



CHALMERS
UNIVERSITY OF TECHNOLOGY

DTA-GNN: a toolkit for constructing target-specific drug–target affinity datasets and training graph neural networks

Downloaded from: <https://research.chalmers.se>, 2026-06-07 01:46 UTC

Citation for the original published paper (version of record):

Özsari, G., Rifaioğlu, A., Acar, A. et al (2026). DTA-GNN: a toolkit for constructing target-specific drug–target affinity datasets and training graph neural networks. *SoftwareX*, 34. <http://dx.doi.org/10.1016/j.softx.2026.102671>

N.B. When citing this work, cite the original published paper.



DTA-GNN: a toolkit for constructing target-specific drug–target affinity datasets and training graph neural networks

Gökhan Özsari^{a, b, *}, Ahmet Süreyya Rifaioğlu^c, Aybar Can Acar^d, Tunca Doğan^{e, f}, M. Volkan Atalay^g

^a E-Commons, Department of Physics and Astronomy, Chalmers University of Technology, Gothenburg, Sweden

^b Computer Engineering Department, Middle East Technical University, Ankara, Türkiye

^c Institute for Computational Biomedicine, Heidelberg University and Heidelberg University Hospital, Heidelberg, Germany

^d Cancer Systems Biology Laboratory (KanSiL), Graduate School of Informatics, Middle East Technical University, Ankara, Türkiye

^e Biological Data Science Lab, Department of Computer Engineering, Department of Computer Engineering, Hacettepe University, Ankara, Türkiye

^f Department of Bioinformatics, Graduate School of Health Sciences, Hacettepe University, Ankara, Türkiye

^g Department of Information Systems and Supply Chain Management, Loyola University, Chicago, IL, USA

ARTICLE INFO

Keywords:

Drug–target binding affinity prediction

Graph neural networks

Cheminformatics

Reproducible research

Data leakage

ABSTRACT

Drug–target affinity (DTA) prediction is a key task in computational drug discovery, yet current research is often compromised by data leakage and non-reproducible preprocessing. We present DTA-GNN, an end-to-end Python toolkit that automates the rigorous construction of target-specific datasets and streamlines the training of Graph Neural Network (GNN) based DTA predictors. To address data validity, the toolkit's dataset construction pipeline handles ChEMBL data ingestion and unit standardization, and implements scaffold- and temporal-splitting strategies to prevent overestimation of performance. Integrated leakage audits quantify split integrity prior to modeling. Following dataset construction, DTA-GNN provides a modular trainer that supports ten state-of-the-art GNN architectures and includes built-in hyperparameter optimization. In addition, DTA-GNN supports latent space analysis either by extracting learned molecular embeddings or leveraging molecular fingerprints, and provides interactive visualizations to explore chemical space and interpret model behavior. By unifying robust dataset construction with accessible model training and latent-space analysis via Python library, CLI, and Web UI, DTA-GNN enables researchers to produce standardized, reproducible, and leakage-free DTA benchmarks.

Code metadata

Nr.	Code metadata description	Metadata
C1	Current code version	v0.2.0
C2	Permanent link to code/repository used for this code version	https://github.com/gozsari/DTA-GNN
C3	Permanent link to Reproducible Capsule	https://colab.research.google.com/drive/1REX98XQ_bZk1y17RCBT1zt2necXAvv7
C4	Legal Code License	MIT License
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	Python, PyTorch [2], PyTorch Geometric [17], RDKit [5], scikit-learn [34]
C7	Compilation requirements, operating environments & dependencies	Python 3.10+; pip; optional GPU support via CUDA or MPS (no compilation required)
C8	Link to developer documentation/manual	https://github.com/gozsari/DTA-GNN/tree/main/docs
C9	Support email for questions	gokhan.ozsari@chalmers.se

* Corresponding author at: E-Commons, Physics Origo Building, Sixth Floor, Room: 6149, Chalmersplatsen 1, 412 96 Gothenburg, Sweden.

Email address: gokhan.ozsari@chalmers.se (G. Özsari).

<https://doi.org/10.1016/j.softx.2026.102671>

Received 6 February 2026; Received in revised form 1 April 2026; Accepted 18 April 2026

Available online 23 April 2026

2352-7110/© 2026 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Motivation and significance

Drug–target affinity (DTA) prediction is a core task in computational drug discovery [3,4,36,37,40], supporting applications such as virtual screening, target prioritization, and lead optimization [44]. Advances in machine learning, particularly the increasing adoption of graph neural networks (GNNs) for molecular representation learning [12,20,42,45], along with recent model-level innovations such as multi-scale feature fusion, transformer-variational architectures, and language model fine-tuning for affinity prediction [19,22,23,39,46], have increased interest in developing predictive models that operate directly on molecular graph structures. In parallel, network-based approaches that encode drugs and targets as nodes in biological networks and apply graph encoders to learn node embeddings represent a complementary direction for DTA prediction [6,26,31]. However, the reliability of all such models depends critically on the quality and construction of the underlying datasets.

Large public bioactivity resources such as ChEMBL [18]—an open-access repository containing millions of bioactivity data points extracted from medicinal chemistry literature—enable large-scale DTA modeling. However, while ChEMBL provides extensively curated data, converting these raw records into a high-quality benchmark dataset is not a straightforward task and requires significant domain expertise. The heterogeneous nature of bioactivity records requires rigorous preprocessing steps—including unit harmonization, normalization (e.g., pChEMBL conversion – negative log₁₀ of molar dose–response measures (IC₅₀, XC₅₀, EC₅₀, AC₅₀) and binding affinity values (K_i, K_d) [1]), aggregation of replicated or conflicting measurements and validation of molecular structures—before the data can be used for model development.

