the assumption that our dataset of equilibrium configurations is contained in $\mathcal{M}$, we would always want to query $f_t(x)$ at $t = \infty$. Ideally, after training a model to learn the idempotent flow map, we would have $f_\infty(x) = x_1$ for any x. However, imperfections may remain, in which case:

$$f_\infty(x) = \hat{x} \tag{16}$$

would land somewhere close to x_1, but not exactly there. In this case, an iterative refinement procedure [35, 15], can also be applied during inference. The vector from x to $\hat{x} = f(x)$ can be used as a step direction in an integration scheme, such as Euler's method. Intuitively, if the magnitude of the vector is large, the integrator will take a large step in that direction, and vice versa, eventually leading to stabilization around the final prediction.

Furthermore, enforcing the flow map f_θ to be idempotent draws an informative connection to the structure refinement regression model [17] which recycles the output for iterative refinement. Hence, in tandem with the condition flow matching loss, we propose our idempotent objective as:

$$L_{\text{ID}}(\theta) := \mathop{\mathbb{E}}_{\substack{t \sim U(0,1) \\ x \sim p_t(x|x_0, x_1) \\ \hat{x}_1 \sim \mathcal{N}(\hat{x}_1 | f_{\theta,t}(x), \sigma^2 I_d)}} \left[||f_{\theta,t}(\hat{x}_1) - x_1||_2^2 \right] \tag{17}$$

where $\hat{x}_1$ is dynamically sampled from the flow model during training.

3.3 Training and Inference

The idempotent objective function enables the network to refine samples iteratively. Theoretically, the sample $\hat{x}_1$ could be refined an infinite number of times. However, excessive refinements increase inference time, a key limitation of sampling-based methods. To mitigate this, we perform only one refinement per step. Specifically, the predictor-refiner sampler makes a prediction $\hat{x}_1$ and refines it, resulting in two network function evaluations (NFEs) per step. To better align training with sampling, we adopt a strategy similar to self-conditioning [7], where the training of the sampler and refiner is separated. For 50% of the training time, the network undergoes flow matching training. In the remaining 50%, the network first predicts $\hat{x}_1$ with the gradient detached, and then trains for the idempotent objective Eq. 17. Empirically, tuning the maximum number of iterations K_{max} helps achieve gradual refinement. Increasing K_{max} beyond 2 provides diminishing returns, with smaller variations observed at $K_{\text{max}} = 2$. This approach also reduces memory overhead, as only $M - 1$ outputs need to be stored when looping the network M times. Further details are provided in Algorithms 1 and 2.

Table 1: SINGLE LIGAND DOCKING. Structure generation (ten samples average for each test example) comparison of methods on the PDBBind for pocket-level docking.

Method	Sequence Similarity Split				Time Split			
	Distance-Pocket		Radius-Pocket		Distance-Pocket		Radius-Pocket	
	%<2	Med.	%<2	Med.	%<2	Med.	%<2	Med.
Product Space Diffusion	27.2	3.2	16.1	4.0	20.8	3.8	15.2	4.3
HarmonicFlow	30.1	3.1	20.5	**3.4**	42.8	2.5	28.3	3.2
IDFlow	**35.6**	**2.9**	**21.0**	3.7	**44.3**	**2.4**	**34.7**	**3.1**

4 Experiments

Datasets and Baselines. We train the models on the PDBBind v2020 dataset [23] for both the time split and the 30% sequence similarity split and the Binding MOAD [13] with 30% sequence similarity split following [35] for pocket level docking to evaluate the structure generation capability of the method. For both the single and multi-ligand docking scenarios, we consider HarmonicFlow [35] and task-associated diffusion models [10, 16] as the baselines. We strictly follow the dataset processing of HarmonicFlow and hyperparameters for fair comparison. While the number of steps used for sampling significantly affects model performance, we maintain a consistent sampling budget with HarmonicFlow, using ten sampling steps equivalent to 20 Number of Function Evaluations (NFEs). All shown results are averaged over three runs. For more experimental details see Appendix C.

Table 2: MULTI-LIGAND DOCKING. Structure generation (ten samples average for each test example) comparison of methods on the Binding MOAD.

