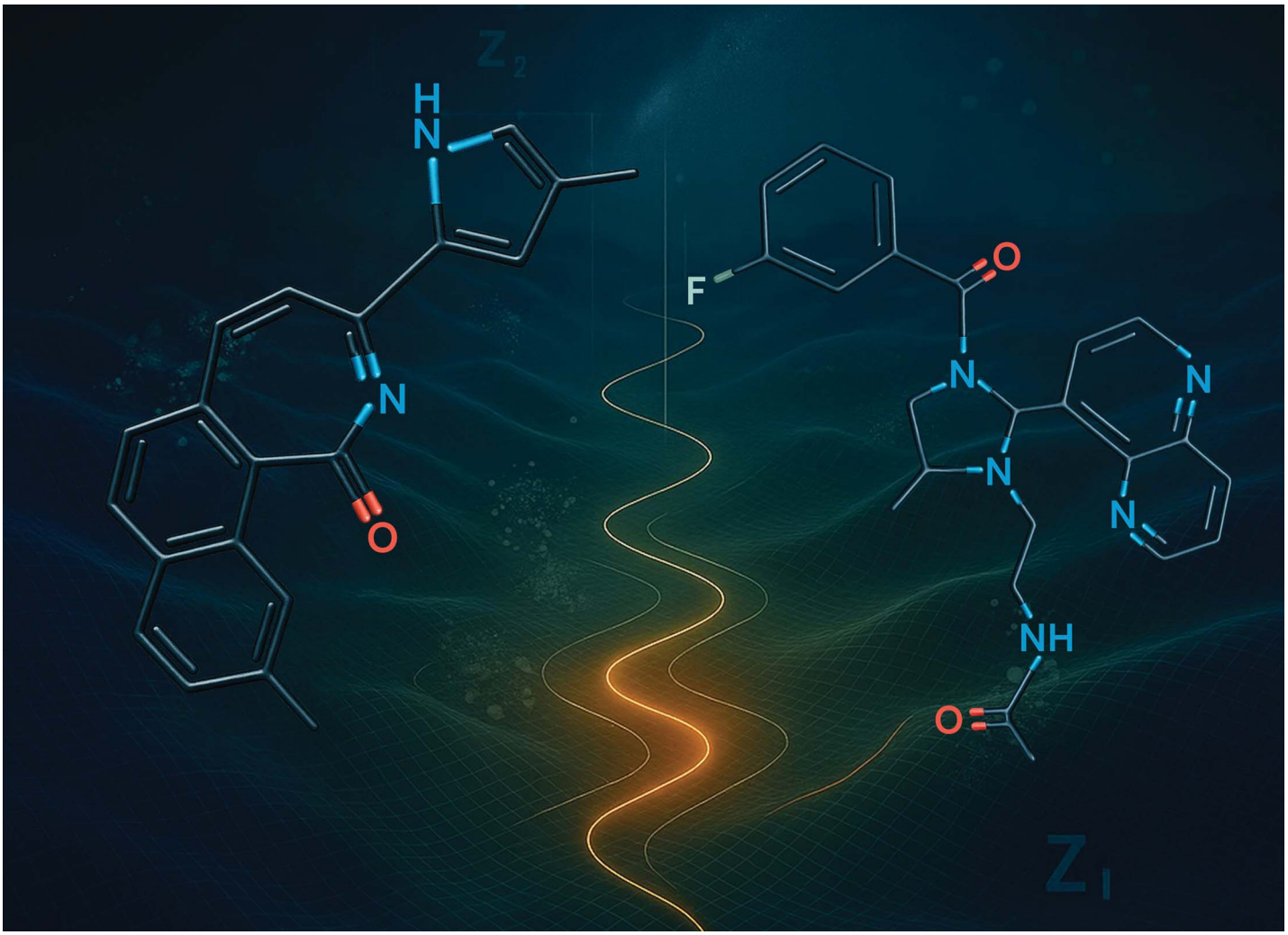

Showcasing research from Amit Kadan, Dr Kevin Ryczko and Dr Takeshi Yamazaki, SandboxAQ, California, USA.

Guided multi-objective generative AI to enhance structure-based drug design

In this work we present IDOLpro, a modular framework for guided diffusion which can generate molecules with a plurality of optimized properties for structure-based drug design, accelerating the drug discovery process.

Image reproduced by permission of SandboxAQ from *Chem. Sci.*, 2025, **16**, 13196.

As featured in:

See Amit Kadan, Kevin Ryczko, Takeshi Yamazaki *et al.*, *Chem. Sci.*, 2025, **16**, 13196.

ROYAL SOCIETY OF CHEMISTRY

15 YEARS ANNIVERSARY

rsc.li/chemical-science

Registered charity number: 207890

Chemical Science

ROYAL SOCIETY OF CHEMISTRY

EDGE ARTICLE

View Article Online
View Journal | View Issue

Cite this: *Chem. Sci.*, 2025, **16**, 13196

All publication charges for this article have been paid for by the Royal Society of Chemistry

Guided multi-objective generative AI to enhance structure-based drug design†

Amit Kadan, ‡*[a] Kevin Ryczko, ‡*[a] Erika Lloyd,[a] Adrian Roitberg[b] and Takeshi Yamazaki *[a]

Generative AI has the potential to revolutionize drug discovery. Yet, despite recent advances in deep learning, existing models cannot generate molecules that satisfy all desired physicochemical properties. Herein, we describe IDOLpro, a novel generative chemistry AI combining diffusion with multi-objective optimization for structure-based drug design. Differentiable scoring functions guide the latent variables of the diffusion model to explore uncharted chemical space and generate novel ligands *in silico*, optimizing a plurality of target physicochemical properties. We demonstrate our platform's effectiveness by generating ligands with optimized binding affinity and synthetic accessibility on two benchmark sets. IDOLpro produces ligands with binding affinities over 10–20% higher than the next best state-of-the-art method on each test set, producing more drug-like molecules with generally better synthetic accessibility scores than other methods. We do a head-to-head comparison of IDOLpro against an exhaustive virtual screen of a large database of drug-like molecules. We show that IDOLpro can generate molecules for a range of important disease-related targets with better binding affinity and synthetic accessibility than any molecule found in the virtual screen while being over 100× faster and less expensive to run. On a test set of experimental complexes, IDOLpro is the first to produce molecules with better binding affinities than the experimentally observed ligands. IDOLpro can accommodate other scoring functions (*e.g.* ADME-Tox) to accelerate hit-finding, hit-to-lead, and lead optimization for drug discovery.

Received 6th March 2025
Accepted 28th April 2025

DOI: 10.1039/d5sc01778e

rsc.li/chemical-science

1 Introduction

The central goal of structure-based drug design (SBDD) is to design ligands with high binding affinities to a target protein pocket given the 3-dimensional information.[1] SBDD is inherently an inverse design problem, where the desired properties (high binding affinity to a target protein, synthesizability, *etc.*) are known, but the design of a molecule with the desired properties is non-trivial. Inverse design problems are prevalent in the materials,[2–4] chemical,[5–7] and life sciences.[8–10] They have two fundamental steps: The sampling of a chemical space, and the evaluation of compounds' abilities to satisfy the set of desired properties. The sampling of chemical space can be done in various ways. For drug discovery, this is typically done by evaluating each entry of a large database of drug-like molecules such as ZINC,[11] Enamine,[12] or GDB,[13] collecting the results and ranking them to yield a shortlist of compounds to be screened in a laboratory. This approach is time-consuming and is restricted to searching through a fraction of drug-like chemical space - despite some of these databases contain hundreds of billions of molecules,[13] drug-like chemical space is estimated to number between 10^{20}–10^{60} molecules.[14]

Deep-learning (DL) based generative models for SBDD can replace virtual screening by directly predicting high-affinity ligands for a given protein pocket.[15–20] Initially, these models were predominantly autoregressive. Ref. 20 develops a sampling scheme based on Monte-Carlo tree search to condition an autoregressive generative model to generate molecules directly into the protein pocket. Ref. 16 trains an equivariant graph neural network called Pocket2Mol to predict novel molecules in a protein pocket by sequentially predicting atoms of the ligand given the current atomic context. Recently, several diffusion-based generative models have been proposed.[18,19] Diffusion models learn to sample molecules by learning to reverse a diffusive noising process.[21] Unlike autoregressive techniques, these models allow for consideration of global relationships between atoms in the ligand throughout generation. Ref. 18 introduces an equivariant diffusion model called TargetDiff which generates ligands from scratch directly in the protein pocket. They also showed that with proper parameterization, their model can be used as an unsupervised binding affinity

[a]SandboxAQ, Palo Alto, CA, USA. E-mail: amit.kadan@sandboxaq.com; kevin.ryczko@sandboxaq.com; takeshi.yamazaki@sandboxaq.com

[b]Department of Chemistry, University of Florida, PO Box 117200, Gainesville, Florida, 32611-7200, USA

† Electronic supplementary information (ESI) available. See DOI: https://doi.org/10.1039/d5sc01778e

‡ Equal Contributions.

predictor, allowing them to filter promising candidates during generation. Similarly, ref. 19 introduces two equivariant diffusion models for SBDD. On top of *de novo* generation, these models are able to perform scaffold-hopping and fragment merging, both of which are facilitated by an inpainting-inspired molecular generation scheme.[22]

DL-based generative models are more efficient than database iteration and can produce molecules that don't currently exist in commercial databases, expanding the chemical space available during the drug-discovery process.[20] However, in some cases they have been shown to produce ligands that do not exhibit desired physicochemical properties, producing compounds with unphysical structures,[23,24] or producing compounds that exhibit poor synthesizability.[25] Just like in database generation, to operate in an inverse design pipeline, molecules generated by these models must be filtered and ranked with physicochemical descriptors, and there is no guarantee of finding a molecule satisfying a set of desired properties.

Alternatively, guidance from physicochemical scores can be directly integrated into the generation process.[6,8,26] Ref. 20 pairs a policy network with Monte Carlo tree search to design ligands with optimized Vina scores. They apply their methodology to design an inhibitor for the main protease of SARS-CoV-2, producing novel ligands with high binding affinity. Ref. 19 pairs a diffusion model with an evolutionary algorithm to optimize the properties of generated ligands. Ref. 27 uses property predictors to guide a generative model composed of an equivariant autoencoder and transformer decoder in generating molecules with good target properties. These works do not make use of gradient information in their optimization schemes. Gradient information forms the basis of many modern optimization algorithms,[28] particularly in DL applications, allowing for optimal scaling with the degrees of freedom of the optimization.[29] Ref. 30 introduces classifier guidance, a technique that allows diffusion models to generate samples conditioned on the gradients of classifiers throughout the reverse diffusion process. Similarly, one can use gradients from regressors trained to quantify properties of interest. Regressor guidance has been implemented in recent works on generative molecular design.[8,10,31] Ref. 31 uses regressor guidance to repurpose a generative model trained to generate gas-phase molecules for structure-based drug design. Ref. 10 uses both regressor and classifier guidance to generate optimized polycyclic aromatic systems - molecules important for the design of organic semiconductors. Ref. 8 uses regressor guidance to design drug-like molecules with optimized target properties including protein–ligand interaction, drug-likeness, and synthesizability. A drawback of regressor guidance is that it requires training ad-hoc property predictors on intermediate states in the reverse diffusion of a particular generative model, and cannot be used with an abundance of state-of-the-art physicochemical scores and models designed to make predictions on three-dimensional molecules.[32–34]

In this work, we present IDOLpro (**I**nverse **D**esign of **O**ptimal **L**igands for **Pro**tein pockets), a generative workflow that actively guides a diffusion model to generate an optimized set of ligands for a target protein pocket. Specifically, we modify the latent variables of a diffusion model at a carefully chosen time step in the reverse diffusion process to optimize one or more objectives of interest simultaneously. We achieve this by backpropagating gradients obtained by evaluating physicochemical scores directly on the generated molecular structures. This avoids the need for developing ad-hoc regressors capable of making predictions on intermediate noisy representations, and allows us to directly leverage common scores for assessing ligand quality. In this report, we include binding affinity and synthetic accessibility, but our framework is highly modular and can incorporate alternative generators and additional scores. All metrics are written in PyTorch[35] and are fully differentiable with respect to the latent variables within the diffusion model, allowing for the use of gradient-based optimization strategies to design optimal ligands.