Second, evaluation protocols often fail to account for prospective generalization. A particularly important and well-documented issue is data leakage arising from inappropriate dataset splitting strategies [28]. Random training–test splits can retain substantial overlap in molecular scaffolds between training and evaluation sets, leading to overly optimistic performance estimates and models that generalize poorly to genuinely novel chemical series—a key requirement in prospective drug discovery settings. Despite this being a known issue, leakage-aware splitting and systematic leakage auditing are not consistently treated as standard steps in DTA dataset construction and evaluation.

Third, reported model performance is often difficult to compare across studies because model development and evaluation are fragmented: architectural choices, training procedures, hyperparameter optimization, and evaluation protocols vary substantially and are frequently not reproducible. While many cheminformatics and machine learning libraries support individual components of DTA workflows (e.g., molecular featurization, model training utilities, or access to public bioactivity data) [35,47], they typically do not provide an end-to-end workflow in which dataset curation, leakage-aware splitting, leakage auditing, and artifact tracking are tightly coupled and enforced as part of a single evaluation protocol. As a result, assembling realistic and reproducible DTA benchmarks often requires additional attention to querying and curating ChEMBL records, generating consistent training–validation–test splits, and tracking dataset and model artifacts. For instance, DeepPurpose [25] offers a deep learning library supporting several encoder–decoder architectures for drug–target interaction prediction, but operates on pre-existing datasets and does not provide target-specific dataset construction from raw ChEMBL records or automated leakage auditing. Similarly, the Therapeutics Data Commons (TDC) [24] provides AI-ready datasets, standardized splitting strategies, and leaderboards across diverse therapeutic tasks, yet it serves as a benchmarking and data access platform rather than an end-to-end pipeline that couples dataset curation from source databases with model training and artifact tracking.

Here, we present DTA-GNN, a unified open-source toolkit designed to address these challenges by enabling the construction of target-specific drug–target affinity datasets and the training of predictive

models within a single framework. DTA-GNN comprises four modular components—data ingestion, preprocessing, modeling/training, and visualization—organized into a seven-phase end-to-end workflow. This workflow covers dataset construction (including leakage-aware splitting and leakage audits), model training (including hyperparameter optimization), evaluation of both baseline machine learning methods and graph neural networks, and post hoc analysis through latent-space inspection and interactive visualization of learned molecular embeddings. By providing these capabilities in a single open-source toolkit, DTA-GNN aims to enhance the transparency, reliability, and reusability of DTA modeling workflows in computational drug discovery research.

2. Software description

DTA-GNN is an open-source toolkit designed to support the end-to-end construction of target-specific drug–target affinity datasets and the training of predictive models within a unified and reproducible workflow. The toolkit integrates data ingestion, rigorous preprocessing, leakage-aware splitting, automated auditing, and model training into a single toolkit, reducing the need for ad hoc scripts and manual coordination between separate tools.

2.1. Software architecture

DTA-GNN was developed as a modular Python package comprising four components that form a pipeline with seven phases, as shown in Fig. 1.

- Component 1 - Data Ingestion:** This component isolates I/O operations from processing logic. It implements the `ChEMBLSource` interface with two concrete implementations: `ChEMBLWebSource` for REST API interaction and `ChEMBLSQLiteSource` for optimized SQL queries against local database dumps. This abstraction allows the backend to be swapped without modifying the downstream pipeline.
- Component 2 - Preprocessing:** The core processing logic resides here. The `Preprocessing` module handles vectorization of scalar values (e.g., normalization to pChEMBL), while the `Featurizer` module operates in parallel branches: one generating Morgan fingerprints (implemented in RDKit library) and another constructing PyTorch Geometric data objects (node features, edge indices) for graph-based learning.
To ensure scientific validity, the splitting logic is encapsulated in this module rather than embedded in the data loaders. The `Splitter` class implements “Cold-Drug” splitting (clustering by Murcko scaffolds to enforce structural distinctness) and temporal splitting (partitioning by publication year to simulate prospective validation). Crucially, a distinct `Audit` component serves as a quality-assurance layer, computing intersection metrics between data partitions to detect leakage prior to model instantiation.
- Component 3 - Modeling and Trainer:** The training component is built on PyTorch, PyTorch Geometric and scikit-learn. It uses a dynamic registry to instantiate models (e.g., GIN, GAT, SVR) from configuration strings. This component integrates an HPO wrapper that interfaces with the Weights & Biases API for Bayesian hyperparameter search.
- Component 4 - Visualizer:** This component is architected as a library of functions in the visualization module for latent space analysis and visualization. It exposes a unified interface for dimensionality reduction—accepting either ECFP4 features, a specific variant of Morgan fingerprints, or learned GNN embeddings—and delegates projection to scikit-learn’s t-SNE and PCA implementations.

