JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim

Application Note

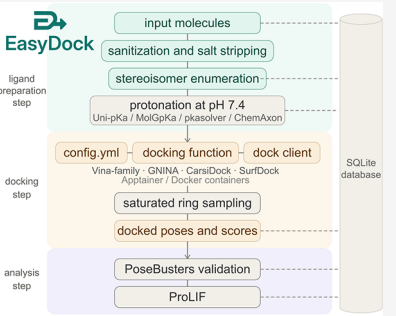
EasyDock 1.3: An Automated Pipeline for Molecular Docking

Guzel Minibaeva, Veincent Yap, and Pavel Polishchuk*

Cite This: https://doi.org/10.1021/acs.jcim.6c01221

Read Online

ABSTRACT: Molecular docking is widely used in drug design, particularly for large compound libraries. We previously developed EasyDock, an automated docking pipeline with multiserver task distribution to support such large-scale campaigns, and present here its extended version. Supported docking engines now include Vina-family CPU- and GPU-based variants (QVina2, Vina-GPU, etc.) and deep-learning-based engines (CarsiDock, SurfDock) via a built-in client−server architecture. Ligand preparation was enriched with salt stripping, stereoisomer enumeration, and conformational sampling of saturated ring systems. Integration of open-source protonation tools (pkasolver, MolGpKa, Uni-pK_a) replaces previously required commercial software, making the pipeline fully open source. Postdocking analysis now includes protein−ligand interaction fingerprint (PLIF) computation and pose quality assessment via PoseBusters. We provide Apptainer/Docker containers for the docking engines and protonation tools, simplifying installation and HPC deployment. The source code is available at https://github.com/ci-lab-cz/easydock.



INTRODUCTION

Computational methods have become central to modern drug discovery by streamlining and accelerating the identification of promising molecules.[1−3] Among them, molecular docking is a key technique for predicting ligand binding poses and affinities, enabling rapid compound prioritization, reducing experimental costs, and supporting virtual screening, lead optimization, and drug repurposing.[3,4] As computational power, scoring functions, and structural biology continue to advance, its role in modern drug discovery is expected to grow further.

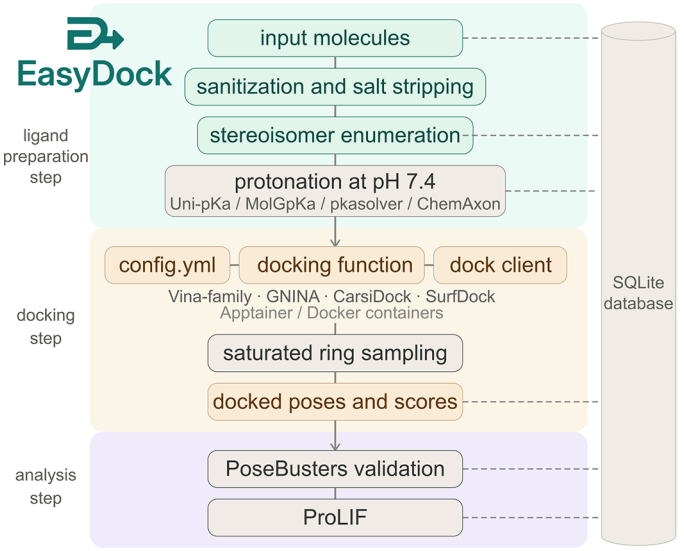
A wide range of open-source docking tools have emerged recently. Empirical-based engines such as AutoDock Vina,[5] Smina,[6] QVina2,[7] and Uni-Dock,[8] along with CNN-scoring tools like GNINA,[9] differ primarily in their scoring functions and GPU acceleration. More recently, deep-learning-based approaches including DiffDock,[10] CarsiDock,[11] and SurfDock[12] have introduced, offering alternative paradigms for pose prediction based on generative models and learned representations. Despite this diversity, the practical adoption in large-scale workflows remains hindered by several factors.

First, most tools lack fully automated pipelines covering all preparatory steps: protonation state assignment, stereoisomer enumeration and format conversion. Existing wrappers such as VirtualFlow,[13] DockStream,[14] ChemFlow,[15] and DockString[16] address some of these steps, but each has some shortcomings. VirtualFlow relies on the commercial ChemAxon cxcalc utility for protonation; DockStream and ChemFlow depend on the Schrödinger's Epik;[17] and DockString uses OpenBabel,[18] whose rule-based protonation is unsuitable for pH-dependent state assignment, especially with multiple or coupled ionizable groups. This dependence on commercial or oversimplified components restricts accessibility and reproducibility, particularly for academic groups without institutional licenses.

Second, deploying modern deep-learning docking tools is challenging. SurfDock, for example, requires specific versions of PyTorch, geometric deep-learning libraries, protein language model embeddings, and surface fingerprint computation, making environment setup error-prone and time-consuming. Many of these tools lack robust containerized distributions, and their integration into automated screening pipelines is often nontrivial. Third, existing pipelines rarely include postdocking analysis, leaving protein−ligand interaction fingerprint (PLIF) computation and pose quality assessment to the user.

