S Supplementary Information

S.1 Computational Resources

All experiments in Section 2.3, Section 2.5, Section 2.7, Section 2.8 were run on an AWS instance with 8 CPU cores and an NVIDIA A10G GPU with 24 GB of VRAM. In Section 2.6 Virtual screening with QVINA on ZINC50K was run a compute-optimized AWS instance with 8 CPU cores, while IDOLpro was run on an AWS instance with 8 CPU cores and an NVIDIA A10G GPU with 24 GB of VRAM. Experiments in Section 2.4 were run on a Google Cloud compute virtual machine instance with 8 cpu cores and an NVIDIA T4 GPU with 16 GB VRAM.

S.2 More Details on Training the TorchSA Model

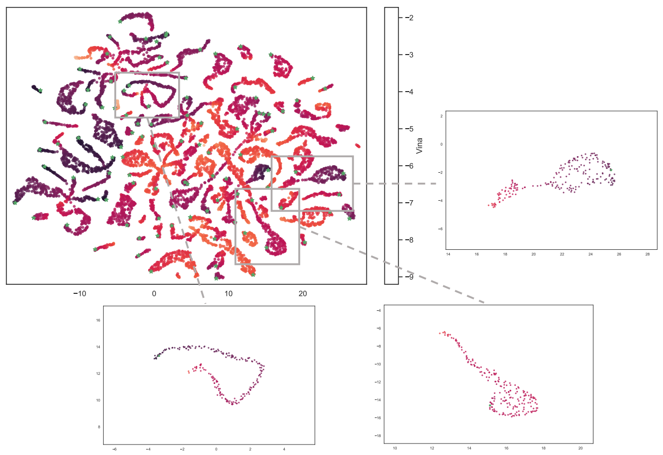

Fig. S1 Training and validation curves for training PaiNN to predict the SA score on 3D atomic point clouds. The MSE at each epoch is plotted for the training set, the ChemBL validation set, the DiffSBDD validation set, and the CrossDocked validation set.

To train the SA model, we prepare a dataset consisting of all molecules with structural information in ChemBL[80] (2,409,270 structures), and ligands used to train DiffSBDD[19] on CrossDocked2020 [38] (183,468 structures). Although the SA score is fully determined by the chemical graph of a molecule, we keep molecules with different conformations from CrossDocked2020 to aid the model in learning the redundancy of pose in determining the SA score. To improve the model's performance on ligands produced by DiffSBDD, we generate nearly 1,000,000 (877,284) ligands with DiffSBDD which are included in the training data. We generate several ligands for each of the protein pockets in the DiffSBDD training set and then filter them using the same validity checks described in the Methods section. We put a higher emphasis on modelling ligands from CrossDocked2020 and DiffSBDD, sampling from one of these datasets during training with a $5\times$ higher likelihood than ChEMBL.

We train the polarizable atomic interaction neural network[81] (PaiNN) from the Open Catalyst Project[82,83] to predict the SA score given the atomic coordinates and atom types. To allow PaiNN to make predictions on atom types coming out of DiffSBDD, we encode atom types as one-hot vectors. We first optimize the hyperparameters of the model using Ray Tune[84]. The hyperparameters chosen were *num_rbf* = 64 *num_layers* = 4, *max_neighbor* = 30, *cutoff* = 8.0, *hidden_channels* = 256. We use a 95%/5% training/validation split for each dataset. The model is trained for 100 epochs to minimize the MSE loss with the AdamW optimizer[85] with a learning rate of 5×10^{-4}. Training and validation curves are plotted in Fig. S1.

S.3 More Details on Training the Regressor Guidance Model

To train the regressor used in Section 2.4, we prepare a dataset of latent vector trajectories from the reverse diffusion process of DiffSBDD. The training set for DiffSBDD consists of 100,000 protein-ligand pairs from the CrossDocked[38] dataset. Each reverse diffusion trajctory of DiffSBDD consists of 500 denoising diffusion steps. In order to reduce the size of the training data, we take a random 10% sample of DiffSBB's training data, resulting in 10,000 protein pockets which can be used to seed the generation of DiffSBDD. For each protein pocket, we generate a full reverse diffusion trajectory (500 denoising diffusion steps). Each latent vector in the trajectory $\mathbf{z}_\ell^t$ is labeled with the SA score of the final generated molecule, i.e., $SA(\mathbf{z}_\ell^\mathbf{0})$. This results in 5,000,000 data points for training the regressor.

We train an equivariant graph neural network (EGNN)[65] with 4 layers and 256 hidden channels to predict the SA score of the final DiffSBDD molecule given an intermediate noisy latent vector. We also supply the model with the corresponding timestep in the reverse diffusion process. The model is trained for 10 epochs to minimize the MSE loss with the Adam optimizer[85] with a learning rate of 1×10^{-4}. The training curve is plotted in Fig. S2.

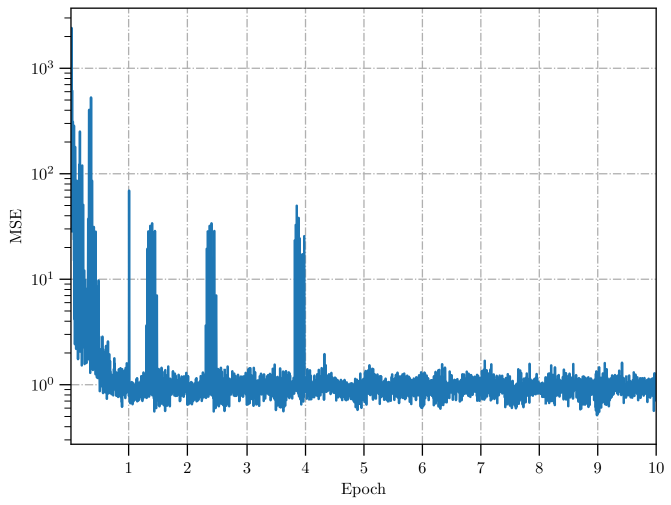

Fig. S2 Loss curve for training EGNN to predict the SA score of molecules produced by DiffSBDD given intermediate noisy latent vectors. The MSE at each step is plotted.

S.4 Hyperparameter Tuning

S.4.1 Accelerating Diffusion

In DiffSBDD, the models are trained to generate ligands over 500 reverse diffusion steps according to some noise schedule. At each

PDB ID	Ligand ID
2ah9	cto
5lvq	p2l
5g3n	u8d
1u0f	g6p
4bnw	fxe
4i91	cpz
2ati	ihu
2hw1	lj9
1bvr	geq
1zyu	k2q

Table S1 Validation set used to choose hyper-parameters in IDOLpro. All proteins from the test set of LiGAN[?] were used, and a single protein pocket for each protein was selected at random.

Diffusion steps	Vina [kcal / mol]	SA	QED	Time [s]
5	-6.25 ± 2.08	3.72 ± 0.46	0.53 ± 0.10	60.60 ± 49.75
50	-6.72 ± 2.45	3.92 ± 0.58	0.54 ± 0.10	196.96 ± 102.97

Table S2 Results when reducing the number of rollout steps from 50 to 5. The average Vina, SA, and QED across the validation set is reported.

t_{hz}	Δ Vina	Δ SA	Δ QED
50	-1.21	-0.34	-0.028
100	-1.17	-0.82	0.004
200	-1.84	-0.76	0.048

Table S3 Results when varying the optimization horizon t_{hz} in IDOLpro. The difference in Vina, SA, and QED for the final optimized ligands produced by IDOLpro relative to the initial ligands produced by DiffSBDD are reported.

time step an equivariant network takes in the noised coordinates and atom types, as well as the time step, and returns a denoised representation of the atoms and coordinates[19]. One can reduce the number of reverse diffusion steps by skipping time steps in the noise schedule.

In IDOLpro, the majority of the reverse diffusion process is run to generate $\mathbf{z}_{t_{hz}}$, i.e., to seed the initial latent vectors. When optimizing $\mathbf{z}_{t_{hz}}$, the rollout of $\mathbf{z}_{t_{hz}}, \ldots, \mathbf{z}_0$ needs to be repeated many times. We run an experiment to determine whether we can reduce the number of steps during this final rollout 10-fold without a significant degradation in performance. We run an experiment with the smallest horizon considered $t_{hz} = 50$. Results are shown in table Table S2. Running the rollout with reduced diffusion steps results in over a $3\times$ speedup, with a slight decrease in average Vina score ($<$ 0.5 kcal /mol), while preserving average SA and QED. We adopt this setting in our pipeline for this reason.