2 Results

2.1 Workflow

IDOLpro takes the protein pocket information as input and iteratively modifies the predictions of a generator that generates molecules directly into the pocket to produce optimal ligands according to a set of differentiable scores defining target properties. IDOLpro accomplishes this by modifying the latent vectors of the generative model with gradients from the property predictors. As shown in Fig. 1, IDOLpro freezes the latent vectors at a specified time step in reverse diffusion - the optimization horizon denoted by t_{hz}. IDOLpro then unwinds the rest of the reverse diffusion process, using gradients from differentiable property predictors to update the frozen latent vectors. This procedure is repeated iteratively, resulting in an improved sample with respect to the scores at the end of the optimization. Once a set of optimized ligands is produced, their binding poses are refined by local structural optimization within the protein pocket. Structural refinement interfaces with the same differentiable scores used for latent optimization and iteratively modifies a ligand's coordinates with gradients from the property predictors.

In this report, our method uses DiffSBDD[19] as the baseline generative method for predicting ligands within a protein pocket. We use a specific variant of the model, DiffSBDD-Cond, which we found was able to generate ligands within the protein pocket without clashes more reliably than the alternative, DiffSBDD-inpaint. We assess the ability of our framework to discover novel ligands with improved binding affinity and synthetic accessibility. To evaluate and be able to gather gradient information for binding affinity, we have developed a torch-based implementation of the Vina score,[32] which we refer to as torchvina. To evaluate synthetic accessibility, we have trained an equivariant neural network model to predict the synthetic accessibility (SA) score first proposed in ref. 36, which we refer to as torchSA. During structural refinement, we also make use of the ANI2x[37] neural network potential, which we use to optimize intramolecular forces, allowing IDOLpro to produce physically valid molecules.

2.2 Datasets

We assess the performance of IDOLpro on three different test sets - a subset of CrossDocked,[38] a subset of the Binding MOAD (Mother of all Databases),[39] and on a test set consisting of

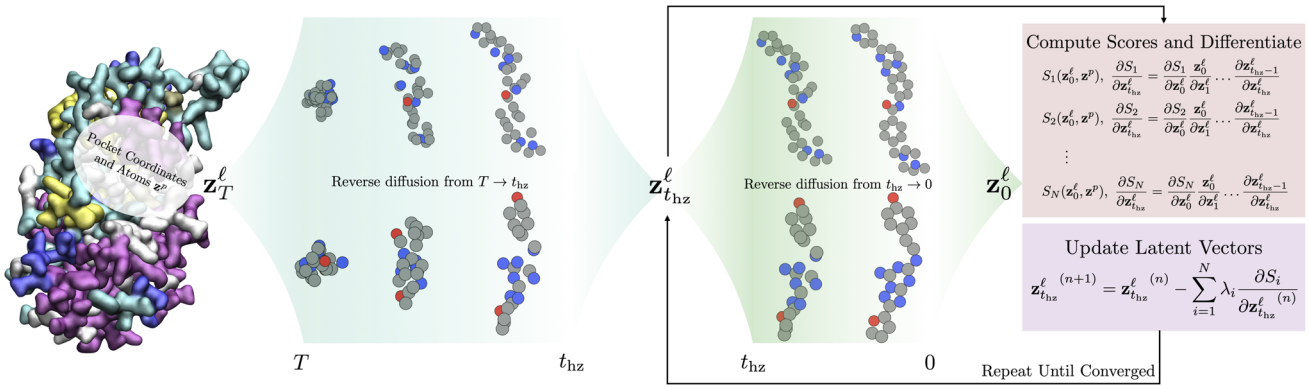

Fig. 1 Visual overview of IDOLpro. A random latent vector, $\mathbf{z}^\ell_T$, is sampled for each ligand in the batch conditioned on the pocket coordinates, $\mathbf{z}^p$. Reverse diffusion is run from time T to the optimization horizon, time t_{hz}. The rest of the diffusion process is completed, and the ligands are scored by evaluating a set of differentiable scores defining target physicochemical properties. The gradient with respect to each latent vector at the optimization horizon, $\partial S_i/\partial z_{t_{hz}}{}^\ell$, is used to take a gradient step in the latent space. This process is iterated until a maximum number of steps have been reached, or a valid ligand cannot be generated with the current latent vector.

disease-related proteins first proposed in ref. 40, which we refer to as the RGA test set.

All three databases contain protein pocket–ligand pairs. The CrossDocked dataset contains 100 pocket–ligand pairs which were derived *via* re-docking ligands to non-cognate receptors with smina.[41] This test set was used to validate the performance of tools in several other papers,[15,16] including DiffSBDD.[19] The Binding MOAD contains 130 high-resolution (<2.5 Å) experimentally derived pocket-ligand pairs extracted from the Protein Data Bank (PDB). This test set was also used to assess the performance of DiffSBDD. The RGA test set consists of 10 disease-related protein targets including G-protein coupling receptors (GPCRs) and kinases from the DUD-E dataset,[42] as well as the SARS-CoV-2 main protease.[43] This is also an experimentally derived test set and was used to assess the performance of several non-deep learning-based methods in ref. 40.

2.3 Validation of latent vector optimization

To assess the ability of IDOLpro to augment the performance of the baseline model *via* the optimization of latent vectors, we run IDOLpro while optimizing torchvina and torchSA, and analyze its capability to improve the Vina and SA scores relative to DiffSBDD-Cond. For each of the protein pockets in the CrossDocked and Binding MOAD test sets, we generate 100 optimized ligands using IDOLpro. Generated ligands do not undergo structural refinement to isolate the effect of optimizing latent vectors.

We report a number metrics averaged across the protein pockets in each test set. Each metric is recorded before and after optimization with IDOLpro. Reported metrics include the Vina and top-10% Vina scores of generated ligands, and three metrics using the SA score - the average SA, top-10% SA and the percentage of synthesizable molecules generated. In this work, we define a ligand as synthesizable if it achieves an SA score of less than 3.5. Although the inventors of the SA score suggest 6 as the cutoff for synthesizability, a number of papers have found SA scores between 3.5 and 6 to be ambiguous.[25,33] A cutoff of 3.5 was used to determine synthesizability in ref. 44. We make note of the average QED (quantitative estimate of drug-likeness),[45] a metric combining several desirable molecular properties for screening drug-like molecule, and pocket diversity - the average pairwise dissimilarity between all generated molecules for a given protein pocket. Dissimilarity is measured as 1–Tanimoto similarity. Lastly, we use PoseBusters[23] to assess the validity of generated molecules. PoseBusters assesses the validity of generated molecules by running a series of checks using RDKit,[46] ensuring stereochemistry and intra- and intermolecular constraints such as appropriate bond lengths, planarity of aromatic rings, and no overlap with the protein pocket are satisfied.

The results are shown in Table 1. We find that IDOLpro generates ligands with better Vina and SA scores than DiffSBDD-cond, yielding molecules with ≈20% better Vina scores and ≈21% better SA scores on the CrossDocked test set, and ≈26% better Vina scores and ≈21% better SA scores on the Binding MOAD test set. Furthermore, IDOLpro yields more than double the amount of synthesizable ligands compared with DiffSBDD-cond for each dataset (51.2% *vs.* 23.5% and 56.9% *vs.* 22.6%). Lastly, despite not optimizing for these directly, we find that IDOLpro generates molecules with slightly higher QED and PoseBusters validity (PB-valid in Table 1) than DiffSBDD-Cond, achieving higher values on both the CrossDocked and Binding MOAD test sets. Despite IDOLpro performing an optimization in latent space, it does not decrease the diversity of generated molecules, achieving equal average pocket diversity to DiffSBDD-Cond on each test set.

We visualize a single example from each benchmark test set in Fig. 3. The figure shows an example of a molecule produced by DiffSBDD prior to latent vector optimization, and after latent vector optimization by IDOLpro. We also picture the corresponding reference molecule, showing that latent vector optimization allows IDOLpro to produce molecules with better metrics than the reference molecule. Overall, IDOLpro is able to co-optimize Vina and SA, producing molecules with

Table 1 Performance of IDOLpro when used to optimize torchvina and torchSA relative to the baseline model, DiffSBDD-cond on the CrossDocked and Binding MOAD test sets. The average Vina score, top-10% Vina score, SA score, top-10% SA score, percent of synthesizable molecules, QED, diversity, and PoseBusters validity are reported

Dataset	Method	Vina [kcal mol^{-1}]	Vina$_{10\%}$ [kcal mol^{-1}]	SA	SA$_{10\%}$	Synth [%]	QED	Diversity	PB-valid [%]
CrossDocked	DiffSBDD-cond[19]	−5.37 ± 1.93	−7.70 ± 2.25	4.30 ± 0.50	2.56 ± 0.63	23.5 ± 15.5	0.56 ± 0.05	0.78 ± 0.07	55.9 ± 11.1
	IDOLpro	−6.47 ± 2.10	−8.69 ± 2.45	3.41 ± 0.70	1.73 ± 0.51	51.2 ± 22.7	0.63 ± 0.06	0.79 ± 0.07	64.3 ± 8.3
MOAD	DiffSBDD-cond[19]	−5.38 ± 2.55	−7.66 ± 2.37	4.18 ± 0.50	2.50 ± 0.53	26.5 ± 17.8	0.58 ± 0.07	0.77 ± 0.07	53.6 ± 15.7
	IDOLpro	−6.77 ± 2.24	−8.68 ± 2.56	3.32 ± 0.66	1.79 ± 0.46	56.9 ± 22.6	0.63 ± 0.08	0.77 ± 0.07	60.4 ± 16.3

Table 2 Performance of IDOLpro compared to DiffSBDD with regressor guidance on the CrossDocked test set. The average SA, QED, diversity, PoseBusters validity, and average time to generate an individual molecule are reported

Method	SA	SA$_{10\%}$	Synth [%]	QED	Diversity	PB-valid [%]	Time [s/ligand]
Regressor guidance	3.53 ± 0.37	1.38 ± 0.18	50.6 ± 10.5	0.37 ± 0.01	0.90 ± 0.02	7.6 ± 3.6	8.11 ± 1.64
IDOLpro	2.79 ± 0.55	1.53 ± 0.40	71.51 ± 18.6	0.62 ± 0.06	0.81 ± 0.06	43.7 ± 16.2	82.33 ± 52.51

significantly better binding affinities and synthetic accessibility when compared to the baseline model, DiffSBDD-cond.

2.4 Comparison to regressor guidance

Next, we compare IDOLpro to regressor guidance - a standard technique for steering the predictions of a diffusion model to optimize a set of scores.[8,10,30,31] We run IDOLpro and DiffSBDD with regressor guidance to generate pocket-bound ligands while optimizing the SA score of generated moelcules. For each method, we generate 100 ligands for each protein pocket in the CrossDocked test set. We do not perform structural refinement after generation for either method to isolate each technique's ability to generate optimized molecules.

For each method, we report the SA score, top-10% SA score, percentage of synthesizable molecules, QED, pocket diversity, and PoseBusters validity averaged across all protein pockets in the test set. We also report the average time required by each technique for generating a single ligand. Full results are showing in Table 2. We find that IDOLpro generates ligands with better average SA scores than DiffSBDD with regressor guidance, yielding molecules with ≈27% better SA scores (2.79 *vs.* 3.53). IDOLpro also produces a significantly higher number of synthesizable molecules (>20% more) than DiffSBDD with regressor guidance. On the other hand, DiffSBDD with regressor guidance achieves a better top-10% SA score than IDOLpro. A major advantage of regressor guidance over IDOLpro is that it is ≈10× faster for generating a single ligand. This is expected as regressor requires running reverse diffusion once, while IDOLpro unwinds the latter portion of reverse diffusion multiple times. IDOLpro yields molecules with significantly better QED and PoseBusters validity than DiffSBDD with regressor guidance, suggesting that adding gradients throughout reverse diffusion hinders the model's ability to adequately account for both intra- and inter-molecular constraints. This is corroborated by the fact that DiffSBDD without regressor guidance produces significantly more PoseBusters-valid molecules than

Table 3 Evaluation of DL tools on targets from the CrossDocked and Binding MOAD datasets. The average, along with the standard deviation of each metric across the protein pockets in each dataset is reported. The top performing model on each metric is bolded in the corresponding column. Numbers for other models are taken from ref. 19. Time is based on running DiffSBDD-cond on our hardware, and adjusting the times reported in ref. 19 accordingly

	Method	Vina [kcal mol^{-1}]	Vina$_{10\%}$ [kcal mol^{-1}]	SA	QED	Diversity	Time [s/ligand]
CrossDocked	Test set	−6.87 ± 2.32	—	3.45 ± 1.26	0.48 ± 0.20	—	—
	3D-SBDD[15]	−5.89 ± 1.91	−7.29 ± 2.34	3.93 ± 1.26	0.50 ± 0.17	0.74 ± 0.09	328.13 ± 245.43
	Pocket2Mol[16]	−7.06 ± 2.80	−8.71 ± 3.18	**3.23 ± 1.08**	0.57 ± 0.16	0.74 ± 0.15	41.79 ± 36.84
	GraphBP[17]	−4.72 ± 4.03	−7.17 ± 1.40	7.24 ± 0.81	0.50 ± 0.12	**0.84 ± 0.01**	**0.17 ± 0.02**
	TargetDiff[18]	−7.32 ± 2.47	−9.67 ± 2.55	4.74 ± 1.17	0.48 ± 0.20	0.72 ± 0.09	∼57.22
	DiffSBDD-cond[19]	−6.95 ± 2.06	−9.12 ± 2.16	4.80 ± 1.17	0.47 ± 0.21	0.73 ± 0.07	2.27 ± 0.86
	DiffSBDD-inpaint[19]	−7.33 ± 2.56	−9.93 ± 2.59	5.01 ± 1.08	0.47 ± 0.18	0.76 ± 0.05	2.67 ± 1.22
	IDOLpro	**−8.04 ± 2.55**	**−10.96 ± 3.02**	3.41 ± 0.70	**0.63 ± 0.06**	0.79 ± 0.07	58.80 ± 32.97
MOAD	Test set	−8.41 ± 2.03	—	3.77 ± 1.08	0.52 ± 0.17	—	—
	GraphBP[17]	−4.84 ± 2.24	−6.63 ± 0.95	7.21 ± 0.81	0.51 ± 0.11	**0.83 ± 0.01**	**0.23 ± 0.03**
	DiffSBDD-cond[19]	−7.17 ± 1.89	−9.18 ± 2.23	4.89 ± 1.08	0.44 ± 0.20	0.71 ± 0.08	5.61 ± 1.42
	DiffSBDD-inpaint[19]	−7.31 ± 4.03	−9.84 ± 2.18	4.47 ± 1.08	0.54 ± 0.21	0.74 ± 0.05	6.17 ± 2.08
	IDOLpro	**−8.74 ± 2.59**	**−11.23 ± 3.12**	**3.32 ± 0.66**	**0.63 ± 0.08**	0.77 ± 0.07	82.30 ± 45.07

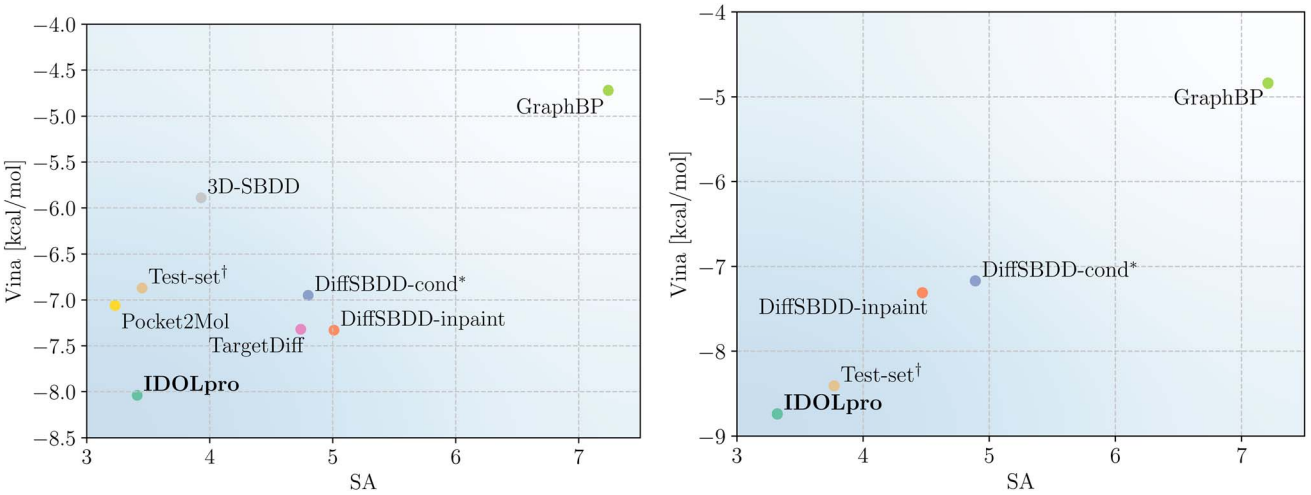

Fig. 2 Performance of DL tools on two benchmark test sets. The scatter plot shows the average Vina and SA score for each method for targets in CrossDocked (left), and the Binding MOAD (right). IDOLpro is at the bottom left of each scatter plot, showing it can co-optimize Vina and SA for generated ligands. † Vina scores of reference ligands in the test set, redocked with QuickVina2. * Baseline method for IDOLpro.

DiffSBDD with regressor guidance (see Table 1 for vanilla DiffSBDD metrics).

2.5 Comparison to deep learning

We compare IDOLpro to other deep learning tools in the literature on the CrossDocked and Binding MOAD test sets. As was done in ref. 19, we generate 100 optimized ligands using IDOLpro for each protein pocket. We report the Vina score, top-10% Vina score, SA score, QED, pocket diversity, and time taken to generate a single ligand averaged across all protein pockets in each test. These metrics are evaluated for six other DL tools in the literature: 3D-SBDD,[15] Pocket2Mol,[16] GraphBP,[17] TargetDiff,[18] DiffSBDD-Cond, and DiffSBDD-inpaint.[19] Results for other tools are taken from ref. 19.

In ref. 19, the generated ligands of each deep learning model are re-docked to the target protein pocket with QuickVina2.[47] In our workflow, we replace re-docking with structural refinement. In QuickVina2, molecular conformations are optimized using both global and local optimization, while structural refinement only performs local optimization on the ligand coordinates *via* L-BFGS. For more information on structural refinement, see Section 4.6.

Full results are shown in Table 3. The performance of the models on the two optimized metrics - Vina and SA is visualized in Fig. 2. For CrossDocked, IDOLpro achieves greatly improved Vina scores relative to other DL tools, with a 0.71 kcal mol^{-1} improvement in average Vina score and 1.03 kcal mol^{-1} improvement in top-10% Vina score compared to the next best tool in the literature, DiffSBDD-inpaint. Despite not optimizing QED directly, we find that IDOLpro produces ligands with the best QED out of all DL tools compared. IDOLpro ranks second for producing molecules with good SA scores, showing the ability of the platform to perform multi-objective optimization. Despite needing to run an entire optimization procedure for each ligand, IDOLpro is still computationally tractable, achieving run times competitive with two other tools - TargetDiff and Pocket2Mol, while achieving a faster runtime than 3D-SBDD (Fig. 3).

IDOLpro is almost always able to find ligands that improve upon both the Vina and SA scores relative to the reference ligand, failing to find such a ligand for only one target - the protein pocket defined by ligand x2p bound to the protein with PDB ID 5mma. For this protein pocket, IDOLpro generated molecules with significantly better SA scores than the reference molecule, but did not generate a single molecule with a better Vina score. We hypothesize that re-weighting the objective in the optimization to more heavily favour the Vina score would allow IDOLpro to generate a ligand that improves upon both metrics.

For the Binding MOAD, the advantage of IDOLpro for generating molecules with high binding affinity is even more pronounced, with a 1.43 kcal mol^{-1} improvement in average Vina score, and 1.40 kcal mol^{-1} improvement in top-10% Vina score compared to the next best method, DiffSBDD-inpaint. In particular, IDOLpro is the first ML tool to generate molecules with a better average Vina score than those of the reference molecules in the Binding MOAD test set. This is noteworthy, because unlike molecules in CrossDocked, molecules in the Binding MOAD were derived through experiment. Out of the four methods compared, IDOLpro also achieves the best SA, improving upon the next best method by 1.15, while also achieving the best QED despite not optimizing for it directly. The time to generate a single ligand for a protein pocket in the Binding MOAD test set is slower than for the CrossDocked test set, and reflects DiffSBDD's slowdown in proposing novel molecules for targets in this set.

IDOLpro is able to find a molecule with a better Vina and SA score than the reference ligand for 126/130 targets (≈97% of the time). Similar to the one case in CrossDocked for which IDOLpro did not find a ligand with improved Vina and SA score relative to the reference ligand, for these 4 targets IDOLpro finds

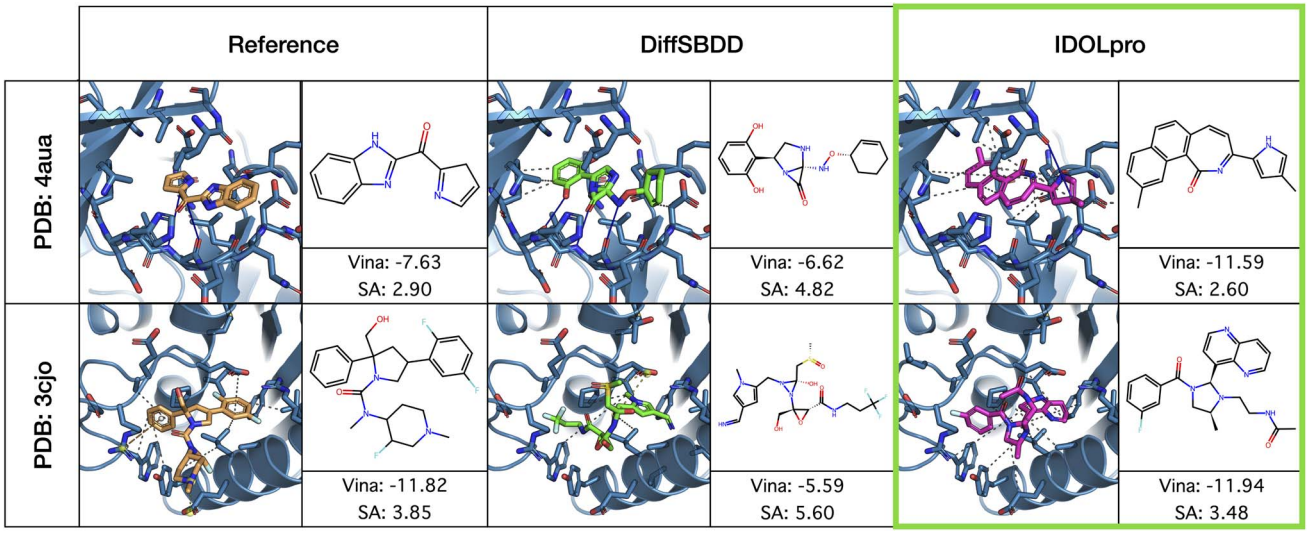

Fig. 3 Molecules produced by IDOLpro when optimizing torchvina and torchSA. One example from each test set is shown - protein 4aua from CrossDocked, and protein 3cjo from the Binding MOAD. Left column: reference molecules from the test sets. Middle column: initial ligand produced by DiffSBDD prior to latent vector optimization. Right column: molecule produced by IDOLpro after optimizing torchvina and torchSA. For each example, the molecule produced by DiffSBDD has worse Vina and SA scores than the reference molecule. After optimization with IDOLpro, both the Vina and SA scores of the generated molecule are better than the reference. Visualizations were created with PyMol,[48] and interactions were visualized with the protein-ligand interaction profiler (PLIP).[49]

molecules with significantly better SA scores than the reference, but fails to find a molecule with a better Vina score. All four of these cases have reference molecules with SA scores of over 7, whereas IDOLpro does not produce a single molecule with an SA score over 6. We again hypothesize that this could be solved by re-weighting the optimization objective to more heavily favor Vina score.