These components are managed by the `Pipeline` class, which handles state transitions and artifact serialization, ensuring that every

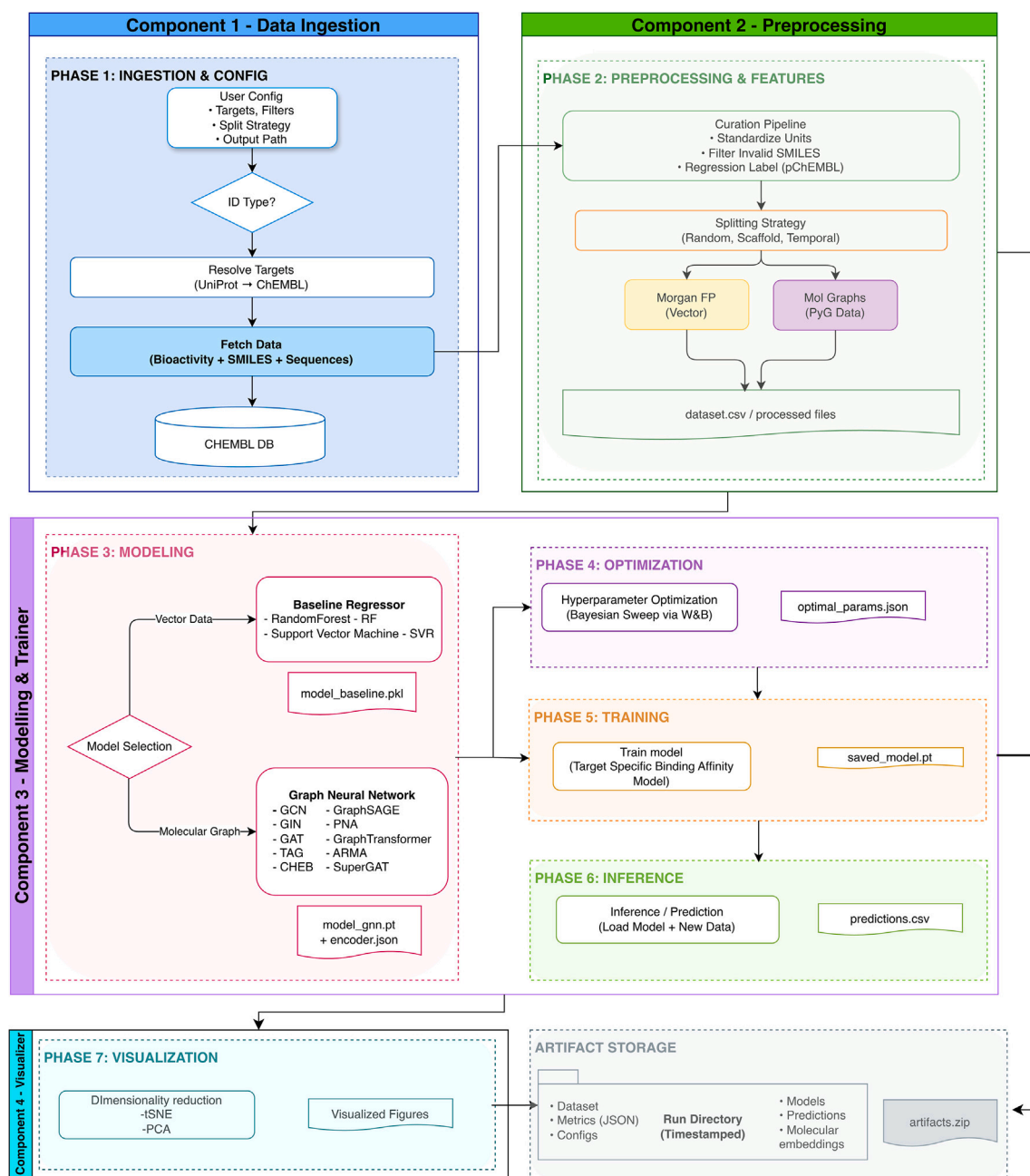


Fig. 1. Overview of the DTA-GNN software architecture. The pipeline integrates data ingestion, preprocessing, and feature generation, leakage-aware dataset splitting, model training using baseline models or graph neural networks, and inference within a unified workflow. All runs produce timestamped artifacts to support reproducibility and reuse.

execution produces a strictly versioned run directory containing data, logs, and model weights.

2.2. Software functionalities

DTA-GNN exposes its functionality through a Python library, a Command Line Interface (CLI), and an interactive Web UI. The key capabilities available to the user include:

- Flexible Data Ingestion:** Users can seamlessly switch between the ChEMBL Web API for lightweight queries and a local SQLite database for high-performance, offline dataset construction. (Phase 1 in Fig. 1)
- Standardized Preprocessing:** All fetched bioactivity measurements are converted to the pChEMBL scale ($-\log_{10}$ of molar dose-response values). Multiple measurements for the same compound-target pair are aggregated using the median by default, with mean, minimum, and maximum also available as configurable options. Censored measurements (e.g., $IC_{50} > 10 \mu\text{M}$) are retained by default but can be excluded. For temporal splits, the earliest publication year among duplicated records is preserved to prevent future-information leakage.
- Leakage-Aware Splitting:** The toolkit implements specific splitting strategies designed for drug discovery, including “Cold-Drug” scaffold splitting (clustering by Murcko scaffolds) and temporal

Table 1
Graph neural network architectures supported in DTA-GNN.

Architecture	Description	Key Characteristics
GIN	Graph Isomorphism Network [43]	A strong general-purpose model that can learn subtle structural differences between molecules
GCN	Graph Convolutional Network [30]	A simple baseline that averages information from directly connected atoms
GAT	Graph Attention Network [41]	Learns which neighbors are more important and gives them higher weight
GraphSAGE	Sample and Aggregate [21]	Scales to large graphs by sampling neighbors instead of using all of them
PNA	Principal Neighbourhood Aggregation [10]	Uses several ways of aggregating neighbors (mean/max/etc.) for richer representations
Transformer	Masked Label Prediction: Unified Message Passing [38]	A Transformer-style model adapted to graphs; uses attention-based message passing
TAG	Topology Adaptive Graph Convolution [14]	Scans the graph with a set of fixed-size learnable filters that adapt to the local topology
ARMA	Auto-Regressive Moving Average [7]	Uses recursive filtering to capture wider, global context and diverse patterns beyond local neighborhoods.
Cheb	Chebyshev Spectral Graph Convolution [13]	Captures information up to K steps away using an efficient approximation
SuperGAT	Supervised Graph Attention Network [29]	Makes attention models more robust to noisy edges via extra supervision/regularization

splitting (based on publication year) to simulate prospective validation. (Phase 2 in Fig. 1)

- **Automated Leakage Audits:** Built-in auditing functions quantify the overlap of scaffolds and targets between training and test sets, providing immediate feedback regarding dataset integrity. (Phase 2 in Fig. 1)
- **Comprehensive Model Zoo:** DTA-GNN supports baseline regressors (Random Forest, SVR) and ten distinct Graph Neural Network architectures (Table 1) with configurable hyperparameters (Phase 3 in Fig. 1).
- **Integrated Experiment Tracking:** The toolkit streamlines Hyperparameter Optimization (HPO) via a seamless integration with **Weights & Biases** [8]. By simply providing an API key, users can trigger Bayesian sweeps and automatically track training metrics, loss curves, and artifact versioning through a cloud-based dashboard without writing custom logging code (Phase 4 in Fig. 1).
- **Model Training:** Using the selected hyperparameters, the toolkit retrains the chosen model to produce a final optimized checkpoint for downstream prediction and embedding extraction. (Phase 5 in Fig. 1)
- **Inference with Trained Models:** Trained models are saved as reusable checkpoints and can be reloaded to perform inference on unseen compounds, enabling binding affinity prediction for the target of interest without retraining (Phase 6 in Fig. 1).
- **Latent Space Analysis & Visualization:** Unlike standard libraries that only output predictions, DTA-GNN allows users to extract learned molecular representations (embeddings) from the GNN encoder into NumPy format (.npz). The Web UI includes a dedicated visualization module for dimensionality reduction. Users can project high-dimensional molecular embeddings using **t-SNE** [32] or **PCA** [33] and color-code points by dataset split, ground-truth affinity, or model affinity predictions in order to visually audit the separation of active and inactive compounds (Phase 7 in Fig. 1).
- **No-Code Web Interface:** A Gradio-based UI allows users to visually configure pipelines, conduct hyperparameter optimization, monitor training progress, and visualize chemical space distributions without writing code, allowing subject matter experts without computational expertise to access its functionality. It is available on Hugging Face Spaces¹ and SciLifeLab Serve.²

2.3. Sample code snippet

This sample code snippet demonstrates the DTA-GNN workflow: building a drug-target affinity dataset for a specific target (ChEMBL1862-Tyrosine-protein kinase ABL1) using scaffold-based splitting, and training baseline models (Random Forest, SVR) and a Graph Neural Network (GIN) to predict binding affinities.

¹ <https://huggingface.co/spaces/gozsari/dta-gnn>

² <https://dta-gnn.serve.scilifelab.se/>

3. Illustrative examples

We present an end-to-end example that demonstrates the functionalities of the proposed toolkit on EGFR (UniProt: P00533), a clinically important human receptor tyrosine kinase implicated in multiple cancers. Using ChEMBL bioactivity data, we build target-specific datasets, audit scaffold leakage, train and tune a GraphSAGE regressor to predict binding affinity, and evaluate performance on held-out splits. The example was executed in a Google Colab notebook³ using the ChEMBL web API (or available dataset files) and produces all the tables/figures used in this section.

3.1. Dataset construction and leakage auditing

Two target-specific datasets were constructed from ChEMBL bioactivity records (standard types: IC50, Ki, Kd) using (i) a conventional random split and (ii) a scaffold-based “cold-drug” split (Table 2). We quantified scaffold overlap between the training and test sets using a leakage ratio (the fraction of test scaffolds also present in the training set). The random split exhibits substantial scaffold leakage [9], while the scaffold split achieves zero overlap for this target (Table 3).

3.2. Hyperparameter optimization and training GraphSAGE model

We then optimized a GraphSAGE regressor on the scaffold split using **Weights & Biases** (WandB) Bayesian sweeps over learning rate and number of message-passing layers (fixed seed: 42). Table 4 presents top 20 hyperparameter configurations from GraphSAGE HPO on the P00533 scaffold split, ranked by validation R^2 . The validation R^2 values across the top 20 trials range from 0.62 to 0.66, indicating relatively stable performance across a range of hyperparameter settings. The best configuration found was a learning rate of 1.40×10^{-3} with 5 GNN layers (validation score R^2 : 0.66; Table 4). This HPO is documented as a report and is available on WandB.⁴

Using the best performing hyperparameters on the validation dataset, we trained a final GraphSAGE model on the scaffold split and evaluated it on the held-out validation and test sets. Performance is reported as RMSE, MAE, Pearson correlation, R^2 , and Spearman’s rank correlation (Table 5), and the predicted-versus-true scatter plot illustrates the calibration of the test set (Fig. 2).

3.3. Model inference, and embeddings

To demonstrate the inference capability of the toolkit, we obtained the predictions of molecules in the test-set and listed the top-ranked ones according to predicted affinity (Table 6). Finally, we extracted the molecular embeddings learned from the trained encoder and visualized the chemical space of the test set using PCA, color-coded by predicted and ground-truth affinity (Fig. 3).