Method	%<2	%<5	Med.
EigenFold Diffusion	39.7	73.5	2.4
HarmonicFlow	**44.4**	75.0	2.2
IDFlow	43.8	**83.1**	**2.1**

Evaluation Metrics. Following [35, 10], we use the fraction of the test examples that have root mean squared deviation (RMSD) below 2 or 5 Ångström ($\% < 2$ and $\% < 5$) and the RMSD median (Med.) for evaluating the docking performance.

Results. We first investigate the method in single ligand docking for structure generation and modes capturing. From Table 1, the structure generation of IDFlow consistently outperforms HarmonicFlow and product space diffusion models except for one entry with the same inference time. From Table 4, the improvements on the top-40, 10 and 5 accuracies are obvious for both the RMSD < 1Å and the RMSD < 2Å, demonstrating the improved prediction accuracy without loss of mode covering. Table 2 shows the results on multi-ligand docking, where IDFlow maintains the same level of performance with the HarmonicFlow on RMSD < 2Å and improves 8.1% for RMSD < 5Å.

Ablations. We investigate some of the design choices of IDFlow on the timesplit radius pocket docking in Table 3. First, we ablate the number iterations K_{max} training for idempotency. Just one iteration improves quite well, and more iterations do not yield big performance gains but less variation. The timestep information t appears to be important for both the diffusion models and flow matching, as it explicitly delivers the noise level in the data. We surprisingly find out the performance does not degrade significantly with t being removed from the model. This may open the opportunity to bridge together the idea of a universal generator and flow models.

5 Conclusion and Future Work

We present energy-based flow matching, an enhanced training framework for flow matching combined with an energy-based model for molecular docking, which better shapes the loss landscape with contrastive samples produced by the flow model. We provide a specific instance of the proposed framework, IDFlow, which considers the reconstruction error as the energy function and builds the flow map to predict and refine the sampling trajectory iteratively. Empirical results show improved performance over HarmonicFlow and task-specific diffusion models for molecular docking. Future work could combine the EBMs formulation with biophysically informed energy to generate chemically plausible structures and apply EBMs as scoring functions in realistic docking scenario.

Acknowledgments and Disclosure of Funding

We thank Matteo Tadiello and Vsevolod Viliuga for insightful discussions and feedback.

This work was partially supported by the Knut and Alice Wallenberg Foundation and the Wallenberg AI, Autonomous Systemsand Software Program (WASP). Scientific computation was enabled by the supercomputing resource Berzelius provided by National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation.

References

[1] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pages 1–3, 2024.

[2] Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023.

[3] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[4] Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian FATRAS, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael M. Bronstein, and Alexander Tong. SE(3)-stochastic flow matching for protein backbone generation. In *The Twelfth International Conference on Learning Representations*, 2024.

[5] Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *The Twelfth International Conference on Learning Representations*, 2024.

[6] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[7] Ting Chen, Ruixiang ZHANG, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations*, 2023.

[8] Gabriele Corso, Arthur Deng, Nicholas Polizzi, Regina Barzilay, and Tommi S. Jaakkola. Deep confident steps to new pockets: Strategies for docking generalization. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] Gabriele Corso, Vignesh Ram Somnath, Noah Getz, Regina Barzilay, Tommi Jaakkola, and Andreas Krause. Flexible docking via unbalanced flow matching. In *ICML'24 Workshop ML for Life and Material Science: From Theory to Industry Applications*.

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

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[13] Liegi Hu, Mark L Benson, Richard D Smith, Michael G Lerner, and Heather A Carlson. Binding moad (mother of all databases). *Proteins: Structure, Function, and Bioinformatics*, 60(3):333–340, 2005.

[14] Yufei Huang, Odin Zhang, Lirong Wu, Cheng Tan, Haitao Lin, Zhangyang Gao, Siyuan Li, Stan Li, et al. Re-dock: Towards flexible and realistic molecular docking with diffusion bridge. *arXiv preprint arXiv:2402.11459*, 2024.

[15] Bowen Jing, Bonnie Berger, and Tommi Jaakkola. Alphafold meets flow matching for generating protein ensembles. *arXiv preprint arXiv:2402.04845*, 2024.

[16] Bowen Jing, Ezra Erives, Peter Pao-Huang, Gabriele Corso, Bonnie Berger, and Tommi Jaakkola. Eigenfold: Generative protein structure prediction with diffusion models, 2023.