We previously developed EasyDock,[19] a modular open-source pipeline which automates ligand docking except the protein preparation step. Here, we present an extended version. Supported docking engines now include deep-learning-based methods (SurfDock and CarsiDock) and additional Vina-family tools (QVina2, Vina-GPU, etc.) complementing the previous AutoDock Vina support. To streamline such integrations, we developed a client-server architecture in which each docking engine is distributed as an Apptainer/Docker container, simplifying installation, distribution, and HPC deployment; this modular design also allows new engines to be added with relatively little effort. Ligand preparation was extended with open-source protonation via MolGpKa,[20] pkasolver,[21] and Uni-

Received: April 20, 2026
Revised: June 3, 2026
Accepted: June 4, 2026




```python
from easydock.run_dock import docking
from easydock.dock.vina_dock import mol_dock
from easydock.protonation import protonate_molgpka
from rdkit import Chem

# protonation
data = [('NCC(=O)O', 'glycine'), ('C1CNCCN1',  'piperazine'])

for smi, name in protonate_molgpka(data):
    mol = Chem.MolFromSmiles(smi)
    if mol:
        mol.SetProp('_Name', name)
        mols.append(mol)

# docking
for mol_name, res in docking(mols, dock_func=mol_dock,
    dock_config='config.yml', ncpu=4):
    print(f'Molecule: 'mol_name')
    print(f'Results: 'res')
```

Figure 1. (Left) Overview of the EasyDock pipeline. Gray blocks indicate optional steps. A docking function is imported for each supported program. The config.yml file contains all docking settings. (Right) Example of using the EasyDock Python API for the docking stage.

pK_a,[22] removing the need for commercial software. PLIF computation[23] and PoseBusters-based pose validation[24] were integrated into the pipeline. EasyDock remains accessible through both a command-line interface and a Python API for straightforward integration into third-party workflows.

EASYDOCK ARCHITECTURE AND NEW FEATURES

EasyDock automates ligand preparation, docking, and post-docking analysis, but does not cover protein preparation. EasyDock is accessible through a command-line interface (*easydock*) and a Python API (*easydock.run_dock.docking*). The screening pipeline is divided into two independent stages: ligand preparation and ligand docking (Figure 1) so that ligands can be prepared once and reused with different docking programs or protein targets without repeating a preparation step.

EasyDock operates through a generator function that takes as input a list of molecules, a docking function wrapping a specific docking program, and a YAML configuration file (config.yml). The configuration file specifies the receptor, docking box coordinates, other program-specific parameters, and therefore varies by docking program. Example configuration files for all supported programs are provided in the repository.

Through its Python API, EasyDock can also be embedded in third-party software. On this basis, we implemented tools for automated *de novo* generation and structure optimization - CReM-dock[25] (https://github.com/ci-lab-cz/crem-dock) and CReM-opt (https://github.com/ci-lab-cz/crem-opt). EasyDock has also been used in combination with REINVENT[26] and ChemTS[27] frameworks for *de novo* molecular generation, demonstrating compatibility with external generative tools.

Client–Server Architecture for Integration of Docking Engines

EasyDock implements a lightweight client-server architecture based on interprocess communication to support docking programs with complex runtime dependencies, including deep-learning engines such as CarsiDock and SurfDock and also GPU-accelerated Vina family tools. This design separates the EasyDock pipeline from each program runtime environment, allowing engines to run in self-contained Apptainer/Docker containers without exposing dependencies to the host. It also separates one-time initialization from the repeatedly executed docking step.

The protocol uses JSON strings (one JSON object per line) sent through a subprocess's standard input and output. The EasyDock client starts the docking server as a long-lived child process and communicates with it using three commands: *info*, *init*, and *dock*. *info* retrieves server defaults, such as ligand and output pose formats (PDBQT, MOL, SDF, etc.) and preferred batch size. *init* sends docking settings, including receptor and auxiliary file paths, to initialize the program once at startup and avoid repeated setup. *dock* sends a batch of ligands and returns docking results with predicted binding scores and poses.

To minimize overhead, a single server process is kept alive within a worker thread for the duration of a screening campaign; the connection is automatically re-established if the server process terminates unexpectedly. On the server side, each supported engine implements the same three-command interface, allowing new docking programs to be integrated via a minimal server script that handles initialization and per-batch docking calls.

Three docking engines were integrated through this protocol, each running as a self-contained server inside its own container. Selection was guided by user requests, with priority given to tools covering different hardware settings (CPU and GPU) so users can match workflows to available resources. Using methods with different scoring functions and modeling strategies also provides more orthogonal predictions, strengthening consensus ranking and confidence in docking results.

CarsiDock[11] is a deep-learning docking method that predicts protein–ligand distance matrices and converts them into binding poses by geometry optimization. In EasyDock, it takes ligands as SMILES strings. During initialization, the receptor PDB file and a reference ligand defining the binding pocket are provided, and the CarsiDock network and RTMScore model[28] are loaded into GPU memory once. Ligand conformers are then generated during docking, and the best pose is scored with RTMScore.

SurfDock[12] is a deep-learning docking method that combines protein sequence, 3D structural graphs, and surface features in an equivariant diffusion model to predict ligand binding poses.

https://doi.org/10.1021/acs.jcim.6c01221
J. Chem. Inf. Model. XXXX, XXX, XXX–XXX

Table 1. Screening Run Times for a Set of 819 Ligands of Estrogen Receptor 1 on CPU (128 Cores) and GPU Nodes (16 cores, 1 A100)[a]

Program	Compute node	Docking wall time (h:mm:ss)	CPU utilization	Median docking time of a single ligand	AUC	PoseBusters
Vina	CPU	0:12:30	66%	19.1 s (4 cores)	0.567	99.4%
QVina	CPU	0:05:48	85%	13.9 s (4 cores)	0.565	99.4%
Vina-GPU	GPU	0:31:13	4%	1.7 s	0.558	99.0%
GNINA	GPU	0:41:50	88%	32.9 s	0.587	95.4%
CarsiDock	GPU	0:51:59	65%	3.8 s	0.566	41.1%
SurfDock	GPU	2:15:14	75%	10.1 s	0.559	17.8%

[a]Vina, QVina and Gnina used exhaustiveness 16.

In EasyDock, it accepts ligands as MOL blocks. During initialization, the receptor and reference ligand are placed in the required directory structure and preprocessed for surface and pocket features. Ligand batches are then docked by the diffusion model. Returned poses are ranked by pose confidence, and screen confidence is used as the final docking score.

The Vina-family server supports six programs: Vina-GPU, QVina2-GPU, QVinaW-GPU, and their CPU counterparts.[5,7,29,30] The engine is chosen by a configuration parameter. All use the same scoring function but different search algorithms: QVina2 speeds up Vina with similar accuracy, while QVinaW extends QVina2 with interprocess spatiotemporal integration for blind docking. Ligands are provided in PDBQT format, and each molecule is docked by running the selected binary as a subprocess, with scores and poses extracted from the PDBQT output.

We measured docking runtimes for 819 estrogen receptor 1 ligands from LIT-PCBA[31] (25 agonists, 794 inactives; Table 1). Although this is a challenging docking benchmark and all programs showed low AUC values, it was suitable for runtime comparison. QVina and Vina were the fastest, but they used 128 CPU cores, whereas the other programs ran on GPU nodes with one A100 GPU and 16 cores. Settings for all programs are provided in the examples directory of the repository. Vina and QVina used 4 cores per molecules, for Gnina the number of cores was set to 1, other programs do not provide control over the number of cores. Vina's lower CPU utilization (66%) was caused by a few molecules with long docking times (3−5 min) finishing last when most CPUs were idle, which reduced overall statistics. In our experience, screening time scales nearly linearly with the number of compounds, so these timings can be reasonably extrapolated to larger sets, where the effect of a few slow docking molecules is smaller. Final runtimes depend strongly on settings and available hardware and should be optimized individually. In general, the number of molecules docked in parallel (*-c* argument) and the number of cores per molecule (configurable for some programs) should be set so that their product exceeds the available CPU count.

Integration of Arbitrary External Docking Programs

To extend EasyDock's compatibility beyond its natively integrated engines, we implemented an experimental generic interface that allows any external docking program to be incorporated into the pipeline without modifying the core codebase. The interface is configured entirely through a YAML file specifying the executable or Python script, the conda or virtual environment, ligand input/output formats, argument names, and a score extraction pattern. It supports both file- and stream-based I/O, three input representations (PDBQT, SMILES, MOL block), and three output formats (PDBQT, PDB, SDF). Configuration examples and implementation details are provided in Supporting Information.

Ligand Protonation

The protonation state of ligands affects both the hydrogen bond network formed during docking as well as atom charges, which may be considered by a docking program; these effects collectively influence ligand-protein recognition and the predicted binding affinity.[32,33] Previously, we incorporated Chemaxon tool, but its accessibility is limited as a commercial product requiring a paid license. To address this, we integrated several open-source tools, including MolGpKa,[20] pkasolver,[21] and Uni-pK_a,[22] demonstrating moderate to high accuracy of pK_a predictions.[34,35]

pkasolver uses a graph neural network with transfer learning to predict microstate pK_a values.[21] It represents the protonated and deprotonated forms of an acid−base pair as molecular graphs, processes them with a GIN-based GNN, and predicts the pK_a from the paired graph embeddings. The model was pretrained on ChEMBL structures with pK_a values predicted by Epik, then fine-tuned on experimental pK_a data. MolGpKa is a graph-convolutional neural network that learns pK_a-relevant chemical patterns automatically from the molecular structure and predicts pK_a for each ionizable site.[20] The model was trained on ChEMBL structures with predicted pK_a by ACD and protonation sites predicted by Epik. Both pkasolver and MolGpKa were integrated into EasyDock through their native Python API.

Uni-pK_a is a thermodynamics-aware machine-learning framework built on a modified Uni-Mol molecular encoder (a Transformer-based 3D molecular representation model), combined with (i) a microstate enumerator, (ii) a free-energy-based formulation that predicts microstate free energies, and (iii) a module that converts predicted free energies into macro- or micro-pK_a values while preserving thermodynamic consistency across coupled protonation equilibria.[22] Uni-pK_a was trained on ChEMBL structures with pK_a values predicted by Chemaxon and fine-tuned on experimental data, including DataWarrior pKaInWater[36] and selected entries from the iBond (https://ibond.las.ac.cn/).