³ https://colab.research.google.com/drive/1REX98XQ_bZk1y17RCBT1Zt2necXAvv7

⁴ https://api.wandb.ai/links/lab_bio_inf/qv8jv1sr

```

1 from dta_gnn.pipeline import Pipeline
2 from dta_gnn.models import train_random_forest_on_run,
   train_svr_on_run, train_gnn_on_run, GnnTrainConfig
3
4 # 1. Build a dataset for your target of interest
5 pipeline = Pipeline(source_type="web")
6 dataset = pipeline.build_dta(
7     target_ids=["CHEMBL1862"], # target of interest
8     split_method="scaffold", # leakage-free splitting
9 )
10
11 print(f"Dataset: {len(dataset)} drug-target pairs")
12 # 2. Train a baseline model (Random Forest or SVR)
13 rf_result = train_random_forest_on_run("runs/current",
14     n_estimators=100)
15 print(f"RF Test RMSE: {rf_result.metrics['splits']['test']
16     ['rmse']:.3f}")
17
18 # Or train SVR
19 svr_result = train_svr_on_run("runs/current", C=10.0,
20     epsilon=0.1, kernel="rbf")
21 print(f"SVR Test RMSE: {svr_result.metrics['splits']['test']
22     ['rmse']:.3f}")
23
24 # 3. Train a Graph Neural Network
25 config = GnnTrainConfig(
26     architecture="gin", # GIN, GCN, GAT, GraphSAGE,
27     PNA, Transformer, TAG, ARMA, Chev, SuperGAT
28     hidden_dim=256,
29     num_layers=5,
30     epochs=100,
31 )
32 gnn_result = train_gnn_on_run("runs/current", config=
33     config)
34 print(f"GNN Test RMSE: {gnn_result.metrics['splits']['test']
35     ['rmse']:.3f}")

```

Listing 1. DTA-GNN usage example.

Table 2
Number of samples per dataset split for random and scaffold splitting.

Split	Random	Scaffold
Train	10,171	10,174
Validation	1453	1451
Test	2906	2885
Total	14,530	14,530

Table 3
Scaffold leakage ratio on P00533 (lower is better).

Split	Leakage ratio
Random	0.5106
Scaffold	0.0000

3.4. Baseline model training and evaluation

Additionally, we trained a baseline model to assess the performance of a classical approach and to quantify the improvement achieved by the proposed GNN models. To demonstrate this, we trained a Random Forest

regressor on the P00533 scaffold split and evaluated it on the validation and test sets (Table 7).

When we compare the results, we observe that the Random Forest provides a useful reference point for this target. The metric scores given in Table 5 and 7 are similar, probably due to the limited hyperparameter search conducted for GraphSAGE. If additional improvement

Table 4
Top 20 hyperparameter configurations from GraphSAGE HPO on the P00533 scaffold split, ranked by validation R^2 .

Rank	Trial	Learning Rate (lr)	Number of GNN Layers	Validation R^2
1	34	0.001403	5	0.6574
2	12	0.000845	6	0.6556
3	41	0.001691	2	0.6551
4	39	0.001824	3	0.6507
5	33	0.001770	3	0.6485
6	32	0.002035	5	0.6482
7	8	0.002572	4	0.6457
8	10	0.002887	4	0.6454
9	28	0.003908	3	0.6445
10	7	0.002735	4	0.6444
11	35	0.002810	5	0.6440
12	18	0.002784	6	0.6420
13	9	0.002108	4	0.6412
14	11	0.002869	4	0.6378
15	48	0.000439	5	0.6371
16	42	0.000735	3	0.6357
17	0	0.007006	3	0.6332
18	20	0.005996	3	0.6313
19	30	0.000870	2	0.6254
20	4	0.009403	5	0.6186

Table 5
GraphSAGE performance on the P00533 scaffold split.

Split	RMSE	MAE	R^2	Pearson	Spearman R
val	0.88	0.67	0.66	0.81	0.82
test	0.89	0.66	0.60	0.77	0.78

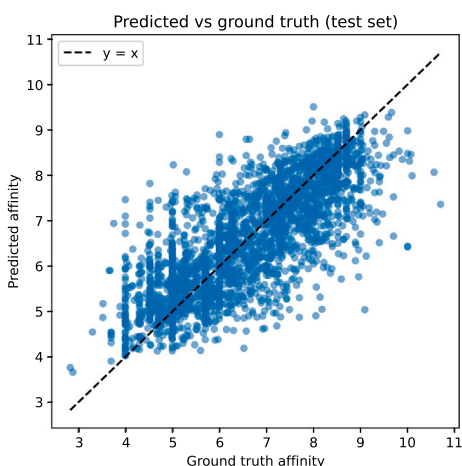


Fig. 2. Predicted vs. ground-truth affinity on the P00533 test split (GraphSAGE).

is required, further GraphSAGE-specific optimization via an expanded hyperparameter search can be pursued to boost the model.

3.5. Additional experiments

To illustrate the applicability of DTA-GNN across diverse protein targets, we provide a complete worked example, benchmarking five

GNN architectures (GCN, GIN, GAT, GraphSAGE, PNA) and two classical baselines (Random Forest, SVR) on three targets from distinct protein families: Androgen Receptor (P10275), Acetylcholinesterase (P22303), and Prostaglandin G/H synthase 2 (P35354). These examples demonstrate how users can employ the toolkit’s end-to-end workflow—from dataset construction, through W&B Bayesian hyperparameter optimization, to model evaluation and embedding extraction—for their own targets of interest. The results highlight practical considerations: GNN architectures ranked highest on all three targets, but the margin over fingerprint-based baselines varied substantially depending on the target’s ligand diversity, illustrating why the toolkit’s systematic comparison capability is valuable. The Supplementary Material also reports wall-clock training times, per-epoch durations, embedding extraction times, and peak GPU memory consumption for each architecture, providing users with practical guidance for hardware planning and architecture selection. All datasets, training scripts, optimized hyperparameters, predictions, runtime and memory profiles, and visualization artifacts are provided in the Supplementary Material as a reference for users adapting the pipeline to new targets.

4. Impact

- Standardizing Evaluation Protocols:** In computational drug discovery, inconsistent dataset construction and random splitting frequently lead to the “Clever Hans” effect, where models exploit data biases. DTA-GNN mitigates this by standardizing the evaluation protocol. By automating “Cold-Drug” scaffold splitting and temporal splitting, the toolkit provides realistic estimates of how models will perform on novel chemical series.
- Systematic Architectural Benchmarking:** The current literature is ambiguous about which GNN architectures are best suited to specific tasks [15,16,27]. DTA-GNN allows researchers to perform controlled ablation studies. Users can systematically benchmark ten distinct architectures (e.g., PNA vs. GCN) on identical, target-specific datasets. This enables targeted analysis of which graph inductive biases are most effective for specific target classes, such as kinases or GPCRs.
- Supporting early-stage DMTA workflows:** DTA-GNN contributes to early-stage Design–Make–Test–Analyze (DMTA) workflows [11] by facilitating access to GNN-based deep learning model development for subject matter experts without computational specializations. By replacing complex, ad hoc scripts with a standardized “no-code” Web UI, the toolkit empowers medicinal chemists to independently construct datasets and train baseline models, significantly shortening the iteration time between data acquisition and decision-making. Crucially, the toolkit enables “Cold-Drug” scaffold splitting to simulate prospective validation, ensuring that generated models provide realistic performance estimates for novel chemical series—a key requirement for the *Design* phase. Furthermore, DTA-GNN strengthens the *Analyze* phase through its interactive visualization module, which allows users to audit learned molecular embeddings and distinguish active scaffolds from decoys in the latent space. This integration enables project teams to prioritize compounds more reliably before committing to experimental synthesis.

Table 6
Top five test-set molecules ranked by predicted affinity (P00533, GraphSAGE).

Rank	SMILES	Predicted Affinity	True Affinity
1	<chem>CCS(=O)(=O)Nc1nc(-c2ccc(F)cc2)c(-c2ccnc(Nc3cc(C)c(N4CCCC(N5CCN(C)CC5)CC4)cc3OC)n2)s1</chem>	9.5069	8.0000
2	<chem>Cn1ncc([N+](=O)[O-])c1C[N+](C)(C)C/C=C/C(=O)Nc1ccc2nnc(Nc3cccc(Br)c3)c2c1.[Br-]</chem>	9.3810	9.6600
3	<chem>O=C/C=C/C(=O)NCCCN1CCOCC1)Cc1cc2c(Nc3cccc(Br)c3)nenc2cn1</chem>	9.3275	9.0900
4	<chem>O=Cc1cccc(Nc2nnc3cc(OC)c(OCCCCN/C(=N\C#N)Nc4ccnc4)cc23)c1</chem>	9.2798	9.5900
5	<chem>Cc1nc(F)ccc1[C@H](Nc1cc(C#N)c2ncc(C#N)c(NCC(C)(C)C)c2c1)C1=CN(C2(C(N)=O)CC2)NN1</chem>	9.2753	9.0000

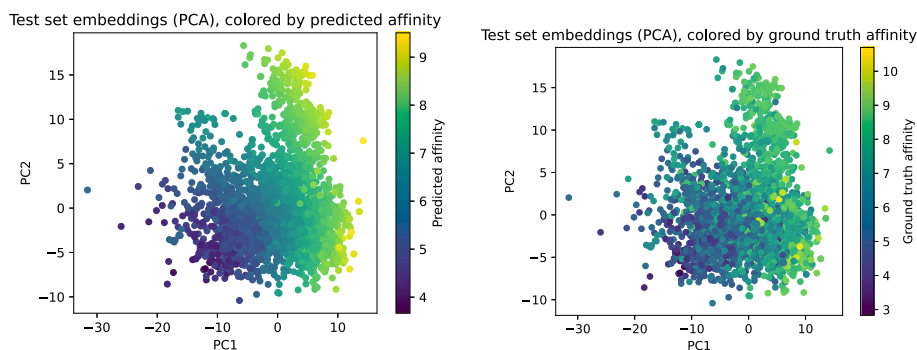


Fig. 3. PCA visualization of test-set embeddings for P00533, color-coded by (left) predicted affinity and (right) ground-truth affinity.

Table 7

Random Forest's performance on the P00533 scaffold split.

Split	RMSE	MAE	R^2	Pearson	Spearman R
val	0.89	0.66	0.65	0.81	0.81
test	0.86	0.62	0.61	0.78	0.79

5. Conclusions

DTA-GNN is an open-source toolkit that addresses key practical challenges in drug–target affinity modeling by providing an end-to-end, reproducible pipeline for dataset construction and model development. The toolkit integrates standardized dataset construction, leakage-aware dataset splitting, automated audits, and support for both baseline machine learning models and graph neural networks.

By emphasizing transparent dataset construction and reproducible evaluation, DTA-GNN supports more reliable comparison of DTA modeling approaches and facilitates reuse across projects. As part of model analysis and validation, DTA-GNN enables latent space analysis and interactive visualization of learned molecular embeddings, helping users interpret model behavior and inspect latent chemical space organization. The availability of a Python application programming interface, a command-line interface, and a graphical user interface enables use by both software-oriented researchers as well as users without programming expertise, supporting adoption across interdisciplinary research teams.

DTA-GNN is designed to complement existing cheminformatics and machine learning libraries by providing infrastructure that bridges raw bioactivity data and model training workflows. The toolkit is intended to serve as a foundation for future methodological development and applied research in drug-target affinity prediction.

Video

Demonstration of the DTA-GNN workflow (dataset construction, leakage audits, model training, and latent-space visualization) and the Web UI. The video is available at the link.⁵

CRedit authorship contribution statement

Gökhan Özsari: Writing – review & editing, Writing – original draft, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Ahmet Süreyya Rifaioğlu:** Writing – review & editing, Investigation, Conceptualization. **Aybar Can Acar:** Writing – review & editing, Investigation, Conceptualization. **Tunca Doğan:** Writing – review & editing, Investigation, Conceptualization.

⁵ https://drive.google.com/file/d/1iz-0t9oaEOrBBzSgP1Lf2yVMCELpT75_/view

M. Volkan Atalay: Writing – review & editing, Supervision, Project administration, Investigation, Conceptualization.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used Grammarly and ChatGPT to improve language and readability. After using these tools, the authors reviewed and edited the content as necessary and assume full responsibility for the publication's content.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We acknowledge that the computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725 [project NAISS2025-22-1609]. We also thank SciLifeLab Serve and Hugging Face for providing hosting of the DTA-GNN app.

Appendix A. Supplementary data

Supplementary data to this article can be found online at doi:10.1016/j.softx.2026.102671.

References

- [1] Allaway RJ, La Rosa S, Guinney J, Gosline SJC. Probing the chemical–biological relationship space with the drug target explorer. *J Cheminformatics* 2018;10(1):41.
- [2] Ansel J, Yang E, He H, Gimelshein N, Jain A, Voznesensky M, Bao B, Bell P, Berard D, Burovski E, Chauhan G, Chourdia A, Constable W, Desmaison A, DeVito Z, Ellison E, Feng W, Gong J, Gschwind M, Hirsh B, Huang S, Kalambarkar K, Kirsch L, Lazos M, Lezcano M, Liang Y, Liang J, Lu Y, Luk CK, Maher B, Pan Y, Puhrsch C, Reso M, Saroufim M, Siraichi MY, Suk H, Suo M, Tillet P, Wang E, Wang X, Wen W, Zhang S, Zhao X, Zhou K, Zou R, Mathews A, Chanan G, Wu P, Chintala S. PyTorch 2: faster machine learning through dynamic Python bytecode transformation and graph compilation. In: 29th ACM International Conference on architectural support for programming languages and operating systems, volume 2 (ASPLOS '24). ACM; Apr 2024.
- [3] Atas Guvenilir H, Doğan T. How to approach machine learning-based prediction of drug/compound–target interactions. *J Cheminformatics* 2023;15(1):16.
- [4] Atas Guvenilir H, Doğan T. Eclipse: exploration of complex ligand–protein interactions through learning from systems-level heterogeneous biomedical knowledge graphs. *Biorxiv* 2025:11–2025.
- [5] Bento AP, Hersey A, Félix E, Landrum G, Gaulton A, Atkinson F, Bellis LJ, De Veij M, Leach AR. An open source chemical structure curation pipeline using rdkit. *J Cheminformatics* 2020;12(1):51.
- [6] Bi X, Zhang S, Ma W, Jiang H, Wei Z. Hisif-Dta: a hierarchical semantic information fusion framework for drug-target affinity prediction. *IEEE J Biomed Health Inform* 2023;29(3):1579–90.
- [7] Bianchi FM, Grattarola D, Livi L, Alippi C. Graph neural networks with convolutional ARMA filters. *IEEE Trans Pattern Anal Mach Intell* 2021;44(7):3496–507.

- [8] Biewald L. Experiment tracking with weights and biases. Software available from wandb.com; 2020.
- [9] Blevins AD, Quigley IK. Clever Hans in Chemistry: chemist style signals confound activity prediction on public benchmarks; 2025.
- [10] Corso G, Cavalleri L, Beaini D, Liò P, Veličković P. Principal neighbourhood aggregation for graph nets. *Adv Neural Inf Process Syst* 2020;33:13260–71.
- [11] Cox PB, Gupta R. Contemporary computational applications and tools in drug discovery. *ACS Medicinal Chemistry Letters* 2022;13(7):1016–29.
- [12] Dalkiran A, Rifaioğlu AS, Cetin-Atalay R, Acar AC, Doğan T, Atalay MV. Molecular contrastive learning with graph attention network (mocl-gat) for enhanced molecular representation. *Biorxiv* 2025:09–2025.
- [13] Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. *Adv Neural Inf Process Syst* 2016;29.
- [14] Du J, Zhang S, Wu G, Moura JMF, Kar S. Topology adaptive graph convolutional networks; 2018.
- [15] Dwivedi VP, Joshi CK, Luu AT, Laurent T, Bengio Y, Bresson X. Benchmarking graph neural networks. *J Mach Learn Res* 2023;24(43):1–48.
- [16] Errica F, Podda M, Bacciu D, Micheli A. A fair comparison of graph neural networks for graph classification. [arXiv preprint]. 2019 arXiv:1912.09893.
- [17] Fey M, Lenses JE. Fast graph representation learning with pytorch geometric. [arXiv preprint]. 2019 arXiv:1903.02428.
- [18] Gaulton A, Hersey A, Nowotka M, Bento AP, Chambers J, Mendez D, Mutowo P, Atkinson F, Bellis LJ, Cibrián-Uhalte E, et al. The ChEMBL database in 2017. *Nucleic acids research* 2017;45(D1):945–54.
- [19] Gorantla R, Gema AP, Yang IX, Serrano-Morrás Á, Suutari B, Juárez-Jiménez J, Mey AS. Learning binding affinities via fine-tuning of protein and ligand language models. *J Chem Inf Model* 2025;65(22):12279–91.
- [20] Guo Z, Guo K, Nan B, Tian Y, Iyer RG, Ma Y, Wiest O, Zhang X, Wang W, Zhang C, et al. Graph-based molecular representation learning. [arXiv preprint]. 2022 arXiv:2207.04869.
- [21] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. *Adv Neural Inf Process Syst* 2017;30.
- [22] He H, Chen G, Chen CY-C. Integrating sequence and graph information for enhanced drug-target affinity prediction. *Sci China Inf Sci* 2024;67(2):129101.
- [23] He H, Chen G, Tang Z, Chen CY-C. Dual modality feature fused neural network integrating binding site information for drug target affinity prediction. *NPJ Digit Med* 2025;8(1):67.
- [24] Huang K, Fu T, Gao W, Zhao Y, Roohani Y, Leskovec J, Coley CW, Xiao C, Sun J, Zitnik M. Artificial Intelligence Foundation for therapeutic science. *Nat Chem Biol* 2022;18(10):1033–6.
- [25] Huang K, Fu T, Glass LM, Zitnik M, Xiao C, Sun J. Deepurpose: a deep learning library for drug–target interaction prediction. *Bioinformatics* 2020;36(22–23):5545–7.
- [26] Huang X, Bi X, Xing N, Ma W, Jiang H, Cai Q, Lu W, Yang F, Wei Z, Zhang S. Lightdta: lightweight drug-target affinity prediction via random-walk network embedding and knowledge distillation. *Mol Divers* 2026:1–24.
- [27] Jiang D, Wu Z, Hsieh C-Y, Chen G, Liao B, Wang Z, Shen C, Cao D, Wu J, Hou T. Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *J Cheminformatics* 2021;13(1):12.
- [28] Joeres R, Blumenthal DB, Kalinina OV. Data splitting to avoid information leakage with datasail. *Nat Commun* 2025;16(1):3337.
- [29] Kim D, Oh A. How to find your friendly neighborhood: Graph attention design with self-supervision. [arXiv preprint]. 2022 arXiv:2204.04879.
- [30] Kipf TN. Semi-supervised classification with graph convolutional networks. [arXiv preprint]. 2016 arXiv:1609.02907.
- [31] Ma W, Zhang S, Li Z, Jiang M, Wang S, Guo N, Li Y, Bi X, Jiang H, Wei Z. Predicting drug-target affinity by learning protein knowledge from biological networks. *IEEE J Biomed Health Inform* 2023;27(4):2128–37.
- [32] Maaten van der L, Hinton G. Visualizing data using t-sne. *J Mach Learn Res* 2008;9(Nov):2579–605.
- [33] Minka T. Automatic choice of dimensionality for PCA. *Adv Neural Inf Process Syst* 2000;13.
- [34] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. Scikit-learn: machine learning in Python. *the Journal of machine Learning research* 2011;12:2825–30.
- [35] Ramsundar B, Eastman P, Walters P, Pande V, Leswing K, Wu Z. Deep learning for the life sciences. O'Reilly Media; 2019. <https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837>.
- [36] Rifaioğlu AS, Cetin Atalay R, Cansen Kahraman D, Doğan T, Martin M, Atalay V. Mdeepred: novel multi-channel protein featurization for deep learning-based binding affinity prediction in drug discovery. *Bioinformatics* 2021;37(5):693–704.
- [37] Rifaioğlu AS, Nalbat E, Atalay V, Martin MJ, Cetin-Atalay R, Doğan T. Deepscreen: high performance drug–target interaction prediction with convolutional neural networks using 2-d structural compound representations. *Chem Sci* 2020;11(9):2531–57.
- [38] Shi Y, Huang Z, Feng S, Zhong H, Wang W, Sun Y. Masked label prediction: unified message passing model for semi-supervised classification; 2021.
- [39] Tang X, Ma W, Yang M, Li W. Mff-dta: multi-scale feature fusion for drug-target affinity prediction. *Methods* 2024;231:1–7.
- [40] Ünü A, Ulusoy E, Yiğit MG, Darcan M, Doğan T. Protein language models for predicting drug–target interactions: novel approaches, emerging methods, and future directions. *Curr Opin Struct Biol* 2025;91:103017.
- [41] Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. [arXiv preprint]. 2017 arXiv:1710.10903.
- [42] Wu X, Wang H, Gong Y, Fan D, Ding P, Li Q, Qian Q. Graph neural networks for molecular and materials representation. *J. Mater. Inf.* 2023;3(2):12.
- [43] Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? [arXiv preprint]. 2018 arXiv:1810.00826.
- [44] Zeng X, Li S-J, Lv S-Q, Wen M-L, Li Y. A comprehensive review of the recent advances on predicting drug-target affinity based on deep learning. *Front Pharmacol* 2024;15:1375522.
- [45] Zhang Y, Hu Y, Han N, Yang A, Liu X, Cai H. A survey of drug-target interaction and affinity prediction methods via graph neural networks. *Comput Biol Med* 2023;163:107136.
- [46] Zhou C, Li Z, Song J, Xiang W. Transvae-Dta: transformer and variational autoencoder network for drug-target binding affinity prediction. *Comput Methods Programs Biomed* 2024;244:108003.
- [47] Zhu Z, Shi C, Zhang Z, Liu S, Xu M, Yuan X, Zhang Y, Chen J, Cai H, Lu J, Ma C, Liu R, Xhonneux L-P, Qu M, Tang J. Torchdrug: A powerful and flexible machine learning platform for drug discovery. [arXiv preprint]. 2022 arXiv:2202.08320.