[17] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[18] Yann LeCun. From machine learning to autonomous intelligence: Lecture 2, 2022. Accessed: 2024-10-02.

[19] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

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

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

[22] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[23] Zhihai Liu, Minyi Su, Li Han, Jie Liu, Qifan Yang, Yan Li, and Renxiao Wang. Forging the basis for developing protein–ligand interaction scoring functions. *Accounts of chemical research*, 50(2):302–309, 2017.

[24] Wei Lu, Qifeng Wu, Jixian Zhang, Jiahua Rao, Chengtao Li, and Shuangjia Zheng. TANKBind: Trigonometry-aware neural networks for drug-protein binding structure prediction. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[25] Wei Lu, Jixian Zhang, Weifeng Huang, Ziqiao Zhang, Xiangyu Jia, Zhenyu Wang, Leilei Shi, Chengtao Li, Peter G Wolynes, and Shuangjia Zheng. Dynamicbind: predicting ligand-specific protein-ligand complex structure with a deep equivariant generative model. *Nature Communications*, 15(1):1071, 2024.

[26] Qizhi Pei, Kaiyuan Gao, Lijun Wu, Jinhua Zhu, Yingce Xia, Shufang Xie, Tao Qin, Kun He, Tie-Yan Liu, and Rui Yan. Fabind: Fast and accurate protein-ligand binding. *Advances in Neural Information Processing Systems*, 36, 2024.

[27] Zhuoran Qiao, Weili Nie, Arash Vahdat, Thomas F Miller III, and Animashree Anandkumar. State-specific protein–ligand complex structure prediction with a multiscale deep generative model. *Nature Machine Intelligence*, 6(2):195–208, 2024.

[28] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[29] Assaf Shocher, Amil V Dravid, Yossi Gandelsman, Inbar Mosseri, Michael Rubinstein, and Alexei A Efros. Idempotent generative network. In *The Twelfth International Conference on Learning Representations*, 2024.

[30] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[31] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

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

[33] Christopher Iliffe Sprague, Arne Elofsson, and Hossein Azizpour. Incorporating stability into flow matching. In *ICML 2024 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2024.

[34] Hannes Stärk, Octavian Ganea, Lagnajit Pattanaik, Regina Barzilay, and Tommi Jaakkola. Equibind: Geometric deep learning for drug binding structure prediction. In *International conference on machine learning*, pages 20503–20521. PMLR, 2022.

[35] Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Harmonic self-conditioned flow matching for multi-ligand docking and binding site design. *arXiv preprint arXiv:2310.05764*, 2023.

[36] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[37] Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. Expert Certification.

[38] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[39] Yilun Xu, Ziming Liu, Max Tegmark, and Tommi Jaakkola. Poisson flow generative models. *Advances in Neural Information Processing Systems*, 35:16782–16795, 2022.

[40] Yilun Xu, Ziming Liu, Yonglong Tian, Shangyuan Tong, Max Tegmark, and Tommi Jaakkola. Pfgm++: Unlocking the potential of physics-inspired generative models. In *International Conference on Machine Learning*, pages 38566–38591. PMLR, 2023.

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

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

[43] Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023.

[44] Yangtian Zhang, Huiyu Cai, Chence Shi, and Jian Tang. E3bind: An end-to-end equivariant network for protein-ligand docking. In *The Eleventh International Conference on Learning Representations*, 2023.

[45] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial networks. In *International Conference on Learning Representations*, 2017.

A More Discussions

A.1 Vector Field of the Data Parameterization

With the data parameterization of the flow matching, the exact format of the vector field $v_t(x)$ could be diverse as long as the predicted data $\hat{x}_1 = f_{\theta,t}(x)$ could be reached following this vector field. As the trajectory sample x_t is often constructed as the linear interpolation, the conditional vector field that generates a single sample at time t is linear in the general format as follows:

$$u_t(x|x_1, x_0) = A(t)(x - x_1) \tag{18}$$

where $A(t)$ is a time-dependent rescaling coefficient. As the marginal vector field $u_t(x)$ is the convex combination of the conditional vector field (Eq. 8 from [21]), we can directly plug into the computation:

$$\begin{aligned}
u_t(x) &= \iint u_t(x|x_1, x_0)\frac{p_t(x|x_1, x_0)q(x_1)p(x_0)}{p_t(x)}\mathrm{d}x_1\mathrm{d}x_0 \\
&= \iint A(t)(x - x_1)\frac{p_t(x|x_1, x_0)q(x_1)p(x_0)}{p_t(x)}\mathrm{d}x_1\mathrm{d}x_0 \\
&= A(t)\iint (x - x_1)p(x_1, x_0|x)\mathrm{d}x_1\mathrm{d}x_0 \\
&= A(t)\int (x - x_1)p(x_1|x)\mathrm{d}x_1 \\
&= A(t)\mathbb{E}_{x_1\sim p(x_1|x)}\left[x - x_1\right] \\
&= A(t)(x - \mathbb{E}_{x_1\sim p(x_1|x)}[x_1])
\end{aligned}$$

where the thrid equality we use the Bayes rule assuming the $q(x_1)$ and $p(x_1)$ are independent. This yields a general format of vector field supposing the conditional vector field is linear. In practice, the flow map $f_{\theta,t}(x)$ is trained to predict the data, implicitly learning the expected value of x_1 given x. Hence, the form of the marginal vector field with data parameterization is:

$$\begin{aligned}
u_t(x) &= A(t)(x - \mathbb{E}_{x_1\sim p(x_1|x)}[x_1]) \\
&\approx A(t)(x - f_{\theta,t}(x))
\end{aligned}$$

With $A(t) = -\frac{1}{1-t}$, the marginal vector field trained by the conditional optimal transport is recovered as:

$$u_t(x) = \frac{f_{\theta,t}(x) - x}{1 - t} \tag{19}$$

Interestingly, with the learned flow map approximates to be the expectation over $p(x_1|x)$, the flow map's idempotency is clear as the expectation is an idempotent operator:

$$f_{\theta,t}(f_{\theta,t}(x)) = \mathbb{E}[\mathbb{E}_{x_1\sim p(x_1|x)}[x_1]] = \mathbb{E}_{x_1\sim p(x_1|x)}[x_1] \tag{20}$$

A.2 Conditional Negative likelihood Energy

Conditional flow matching assumes a smooth delta function for the training example $x \sim \mathcal{N}(x|x_1, \sigma^2 I_d)$:

$$p(x|x_1) \sim \exp\left(-\frac{1}{2\sigma^2}(x - x_1)^\top(x - x_1)\right) \tag{21}$$

where x_1 is a single sample from the training set. However, the true x_1 is unknown during sampling, and $\hat{x}_1 = f_\theta(x, t)$ is estimated to refine the trajectory of the sampling path. We can approximately assume the $\hat{x}_1$ distribution follows a similar format:

$$p(\hat{x}_1|x_1) \sim \exp\left(-\frac{1}{2\sigma^2}(\hat{x}_1 - x_1)^\top(\hat{x}_1 - x_1)\right) \tag{22}$$

Ideally, each sampling step intends to find an estimate of $\hat{x}_1$ such that it lies in the high-density region and approximates to be stable around a certain area of the energy function. With the exact energy format as the negative log-likelihood of Eq. 23, the energy of $\hat{x}_1$ could be simply computed as:

$$E(\hat{x}_1) = -\log p(\hat{x}_1|x_1) = \frac{1}{2\sigma^2}(\hat{x}_1 - x_1)^\top(\hat{x}_1 - x_1) \tag{23}$$

Algorithm 1 Idempotent Flow Map Training

```
Require: prior distribution $p_0$, data distribution $p_1$
  while Training do
      $x_0 \sim p_0(x_0), x_1 \sim p_1(x_1)$
      $t \sim U(0,1), m \sim U(0,1)$
      $\mu_t \leftarrow t \cdot x_1 + (1-t) \cdot x_0$
      $x \sim \mathcal{N}(\mu_t, \sigma^2 I)$
      if $m \leq 0.5$ then
          $k \sim \text{randint}(1, K_{\text{max}})$
          With torch.no_grad():
            $\hat{x}_1 = f_{\theta,t}(x)$
          $x_1\_list = [\,]$
          for $i = 0, \ldots, k$ do
              $\hat{x}_1 \leftarrow f_{\theta,t}(\hat{x}_1.detach())$
              $x_1\_list.append(\hat{x}_1)$
          end for
          $L_{\text{R}} \leftarrow \frac{1}{|\text{x1\_list}|} \sum_{\hat{x}_1 \in \text{x1\_list}} \|\hat{x}_1 - x_1\|^2$
      else
          $L_{\text{G}} \leftarrow \|f_{\theta,t}(x) - x_1\|^2$
      end if
      $\theta \leftarrow \text{Update}(\theta, \nabla_\theta L_{\text{G/R}})$
  end while
  return $f_\theta$
```

Algorithm 2 Predictor Refiner Sampler

```
Require: prior distribution $p_0$, number of integration steps $T$, and trained function $f_\theta$
  steps $\leftarrow 1$
  $\Delta t \leftarrow \frac{1}{T}$
  $t \leftarrow 0$
  $x_0 \sim p_0(x_0)$
  $x_t \leftarrow x_0$
  while steps $\leq T - 1$ do
      $\hat{x}_1 \leftarrow f_{\theta,t}(x_t)$
      $\hat{x}_1 \leftarrow f_{\theta,t}(\hat{x}_1)$
      $x_t \leftarrow x_t + \Delta t \cdot \frac{\hat{x}_1 - x_t}{1-t}$
      $t \leftarrow t + \Delta t$
      steps $\leftarrow$ steps + 1
  end while
  return $x_t$
```

Furthermore, considering the energy minimum x' of Eq. 23 is equivalent to the gradient of the energy function equal to zero, and we could obtain the x' by a neural network G$(\hat{x}_1)$ which could be seen as a neural refiner that projects the $\hat{x}_1$ to the high density region of the data manifold. The gradient of the likelihood of $\hat{x}_1$ could be written as:

$$G(\hat{x}_1) = x' \qquad \nabla E(\hat{x}_1) = -\nabla \log p(\hat{x}_1|x_1) = \frac{1}{\sigma^2}(\nabla G(\hat{x}_1))(G(\hat{x}_1) - x_1) \tag{24}$$

Observing Eq. 24, the term could be optimized if the neural refiner G$(\hat{x}_1)$ approximates the x_1, which aligns with the proposed energy objective 13:

$$L_G = ||G(\hat{x}_1) - x_1||^2. \tag{25}$$

A.3 Idempotency and Stability

Idempotency and stability are two fundamental notions over many disciplines, for example, formation control, dynamical systems and molecular dynamics. It has been recently explored for sampling from stable distribution [33] or being used as the building block for generative models [29]. Idempotency helps learn a robust function where additional function evaluations doe not drastically alter the output. This relates to stability in terms of the loss used during training such that the iteratively generated output stays on the data manifold, a desirable property for any generative model. Moreover, the concept of stability could be further expanded into any energy function, combined with flow models. Existing work [39, 40] proposed to learn a stable vector field to be a Poisson field implicitly governed by the potential energy induced by the training sample over the augmented space of the data.

B Algorithm Details

Algorithms 1 and 2 are pseudocode for the training and sampling algorithms for the IDFlow.

C Experimental Details

C.1 Task Formulation and Notation

For the molecular docking task, we represent the protein structure as $\mathbf{y} \in \mathbb{R}^{3\times n_p}$ and the ligand structure as $\mathbf{x} \in \mathbb{R}^{3\times n_l}$, where n_p and n_l are the number of residues of the protein and the number of atoms of the ligand. The aim is to learn the conditional distribution $p(x|y)$, the docking distribution of the ligand x given the protein structure y.

Single ligand docking assumes a unique pair of the protein and ligand (x, y), such that the generative model learns the binding mode of the protein y with *a single sample* from the condition distribution $p(x|y)$.

Multi-ligand docking assumes multiple pairs of the protein y and the ligand x_i: $(x_1, y), (x_2, y), ..., (x_n, y)$, where n is the number of ligands for the protein y. The training signal of multi-ligand docking could be richer with several docking conformation x_i provided for learning the condition distribution $p(x|y)$.

C.2 Datasets

The PDBBind v2020[23] with a total of 19k complexes timesplit is a commonly used benchmark for molecular docking [34, 24, 10, 44, 26, 8]. The split proposed by [34] consists of 17k complexes before 2019 for training and validation and 363 complexes after 2019 for testing without the seen ligand in the training set. The 30% sequence similarity split is constructed from the same dataset but with the constraint of the chain-wise similarity less than 30%, which is considered a more difficult split for the timesplit.