Overall IDOLpro can effectively co-optimize multiple objectives, generating ligands with state-of-the-art binding affinity and synthetic accessibility on two test sets. Improving other metrics with IDOLpro is straightforward, simply requiring a differentiable score for evaluating the desired metric. Part of our future work will be to focus on including differentiable scoring functions for other desirable properties such as solubility, toxicity, *etc.*

2.6 Comparison to virtual screening

We compare the ability of IDOLpro to generate promising compounds when compared to an exahaustive virtual screen of a large library of commercially available drug-like compounds. To do so, we evaluate the performance of IDOLpro for finding ligands with high binding affinity and synthetic accessibility when compared with virtually screening the ZINC250K database - a curated collection of 250 000 commercially available chemical compounds.[26] We evaluate each method on all 10 protein pockets in the RGA test set.

For this experiment, we measure how quickly IDOLpro can generate a ligand with both better binding affinity (Vina score) and synthetic accessibility (SA score) than the ligand with the best binding affinity found when screening ZINC250K. For screening ZINC250K, we use QuickVina2[47] to quickly dock molecules to each of the 10 target protein pockets in the RGA test set. We run docking on an AWS compute-optimized instance with 8 CPU cores. We then use IDOLpro to generate optimized ligands, making note of the number of ligands, time, and cost required for IDOLpro to generate an improved ligand relative to virtually screening ZINC250K. Cost is based on the AWS pricing for the requested instances. The results are shown in Table 4.

Screening ZINC250K using QuickVina2 takes an average of ≈161 hours per protein pocket. On the other hand, IDOLpro is able to find a ligand with a better Vina and SA score than the virtual screen in under an hour (≈56 minutes) on average. This translates to ≈173× speedup in terms of time, and ≈103× reduction in cost. For 4/10 cases, IDOLpro is able to find a ligand with better Vina and SA score within the first 10 ligands generated, and for 9/10 cases within the first 100 ligands generated. For a single case (PDB ID 3eml) IDOLpro needs to produce 153 ligands to find a ligand with better binding affinity and synthetic accessibility than the one found in the virtual screen. Even in this case, running IDOLpro takes ≈2.5 hours to run on the requested GPU instance, translating to a ≈66×

Table 4 Comparison of using IDOLpro to find improved ligands relative to a virtual screen of ZINC250K. We note the average number of ligands, time, and cost it takes for IDOLpro to find a ligand with both better Vina and SA than the best ligand from ZINC250K. Virtual screening was run on an AWS compute-optimized instance with 8 CPU cores, while IDOLpro was run on an NVIDIA A10G GPU with 24 GB of VRAM. Costs are computed based on AWS instance pricing

Method	N_{ligands}	Time [h]	Cost [$]
IDOLpro	44.90 ± 46.06	0.93 ± 0.75	1.06 ± 0.86
Virtual Screen	250 000	160.90 ± 24.12	109.41 ± 16.40

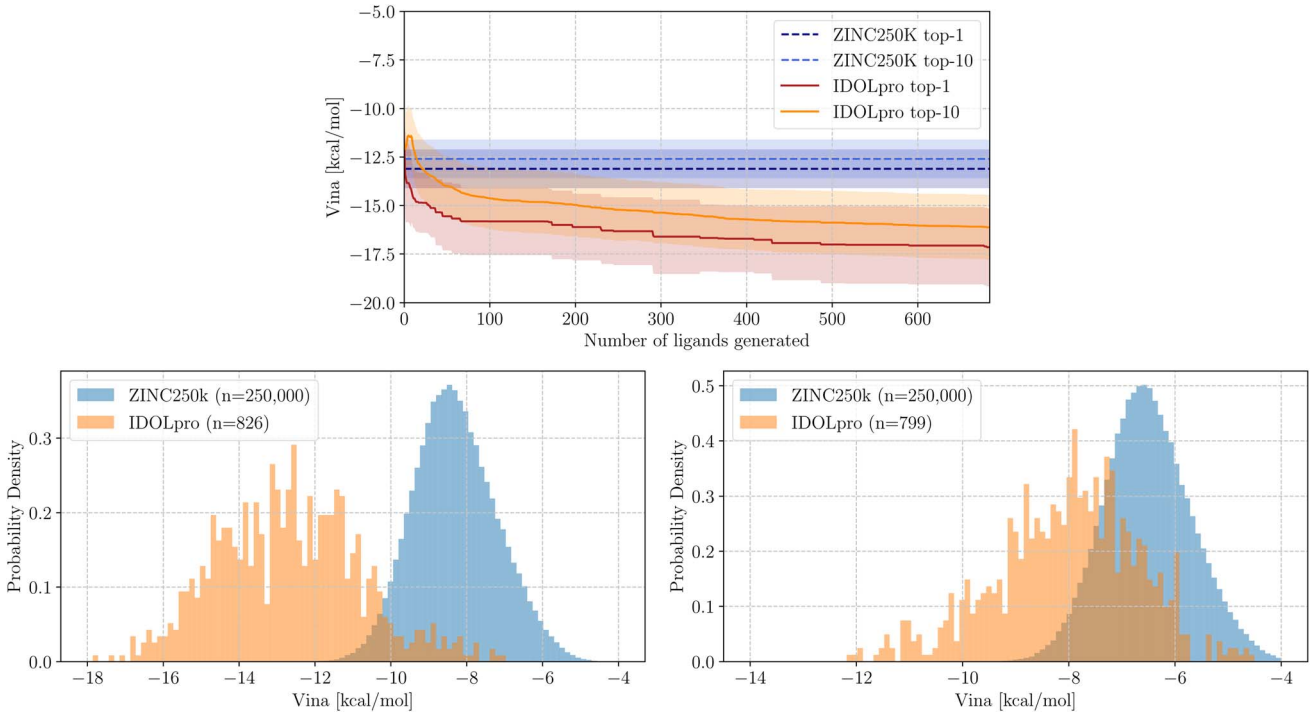

Fig. 4 Comparison of IDOLpro to virtually screening ZINC250K. Top center: The average top-1 and top-10 Vina scores of IDOLpro generated molecules compared to screened molecules from ZINC250K across all 10 targets. Bottom left: The distribution of Vina scores for generated and screened ligands for EGFR, an important oncology target (PDB ID 2rgp). Bottom right: The distribution of Vina scores for generated and screened ligands for the SARS-Cov-2 main protease (PDB ID 7l11).

speedup in terms of time, and ≈39× reduction in cost compared to virtually screening ZINC250K.

In general, for a given protein pocket, IDOLpro produces ligands with significantly better binding affinities than those found when screening a standard drug database such as ZINC250K. In Fig. 4, we track the average running top-1 and top-10 Vina scores for IDOLpro across the 10 protein targets as a function of the number of ligands generated. For both the top-1 and top-10 Vina score, IDOLpro quickly surpasses the virtual screen of ZINC250K - needing to generate on average only 1 ligand to surpass the top-1 Vina score of the virtual screen, and needing to generate on average only 15 ligands to surpass the top-10 Vina score of the virtual screen. We also plot the distribution of Vina scores for both the ligands produced by IDOLpro, and those found when screening ZINC250K for 2 of the targets from the RGA test set - 2rgp, a protein whose overexpression has been associated with human tumor growth,[50] and 7l11 - the SARS-CoV-2 main protease.[43] These plots are shown in Fig. 4. These plots further validate IDOLpro's ability to generate ligands with high binding affinity relative to virtual screening.

2.7 Comparison to other methods

Lastly, we compare IDOLpro to various non-deep learning-based methods in the literature. These methods include genetic algorithms,[40,56–58] reinforcement learning,[52–55] and an MCMC method.[51] We evaluate these methods across the 10 protein pockets in the RGA test set. For each target, as was done in ref. 40, we generate 1000 ligands with IDOLpro and calculate the average top-100, top-10, and top-1 Vina score. We also record the average SA and QED of molecules, along with the average diversity per protein pocket. Numbers for other methods are taken from ref. 40. Results are shown in Table 5.

IDOLpro greatly outperforms non-DL techniques in terms of Vina score, improving on the next best method by ≈23%, ≈29%, and ≈35% in terms of average top-100, top-10, and top-1 Vina score respectively. Unlike when compared to other DL methods, IDOLpro ranks behind most of these methods in terms of average SA, ranking 9th out of the 10 methods compared. IDOLpro is middle-of-the-pack in terms of QED, ranking 5th out of the 10 methods compared. This shows that IDOLpro, and deep learning methods in general, have a ways to go before they can produce molecules with the same synthesizability and drug-likeness as other advanced non-DL methods in the literature. This is an ongoing area of research,[23,25] and is an aspect that we would like to improve in IDOLpro.

2.8 Lead optimization

In addition to *de novo* generation, IDOLpro can be used to optimize a known ligand in the protein pocket. This functionality is useful for a common task in drug discovery pipelines known as lead optimization, where a molecule is progressed from an initial promising candidate towards having optimal properties.[59] Generally, this is accomplished by fixing a large part of the molecule, *i.e.*, the scaffold, while optimizing the rest of the molecule.[60] This is incorporated into generation with

Table 5 Comparison of IDOLpro to various score and sample-based methods on 10 disease-related protein targets. The average top-100, top-10, and top-1 Vina scores across the targets are reported, along with the average SA, average QED, and diversity. The top-performing model on each metric is bolded in the corresponding column. Numbers for other methods are taken from ref. 40

Method	$\text{Vina}_{\text{top-100}}$ [kcal mol^{-1}]	$\text{Vina}_{\text{top-10}}$ [kcal mol^{-1}]	$\text{Vina}_{\text{top-1}}$ [kcal mol^{-1}]	SA	QED	Diversity
MARS[51]	−7.76 ± 0.61	−8.8 ± 0.71	−9.26 ± 0.79	2.69 ± 0.08	0.71 ± 0.01	**0.88 ± 0.00**
MolDQN[52]	−6.29 ± 0.40	−7.04 ± 0.49	−7.50 ± 0.40	5.83 ± 0.18	0.17 ± 0.02	**0.88 ± 0.01**
GEGL[53]	−9.06 ± 0.92	−9.91 ± 0.99	−10.45 ± 1.04	2.99 ± 0.05	0.64 ± 0.01	0.85 ± 0.00
REINVENT[54]	−10.81 ± 0.44	−11.23 ± 0.63	−12.01 ± 0.83	2.60 ± 0.12	0.45 ± 0.06	0.86 ± 0.01
RationaleRL[55]	−9.23 ± 0.92	−10.83 ± 0.86	−11.64 ± 1.10	2.92 ± 0.13	0.32 ± 0.02	0.72 ± 0.03
GA + D[56]	−8.69 ± 0.45	−9.29 ± 0.58	−9.83 ± 0.32	3.45 ± 0.12	0.70 ± 0.02	0.87 ± 0.01
Graph-GA[57]	−10.48 ± 0.86	−11.70 ± 0.93	−12.30 ± 1.91	3.50 ± 0.37	0.46 ± 0.07	0.81 ± 0.04
Autogrow 4.0[58]	−11.37 ± 0.40	−12.21 ± 0.62	−12.47 ± 0.84	2.50 ± 0.05	**0.75 ± 0.02**	0.85 ± 0.01
RGA[40]	−11.87 ± 0.17	−12.56 ± 0.29	−12.89 ± 0.47	**2.47 ± 0.05**	0.74 ± 0.04	0.86 ± 0.02
IDOLpro	**−14.59 ± 1.51**	**−16.26 ± 1.66**	**−17.35 ± 2.10**	3.77 ± 0.33	0.64 ± 0.06	0.72 ± 0.04

Table 6 Results for IDOLpro scaffold fixing on the 71 crossdocked data points with identifiable scaffolds using RDKit's Bemis-Murcko scaffold. The average and top-10 Vina and SA scores are reported for the reference ligands and the optimized ligands

Method	Vina [kcal mol^{-1}]	$\text{Vina}_{10\%}$ [kcal mol^{-1}]	SA	$\text{SA}_{10\%}$
Test Set	−5.58 ± 2.32	—	3.67 ± 1.23	—
IDOLpro	−7.17 ± 2.36	−8.96 ± 2.57	4.12 ± 1.10	2.90 ± 1.12

DiffSBDD[19] using an inpainting-inspired approach. At each denoising time step, fixed atoms are replaced with their noised counterparts, while the rest of the molecule is generated *de-novo*.

We test the capability of our framework for performing lead optimization with examples from the CrossDocked test set. We specify atoms to fix by using RDKit to identify the Bemis-Murcko scaffold[61] of each reference ligand. If no scaffold is found (13 cases), if it is identified to be greater than 90% of the full reference ligand (13 cases), or if it contains atoms not supported by the ANI2x model (3 cases), then these targets are removed from the test set. The lead optimization results with IDOLpro for the remaining 71 protein and scaffold pairs are shown in Table 6 where they are compared to the original reference ligands. We report the Vina and SA scores of the un-docked reference ligands since their scaffolds' coordinates act as the seed for IDOLpro optimization.

We find that the average Vina scores of the optimized ligands exceed the average values of the reference ligands in the test set (1.59 kcal mol^{-1}). Although the average SA is higher, both the

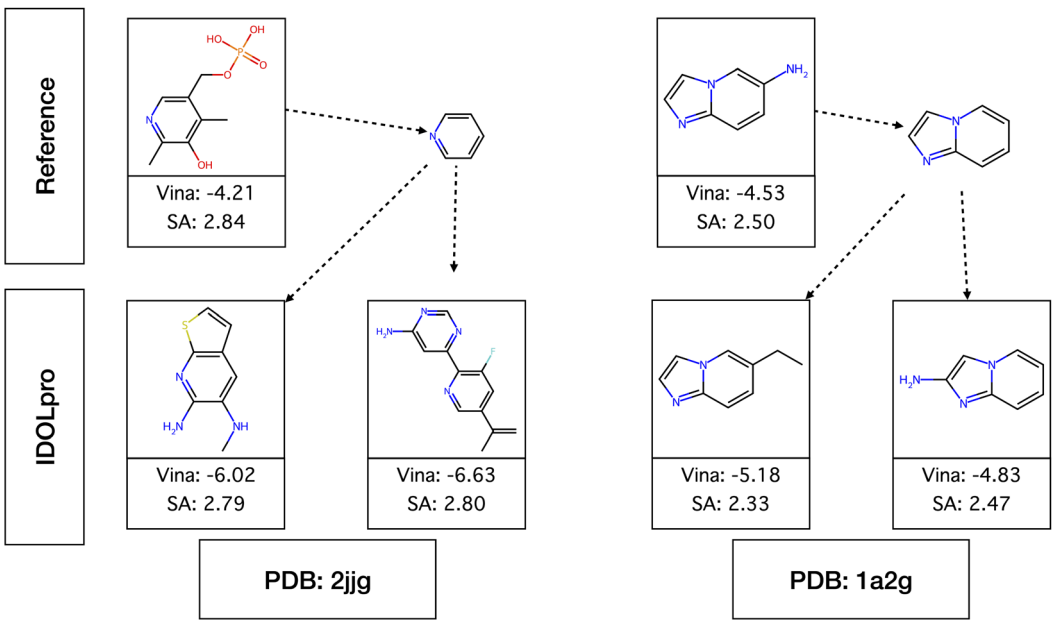

Fig. 5 Molecules produced by IDOLpro during lead optimization. Examples shown are on the ligand plp docked into protein 2jjg, and ligand 1ly docked into protein 1a2g, both examples from the CrossDocked test set. IDOLpro is used to append molecules to the scaffold while optimizing torchvina and torchSA. IDOLpro yields multiple ligands with improved SA and Vina relative to the reference molecule.

top-10% SA score and the top-10% Vina are significantly better than the test set, and are more realistic to consider when assessing lead optimization capability. A molecule with both better SA and Vina scores compared to the reference ligands was found for all but one target, where we were able to improve the SA score but not the Vina score. As discussed previously, a possible remedy would be re-weighting the optimization objective to more heavily favor Vina score.

Two examples of generated molecules that retain the original seed scaffold and successfully improve both the Vina and SA scores compared to the reference ligand are shown in Fig. 5. The first is a case where the identified scaffold is a small portion of the reference (ligand with PDB residue ID plp docked into protein 2jjg), and we see a large diversity in the generated molecules. The second is a case where the scaffold is a large component of the reference (ligand with PDB residue ID 1ly docked into protein 1a2g) where we are effectively optimizing functional groups on a fixed scaffold. We believe our results demonstrate the flexibility of atom fixing in IDOLpro, and its utility in performing lead optimization.

3 Discussion

We have presented a framework that is designed to produce optimal ligands with desired properties for a given protein pocket. We accomplish this by constructing a computational graph that begins with the latent variables of a diffusion model and ends with the evaluation of metrics important in drug discovery. The latent variables can then be modified *via* standard gradient-based optimization routines to optimize the metrics of interest. More specifically, we perform multivariate optimization by optimizing both Vina and SA scores simultaneously. The molecules generated by our platform achieve the best Vina scores when compared to previous, state-of-the-art machine learning methods. When considering the CrossDocked and Binding MOAD test sets, we see a 10% (0.71 kcal mol^{-1}) and 20% (1.43 kcal mol^{-1}) improvement to the next best tool. Our tool ranks second among all tools compared for producing molecules with good SA scores on CrossDocked, and is state-of-the-art for the Binding MOAD, improving upon the SA of the next best tool by 35%. Furthermore, our tool produces molecules with the highest QED on both datasets. For the Binding MOAD, our framework is the first to produce molecules with a better average Vina score than reference molecules in the test set, which were derived through experiment. Our tool shows great promise in the hit-finding stage of a typical drug discovery pipeline, finding molecules with better Vina and SA scores at a fraction of the time and cost when compared to a virtual screen of a library of commercially available compounds. In addition, our tool can perform lead optimization by beginning the optimization with a scaffold taken from a reference molecule. When applied to a relevant subset of the CrossDock dataset, we identified ligands with both better SA and Vina scores than the reference ligand in all but one case. Our framework proposes optimal drugs given particular properties, unlocking the ability to virtually screen optimized molecules from a vast chemical space without the need of searching through a large database, or generating molecules randomly until one with desired properties is found, therefore accelerating the drug discovery process. Our future work will include other important metrics such as toxicity and solubility within the objective to ensure the generation of feasible ligands, along with the consideration of other binding affinity metrics such as free energy perturbation (FEP) based affinity.[62]

4 Methods

4.1 Generator module

When optimizing latent vectors, we utilize a state-of-the-art denoising diffusion probabilistic model (DDPM),[21] DiffSBDD,[19] for generating novel ligands with high binding affinity. DDPMs generate samples from a target distribution by learning the a denoising process. For some $\tilde{\mathbf{z}}_{\text{data}}$ sampled from the target distribution, a diffusion process adds noise to $\tilde{\mathbf{z}}_{\text{data}}$ to elicit the latent vector at time-step t for $t = 0, \ldots, T$ (where T is the length of the noising process) according to the transition distributions defined by:

$$p(\mathbf{z}_t|\tilde{\mathbf{z}}_{\text{data}}) = \mathcal{N}\left(\mathbf{z}_t\middle|\alpha_t\tilde{\mathbf{z}}_{\text{data}},\ \sigma_t^2\mathbf{I}\right), \tag{1}$$

α_t controls the level of noise in $\mathbf{z}_t$ and follows a pre-defined schedule from $\alpha_0 \approx 1$ (no noise) to $\alpha_T \approx 0$ (pure noise). In DiffSBDD, the variance-preserving noising process is used, *i.e.*, $\alpha_t = \sqrt{1-\sigma_t^2}$. The posterior of the transitions conditioned on $\tilde{\mathbf{z}}_{\text{data}}$ define the inverse of the noising process, *i.e.*, the denoising process, and can be written in closed form for any $s < t$:

$$q(\mathbf{z}_s|\tilde{\mathbf{z}}_{\text{data}},\ \mathbf{z}_t) = \mathcal{N}\left(\mathbf{z}_s\middle|\frac{\alpha_{t|s}\sigma_s^2}{\sigma_t^2}\mathbf{z}_t + \frac{\alpha_s\sigma_{t|s}^2}{\sigma_t^2}\tilde{\mathbf{z}}_{\text{data}},\ \frac{\sigma_{t|s}^2\sigma_s^2}{\sigma_t^2}\mathbf{I}\right), \tag{2}$$

where $\alpha_{t|s} = \dfrac{\alpha_t}{\alpha_s}$, and $\sigma_{t|s} = \sigma_t^2 - \alpha_{t|s}^2\sigma_s^2$ following the notation of ref. 63. This denoising process relies on $\tilde{\mathbf{z}}_{\text{data}}$, which is not available during inference. Instead, a neural network, ψ_θ, is used to make an inference of $\hat{\mathbf{z}}_{\text{data}}$. Ref. 21 found that optimization is easier when predicting the noise in the signal. One can reparameterize eqn (1) such that $\tilde{\mathbf{z}}_{\text{data}} = \dfrac{1}{\alpha_t}\mathbf{z}_t - \dfrac{\sigma_t}{\alpha_t}\varepsilon$ where $\varepsilon \sim \mathcal{N}(0,\ \mathbf{I})$, and get the neural network to predict $\hat{\varepsilon}_\theta(\mathbf{z}_t,\ t)$, yielding a prediction $\hat{\mathbf{z}}_{\text{data}}$. Thus, given $\mathbf{z}_t$ and using eqn (2), one can sample $\mathbf{z}_s$ for any $s < t$. In practice, denoising is done in successive time-steps, *i.e.*, $s = t - 1$ in eqn (2).

DiffSBDD is an SE(3)-equivariant[64] 3D-conditional DDPM which respects translation, rotation, and permutation symmetries. DiffSBDD was trained to create ligands with high binding affinity given a target protein pocket. In DiffSBDD, data samples consist of protein pocket and ligand point clouds, *i.e.*, $\mathbf{z} = [\mathbf{z}^\ell, \mathbf{z}^p]$. Each point cloud consists of atom types and coordinates, $\mathbf{z} = [\boldsymbol{r}, \boldsymbol{h}]$ where $\boldsymbol{r} \in \mathbb{R}^{N\times 3}$ is a tensor of atomic coordinates, and $\boldsymbol{h} \in \mathbb{R}^{N\times 10}$ is a tensor of atomic probabilities over the atom types which the model can generate. Within the model, each $\mathbf{z}_t$ is converted to a graph, and processed by an EGNN[65] to produce a prediction $\hat{\varepsilon}_\theta(\mathbf{z}_t,\ t)$. DiffSBDD contains two different models for 3D pocket conditioning - a conditional DDPM that receives a fixed pocket representation as the context in each denoising step, and a model that is trained to approximate the joint

distribution of ligand-protein pocket pairs and is combined with an inpainting-inspired[22,66] sampling procedure at inference time to yield conditional samples of ligands given a fixed protein pocket. In this work, we focus only on the conditional DDPM for ligand generation. Our framework requires that generated ligands do not overlap with the target protein pocket, as the ligands' poses later undergo local structural refinement. We found that the conditional DDPM model can consistently generate ligands satisfying this constraint.

For each pocket-conditioning scheme, DiffSBDD contains two versions of the model - one trained on a subset of CrossDocked,[38] and another trained on a subset of the Binding MOAD.[39] For CrossDocked, ref. 19 used the same train/test splits as in ref. 15 and ref. 16, resulting in 100 000 complexes for training, and 100 protein pockets for testing. For the Binding MOAD, ref. 19 filtered the database to contain only molecules with atom types compatible with their model, and removed corrupted entries, resulting in 40 344 complexes for training, and 130 protein pockets for testing. In this work we use only version of DiffSBDD trained on CrossDocked as we found we were able to achieve good results on both test sets (CrossDocked and the Binding MOAD) with just this model.

DiffSBDD was shown to achieve state-of-the-art performance on both test sets. In particular, DiffSBDD achieved the best average, and best top-10% Vina score when compared with other state-of-the-art models in the literature - 3D-SBDD,[15] Pocket2Mol,[16] GraphBP,[17] and TargetDiff.[18] We note, that although DiffSBDD is used as our baseline model in this report, our framework is not limited to the use of this specific model - any other diffusion model can take its place.

4.2 Molecule size

For determining the number of atoms in a generated ligand n_ℓ, we employ the same sampling procedure as is used when sampling structures from DiffSBDD.[19] We sample from an empirical distribution of the number of heavy atoms in the ligand given the number of heavy atoms in the pocket - $p(n_\ell|n_p)$. This empirical distribution is based on the data used to train the CrossDocked model in ref. 19. We then bias the model in the same way that is done in ref. 19, adding 5 heavy atoms to each sample of n_ℓ - this allows us to have an apples-to-apples comparison of our tool *versus* tools compared in that work. In lead optimization, the number of heavy atoms in the ligand is sampled from a normal distribution centered at the size of the reference ligand, and whose lower limit is clamped to be no lower than the number of heavy atoms in the scaffold. The number of atoms in the scaffolds relative to their reference values varied in the test set. To ensure a balanced sampling of scaffolds with more or fewer atoms than the reference, in this study we set the standard deviation to the greater of 5 (matching the heavy atom bias) or half the difference between the number of heavy atoms in the scaffold and the reference ligand.

4.3 Ligand validity checks

When generating ligands using DiffSBDD, we perform several chemical and structural checks to ensure that the generated ligand is valid. A number of these checks use RDKit.[46] These include a valency check, verifying that hydrogens can be added to the ligand and assigned a Cartesian coordinate (using the *addCoords* option in *Chem.AddHs*), that the ligand is not fragmented, and that the ligand can be sanitized. All of these except for the valency check are also done within DiffSBDD.[67]

In addition, we have four more checks to ensure the structural validity of the ligand. These four checks are necessary to be able to run structural refinement with IDOLpro, which makes use of the ANI2x model.[37] Structural refinement is described in Section 4.6. We first make sure that the ligand contains only atoms compatible with ANI2x. DiffSBDD can generate ligands with four atom types that are incompatible with ANI2x - B, P, Br, and I. We also make sure that the bond lengths in the ligand are reasonable by referring to covalent radii, and that the ligand does not overlap with the protein pocket. This is done *via* ASE's[68] (Atomic Simulation Environment) NeighborList class. Lastly, We make sure that the atoms do not have significant overlap within the ligand itself. This is done *via* pymatgen's[69] Molecule class.

4.4 Scoring module

After generating a set of ligands, we pass them to a scoring module. In this work, we include a custom torch-based Vina score[32] which we refer to as torchvina, an equivariant neural network trained to predict the synthetic accessibility of molecules with 3D information[36] which we refer to as torchSA, and the ANI2x model.[37] These objectives are all written using Pytorch[35] with differentiable operations and hence can be differentiated automatically using autograd.

4.4.1 Torchvina. We re-implement the Vina force field[32] using Pytorch to allow for automatic differentiation with respect to the latent parameters of the generator. Our work is not the first to produce a Pytorch-based version of Vina to facilitate automatic differentiation, a similar implementation was presented by ref. 70. Our motivation for implementing a differentiable Vina score is that docking with Vina was shown to outperform state-of-the-art ML models such as DiffDock[34] when stricter chemical and physical validity checks were enforced on docked molecules, or when these procedures were evaluated on a dataset composed of examples distinct from the ML models' training data.[23]

The Vina force field is composed of a weighted sum of atomic interactions. Steric, hydrophobic, and hydrogen bonding interactions are calculated and weighted according to a nonlinear fit to structural data.[32] The final score is re-weighted by the number of rotatable bonds to account for entropic penalties.[71] The Vina score is composed of a sum of intramolecular and intermolecular terms, both of which are integrated into our implementation.

To validate our implementation of torchvina, we took 4 systems from our validation set (details in the ESI†). The Vina score consists of a weighted sum of five individual terms, three steric terms - two attractive Gaussian terms (gauss1 and gauss2), and a repulsion term, as well as hydrophobic interactions, and where applicable, hydrogen bonding. For these four systems, we

made sure that torchvina produced the same value on each of these sub-terms, as well as the total intermolecular energy and the total intramolecular energy as the smina[41] tool. These checks are included in our github repository in the tests directory.

4.4.2 torchSA. To have an evaluator model capable of estimating synthesizability, we train an equivariant neural network to predict the synthetic accessibility (SA) score. SA score was first proposed by ref. 36, ranges from 1 (easy to make) and 10 (very difficult to make), and shown to be effective for biasing generative pipelines towards synthesizable molecules.[25,72] Moreover, it was used directly in DiffSBDD to measure the performance of the pipeline.[19] To be able to guide latent parameters in DiffSBDD towards generating ligands with high synthesizability requires designing a model that can handle the outputs of DiffSBDD in a differentiable manner. In particular, we construct a machine learning model that can take in atomic point clouds, $\mathbf{z} = [\boldsymbol{r}, \boldsymbol{h}]$, where $\boldsymbol{r}$ is a set of coordinates, and $\boldsymbol{h}$ is corresponding probability distributions over atom types. We train this model by creating a dataset of atomic point clouds of ligands labeled with SA score. To allow for predictions on probability distributions of atom types, we encode atom types as one-hot vectors. For more details on the training of this model, we refer the reader to the ESI.†

To validate the torchSA model for guiding DiffSBDD towards generating molecules with high synthetic accessibility, we generated 100 molecules using IDOLpro with only torchvina guidance for each target in our validation set (see ESI† for details on structures in this set). For each generated molecule, we save the entire trajectory of point clouds required to produce the final IDOLpro molecule, *i.e.*, the point cloud produced by DiffSBDD during each gradient update of the latent vectors. For each of these intermediate point clouds, we store the coordinates, and raw atomic probabilities. For each point cloud, we also store their corresponding RDKit molecule. This is done by taking the most likely atom type for each probabilty distribution, and adding bonds using OpenBabel.[73] We generate 25 926 individual point cloud–molecule pairs for evaluating the torchSA model. For each pair, we compare the output of torchSA on the point cloud to the SA score on the corresponding molecule. We plot the correlation between the two in Fig. 6, and include both the Pearson's ρ, and mean absolute error in the figure.

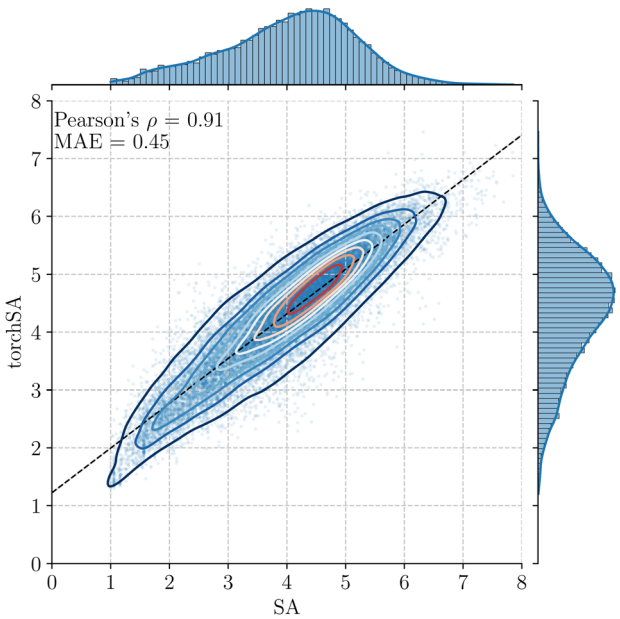

Fig. 6 Correlation plot between the predictions of the torchSA model and the SA score as computed in RDKit.[46] The torchSA model is validated on ≈25 000 atomic point clouds (coordinates and atomic probability distributions) generated by DiffSBDD. For each atomic point cloud, we construct a corresponding RDKit molecule to assess the SA score. TorchSA displays a high-correlation ($\rho = 0.91$) and low mean absolute error (MAE = 0.45) relative to the SA score on these data points. The histogram on each axis represents the distribution of molecules that achieved corresponding scores as predicted by torchSA (*y*-axis) and SA score (*x*-axis).

4.4.3 ANI2x. ANI2x is a neural network ensemble model that is part of the ANI suite of models.[74] The ANI models are trained on quantum chemistry calculations (at the density functional theory level) and they predict the total energy of a target system. The ANI models are trained on millions of organic molecules and are accurate across different domains.[37,75–77] In addition, they have been shown to outperform many common force fields in terms of accuracy.[78] The ANI models make use of atomic environment descriptors, which probe their local environment, as input vectors. An individual ANI model contains multiple neural networks, each specialized for a specific atom type, predicting the energy contributed by atoms of that type in the molecular system. The total energy of the system is obtained by performing a summation over the atomic contributions.[75] The ANI2x model is an ensemble model consisting of 8 individual ANI models. Each sub-model is trained on a different fold of the ANI2x dataset, composed of gas-phase molecules containing seven different atom types - H, C, N, O, F, Cl, and S.[37] These seven atom types cover ≈90% of drug-like molecules, making ANI2x a suitable ML model for usage in our framework.

4.5 Latent Vector Optimization

The main optimization in IDOLpro occurs *via* the modification of latent vectors used by the generator to generate novel ligands. We do this by repeatedly evaluating generated ligands with an objective composed of a sum of differentiable scores, calculating the gradient of the objective with respect to the latent vectors (facilitated by automatic differentiation with Pytorch[35]), and modifying the latent vectors *via* a gradient-based optimizer.

When optimizing latent vectors in DiffSBDD, we do not modify the initial latent vectors used by the model. Instead, we define an optimization horizon, t_{hz}. First latent vectors are generated up to the optimization horizon $\mathbf{z}_T^{\ell}, \ldots, \mathbf{z}_{t_{\mathrm{hz}}}^{\ell}$. This latent vector is saved, and the remaining latent vectors $\mathbf{z}_{t_{\mathrm{hz}}-1}^{\ell}, \ldots, \mathbf{z}_0^{\ell}$, are generated. Upon passing the ligand validity checks (Section 4.3), the gradient of the objective with respect to $\mathbf{z}_{t_{\mathrm{hz}}}$ is evaluated, and $\mathbf{z}_{t_{\mathrm{hz}}}$ is modified using a gradient-based optimizer. When re-generating ligands, rather than starting from $\mathbf{z}_T^{\ell}$, only latent vectors proceeding the optimization horizon are re-generated,

Algorithm 1 Optimization with IDOLpro

parameter description
N_{max}, maximum number of optimization steps.
$\tilde{\mathbf{z}}^p$, protein atom embeddings (types and coordinates).
n_ℓ, number of atoms in generated ligand.
$\tilde{\mathbf{z}}_0^\ell$, fixed ligand atom embeddings (for lead optimization).
$\mathbf{m}_\ell$, mask for fixed ligand atoms (for lead optimization).
end parameter description

function IDOLPRO($\tilde{\mathbf{z}}^p$, n_ℓ, $\tilde{\mathbf{z}}_0^\ell$ = None, $\mathbf{m}_\ell = \mathbf{1}$)
  Sample $\mathbf{z}_T^\ell \sim \mathcal{N}(\mathbf{0}_{n_\ell}, \mathbf{I}_{n_\ell})$
  for $t = T, \ldots, t_{hz}+1$ **do**
    $\mathbf{z}_{t-1}^\ell \leftarrow$ DENOISE($\mathbf{z}_t^\ell, \tilde{\mathbf{z}}^p, \mathbf{m}_\ell, \tilde{\mathbf{z}}_0^\ell$)
  end for
  for $i = 0, \ldots, N_{max}$ **do**
    for $t = t_{hz}, \ldots, 1$ **do**
      $\mathbf{z}_{t-1}^\ell \leftarrow$ DENOISE($\mathbf{z}_t^\ell, \tilde{\mathbf{z}}^p, \tilde{\mathbf{z}}_0^\ell, \mathbf{m}_\ell$)
    end for
    if VALIDITY($\mathbf{z}_0^\ell, \tilde{\mathbf{z}}^p$) **then**
      $\mathbf{z}_{t_{hz}}^\ell \leftarrow$ ADAM($\mathbf{z}_{t_{hz}}^\ell, \frac{\partial(\mathtt{Vina}(\mathbf{z}_0^\ell, \tilde{\mathbf{z}}^p) + \mathtt{SA}(\mathbf{z}_0^\ell))}{\partial \mathbf{z}_{t_{hz}}^\ell}$)
    end if
  end for
  for $t = t_{hz}, \ldots, 1$ **do**
    $\mathbf{z}_{t-1}^\ell \leftarrow$ DENOISE($\mathbf{z}_t^\ell, \tilde{\mathbf{z}}^p, \tilde{\mathbf{z}}_0^\ell, \mathbf{m}^\ell$)
  end for
  $\mathbf{z}_0^\ell \leftarrow$ STRUCTURALREFINEMENT($\mathbf{z}_0^\ell$)
  return $\mathbf{z}_0^\ell$
end function

function VALIDITY($\mathbf{z}^\ell, \mathbf{z}^p$)
  $v \leftarrow$ VALENCECHECK($\mathbf{z}^\ell$)
  $v \leftarrow v$ and CANSANITIZE($\mathbf{z}^\ell$)
  $v \leftarrow v$ and CANADDHS($\mathbf{z}^\ell$)
  $v \leftarrow v$ and FRAGMENTS($\mathbf{z}^\ell$) = 1
  $v \leftarrow v$ and $h_i^\ell \in \{H, C, N, O, F, Cl, S\}$
  $v \leftarrow v$ and CONNECTED($\mathbf{z}^\ell$)
  $v \leftarrow v$ and NOOVERLAPINTRA($\mathbf{z}^\ell$)
  $v \leftarrow v$ and NOOVERLAPINTER($\mathbf{z}^\ell, \mathbf{z}^p$)
  return v
end function

function DENOISE($\mathbf{z}_t^\ell, \tilde{\mathbf{z}}^p, \mathbf{z}_0^\ell, \mathbf{m}_\ell$)
  if $\mathbf{z}_0^\ell$ is None **then** ▷ de-novo
    $\mathbf{z}_{t-1}^\ell \leftarrow q(\mathbf{z}_{t-1}^\ell | \mathbf{z}_t^\ell, \mathbf{z}^p)$
  else ▷ lead-opt
    $\mathbf{z}_{t-1}^\ell \leftarrow q(\mathbf{z}_{t-1}^\ell | \mathbf{z}_t^\ell, \mathbf{z}^p) \cdot \mathbf{m}_\ell + p(\mathbf{z}_{t-1}^\ell | \mathbf{z}_0^\ell, \mathbf{z}^p) \cdot (1 - \mathbf{m}_\ell)$
  end if
end function

i.e., $\mathbf{z}_{t_{hz}-1}^\ell, \ldots, \mathbf{z}_0^\ell$. Optimization continues until a maximum number of steps, $N_{\max}$, have been taken in the latent space. When de-noising $\mathbf{z}_t^\ell$ during lead optimization, a mask $\mathbf{m}_\ell \in \{0, 1\}^{n_\ell}$ is used to keep specific atoms fixed during the inpainting procedure. We employ backtracking, early-stopping, and learning-rate decay, all described in the ESI.† The optimization of a single ligand is provided in Algorithm 1.

In this work, we focus on using two combinations of evaluators: torchvina on its own, and torchvina in combination with torchSA. We use the Adam[79] optimizer with a learning rate of 0.1, $\beta_1 = 0.5$ and $\beta_2 = 0.999$ to modify latent vectors. We perform hyperparameter optimization to choose the optimization horizon, described in the SI.

4.6 Structural refinement

After an optimized ligand has been generated into a protein pocket with latent vector optimization, its bound pose is further refined *via* structural refinement. Structural refinement in IDOLpro proceeds in a similar fashion to latent vector optimization. Differentiable scores are used to repeatedly evaluate the ligand's bound pose, and the derivatives of these scores with respect to the molecular coordinates are used to update the pose in the protein pocket with a gradient-based optimizer.

We use the L-BFGS optimizer in Pytorch[35] to perform coordinate optimization. Our optimization algorithm is implemented with Pytorch and is parallelizable on a GPU. Using structural refinement instead of re-docking each individually generated ligand using an auto-docking software such as AutoDock Vina[32] or QuickVina2[47] affords us an increase in overall computational efficiency.In this work, we only use one combination of evaluators to perform coordinate optimization: torchvina and ANI2x.[37]

To balance the validity of relaxed molecules with docking performance, We consider various combinations of intra- and inter-molecular forces derived from the torchvina and ANI2x potentials. We run structural refinement on a set of 200 ligands generated by IDOLpro when seeded with pockets in our validation set (see SI for details). We tabulate the results in Table 7 and include metrics when QuickVina2[47] is used for re-docking. For each setting, we compute the average Vina and top-10% Vina scores of relaxed molecules, as well as percent of molecules which remain valid according to our validity checks after structural refinement is applied. We initially found that invalid molecules were often caused by bonded atoms being pulled apart during structural refinement. To remedy this, we add an L_1 penalty for violating bonds in the molecule. We include the effect of adding this penalty in Table 7.

4.7 Regressor guidance

Classifier guidance was first proposed in ref. 30. Classifier guidance can be straightforwardly re-interpreted for regressor guidance, which has been implemented for molecular generation in a number of works.[8,10,31] To apply regressor guidance to molecular generation, a regressor is trained to predict the physicochemical score for the final generated molecule given an intermediate latent vector in the denoising process. In particular, given some molecule $\mathbf{z}$ produced by a generative diffusion model, and some physicochemical score $f : Z \to \mathbb{R}$, a regressor is trained to approximate $f(\mathbf{z})$ with $\hat{f}_\theta(\mathbf{z}_t, t)$.

To implement regressor guidance in DiffSBDD, we train a regressor to predict the SA score of a molecule produced by DiffSBDD given an intermediate noisy latent vector. To do so, we run DiffSBDD, and save the entire trajectory of latent vectors, $\mathbf{z}_\ell^T$, $\mathbf{z}_\ell^{T-1}$, …, $\mathbf{z}_\ell^0$. We then train an equivariant graph neural network (EGNN)[65] to minimize the L_2 loss between its prediction of the SA score given an intermediate noisy latent vector, and the SA score of the final molecule. More information about the training of the EGNN regressor can be found in the SI.

At each step in the denoising process, a gradient term from the trained regressor is added to the sampled latent vector, *i.e.*,

Table 7 Results when running structural refinement with various combinations of inter- and intra-molecular forces derived from the Vina and ANI2x potentials. Additionally, we note whether an $L1$ penalty for violating bonds was used. For each experiment, we report the Vina score, top-10 Vina score, and the percent of valid structures produced. Validity checks are performed according to the checks described in Section 4.3. Lastly, we report the average time to run structural refinement on the 20 ligands generated during each run. Experiments were run on an NVIDIA A10G GPU with 24 GB of memory

Method	Torchvina	ANI2x	$L1$ bond penalty	Vina [kcal mol^{-1}]	$\text{Vina}_{10\%}$ [kcal mol^{-1}]	Validity [%]	Time [s]
QuickVina2	—	—	—	−8.51	−9.51	95.0	75.0
IDOLpro	Inter + Intra	Inter + Intra	✗	−9.26	−10.35	80.1	19.34
IDOLpro	Inter + Intra	Intra	✗	−9.33	−10.41	77.2	22.49
IDOLpro	Inter	Inter + Intra	✗	−9.38	−10.53	82.1	18.53
IDOLpro	Inter	Intra	✗	−9.53	−10.70	81.6	23.90
IDOLpro	Inter	Intra	✓	−9.39	−10.68	86.6	24.45

$$\tilde{\mathbf{z}}_{t-1}^{\ell} \leftarrow q\left(\mathbf{z}_{t-1}^{\ell} \middle| \mathbf{z}_{t}^{\ell},\ \mathbf{z}^{p}\right)$$
$$\mathbf{z}_{t-1}^{\ell} \leftarrow \tilde{\mathbf{z}}_{t-1}^{\ell} + \lambda \nabla \hat{f}_{\theta}\left(\mathbf{z}_{t}^{\ell}, t\right).$$

The first equation samples a new ligand given the current noisy ligand and pocket atoms, and the second equation applies regressor guidance with a factor of $\lambda > 0$, maximizing the SA score of the molecule. In our experiment (Section 2.4) we set $\lambda = 0.5$.

Data availability

Our source code is publicly available at **https://github.com/sandbox-quantum/idolpro**. We used the following publicly available datasets for validation and testing: CrossDocked2020 (**https://bits.csb.pitt.edu/files/crossdock2020/**), and Binding MOAD (**http://www.bindingmoad.org/**). Detailed testing splits for generating each of the test benchmarks used in this work are described in **https://github.com/pengxingang/Pocket2Mol/tree/main/data**. A subset of CrossDocked2020 was used to select hyperparameters as described in the SI, and is available at **https://github.com/mattragoza/LiGAN/tree/master/data/crossdock2020**.

Author contributions

AK, KR, and TY conceived the study, designed IDOLpro, and planned the experiments. AK and KR implemented IDOLpro, ran experiments, and wrote the manuscript. EL implemented the lead optimization capability in IDOLpro and wrote the corresponding section in the manuscript. AR and TY advised and reviewed the manuscript.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

The authors thank Andrea Bortolato, Andrew Wildman, Arman Zaribafiyan, Benjamin Shields, and Jordan Crivelli-Decker for their feedback on the manuscript.