S.4.2 Tuning Optimization Horizon

We tune the value of the optimization horizon, t_{hz}, to optimize both the Vina and SA scores of generated molecules. We consider $t_{hz} \in 50, 100, 200$. For each setting of the optimization horizon, we track the difference in Vina score, SA score, and QED. Results are reported in Table S3. Based on these results we set the optimization horizon to 200, since that setting resulted in by far the best difference in Vina score and QED, albeit a slightly worse improvement in SA score relative to setting $t_{hz} = 100$.

S.4.3 Structural Refinement with Torchvina and ANI2x

We use the following parameters in the L-BFGS optimization algorithm: *max_iter*=100, *tolerance_grad*=10^{-3}, and *line_search_fn*="strong_wolfe". On top of using intra- and inter-molecular forces derived from the ANI2x[37] and torchvina potentials, we inlcude an $L1$ penalty for violating the bonds in the molecule produced by IDOLpro. To do so, we use ASE's[68] natural cutoffs. We find that setting a weight of 0.01 on this $L1$ penalty result in the best balance of Vina score and validity.

S.4.4 Stopping Criteria, Backtracking, and Decaying Learning Rate

We use per-parameter options in Pytorch[35] to allow for individualized learning rates for different ligands. For each ligand, we optimize it with Adam with the chosen hyperparameters. We optimize each latent vector for 10-200 optimization steps. Often, during latent vector optimization, a ligand will be pushed to a part of latent space such that it becomes invalid. In such a case, we attempt to generate a ligand 10 times with the given latent vector. If after 10 attempts, reverse diffusion has not produced a valid ligand, we backtrack to the previous latent vector in the optimization trajectory, reduce the learning rate by a factor of 10, and restart the optimization. If at another point in the optimization, with the reduced learning rate, another latent vector fails to generate a valid ligand over 10 attempts, the optimization of that trajectory is stopped.

S.5 Visualization of Latent Vectors

Latent vector visualization was performed with UMAP[86] and visualized in Fig. S3.

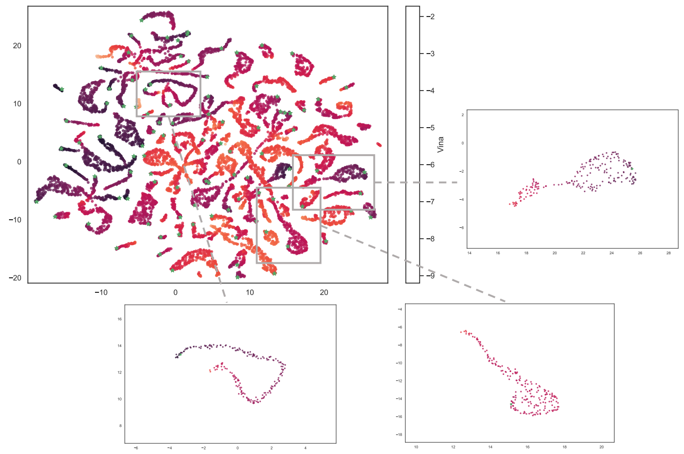

Fig. S3 Latent vector visualizations of IDOLpro when generating ligands for 14gs. The points are coloured by Vina score (darker implies lower scores), and a green star marks the end of each optimization trajectory.

Supplementary References

80 B. Zdrazil, E. Felix, F. Hunter, E. J. Manners, J. Blackshaw, S. Corbett, M. de Veij, H. Ioannidis, D. M. Lopez, J. F. Mosquera *et al.*, *Nucleic acids research*, 2024, **52**, D1180–D1192.

81 K. Schütt, O. Unke and M. Gastegger, International Conference on Machine Learning, 2021, pp. 9377–9388.

82 L. Chanussot, A. Das, S. Goyal, T. Lavril, M. Shuaibi, M. Riviere, K. Tran, J. Heras-Domingo, C. Ho, W. Hu *et al.*, *Acs Catalysis*, 2021, **11**, 6059–6072.

83 R. Tran, J. Lan, M. Shuaibi, B. M. Wood, S. Goyal, A. Das, J. Heras-Domingo, A. Kolluru, A. Rizvi, N. Shoghi *et al.*, *ACS Catalysis*, 2023, **13**, 3066–3084.

84 R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez and I. Stoica, *arXiv preprint arXiv:1807.05118*, 2018.

85 I. Loshchilov and F. Hutter, International Conference on Learning Representations, 2019.

86 L. McInnes, J. Healy and J. Melville, *arXiv preprint arXiv:1802.03426*, 2018.