BindingMOAD [13] is another curated dataset from the PDB, with a different preprocessing pipeline from the PDBBind, ending up with 41k complexes. The dataset has been recently explored for more challenging benchmark construction and multi-ligand docking. Similar to the PDBBind, the maximum 30% sequence similarity split provides 56649, 1136 and 1288 for training, validation and testing. Only one biounit for each complex is used for training. The complex with only one contact (protein residue ligand atom distance less than 4) is further filtered out, retrieving 36203, 734 and 756 training, validation and test examples.

C.3 Pocket Definition

Radius Pocket. The pocket center is the mean position of the protein residues of which the minimum distance to any ligand atoms is less than 8Å. The radius is computed as the maximum between 5Å and the ligand radius (half of the largest distance between the ligand atoms) plus the radius pocket buffer (set to 7Å in all experiments). The pocket residues are selected based on the comparison between the residue pocket center distance and the radius. The residue pocket center distance is randomly flipped by the $\sigma = 2$.

Distance Pocket. The protein residue ligand atoms distances are extracted by the minimum distance between the residue and any ligand atom positions. Again, these protein-ligand distances are further randomly shifted with the $\sigma = 2$. The final pocket residues are the ones whose distances are below 14Å.

C.4 Architecture and Hyperparameters.

We maintain the same backbone architecture as [35], Equivariant Tensor Field Networks (TFN [36]) are leveraged for predicting the ligand atom coordinates. The equivariant TFN refinement layer maintains an EGNN [28] like message passing paradigm, but the learned path weights are parametrized via the tensor products by the spherical harmonics of the edge vector and the node features. There is no higher order representation (>1) being used in the experiment and we do not use the batch normalization and residual connection for the aggregated messages, but only layernorm the input features for each layer. We strictly follow the hyperparameter settings in HarmonicFlow

Table 3: Ablations on timesplit radius pocket docking. "%<2" is the fraction of the prediction that the root mean squared deviation (RMSD) is less than 2 Ångström. "Meds" refers to the median RMSD.

	%<2	Med.
IDFlow	34.7 (0.17)	3.1 (0.09)
number of iterations $k_{max} = 0$ (HarmonicFlow)	28.3	3.2
number of iterations $k_{max} = 1$	34.2 (2.42)	3.0 (0.05)
no timestep information	34.0 (0.54)	3.0 (0.15)

Table 4: Top-40, 10 and 5 accuracy comparison of methods on the PDBBind splits for pocket level docking based on different metrics. "%<2" or "%<1" is the fraction of the prediction that the root mean squared deviation (RMSD) is less than 2 or 1 Ångström. "Meds" refers to the median RMSD.

	Sequence Similarity Split						Time Split					
	Distance-Pocket			Radius-Pocket			Distance-Pocket			Radius-Pocket		
	%<1	%<2	Med.	%<1	%<2	Med.	%<1	%<2	Med.	%<1	%<2	Med.
HarmonicFlow (40)	30.7	63.2	1.5	16.1	46.8	2.1	35.8	66.9	1.3	21.9	52.4	1.9
IDFlow (40)	34.3	64.0	1.4	18.6	48.0	2.1	37.6	67.1	1.2	27.8	53.4	1.8
HarmonicFlow (10)	22.0	53.9	1.8	9.1	36.7	2.4	25.8	58.8	1.6	15.8	45.4	2.2
IDFlow (10)	25.5	56.2	1.7	14.2	40.9	2.4	29.6	60.9	1.5	22.3	47.4	2.2
HarmonicFlow (5)	15.4	47.5	2.1	7.7	31.2	2.8	19.3	55.0	1.7	11.8	42.0	2.5
IDFlow (5)	20.6	52.0	1.9	10.0	34.3	2.6	24.7	57.9	1.6	17.8	44.9	2.4

to ensure a fair comparison. All experiments are conducted on 8 NVIDIA Tesla V100 GPUs. The detailed setting can be found here[1].

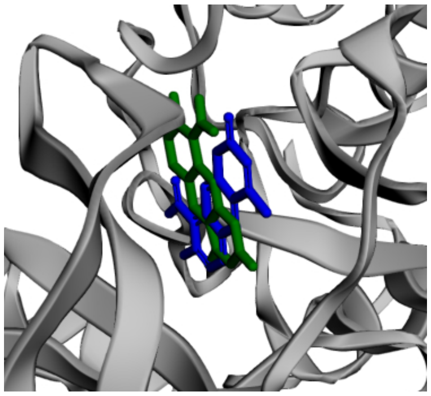
D More Results

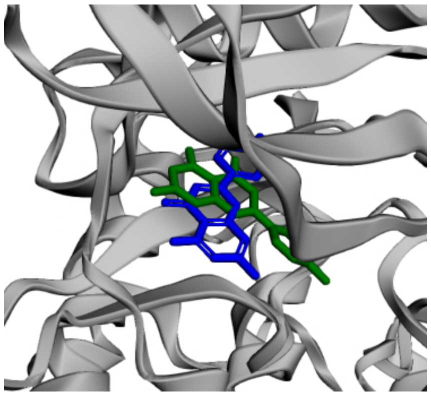
Table 3 shows the ablation on the design choice of the IDFlow. The experiments are averaged over three runs, and the value in the parenthesis is the standard deviation. Table 4 shows the top-k accuracy on the PDBBind of single ligand docking.

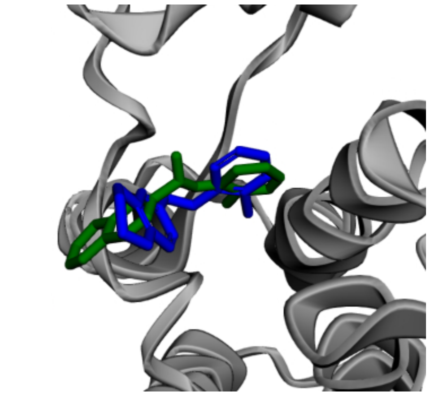
[1]https://github.com/HannesStark/FlowSite

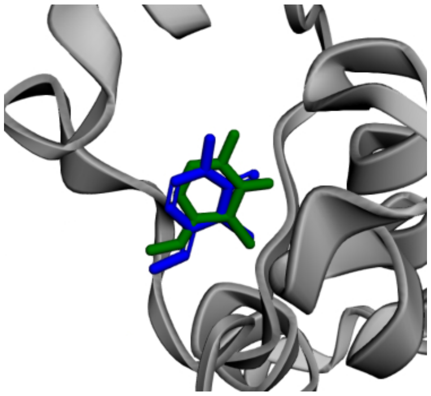
E Visualizations

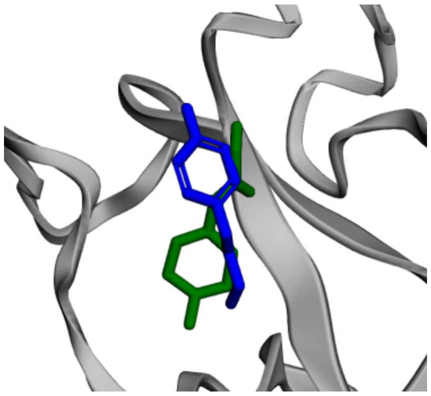
Figure 1 exhibits some randomly selected generated molecules compared with the Ground Truth. The sample is generated by 20 NFEs.


(a) 3pwd


(b) 4dgm


(c) 5l98


(d) 5mkx


(e) 6fap

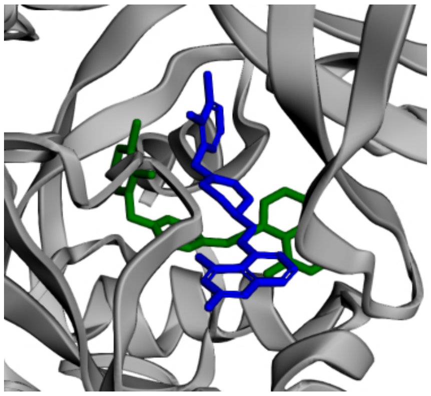

(f) 3bl9

Figure 1: Six randomly selected generated complexes on the radius pocket docking on 30% sequence similarity split. The one in blue is the ground truth, and the one in green is generated from IDFlow.