nature protocols

https://doi.org/10.1038/s41596-024-01084-x

Protocol

Check for updates

An explainable deep learning platform for molecular discovery

Felix Wong[1,2,3], Satotaka Omori[1,3], Alicia Li[3], Aarti Krishnan[1,2], Ryan S. Lach[3], Joseph Rufo[4,5,6,7], Maxwell Z. Wilson[3,4,5,6,7] & James J. Collins[1,2,8]

Abstract

Deep learning approaches have been increasingly applied to the discovery of novel chemical compounds. These predictive approaches can accurately model compounds and increase true discovery rates, but they are typically black box in nature and do not generate specific chemical insights. Explainable deep learning aims to 'open up' the black box by providing generalizable and human-understandable reasoning for model predictions. These explanations can augment molecular discovery by identifying structural classes of compounds with desired activity in lieu of lone compounds. Additionally, these explanations can guide hypothesis generation and make searching large chemical spaces more efficient. Here we present an explainable deep learning platform that enables vast chemical spaces to be mined and the chemical substructures underlying predicted activity to be identified. The platform relies on Chemprop, a software package implementing graph neural networks as a deep learning model architecture. In contrast to similar approaches, graph neural networks have been shown to be state of the art for molecular property prediction. Focusing on discovering structural classes of antibiotics, this protocol provides guidelines for experimental data generation, model implementation and model explainability and evaluation. This protocol does not require coding proficiency or specialized hardware, and it can be executed in as little as 1–2 weeks, starting from data generation and ending in the testing of model predictions. The platform can be broadly applied to discover structural classes of other small molecules, including anticancer, antiviral and senolytic drugs, as well as to discover structural classes of inorganic molecules with desired physical and chemical properties.

Key points

- This protocol enables the computational discovery of chemical compounds using a deep learning architecture called graph neural networks, which, given the chemical structure of any compound, can predict whether the compound has a property of interest.
- The platform leverages explainable deep learning to facilitate the identification of structural classes of novel compounds. This approach guides hypothesis generation and makes searching large chemical spaces more efficient compared with previous approaches, which are typically black box in nature and do not generate specific chemical insights.

Key references

Wong, F. et al. *Nature* **626**, 177–185 (2024): https://doi.org/10.1038/s41586-023-06887-8

Wong, F. et al. *Nat. Aging* **3**, 734–750 (2023): https://doi.org/10.1038/s43587-023-00415-z

Liu, G. et al. *Nat. Chem. Biol.* **19**, 1342–1350 (2023): https://doi.org/10.1038/s41589-023-01349-8

Stokes, J. M. et al. *Cell* **180**, 688–702.e13 (2020): https://doi.org/10.1016/j.cell.2020.01.021

[1]Infectious Disease and Microbiome Program, Broad Institute of MIT and Harvard, Cambridge, MA, USA. [2]Institute for Medical Engineering and Science and Department of Biological Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA. [3]Integrated Biosciences, Inc., Redwood City, CA, USA. [4]Center for BioEngineering, University of California Santa Barbara, Santa Barbara, CA, USA. [5]Biomolecular Science and Engineering Program, University of California Santa Barbara, Santa Barbara, CA, USA. [6]Department of Molecular, Cellular, and Developmental Biology, University of California Santa Barbara, Santa Barbara, CA, USA. [7]Neuroscience Research Institute, University of California Santa Barbara, Santa Barbara, CA, USA. [8]Wyss Institute for Biologically Inspired Engineering, Harvard University, Boston, MA, USA. e-mail: jimjc@mit.edu

Introduction

Artificial intelligence (AI) is a subfield of computational science that aims to create systems to perform tasks, and machine learning (ML) is a subfield of AI using computers that model input–output relationships to accomplish this goal[1,2] (see Box 1 for a brief introduction to ML). In recent years, there has been a sharp increase in the scientific applications of AI and ML due to increased computational resources, the development of models that process different forms of data and the increased accessibility of codebases and databases. AI and ML have enabled notable scientific advances, including the development of AlphaGo[3], the development of AlphaFold and other protein sequence-to-structure models[4–6], and the discovery of a structural class of antibiotics[7].

Many scientific advances have been driven by the capability of AI and ML models to perform extensive computations that exploit often subtle patterns in data, enabling combinatorically large spaces to be productively searched[8,9]. Such large search spaces are domain specific but can include, for example, possible Go moves, possible geometric conformations of a protein or possible chemical compound structures. