

TeachOpenCADD 2022: open source and FAIR Python pipelines to assist in structural bioinformatics and cheminformatics research

Dominique Sydow[†], Jaime Rodríguez-Guerra[†], Talia B. Kimber, David Schaller, Corey J. Taylor, Yonghui Chen, Mareike Leja, Sakshi Misra, Michele Wichmann, Armin Ariamajd and Andrea Volkamer*

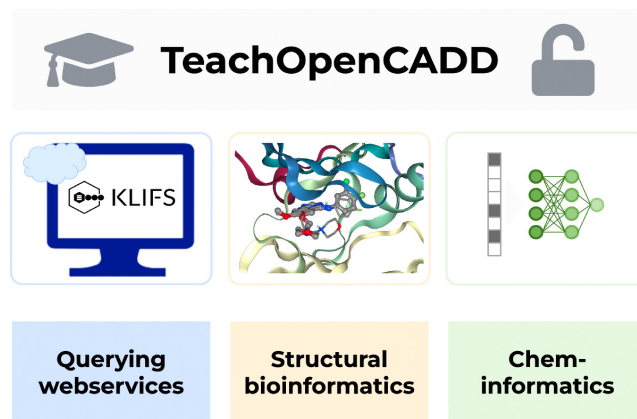
In Silico Toxicology and Structural Bioinformatics, Institute of Physiology, Charité – Universitätsmedizin Berlin, corporate member of Freie Universität Berlin and Humboldt-Universität zu Berlin, Germany

Received March 08, 2022; Revised March 30, 2022; Editorial Decision April 02, 2022; Accepted April 06, 2022

ABSTRACT

Computational pipelines have become a crucial part of modern drug discovery campaigns. Setting up and maintaining such pipelines, however, can be challenging and time-consuming—especially for novice scientists in this domain. TeachOpenCADD is a platform that aims to teach domain-specific skills and to provide pipeline templates as starting points for research projects. We offer Python-based solutions for common tasks in cheminformatics and structural bioinformatics in the form of Jupyter notebooks, based on open source resources only. Including the 12 newly released additions, TeachOpenCADD now contains 22 notebooks that cover both theoretical background as well as hands-on programming. To promote reproducible and reusable research, we apply software best practices to our notebooks such as testing with automated continuous integration and adhering to the idiomatic Python style. The new TeachOpenCADD website is available at <https://projects.volkamerlab.org/teachopencadd> and all code is deposited on GitHub.

GRAPHICAL ABSTRACT



INTRODUCTION

Computational methods play an integral role in the design-make-test-analyze (DMTA) cycle that drives real-world drug design projects (1). To address questions raised during this cycle, a single method does not suffice to deliver an answer; instead, a pipeline combining different methods can produce complementary and useful insights. Setting up such complex pipelines, however, can be difficult and time-consuming for many reasons: the scientist may not have had the training necessary to tackle these tasks (2), tools and their usage are constantly evolving (or becoming deprecated), and feeding the output from one tool into another is often not straightforward. On top of these considerations, sustainable pipelines need to be findable, accessible, interoperable, and reusable (FAIR principles (3))—not only today but in many years from now—to drive reproducible research.

In 2019, we launched the teaching platform TeachOpenCADD (4) on [GitHub](#) to help face these challenges.

*To whom correspondence should be addressed. Email: andrea.volkamer@charite.de

[†]The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.

TeachOpenCADD teaches by example how to build Python pipelines with open source resources used in the fields of cheminformatics and structural bioinformatics to answer central questions in computer-aided drug design (CADD). With these ready-to-use pipelines, we target students and teachers who need training material for CADD-related topics, as well as researchers who need a template or an inspiration to tackle their research questions. The theoretical and practical aspects of each topic are covered in an interactive Jupyter notebook (5). This setup makes it easy for users from different fields to understand the computational concepts and to get started with hands-on Python programming. We call these Jupyter notebooks *talktorials* (talk + tutorial) because their format is suited for presentations as well. The initial stack of talktorials T001–T010 covers common CADD tasks involving webserver queries, cheminformatics, and structural bioinformatics (4). We show how to fetch chemical and structural data from the ChEMBL (6) and PDB (7,8) databases and how to encode, filter, cluster, and screen such datasets to find novel drug candidates and off-targets (4). The talktorials are inspired by several online resources recommended for further reading such as Teach-Discover-Treat and CDK (9,10) and the blogs [Practical Cheminformatics](#), [RDKit blog](#), and [Is live worth living?](#). Over the last two years, the TeachOpenCADD GitHub repository underwent many additions and changes: we now have more than doubled our content and extended the application of software best practices rigorously. The full collection of talktorials is easily accessible on the new [TeachOpenCADD website](#). We comply with software best practices regarding the code style as well as maintenance and facilitate installation with a dedicated conda package.

NEW TALKTORIALS

The new stack of talktorials showcases data acquisition from additional CADD-relevant databases, adds many examples for structure-based tasks, and extends the cheminformatics side with straightforward deep learning (DL) applications. Our example use case is the EGFR kinase (19) but the talktorials are easily adaptable to other targets as long as sufficient data is available. Besides the domain-specific resources described below, we rely in all talktorials on established Python packages for data science and visualization such as NumPy (20), pandas (21), scikit-learn (22), matplotlib (23), and seaborn (24).

Webservices queries

Over the last decades, the scientific community has produced an incredible amount of data and analysis software, and adapted modern technologies to make these resources easily available via online webservices (25). However, it might not always be obvious to the beginner how to use a web application programming interface (API) to access such data and how to integrate them into larger pipelines. TeachOpenCADD dedicates several talktorials to the usage of different webservers relevant for the life sciences.

In the first TeachOpenCADD release from 2019, we already showed how to query the ChEMBL (6) and PDB (7,8)

databases. From the ChEMBL webservice, compounds and bioactivities are fetched for the EGFR kinase using the ChEMBL webresource client (26) (T001). This dataset is used in many downstream talktorials for common cheminformatics tasks (T002–T007). From the PDB webservice, we fetch a set of EGFR kinase structures based on criteria such as ‘ligand-bound structures from X-ray experiments with a resolution <3.0 Å’ using the biotite (27) and PyPDB (28) (T008) packages.

In the latest release, we now have added three more notebooks covering the usage of additional online API webservices (Figure 1, T011–T013).

T011: Querying online API webservices. We added a broad introduction on how to programmatically use online webservices from Python with a focus on REST services and web scraping. The usage of several libraries is demonstrated; e.g. we use [requests](#) to retrieve content from UniProt (29), [bravado](#) to generate a Python client for OpenAPI-compatible services — exemplified for the KLIFS database (11)—, and [Beautiful Soup](#) to scrape (parse) HTML content from the web.

T012: Data acquisition from KLIFS. KLIFS (11) is a kinase database gathering information on experimental kinase structures and interacting inhibitors. The talktorial shows how to quickly fetch data from KLIFS given a query kinase or ligand. For example, we spot frequent key ligand-interactions in EGFR based on KLIFS interaction fingerprints and we assess kinome-wide bioactivity values for the inhibitor gefitinib. These queries are demonstrated by using the [KLIFS OpenAPI](#) directly with [bravado](#), or by using the KLIFS-dedicated wrapper [OpenCADD-KLIFS](#) (30), implemented in the Python package [OpenCADD](#).

T013: Data acquisition from PubChem. PubChem (12) is a database holding chemical information on over 100 million compounds. We demonstrate how to fetch data from PubChem’s PUG-REST API (31), given the name or SMILES (32) of a query ligand. For example, we show how to fetch molecular properties for a ligand of interest by name (aspirin) and how to query PubChem for the most similar compounds given a query SMILES (gefitinib).

Data acquisition case study. A summary of the information that can be acquired automatically for a target of interest using these webservices is exemplified in Figure 2. Using the Uniprot ID of EGFR kinase as input query only, (i) 227 available EGFR structures from the PDB can be obtained and further filtered (T008); (ii) 446 available complex structures and their interaction fingerprints can be fetched from KLIFS (T012), or (iii) a total of 8463 IC50 values of molecules measured against EGFR can be acquired from ChEMBL (T001). Finally, (iv) a PubChem query with the molecule name ‘gefitinib’ showcases how to gather ligand properties or to perform a similarity search (T013).

Pocket detection, ligand–protein docking and interactions

During a drug discovery campaign, frequent questions are: What should I test next? Can you suggest a diverse set of

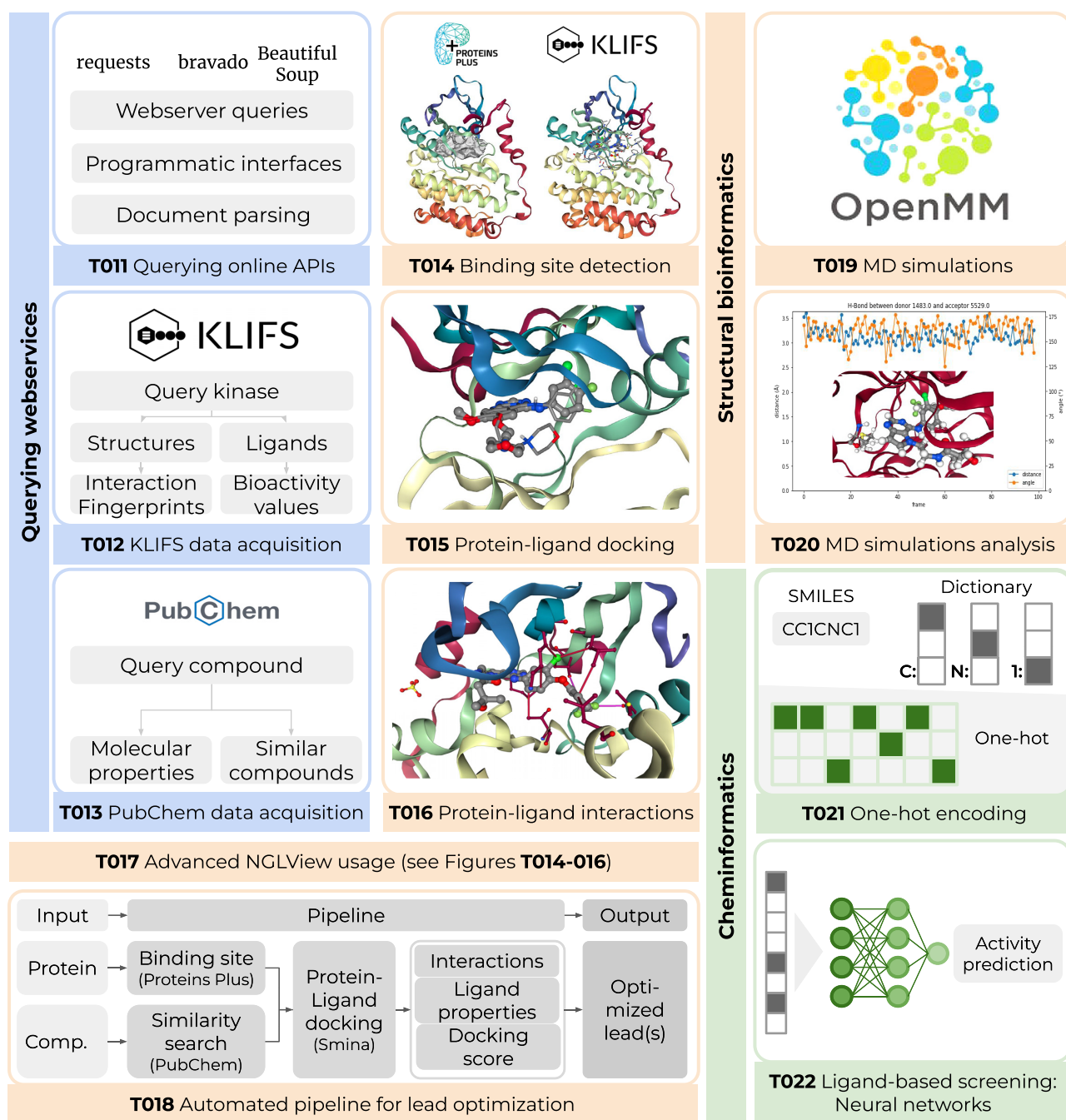


Figure 1. Overview of 12 new talktorials. (i) Querying webservices (blue): T011 gives a broad introduction to programmatic access to webservices from Python, T012 and T013 demonstrate how to query the KLIFS (11) and PubChem (12) databases for kinase and compound data, respectively. (ii) Structural bioinformatics (orange): T014 detects the binding site in an EGFR kinase structure and compares the prediction to the binding site defined by KLIFS (11). T015 performs a re-docking for an EGFR–ligand complex with Smina (13). T016 detects protein–ligand interactions in an EGFR–ligand complex structure with PLIP (14). T017 introduces basic and advanced usages of the molecular visualization tool NGLView (15), used throughout most of TeachOpenCADD’s talktorials. T018 outlines a fully automated lead optimization pipeline: Based on an input structure, the pocket is detected and a set of compounds similar to a selected ligand are fetched from PubChem (12). These compounds are docked into the selected binding site. The most promising compounds w.r.t. docking scores and interaction profiles are proposed as optimized compounds. T019 demonstrates how to set up and run a molecular dynamics (MD) simulation on Google Colab with OpenMM (16). T020 analyzes the resulting MD trajectory with a focus on the root-mean-square deviation (RMSD) between trajectory frames and the dynamics of protein–ligand interactions using MDAAnalysis (17,18). (iii) Cheminformatics (green): T021 exhibits the steps to numerically encode a small molecule from its SMILES representation. T022 lays the groundwork for deep learning and focuses on a simple feed-forward neural network for activity prediction using molecular fingerprints.

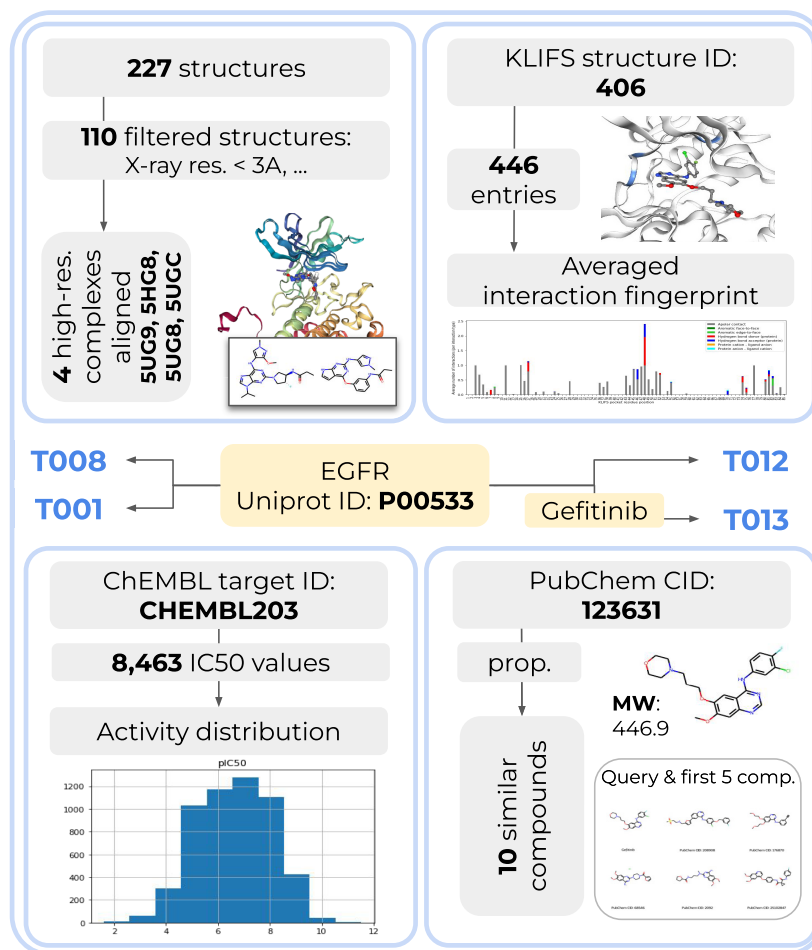


Figure 2. Data and information that can be automatically gathered for the EGFR kinase using the different web query talktorials as of September 2021, created based on ChEMBL v.27 (6) (T001), PDB (8) (T008), PubChem (12) (T013), and KLIFS (11) (T012). Input: yellow boxes, output: gray boxes, plots and molecule visualizations (using NGLView (15) and RDKit).

small molecules likely to bind to this protein? How should I modify the lead compound to increase the binding affinity? Answering these questions involves multiple scientific observations, and thus, multiple computational steps as addressed in talktorials T014–T017. Finally, an automated pipeline is compiled (T018) to process a protein structure and a lead compound, and propose several similar ligands with optimized estimated affinities and interactions based on the docked protein–ligand structures.

T014: Binding site detection. First, we need to know where ligands may bind to a protein of interest. Sometimes the binding site is known from experimental protein–ligand structures. If only experimental apo structures are available, putative binding sites can be predicted with computational methods. We demonstrate how to use the REST API of the ProteinsPlus webserver (33) to detect the main pocket of an EGFR structure using the DoGSiteScorer (34) pocket detection algorithm. To validate our results, the predicted pocket is compared with the KLIFS-defined kinase pocket, which encompasses 85 residues shown to be in contact with ligands based on X-ray complex structures (35).

T015: Protein–ligand docking. Next, we introduce molecular docking to predict the binding mode of a ligand to its protein target by explaining several sampling algorithms and scoring functions, as well as commenting on limitations and interpretation of docking results. The theoretical background is then applied in a re-docking experiment aiming to reproduce the binding mode observed in a published X-ray structure of EGFR. Protein and ligand are prepared using Pybel (36), the ligand is docked into the protein using Smina (13), and finally, the docking poses are visually inspected using NGLView (15). We refer to [JupyterDock](#) for further reading on different docking protocols run from Jupyter notebooks.

T016: Protein–ligand interactions. Understanding which forces and interactions drive molecular recognition is important for drug design (37). In this talktorial, we give an introduction to relevant protein–ligand interactions and their programmatic detection using the protein–ligand interaction profiler PLIP (14). To this end, all interactions in an EGFR–ligand complex fetched from the PDB are detected and visualized in 3D using NGLView.

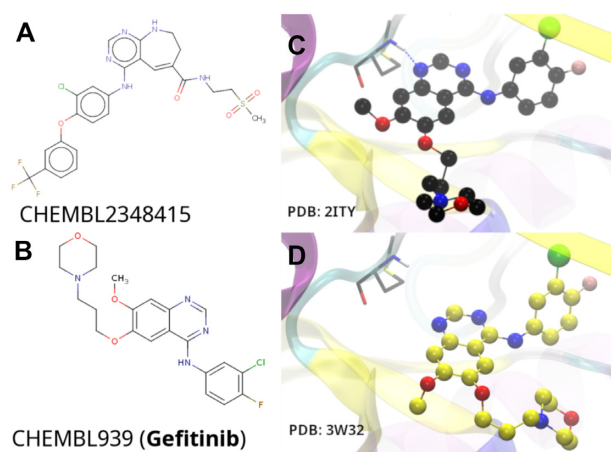


Figure 3. Case study for talktorial T018 depicting (A) 2D structure of the input ligand for the pipeline that was used with an EGFR crystal structure (PDB: 3W32, $IC_{50} = 75\text{nM}$); (B) 2D structure of gefitinib ($IC_{50} = 0.17\text{nM}$), an EGFR ligand found during similarity searches; (C) crystal structure of gefitinib co-crystallized with EGFR (PDB: 2ITY, black CPK representation); (D) docked pose of gefitinib (yellow CPK representation). Some segments of the protein structure have been removed for clarity. The ligand RMSD between (C) and (D) and the discovery of a higher-affinity ligand demonstrate the utility of the fully automated pipeline for early stage drug discovery.

T017: Advanced NGLView usage. Since the molecular visualization package NGLView is invoked in many talktorials, we give a dedicated overview of its usage and show some advanced cases on how to customize residue coloring, and how to create interactive interfaces with IPyWidgets. In addition, access to the JavaScript layer NGL (38,39) is showcased to perform operations that are not exposed to the Python wrapper NGLView.

T018: Automated pipeline for lead optimization. All previous talktorials are composed of stand-alone tasks that can be completed independently. Proposing ligand modifications that will improve interaction patterns with target proteins in a complete end-to-end process, however, necessitates orchestration of code and concepts implemented in the previously discussed talktorials T014–T017. A docking pipeline is constructed in T018 that is comprised of both a step-by-step demonstration and a fully automated procedure. Given a query protein and a lead compound, similar ligands fetched from PubChem are suggested, which show optimized affinity estimates and interaction profiles based on generated docking poses.

Lead optimization case study. As a case study, an EGFR crystal structure (PDB: 3W32) and its co-crystallized ligand were used as inputs for the pipeline. A similarity search led to the generation of a small library of compounds from PubChem for docking and further analysis to find compounds ideally more affine than the co-crystallized ligand. Using the pipeline, an approved breast cancer drug, gefitinib, was found in the top 50 of docked poses (Figure 3). Gefitinib ($IC_{50} = 0.17\text{ nM}$ (40)) is at least an order of magnitude more affine for EGFR than the measured affinity of the input ligand ($IC_{50} = 75\text{ nM}$ (41)). Gefitinib's predicted geometry was $<2\text{ \AA}$ RMSD from a crystal structure of wild-type

EGFR (PDB: 2ITY). This retrospective example demonstrates the utility of a fully automated pipeline and potential application as prospective tool.

Molecular dynamics

Experimentally resolved structures offer immense insights for drug design but can only provide a static snapshot of the full conformational space that represents the flexible nature of biological systems. Molecular dynamics (MD) simulations approximate such flexibility *in silico* with a trajectory of atom positions over a series of time steps (frames). These trajectories thereby reveal a more detailed—albeit still incomplete—picture of drug-target recognition and binding by providing access to protein-ligand interaction patterns over time (42–44). These insights can for example help in lead discovery to examine the stability and validity of a predicted ligand docking pose, and in lead optimization phases to estimate the effect of a chemical modification on binding affinity.

T019: MD simulations. We explain the key concepts behind MD simulations and provide the code to run a short MD simulation of EGFR in complex with a ligand on a local machine or on Google Colab with `condacolab`, which allows for GPU-accelerated simulations. The protein and ligand are thereby separately prepared with `pdbfixer` and `RDKit`, and subsequently combined using `MDTraj` (45) and `openff-toolkit`. The simulation is performed with `OpenMM` (16), a high-performance toolkit for molecular simulation. The talktorial produces a 100 ps trajectory if run on Google Colab. On a local machine, only 20 fs are generated by default to keep computational efforts reasonable. We refer to the work by Arantes *et al.* (46) for further reading on different MD protocols run with `OpenMM` using Jupyter notebooks on Google Colab.

T020: Analyzing MD simulations. We analyze and visualize the trajectory using the Python packages `MDAnalysis` (17,18) and `NGLView`. First, the protein is structurally aligned across all trajectory frames, followed by calculating the root-mean-square deviation (RMSD) for different system components, i.e. protein, backbone, and ligand. Then, we take a closer look at a selected interaction between ligand and protein atoms, showcasing the contribution of distance and angle to the hydrogen bond strengths.

Deep learning

Machine learning and more specifically deep learning have gained in popularity over the last few decades thanks to powerful computational resources (GPUs), novel algorithms, and the growing amount of available data (47). Applications to CADD are diverse, ranging from molecular property prediction (48) to *de novo* molecular design (49). Here, the focus is the featurization of molecular entities (T021) and ligand-based screening (T022).

T021: One-hot encoding. In CADD, machine learning algorithms require as input a numerical representation of small molecules. Besides molecular fingerprints (see T004),

a popular featurization is the SMILES notation (32). However, these representations are composed of strings and therefore cannot simply be input to an algorithm. One-hot encoding provides a solution for SMILES usage, explained in T021.

T022: Ligand-based screening: neural networks. We introduce the basics of neural networks and build a simple two-layer neural network. A model is trained on a subset of ChEMBL data to predict the pIC50 values of compounds against EGFR using MACCS keys as input. This talktorial is meant as groundwork for the understanding of neural networks. More complex architectures such as convolutional and recurrent neural networks will be explored in future notebooks. Such models may use the one-hot encoding of SMILES as input (50).

BEST PRACTICES

We provide reliable and reproducible TeachOpenCADD pipelines, periodically checked via automated testing mechanisms, and a streamlined and easy-to-understand code style across all talktorials.

Testing. Reproducibility is ensured by testing if the notebooks can run without errors and whether the output of specific operations can be reproduced. For this purpose, we use the tools [pytest](#) and [nbval](#).

Continuous integration. We are testing the talktorials regularly for Linux, OSX, and Windows and different Python versions on [GitHub Actions](#). This ensures identical behavior across different operating systems and Python versions and also spots issues like conflicting dependency updates or changing outputs.

Repository structure. The repository structure is based on the [cookiecutter-cms](#) template, which provides a Python-focused project scaffold with pre-configured settings for packaging, continuous integration, [Sphinx](#)-based documentation, and much more. We have adapted the template to our notebook-focused needs.

Code style. We aim to adhere to the [PEP8](#) style guide for Python code, which defines how to write idiomatic Python (Pythonic) code. Such rules are important so that new developers—or in our case talktorial users—can quickly read and understand the code. Furthermore, we use [black-nb](#) to format the Python notebooks compliant with PEP8.

TEACHOPENCADD USAGE

There are many ways to use the talktorials. If users simply want to go through the material, they can use the read-only website version. If users would rather like to execute and modify the Jupyter notebooks, this can be done online thanks to the [Binder](#) integrations or locally using the new conda package.

New website. Firing up Jupyter notebooks can entail unexpected complications if one wants to simply read through a talktorial. To make the access easy and fast, we launched a new [TeachOpenCADD website](#). The website statically renders the talktorials for immediate online reading using [sphinx-nb](#) and provides detailed documentation for local usage, contributions and external resources.

New Binder support. The [Binder project](#) offers a place to share computing environments via a single link. The environment setup of TeachOpenCADD can take a couple of minutes but does not require any kind of action on the user's end. This access option is recommended if the user plans on executing the material but does not need to save the changes.

New conda package. To make the local installation of TeachOpenCADD as easy as possible, we offer a conda package that ships all Jupyter notebooks with all necessary dependencies. The installation instructions are lined out in the [TeachOpenCADD documentation](#). This access option is recommended if the user plans on adapting the material for individual use cases.

CONCLUSION

The increasing amount of data and the focus on data-driven methods call for reproducible and reliable pipelines for computer-aided drug design (CADD). Knowing how to access and use these resources programmatically, however, requires domain-specific training and inspiration. The TeachOpenCADD platform showcases webserver-based data acquisition and common tasks in the fields of cheminformatics and structural bioinformatics. The theoretical and programmatic aspects of each topic are outlined side-by-side in Jupyter notebooks (talktorials) using open source resources only. To foster FAIR research, we apply software best practices such as testing, continuous integration, and idiomatic coding throughout the whole project. The talktorials are accessible via our website, Binder, and conda package to accommodate different use cases such as reading, executing, and modifying, respectively. We believe that TeachOpenCADD is not only a rich resource for CADD pipelines and teaching material on computational concepts and programming but as well a good example of how to set up websites, automated testing, and packaging for notebook-centric repositories. TeachOpenCADD is a living resource; problems can be voiced via GitHub issues and contributions can be made in the form of pull requests on GitHub. TeachOpenCADD is meant to grow; everyone is welcome to add new topics. Whenever you explore a new topic for your work, we invite you to fill our talktorial template with what one learns along the way and to submit it to TeachOpenCADD.

DATA AVAILABILITY

- TeachOpenCADD website: <https://projects.volkamerlab.org/teachopencadd/>.
- TeachOpenCADD GitHub repository: <https://github.com/volkamerlab/teachopencadd>.

ACKNOWLEDGEMENTS

The authors thank Pedro Gerletti (T019), Ahmed Atta (T022), Melanie Vogel (T018), Abishek Laxmanan Ravi Shankar (T014), Maria Trofimova (T015) and Jeffrey R. Wagner (T019) for the initial drafts or contributions to the above-mentioned talktorials.

The authors are grateful to the PyPDB and biotite maintainers for their work on updating their packages according to the new RCSB PDB API (special thanks to Patrick Kunzmann), and Albert Kooistra for helping with questions regarding the KLIFS database. The authors thank Hai Nguyen for dedicated and helpful NGLView support. The authors are thankful for a fruitful hackathon at the RDKit UGM in 2019, where we started to tackle a few TeachOpenCADD enhancements: structural superimposition and visualization without PyMol with Richard Gowers and testing Jupyter notebooks with Floriane Montanari. Finally, the authors thank Pat Walters and Hai Nguyen for endorsing the TeachOpenCADD platform on their websites, and we thank Greg Landrum for giving TeachOpenCADD a spot at the RDKit UGMs 2019, 2020, and 2021.

Author contributions: Conceptualization: D.S., J.R.G., A.V.; Data Curation, Formal Analysis, Investigation, Software, Validation and Visualization: D.S., J.R.G., T.B.K., Da.S., C.T., Y.C., M.L., S.M., M.W., A.A., A.V.; Funding Acquisition: A.V.; Methodology and Maintenance: D.S., J.R.G., T.B.K., Da.S., A.V.; Project Administration: D.S., J.R.G., A.V.; Resources: A.V.; Supervision: D.S., J.R.G., Da.S., T.B.K., A.V.; Writing-Original Draft: D.S., T.B.K., Da.S., J.R.G., A.V.; Writing - Review and Editing: D.S., J.R.G., T.B.K., Da.S., C.T., Y.C., M.L., S.M., M.W., A.A., A.V.

FUNDING

Note that the TeachOpenCADD project has been a group effort and has received no explicit funding, while the positions of individual authors were supported by diverse funding agencies; the Volkamer Lab received funding from the Bundesministerium für Bildung und Forschung [031A262C to A.V.]; Deutsche Forschungsgemeinschaft [VO 2353/1-1 to D.S.]; Stiftung Charité in the context of the Einstein BIH Visiting Fellow Project [to T.B.K., J.R.G. and C.T.]; Bayer in the context of the MIAME project (DaS); China Scholarship Council Project [201906210079 to Y.C.]. We acknowledge financial support from the Open Access Publication Fund of Charité - Universitätsmedizin Berlin and the German Research Foundation (DFG).

Conflict of interest statement. None declared.

REFERENCES

- Schneider,P., Walters,W.P., Plowright,A.T., Sieroka,N., Listgarten,J., Goodnow,R.A., Fisher,J., Jansen,J.M., Duca,J.S., Rush,T.S. *et al.* (2020) Rethinking drug design in the artificial intelligence era. *Nat. Rev. Drug Disc.*, **19**, 353–364.
- Ringer McDonald,A. (2021) In: *Teaching Programming across the Chemistry Curriculum. Teaching Programming across the Chemistry Curriculum: A Revolution or a Revival?* American Chemical Society pp. 1–11.
- Wilkinson,M.D., Dumontier,M., Aalbersberg,I.J., Appleton,G., Axton,M., Baak,A., Blomberg,N., Boiten,J.-W., da Silva Santos,L.B., Bourne,P.E. *et al.* (2016) The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, **3**, 160018.
- Sydow,D., Morger,A., Driller,M. and Volkamer,A. (2019) TeachOpenCADD: A Teaching Platform For Computer-Aided Drug Design Using Open Source Packages And Data. *J. Cheminform.*, **11**, 29.
- Kluyver,T., Ragan-Kelley,B., Pérez,F., Granger,B., Bussonnier,M., Frederic,J., Kelley,K., Hamrick,J., Grout,J., Corlay,S. *et al.* (2016) Jupyter Notebooks - A Publishing Format For Reproducible Computational Workflows. In: Loizides,F. and Schmidt,B. (eds). *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, Netherlands, pp. 87–90.
- Mendez,D., Gaulton,A., Bento,A.P., Chambers,J., De Veij,M., Félix,E., Magariños,M., Mosquera,J., Mutowo,P., Nowotka,M. *et al.* (2018) ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Res.*, **47**, D930–D940.
- Berman,H.M., Westbrook,J., Feng,Z., Gilliland,G., Bhat,T.N., Weissig,H., Shindyalov,I.N. and Bourne,P.E. (2000) The Protein Data Bank. *Nucleic Acids Res.*, **28**, 235–242.
- Burley,S.K., Bhikadiya,C., Bi,C., Bittrich,S., Chen,L., Crichlow,G.V., Christie,C.H., Dalenberg,K., Di Costanzo,L., Duarte,J.M. *et al.* (2020) RCSB Protein Data Bank: powerful new tools for exploring 3D structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. *Nucleic Acids Res.*, **49**, D437–D451.
- Riniker,S., Landrum,G., Montanari,F., Villalba,S., Maier,J., Jansen,J., Walters,W. and Shelat,A. (2018) Virtual-screening workflow tutorials and prospective results from the Teach-Discover-Treat competition 2014 against malaria [version 2; peer review: 3 approved]. *F1000Research*, **6**, 1136.
- Willighagen,E.L., Mayfield,J.W., Alvarsson,J., Berg,A., Carlsson,L., Jeliaskova,N., Kuhn,S., Pluskal,T., Rojas-Chertó,M., Spjuth,O. *et al.* (2017) The Chemistry Development Kit (CDK) v2.0: atom typing, depiction, molecular formulas, and substructure searching. *J. Cheminform.*, **9**, 33.
- Kanev,G.K., de Graaf,C., Westerman,B.A., de Esch,I. J.P. and Kooistra,A.J. (2020) KLIFS: an overhaul after the first 5 years of supporting kinase research. *Nucleic Acids Res.*, **49**, D562–D569.
- Kim,S., Chen,J., Cheng,T., Gindulyte,A., He,J., He,S., Li,Q., Shoemaker,B.A., Thiessen,P.A., Yu,B. *et al.* (2020) PubChem in 2021: new data content and improved web interfaces. *Nucleic Acids Res.*, **49**, D1388–D1395.
- Koes,D.R., Baumgartner,M.P. and Camacho,C.J. (2013) Lessons learned in empirical scoring with smina from the CSAR 2011 benchmarking exercise. *J. Chem. Inf. Model.*, **53**, 1893–1904.
- Salentin,S., Schreiber,S., Haupt,V.J., Adasme,M.F. and Schroeder,M. (2015) PLIP: fully automated protein–ligand interaction profiler. *Nucleic Acids Res.*, **43**, W443–W447.
- Nguyen,H., Case,D.A. and Rose,A.S. (2017) NGLView - Interactive Molecular Graphics For Jupyter Notebooks. *Bioinformatics*, **34**, 1241–1242.
- Eastman,P., Swails,J., Chodera,J.D., McGibbon,R.T., Zhao,Y., Beauchamp,K.A., Wang,L.-P., Simmonett,A.C., Harrigan,M.P., Stern,C.D. *et al.* (2017) OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS Computat. Biol.*, **13**, e1005659.
- Michaud-Agrawal,N., Denning,E.J., Woolf,T.B. and Beckstein,O. (2011) MDAnalysis: a toolkit for the analysis of molecular dynamics simulations. *J. Comput. Chem.*, **32**, 2319–2327.
- Gowers,R.J., Linke,M., Barnoud,J., Reddy,T.J.E., Melo,M.N., Seyler,S.L., Domański,J., Dotson,D.L., Buchoux,S., Kenney,I.M. *et al.* (2016) MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In: Sebastian,B. and Scott,R. (eds). *Proceedings of the 15th Python in Science Conference*. pp. 98–105.
- Herbst,R.S. (2004) Review of epidermal growth factor receptor biology. *Int. J. Radiat. Oncol.*, **59**(Suppl. 2), S21–S26.
- Harris,C.R., Millman,K.J., van der Walt,S.J., Gommers,R., Virtanen,P., Cournapeau,D., Wieser,E., Taylor,J., Berg,S., Smith,N.J. *et al.* (2020) Array programming with NumPy. *Nature*, **585**, 357–362.
- McKinney,W. (2010) Data structures for statistical computing in Python. In: van der Walt, S. and Millman,J. (eds). *Proceedings of the 9th Python in Science Conference*. pp. 56–61.
- Pedregosa,F., Varoquaux,G., Gramfort,A., Michel,V., Thirion,B., Grisel,O., Blondel,M., Prettenhofer,P., Weiss,R., Dubourg,V. *et al.*

- (2011) Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
23. Hunter, J.D. (2007) Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.*, **9**, 90–95.
 24. Waskom, M.L. (2021) seaborn: statistical data visualization. *J. Open Source Softw.*, **6**, 3021.
 25. Ireland, S.M. and Martin, A. C.R. (2021) GraphQL for the delivery of bioinformatics web APIs and application to ZincBind. *Bioinformatics Adv.*, **1**, vbab023.
 26. Davies, M., Nowotka, M., Papadatos, G., Dedman, N., Gaulton, A., Atkinson, F., Bellis, L. and Overington, J.P. (2015) ChEMBL Web Services: Streamlining Access To Drug Discovery Data And Utilities. *Nucleic Acids Res.*, **43**, W612–W620.
 27. Kuzmann, P. and Hamacher, K. (2018) Biotite: a unifying open source computational biology framework in Python. *BMC Bioinformatics*, **19**, 346.
 28. Gilpin, W. (2015) PyPDB: a Python API for the Protein Data Bank. *Bioinformatics*, **32**, 159–160.
 29. Consortium, T.U. (2020) UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.*, **49**, D480–D489.
 30. Sydow, D., Rodríguez-Guerra, J. and Volkamer, A. (2022) OpenCADD-KLIFS: A Python package to fetch kinase data from the KLIFS database. *J. Open Source Softw.*, **7**, 3951.
 31. Kim, S., Thiessen, P.A., Cheng, T., Yu, B. and Bolton, E.E. (2018) An update on PUG-REST: RESTful interface for programmatic access to PubChem. *Nucleic Acids Res.*, **46**, W563–W570.
 32. Weininger, D. (1988) SMILES, A Chemical Language And Information System. 1. Introduction To Methodology And Encoding Rules. *J. Chem. Inf. Model.*, **28**, 31–36.
 33. Fährrolfes, R., Bietz, S., Flachsenberg, F., Meyder, A., Nittinger, E., Otto, T., Volkamer, A. and Rarey, M. (2017) ProteinsPlus: a web portal for structure analysis of macromolecules. *Nucleic Acids Res.*, **45**, W337–W343.
 34. Volkamer, A., Kuhn, D., Grombacher, T., Rippmann, F. and Rarey, M. (2012) Combining global and local measures for structure-based druggability predictions. *J. Chem. Inf. Model.*, **52**, 360–372.
 35. van Linden, O. P.J., Kooistra, A.J., Leurs, R., de Esch, I. J.P. and de Graaf, C. (2014) KLIFS: a knowledge-based structural database to navigate kinase–ligand interaction space. *J. Med. Chem.*, **57**, 249–277.
 36. O’Boyle, N.M., Banck, M., James, C.A., Morley, C., Vandermeersch, T. and Hutchison, G.R. (2011) Open Babel: an open chemical toolbox. *J. Cheminformatics*, **3**, 33.
 37. Klebe, G. (2013) In: *Drug Design: Methodology, Concepts, and Mode-of-Action chapter Protein–Ligand Interactions as the Basis for Drug Action*. Springer Berlin Heidelberg, pp. 61–88.
 38. Rose, A.S. and Hildebrand, P.W. (2015) NGL Viewer: a web application for molecular visualization. *Nucleic Acids Res.*, **43**, W576–W579.
 39. Rose, A.S., Bradley, A.R., Valasatava, Y., Duarte, J.M., Prlić, A. and Rose, P.W. (2018) NGL viewer: web-based molecular graphics for large complexes. *Bioinformatics*, **34**, 3755–3758.
 40. Kawakita, Y., Seto, M., Ohashi, T., Tamura, T., Yusa, T., Miki, H., Iwata, H., Kamiguchi, H., Tanaka, T., Sogabe, S. *et al.* (2013) Design and synthesis of novel pyrimido[4,5-b]zazepine derivatives as HER2/EGFR dual inhibitors. *Bioorg. Med. Chem.*, **21**, 2250–2261.
 41. Yang, J., Tu, Z., Xu, X., Luo, J., Yan, X., Ran, C., Mao, X., Ding, K. and Qiao, C. (2017) Novel conjugates of endoperoxide and 4-anilinoquinazoline as potential anticancer agents. *Bioorgan. Med. Chem. Lett.*, **27**, 1341–1345.
 42. Mortier, J., Rakers, C., Bermudez, M., Murgueitio, M.S., Riniker, S. and Wolber, G. (2015) The impact of molecular dynamics on drug design: applications for the characterization of ligand–macromolecule complexes. *Drug Discov. Today*, **20**, 686–702.
 43. De Vivo, M., Masetti, M., Bottegoni, G. and Cavalli, A. (2016) Role of molecular dynamics and related methods in drug discovery. *J. Med. Chem.*, **59**, 4035–4061.
 44. Salmaso, V. and Moro, S. (2018) Bridging molecular docking to molecular dynamics in exploring ligand–protein recognition process: an overview. *Front. Pharm.*, **9**, 923.
 45. McGibbon, R.T., Beauchamp, K.A., Harrigan, M.P., Klein, C., Swails, J.M., Hernández, C.X., Schwantes, C.R., Wang, L.-P., Lane, T.J. and Pande, V.S. (2015) MDTraj: a modern open library for the analysis of molecular dynamics trajectories. *Biophys. J.*, **109**, 1528–1532.
 46. Arantes, P.R., Polêto, M.D., Pedebos, C. and Ligabue-Braun, R. (2021) Making it rain: cloud-based molecular simulations for everyone. *J. Chem. Inf. Model.*, **61**, 4852–4856.
 47. Goodfellow, I., Bengio, Y. and Courville, A. (2016) In: *Deep Learning*. MIT Press.
 48. Wu, Z., Ramsundar, B., Feinberg, E.N., Gomes, J., Geniesse, C., Pappu, A.S., Leswing, K. and Pande, V. (2018) MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.*, **9**, 513–530.
 49. Brown, N., Fiscato, M., Segler, M.H. and Vaucher, A.C. (2019) GuacaMol: benchmarking models for de novo molecular design. *J. Chem. Inf. Model.*, **59**, 1096–1108.
 50. Kimber, T.B., Engelke, S., Tetko, I.V., Bruno, E. and Godin, G. (2018) Synergy effect between convolutional neural networks and the multiplicity of SMILES for improvement of molecular prediction. arXiv doi: <https://arxiv.org/abs/1812.04439>, 11 December 2018, preprint: not peer reviewed.