References

1 A. C. Anderson, *Chem. Biol.*, 2003, **10**, 787–797.
2 A. Zunger, *Nat. Rev. Chem.*, 2018, **2**, 0121.
3 K. Ryczko, P. Darancet and I. Tamblyn, *J. Phys. Chem. C*, 2020, **124**, 26117–26123.
4 F. R. J. Cornet, B. Benediktsson, B. A. Hastrup, A. Bhowmik and M. N. Schmidt, *37th Conference on Neural Information Processing Systems*, 2023.
5 B. Sanchez-Lengeling, C. Outeiral, G. L. Guimaraes and A. Aspuru-Guzik, *ChemRxiv*, 2017, preprint, DOI: **10.26434/chemrxiv.5309668.v3**.
6 N. W. Gebauer, M. Gastegger, S. S. Hessmann, K.-R. Müller and K. T. Schütt, *Nat. Commun.*, 2022, **13**, 973.
7 B. Sridharan, M. Goel and U. D. Priyakumar, *Chem. Commun.*, 2022, **58**, 5316–5331.
8 S. Lee, J. Jo and S. J. Hwang, *Proceedings of the 40th International Conference on Machine Learning*, 2023, pp. 18872–1889203.
9 S. Zaman, D. Akhiyarov, M. Araya-Polo and K. Chiu, *arXiv*, 2023, preprint, arXiv:2311.06297, DOI: **10.48550/arXiv.2311.06297**.
10 T. Weiss, E. Mayo Yanes, S. Chakraborty, L. Cosmo, A. M. Bronstein and R. Gershoni-Poranne, *Nat. Comput. Sci.*, 2023, **3**, 873–882.
11 J. J. Irwin and B. K. Shoichet, *J. Chem. Inf. Model.*, 2005, **45**, 177–182.
12 A. Shivanyuk, S. Ryabukhin, A. Tolmachev, A. Bogolyubsky, D. Mykytenko, A. Chupryna, W. Heilman and A. Kostyuk, *Chem. Today*, 2007, **25**, 58–59.
13 L. Ruddigkeit, R. Van Deursen, L. C. Blum and J.-L. Reymond, *J. Chem. Inf. Model.*, 2012, **52**, 2864–2875.
14 P. G. Polishchuk, T. I. Madzhidov and A. Varnek, *J. Comput.-Aided Mol. Des.*, 2013, **27**, 675–679.
15 S. Luo, J. Guan, J. Ma and J. Peng, *Advances in Neural Information Processing Systems*, 2021, vol. 34, pp. 6229–623903.
16 X. Peng, S. Luo, J. Guan, Q. Xie, J. Peng and J. Ma, *Proceedings of the 39th International Conference on Machine Learning*, 2022, pp. 17644–1765503.
17 M. Liu, Y. Luo, K. Uchino, K. Maruhashi and S. Ji, *Proceedings of the 39th International Conference on Machine Learning*, 2022, pp. 13912–1392403.

18 J. Guan, W. W. Qian, X. Peng, Y. Su, J. Peng and J. Ma, *International Conference on Learning Representations*, 2023.
19 A. Schneuing, Y. Du, C. Harris, A. Jamasb, I. Igashov, W. Du, T. Blundell, P. Lió, C. Gomes and M. Welling *et al.*, *International Conference on Learning Representations*, 2023.
20 Y. Li, J. Pei and L. Lai, *Chem. Sci.*, 2021, **12**, 13664–13675.
21 J. Ho, A. Jain and P. Abbeel, *Advances in neural information processing systems*, 2020, vol. 33, pp. 6840–685103.
22 A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte and L. Van Gool, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11461–1147103.
23 M. Buttenschoen, G. M. Morris and C. M. Deane, *Chem. Sci.*, 2024, **15**, 3130–3139.
24 Y. Yu, S. Lu, Z. Gao, H. Zheng and G. Ke, *arXiv*, 2023, preprint, arXiv:2302.07134, DOI: **10.48550/arXiv.2302.07134**.
25 W. Gao and C. W. Coley, *J. Chem. Inf. Model.*, 2020, **60**, 5714–5723.
26 R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams and A. Aspuru-Guzik, *ACS Cent. Sci.*, 2018, **4**, 268–276.
27 O. Dollar, N. Joshi, J. Pfaendtner and D. A. Beck, *J. Phys. Chem. A*, 2023, **127**, 7844–7852.
28 S. Bubeck *et al.*, *Foundations and Trends® in Machine Learning*, 2015, vol. 8, pp. 231–35703.
29 J. C. Duchi, M. I. Jordan, M. J. Wainwright and A. Wibisono, *IEEE Trans. Inf. Theory*, 2015, **61**, 2788–2806.
30 P. Dhariwal and A. Nichol, *Advances in neural information processing systems*, 2021, vol. 34, pp. 8780–879403.
31 Y. Ziv, F. Imrie, B. Marsden and C. M. Deane, *J. Chem. Inf. Model.*, 2025, **65**(9), 4263–4273.
32 O. Trott and A. J. Olson, *J. Comput. Chem.*, 2010, **31**, 455–461.
33 A. Thakkar, V. Chadimová, E. J. Bjerrum, O. Engkvist and J.-L. Reymond, *Chem. Sci.*, 2021, **12**, 3339–3349.
34 G. Corso, H. Stärk, B. Jing, R. Barzilay and T. Jaakkola, *International Conference on Learning Representations*, 2023.
35 A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein and L. Antiga *et al.*, *Advances in neural information processing systems*, 2019, vol. 32, pp. 8026–038037.
36 P. Ertl and A. Schuffenhauer, *J. Cheminf.*, 2009, **1**, 1–11.
37 C. Devereux, J. S. Smith, K. K. Huddleston, K. Barros, R. Zubatyuk, O. Isayev and A. E. Roitberg, *J. Chem. Theory Comput.*, 2020, **16**, 4192–4202.
38 P. G. Francoeur, T. Masuda, J. Sunseri, A. Jia, R. B. Iovanisci, I. Snyder and D. R. Koes, *J. Chem. Inf. Model.*, 2020, **60**, 4200–4215.
39 L. Hu, M. L. Benson, R. D. Smith, M. G. Lerner and H. A. Carlson, *Proteins: Struct., Funct., Bioinf.*, 2005, **60**, 333–340.
40 T. Fu, W. Gao, C. Coley and J. Sun, *Advances in Neural Information Processing Systems*, 2022, vol. 35, pp. 12325–0312338.
41 D. R. Koes, M. P. Baumgartner and C. J. Camacho, *J. Chem. Inf. Model.*, 2013, **53**, 1893–1904.
42 M. M. Mysinger, M. Carchia, J. J. Irwin and B. K. Shoichet, *J. Med. Chem.*, 2012, **55**, 6582–6594.
43 C.-H. Zhang, E. A. Stone, M. Deshmukh, J. A. Ippolito, M. M. Ghahremanpour, J. Tirado-Rives, K. A. Spasov, S. Zhang, Y. Takeo, S. N. Kudalkar, *et al.*, *ACS Cent. Sci.*, 2021, **7**, 467–475.
44 J. Yu, J. Wang, H. Zhao, J. Gao, Y. Kang, D. Cao, Z. Wang and T. Hou, *J. Chem. Inf. Model.*, 2022, **62**, 2973–2986.
45 G. R. Bickerton, G. V. Paolini, J. Besnard, S. Muresan and A. L. Hopkins, *Nat. Chem.*, 2012, **4**, 90–98.
46 *RDKit: Open-source cheminformatics*, **https://www.rdkit.org**.
47 A. Alhossary, S. D. Handoko, Y. Mu and C.-K. Kwoh, *Bioinformatics*, 2015, **31**, 2214–2216.
48 Schrödinger, LLC, *The PyMOL Molecular Graphics System*, 2015.
49 M. F. Adasme, K. L. Linnemann, S. N. Bolz, F. Kaiser, S. Salentin, V. J. Haupt and M. Schroeder, *Nucleic Acids Res.*, 2021, **49**, W530–W534.
50 G. Xu, M. C. Abad, P. J. Connolly, M. P. Neeper, G. T. Struble, B. A. Springer, S. L. Emanuel, N. Pandey, R. H. Gruninger, M. Adams, *et al.*, *Bioorg. Med. Chem. Lett.*, 2008, **18**, 4615–4619.
51 Y. Xie, C. Shi, H. Zhou, Y. Yang, W. Zhang, Y. Yu and L. Li, *International Conference on Learning Representations*, 2021.
52 Z. Zhou, S. Kearnes, L. Li, R. N. Zare and P. Riley, *Sci. Rep.*, 2019, **9**, 10752.
53 S. Ahn, J. Kim, H. Lee and J. Shin, *Advances in neural information processing systems*, 2020, vol. 33, pp. 12008–0312021.
54 M. Olivecrona, T. Blaschke, O. Engkvist and H. Chen, *J. Cheminf.*, 2017, **9**, 1–14.
55 W. Jin, R. Barzilay and T. Jaakkola, *International conference on machine learning*, 2020, pp. 4849–034859.
56 A. Nigam, P. Friederich, M. Krenn and A. Aspuru-Guzik, *International Conference on Learning Representations*, 2020.
57 J. Jensen, A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space, *Chem. Sci.*, 2019, **10**(12), 3567–3572.
58 J. O. Spiegel and J. D. Durrant, *J. Cheminf.*, 2020, **12**, 1–16.
59 J. P. Hughes, S. Rees, S. B. Kalindjian and K. L. Philpott, *Br. J. Pharmacol.*, 2011, **162**, 1239–1249.
60 H.-J. Böhm, A. Flohr and M. Stahl, *Drug Discov. Today Technol.*, 2004, **1**, 217–224.
61 G. W. Bemis and M. A. Murcko, *J. Med. Chem.*, 1996, **39**, 2887–2893.
62 J. E. Crivelli-Decker, Z. Beckwith, G. Tom, L. Le, S. Khuttan, R. Salomon-Ferrer, J. Beall, R. Gómez-Bombarelli and A. Bortolato, *J. Chem. Theory Comput.*, 2024, **20**, 7188–7198.
63 E. Hoogeboom, V. G. Satorras, C. Vignac and M. Welling, *International conference on machine learning*, 2022, pp. 8867–038887.
64 F. Fuchs, D. Worrall, V. Fischer and M. Welling, *Advances in neural information processing systems*, 2020, vol. 33, pp. 1970–031981.
65 V. G. Satorras, E. Hoogeboom and M. Welling, *Proceedings of the 38th International Conference on Machine Learning conference on machine learning*, 2021, pp. 9323–039332.

66 Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon and B. Poole, *International Conference on Learning Representations*, 2021.
67 A. Schneuing, *DiffSBDD*, 2023, **https://github.com/arneschneuing/DiffSBDD**.
68 A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, *et al.*, *J. Phys.: Condens. Matter*, 2017, **29**, 273002.
69 S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson and G. Ceder, *Comput. Mater. Sci.*, 2013, **68**, 314–319.
70 Z. Wang, L. Zheng, S. Wang, M. Lin, Z. Wang, A. W.-K. Kong, Y. Mu, Y. Wei and W. Li, *Briefings Bioinf.*, 2023, **24**, bbac520.
71 A. T. McNutt, P. Francoeur, R. Aggarwal, T. Masuda, R. Meli, M. Ragoza, J. Sunseri and D. R. Koes, *J. Cheminf.*, 2021, **13**, 1–20.
72 G. Skoraczyński, M. Kitlas, B. Miasojedow and A. Gambin, *Briefings Bioinf.*, 2023, **15**, 6.
73 N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch and G. R. Hutchison, *J. Cheminf.*, 2011, **3**, 1–14.
74 X. Gao, F. Ramezanghorbani, O. Isayev, J. S. Smith and A. E. Roitberg, *J. Chem. Inf. Model.*, 2020, **60**, 3408–3415.
75 J. S. Smith, O. Isayev and A. E. Roitberg, *Chem. Sci.*, 2017, **8**, 3192–3203.
76 J. S. Smith, B. Nebgen, N. Lubbers, O. Isayev and A. E. Roitberg, *J. Chem. Phys.*, 2018, **148**, 241733.
77 J. S. Smith, B. T. Nebgen, R. Zubatyuk, N. Lubbers, C. Devereux, K. Barros, S. Tretiak, O. Isayev and A. E. Roitberg, *Nat. Commun.*, 2019, **10**, 2903.
78 D. Folmsbee and G. Hutchison, *Int. J. Quantum Chem.*, 2021, **121**, e26381.
79 D. P. Kingma and J. Ba, *International Conference on Learning Representations*, 2015.