Uni-pK_a integration required a container to isolate dependencies conflicting with the main EasyDock environment. Several fixes were applied that had not been addressed by the Uni-pK_a developers in open issues on the GitHub repository. The entire pipeline was substantially refactored to enable parallel execution across multiple CPUs and to support stream-based processing via stdin/stdout pipes. The same containerized approach can be applied to integrate additional protonation tools.

By default, the major microspecies is predicted at pH 7.4. This value can be changed via a command line for all three

https://doi.org/10.1021/acs.jcim.6c01221
J. Chem. Inf. Model. XXXX, XXX, XXX−XXX

protonation tools. All tools treat input tautomeric forms explicitly and do not modify them.

Our goal was not a comprehensive comparison of different tools in predicting the major microspecies at pH 7.4, but rather to identify possible discrepancies and notify users about them. As a common denominator, we used predictions from Chemaxon, the only commercial tool accessible to us. For the study, we selected structures from ChEMBL33 with molecular mass of 500 Da or less. Stereochemistry was stripped, as all evaluated tools except Uni-pK_a operate at 2D level. For Uni-pK_a a single randomly chosen stereoisomer was used. After duplicate removal 1,748,091 structures remained. Chemaxon (version 24.2.2) was used to obtain reference protonated forms. In total, 807,407 structures were assigned protonation states differed from the initial ones by at least one of the programs (Table 2).

Table 2. Comparison of Ligand Protonation State Predictions by Five Tools against Chemaxon (version 24.2.2) as the Reference[a]

Program	Total molecules changed by at least one of the programs	Number of molecules with different protonation states from Chemaxon	Protonation wall time of 10,000 molecules (CPU utilization)
OpenBabel	807,407	278,660 (34.5%)	2 s (1 CPU)
MolGpKa		236,156 (29.2%)	743 s (CPU node) (74%)
			328 s (GPU node) (68%)
MolGpKa fix		209,798 (26%)	782 s (CPU node) (72%)
			306 s (GPU node) (75%)
pkasolver fix		378,257 (46.8%)	171 s (CPU node) (48%)
Uni-pK_a		172,323 (21.3%)	1781 s (CPU node) (93%)
			834 s (GPU node) (77%)

[a]Dataset: ChEMBL structures with molecular weight ≤ 500 Da, restricted to molecules whose protonation state was modified by at least one tool. Runtimes were measured on a random subset of 10,000 molecules. Protonation was run on CPU nodes (128 cores, 256 GB RAM) and GPU nodes (16 CPU cores, A100 GPU, 256 GB RAM).

OpenBabel served as the baseline - a simple, fast, rule-based method. It disregards the mutual influence of ionized groups and therefore tends to overestimate the number of charged states. It produced 278,660 differently protonated molecules relative to Chemaxon.

MolGpKa exhibited a better agreement with Chemaxon (Table 2); nevertheless, 236,156 structures showed protonation states different from the Chemaxon. To address several common discrepancies, we implemented manually crafted rules, which improved consistency. These rules were created specifically to fix protonated centers at pH 7.4 and cannot be applied at other pH values. The full list is provided in Supporting Information. Examples include correcting deprotonation of multiple OH groups within a single ring of polyphenols, overprotonation of closely placed nitrogens, the protonation of amide groups, etc. Since these rules rely on subjective judgment, the original version of MolGpKa is also available via *molgpka* protonation argument, while the fixed version is invoked via *molgpka_fix*.

pkasolver showed substantially more discrepancies relative to Chemaxon. For this reason, we incorporated only one obvious fix − reverting protonated nitrogens in amide/sulfonamide groups to their neutral states, enabled by default for all pkasolver predictions.

Among the evaluated methods, Uni-pK_a demonstrated the best agreement with Chemaxon, with 172,323 structures showing differing protonation states. To characterize these discrepancies, we performed a functional group-level analysis (see Supporting Information). We identified several groups that Chemaxon systematically deprotonated while Uni-pK_a did not, including N-acetyl-N-arylamine (cNC(C)=O), N′-acetyl aroylhydrazide (cC(=O)NNC(C)=O), diaroylhydrazine (cC(=O)NNC(c)=O), etc. A number of discrepancies also arose from competing protonation sites with close pK_a values. Uni-pK_a results were reproducible for most molecules because a fixed seed is used for conformer embedding. However, the predicted protonation states for 13,376 (1.7%) out of 793,431 compounds having multiple stereoisomers were different for different stereoisomers indicating sensitivity of Uni-pK_a to 3D structures. Multiple runs also revealed that 108 compounds yielded different protonation states for some stereoisomers across runs. The list of these stereoisomers is given in Supporting Information.

Protein−Ligand Interaction Fingerprints

The binding mode of a docked ligand can provide valuable information for compound prioritization, particularly in lead optimization and when selecting compounds expected to recapitulate interactions observed in a reference cocrystal structure.[37−39] To support this, EasyDock integrates protein−ligand interaction fingerprint (PLIF) computation based on the ProLIF library.[23]

Ten interaction types are detected for each docked pose: hydrophobic contacts, hydrogen bond donation and acceptance, cationic and anionic interactions, cation−π and π−cation contacts, face-to-face and edge-to-face $\pi-\pi$ stacking, and metal coordination. The resulting binary fingerprint encodes the presence or absence of each interaction with each protein residue are stored in the EasyDock SQLite database, supporting incremental calculation and later retrieval without reprocessing.

A key practical feature is similarity scoring against a user-defined reference fingerprint, typically derived from a known active compound or a cocrystallized ligand. Similarity is computed as the fraction of reference interactions reproduced by the docked pose, ignoring additional contacts not present in the reference. This allows compounds to be filtered or ranked by their ability to reproduce the pharmacophoric interactions of interest, complementing the docking score as an orthogonal selection criterion.

Pose Validation with PoseBusters

A known limitation of deep learning-based molecular docking approaches, is that high-ranking poses may contain physically implausible features - steric clashes with the protein, distorted bond geometries, or violations of fundamental chemical rules - that are invisible to the scoring function but would be rejected on physical grounds upon manual inspection. These artifacts arise from the simplifications inherent in docking search algorithms and scoring functions. Filtering such poses before downstream analysis reduces the risk of pursuing false positives.

EasyDock integrates PoseBusters,[24] a rule-based pose validation tool that applies geometry, chemistry, and protein−ligand compatibility checks to docked structures. Validation is performed in the *dock_fast* mode, which evaluates ligand geometry and clashes with the receptor without a cocrystal reference structure. Results − a pass/fail flag per molecule and

https://doi.org/10.1021/acs.jcim.6c01221
J. Chem. Inf. Model. XXXX, XXX, XXX−XXX

pose − are stored in the EasyDock database, enabling their use as filters at the retrieval stage. Molecules can be extracted from the database with an option to retain only those that passed all PoseBusters checks, enabling one-step filtering. As with PLIF computation, PoseBusters validation is performed incrementally: previously processed molecules are skipped on subsequent runs, making the check practical for large databases in stages.

Other Features

Ligand preparation was extended with salt stripping and stereoisomer enumeration. Salt stripping is performed with the RDKit SaltRemover, which removes counterions and small fragments commonly present in compound library records. Molecules remaining multicomponent after salt removal are flagged and excluded from docking. The original unmodified SMILES is retained in the database alongside the processed structure to ensure traceability.

For molecules supplied as 2D structures or SMILES having undefined chiral centers or double bonds, stereoisomers are enumerated using RDKit EnumerateStereoisomers, with the number of generated isomers controlled by a user-defined parameter. Each stereoisomer receives a unique stereo ID and is treated as an independent entity: stored separately and docked independently. Molecules provided as 3D SDF structures bypass stereoisomer enumeration, and their input geometry is used directly for docking. Some complex molecules require considerable time to generate physically plausible 3D structures; if plausible 3D coordinates cannot be generated within 300 s, enumeration is interrupted and the molecule is skipped.

For docking programs requiring 3D coordinates for input ligands, starting conformers are generated with RDKit's ETKDGv3 algorithm followed by MMFF94 force field minimization. To improve docking of compounds with saturated rings systems in programs treating rings as rigid bodies, multiple input conformers can be generated optionally. Up to 50 initial conformers are generated, clustered by RMSD of ring atoms and their directly bonded neighbors using agglomerative clustering. A representative subset of structurally diverse conformers is retained. Each conformer is submitted to docking independently, and all poses from the best-scoring across all input conformers are reported.

We evaluated the performance of ring sampling in redocking on a set of 1410 protein−ligand complexes from the refined subset of PDBbind (version 2020)[40] containing at least one saturated ring system of up to 8 atoms. Redocking was performed using Vina and GNINA. In both cases, ring sampling increased the number of successfully redocked structures, defined as poses within 2Å RMSD of the X-ray reference. While the absolute differences were small (Figure S1), the paired *t*-test showed the statistical significance of the improvements (Table S2). The computational runtime increased approximately 3-fold due to the docking of additional starting conformers, that should be considered when applying ring sampling to large-scale virtual screening.

Other enhancements include more detailed logging and documentation. The logging level is controlled via a command line argument; setting it to DEBUG mode produces additional messages that help identify the source of potential issues. The documentation covers not only installation and usage with examples but also includes a developer section describing internal conventions and interfaces (https://easydock.readthedocs.io/en/latest/).

To extract docked poses and scores, a command line utility *get_sdf_from_easydock* retrieves selected poses as an SDF file or returns structures as SMILES together with docking scores.

Interrupted screenings (due to system failures, timeouts, or user intervention) can be resumed by restarting the same command; results are committed incrementally to the SQLite database, and completed molecules are automatically skipped.

LIMITATIONS

Protein structure preparation and docking grid box definition are not integrated in EasyDock, as curation of protein structures may be nontrivial and often requires manual intervention. We recommend preparing the protein manually and carefully inspecting the structure, in particular for missing side chains or residues and misplaced hydrogen atoms (especially on histidines), since such issues affect the entire docking campaign.

All EasyDock features are available on Linux and macOS systems. On Windows, only docking programs with native Windows binaries (for example, a standalone Vina binary) can be run by substituting *script_file* with a Windows binary in the configuration. Containerized features can be run via Docker, although EasyDock on Windows has not been thoroughly tested.

The client-server docking protocol does not support distributed virtual screening, which was implemented in the previous EasyDock version via the Dask library.[41] Dask setup was found to be relatively complex, therefore we are looking for a more convenient alternative rather than reimplementing Dask support. The previously integrated docking engines, however, can still use it.

FUTURE ENHANCEMENTS

Currently, docking results for the same ligand set are stored in separate databases for different proteins or docking programs. In future versions, we plan to combine all results for a given molecule set into a single database to facilitate consensus predictions and better assessment of ligand selectivity. We also aim to add pose rescoring with alternative docking programs as a preliminary step before more computationally intensive methods such as MM-GBSA/PBSA[42−44] or free energy perturbation.[45] We are also considering the incorporation of more advanced protonation prediction tools that may improve accuracy at the cost of longer computational times. This would be particularly beneficial for smaller compound libraries, where extended protonation runtimes remain manageable.

CONCLUSIONS

The current version of EasyDock provides an automated, open-source pipeline covering all stages of ligand-protein docking except protein preparation. The pipeline is accessible via both the command line and a Python API. Its client-server architecture enables seamless integration with multiple docking engines, substantially broadening the range of supported tools compared to the previous version. Newly added engines include deep-learning-based methods (CarsiDock and SurfDock) as well as additional Vina-family tools (QVina2, Vina-GPU, QVina2-GPU), extending the GNINA and AutoDock Vina support already present in the earlier release. Several open-source protonation tools were integrated and evaluated; among them, Uni-pK_a demonstrated the highest concordance with the commercial ChemAxon software. Other key additions include protein−ligand interaction fingerprint (PLIF) computation for

https://doi.org/10.1021/acs.jcim.6c01221
J. Chem. Inf. Model. XXXX, XXX, XXX−XXX

downstream pose analysis, PoseBusters-based pose validation, and sampling of saturated ring conformations to improve the accuracy of docking poses. EasyDock is accompanied by comprehensive documentation covering installation, usage, advanced workflows, analysis, and customization.

■ ASSOCIATED CONTENT

Data Availability Statement

The source code of EasyDock is available under the BSD-3 license at https://github.com/ci-lab-cz/easydock. The third-party integrated protonation and docking tools are distributed under Apache 2.0 or MIT licenses, with the exception of Meeko, which is licensed under the LGPL-2.1 license. The validation data for protonation tools are available in the Zenodo repository at 10.5281/zenodo.17179256. Prebuilt docking and protonation containers are available at 10.5281/zenodo.19652220 and 10.5281/zenodo.17506577.

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.6c01221.

Additional details of integration of arbitrary external docking programs with examples; the list of SMARTS patterns used to fix MolGpKa outputs; detailed statistics on computational performance of protonation tools and discrepancies in ligand protonation patterns between Uni-pK_a and Chemaxon; performance and statistics of docking using ring sampling (PDF)

Performance and statistics of docking using ring sampling (ZIP)

■ AUTHOR INFORMATION

Corresponding Author

Pavel Polishchuk − *Institute of Molecular and Translational Medicine, Faculty of Medicine and Dentistry, Palacky University, Olomouc 779 00, Czech Republic*; orcid.org/0000-0001-5088-8149; Email: pavlo.polishchuk@upol.cz

Authors

Guzel Minibaeva − *Institute of Molecular and Translational Medicine, Faculty of Medicine and Dentistry, Palacky University, Olomouc 779 00, Czech Republic*

Veincent Yap − *Nanyang Technological University, Singapore 639798, Singapore*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.6c01221

Author Contributions

G.M.: software, validation, investigation, writing − original draft, visualization, writing − review and editing; V.Y.: software, validation; P.P.: conceptualization, methodology, validation, writing - original draft, writing − review and editing, visualization, supervision, funding acquisition.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through INTER-EXCELLENCE II LUAUS23262 and the e-INFRA CZ (ID:90254). Authors acknowledge EuroHPC Joint Undertaking for awarding the project ID EHPC-BEN-2025B12-028 access to Karolina at IT4Innovations, Czech Republic.

■ REFERENCES

(1) Sadybekov, A. V.; Katritch, V. Computational approaches streamlining drug discovery. *Nature* **2023**, *616* (7958), 673−685.

(2) Macalino, S. J. Y.; Gosu, V.; Hong, S.; Choi, S. Role of computer-aided drug design in modern drug discovery. *Archives of Pharmacal Research* **2015**, *38* (9), 1686−1701.

(3) Kitchen, D. B.; Decornez, H.; Furr, J. R.; Bajorath, J. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nat. Rev. Drug Discovery* **2004**, *3* (11), 935−949.

(4) Pinzi, L.; Rastelli, G. Molecular Docking: Shifting Paradigms in Drug Discovery. *International Journal of Molecular Sciences* **2019**, *20* (18), 4331.

(5) Trott, O.; Olson, A. J. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J. Comput. Chem.* **2010**, *31* (2), 455−461.

(6) Koes, D. R.; Baumgartner, M. P.; Camacho, C. J. Lessons Learned in Empirical Scoring with smina from the CSAR 2011 Benchmarking Exercise. *J. Chem. Inf. Model.* **2013**, *53* (8), 1893−1904.

(7) Alhossary, A.; Handoko, S. D.; Mu, Y.; Kwoh, C.-K. Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics* **2015**, *31* (13), 2214−2216.

(8) Yu, Y.; Cai, C.; Wang, J.; Bo, Z.; Zhu, Z.; Zheng, H. Uni-Dock: GPU-Accelerated Docking Enables Ultralarge Virtual Screening. *J. Chem. Theory Comput.* **2023**, *19* (11), 3336−3345.

(9) McNutt, A. T.; Li, Y.; Meli, R.; Aggarwal, R.; Koes, D. R. GNINA 1.3: the next increment in molecular docking with deep learning. *J. Cheminf.* **2025**, *17* (1), 28.

(10) Corso, G.; Deng, A.; Polizzi, N.; Barzilay, R.; Jaakkola, T. Deep Confident Steps to New Pockets: Strategies for Docking Generalization. In *International Conference on Learning Representations (ICLR)*, **2024**.

(11) Cai, H.; Shen, C.; Jian, T.; Zhang, X.; Chen, T.; Han, X.; Yang, Z.; Dang, W.; Hsieh, C.-Y.; Kang, Y.; et al. CarsiDock: a deep learning paradigm for accurate protein−ligand docking and screening based on large-scale pre-training. *Chem. Sci.* **2024**, *15* (4), 1449−1471.

(12) Cao, D.; Chen, M.; Zhang, R.; Wang, Z.; Huang, M.; Yu, J.; Jiang, X.; Fan, Z.; Zhang, W.; Zhou, H.; et al. SurfDock is a surface-informed diffusion generative model for reliable and accurate protein−ligand complex prediction. *Nat. Methods* **2025**, *22*, 310−322.

(13) Gorgulla, C.; Boeszoermenyi, A.; Wang, Z.-F.; Fischer, P. D.; Coote, P. W.; Padmanabha Das, K. M.; Malets, Y. S.; Radchenko, D. S.; Moroz, Y. S.; Scott, D. A.; et al. An open-source drug discovery platform enables ultra-large virtual screens. *Nature* **2020**, *580*, 663−668.

(14) Guo, J.; Janet, J. P.; Bauer, M. R.; Nittinger, E.; Giblin, K. A.; Papadopoulos, K.; Voronov, A.; Patronov, A.; Engkvist, O.; Margreitter, C. DockStream: a docking wrapper to enhance de novo molecular design. *J. Cheminf.* **2021**, *13* (1), 89.

(15) Barreto Gomes, D. E.; Galentino, K.; Sisquellas, M.; Monari, L.; Bouysset, C.; Cecchini, M. ChemFlow—From 2D Chemical Libraries to Protein−Ligand Binding Free Energies. *J. Chem. Inf. Model.* **2023**, *63* (2), 407−411.

(16) García-Ortegón, M.; Simm, G. N. C.; Tripp, A. J.; Hernández-Lobato, J. M.; Bender, A.; Bacallado, S. DOCKSTRING: Easy Molecular Docking Yields Better Benchmarks for Ligand Design. *J. Chem. Inf. Model.* **2022**, *62* (15), 3486−3502.

(17) Johnston, R. C.; Yao, K.; Kaplan, Z.; Chelliah, M.; Leswing, K.; Seekins, S.; Watts, S.; Calkins, D.; Chief Elk, J.; Jerome, S. V.; et al. Epik: pKa and Protonation State Prediction through Machine Learning. *J. Chem. Theory Comput.* **2023**, *19* (8), 2380−2388.

(18) O'Boyle, N. M.; Banck, M.; James, C. A.; Morley, C.; Vandermeersch, T.; Hutchison, G. R. Open Babel: An open chemical toolbox. *J. Cheminf.* **2011**, *3* (1), 33.

(19) Minibaeva, G.; Ivanova, A.; Polishchuk, P. EasyDock: customizable and scalable docking tool. *J. Cheminf.* **2023**, *15* (1), 102.

(20) Pan, X.; Wang, H.; Li, C.; Zhang, J. Z. H.; Ji, C. MolGpka: A Web Server for Small Molecule pKa Prediction Using a Graph-Convolutional Neural Network. *J. Chem. Inf. Model.* **2021**, *61* (7), 3159−3165.
(21) Mayr, F.; Wieder, M.; Wieder, O.; Langer, T. Improving Small Molecule pKa Prediction Using Transfer Learning With Graph Neural Networks. *Frontiers in Chemistry* **2022**, *10*, No. 866585.
(22) Luo, W.; Zhou, G.; Zhu, Z.; Yuan, Y.; Ke, G.; Wei, Z.; Gao, Z.; Zheng, H. Bridging Machine Learning and Thermodynamics for Accurate pKa Prediction. *JACS Au* **2024**, *4* (9), 3451−3465.
(23) Bouysset, C.; Fiorucci, S. ProLIF: a library to encode molecular interactions as fingerprints. *J. Cheminf.* **2021**, *13* (1), 72.
(24) Buttenschoen, M.; Morris, G. M.; Deane, C. M. PoseBusters: AI-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chem. Sci.* **2024**, *15* (9), 3130−3139.
(25) Minibaeva, G.; Du, H.; Clark, F.; Michel, J.; Polishchuk, P. CReM-dock: de novo design of synthetically feasible structures guided by molecular docking. *Digital Discovery* **2026**, *5*, 2271−2291.
(26) Loeffler, H. H.; He, J.; Tibo, A.; Janet, J. P.; Voronov, A.; Mervin, L. H.; Engkvist, O. Reinvent 4: Modern AI−driven generative molecule design. *J. Cheminf.* **2024**, *16* (1), 20.
(27) Ishida, S.; Aasawat, T.; Sumita, M.; Katouda, M.; Yoshizawa, T.; Yoshizoe, K.; Tsuda, K.; Terayama, K. ChemTSv2: Functional molecular design using de novo molecule generator. *WIREs Computational Molecular Science* **2023**, *13* (6), No. e1680.
(28) Shen, C.; Zhang, X.; Deng, Y.; Gao, J.; Wang, D.; Xu, L.; Pan, P.; Hou, T.; Kang, Y. Boosting Protein−Ligand Binding Pose Prediction and Virtual Screening Based on Residue−Atom Distance Likelihood Potential and Graph Transformer. *J. Med. Chem.* **2022**, *65* (15), 10691−10706.
(29) Hassan, N. M.; Alhossary, A. A.; Mu, Y.; Kwoh, C.-K. Protein-Ligand Blind Docking Using QuickVina-W With Inter-Process Spatio-Temporal Integration. *Sci. Rep.* **2017**, *7* (1), 15451.
(30) Tang, S.; Ding, J.; Zhu, X.; Wang, Z.; Zhao, H.; Wu, J. Vina-GPU 2.1: Towards Further Optimizing Docking Speed and Precision of AutoDock Vina and Its Derivatives. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **2024**, *21* (6), 2382−2393.
(31) Tran-Nguyen, V.-K.; Jacquemard, C.; Rognan, D. LIT-PCBA: An Unbiased Data Set for Machine Learning and Virtual Screening. *J. Chem. Inf. Model.* **2020**, *60* (9), 4263−4273.
(32) ten Brink, T.; Exner, T. E. Influence of Protonation, Tautomeric, and Stereoisomeric States on Protein−Ligand Docking Results. *J. Chem. Inf. Model.* **2009**, *49* (6), 1535−1546.
(33) Petukh, M.; Stefl, S.; Alexov, E. The Role of Protonation States in Ligand-Receptor Recognition and Binding. *Curr. Pharm. Des.* **2013**, *19* (23), 4182−4190.
(34) Baikété, J.; Malloum, A.; Conradie, J. pKa prediction for small molecules: an overview of experimental, quantum, and machine learning-based approaches. *J. Comput.-Aided Mol. Des.* **2026**, *40* (1), 5.
(35) Sipos-Szabó, L.; Bajusz, D.; Balogh, G. T.; Keserű, G. M. Benchmarking pKa Prediction Algorithms against an Extensive, Public Data Set. *J. Chem. Inf. Model.* **2026**, *66* (8), 4607−4619.
(36) Sander, T.; Freyss, J.; von Korff, M.; Rufener, C. DataWarrior: An Open-Source Program For Chemistry Aware Data Visualization And Analysis. *J. Chem. Inf. Model.* **2015**, *55* (2), 460−473.
(37) Deng, Z.; Chuaqui, C.; Singh, J. Structural Interaction Fingerprint (SIFt): A Novel Method for Analyzing Three-Dimensional Protein−Ligand Binding Interactions. *J. Med. Chem.* **2004**, *47* (2), 337−344.
(38) Da, C.; Kireev, D. Structural Protein−Ligand Interaction Fingerprints (SPLIF) for Structure-Based Virtual Screening: Method and Benchmark Study. *J. Chem. Inf. Model.* **2014**, *54* (9), 2555−2561.
(39) Wang, D. D.; Chan, M.-T.; Yan, H. Structure-based protein−ligand interaction fingerprints for binding affinity prediction. *Computational and Structural Biotechnology Journal* **2021**, *19*, 6291−6300.
(40) Wang, R.; Fang, X.; Lu, Y.; Yang, C.-Y.; Wang, S. The PDBbind Database: Methodologies and Updates. *J. Med. Chem.* **2005**, *48* (12), 4111−4119.
(41) Rocklin, M. Dask: Parallel Computation with Blocked algorithms and Task Scheduling. In *Proceedings of the 14th Python in Science Conference*, Huff, K., Bergstra, J. Eds.; **2015**; pp 130−136.
(42) Ivanova, A.; Mokshyna, O.; Polishchuk, P. StreaMD: the toolkit for high-throughput molecular dynamics simulations. *J. Cheminf.* **2024**, *16* (1), 123.
(43) Genheden, S.; Ryde, U. The MM/PBSA and MM/GBSA methods to estimate ligand-binding affinities. *Expert Opinion on Drug Discovery* **2015**, *10* (5), 449−461.
(44) Srinivasan, J.; Cheatham, T. E.; Cieplak, P.; Kollman, P. A.; Case, D. A. Continuum Solvent Studies of the Stability of DNA, RNA, and Phosphoramidate−DNA Helices. *J. Am. Chem. Soc.* **1998**, *120* (37), 9401−9409.
(45) Clark, F.; Robb, G. R.; Cole, D. J.; Michel, J. Automated Adaptive Absolute Binding Free Energy Calculations. *J. Chem. Theory Comput.* **2024**, *20* (18), 7806−7828.

https://doi.org/10.1021/acs.jcim.6c01221
J. Chem. Inf. Model. XXXX, XXX, XXX−XXX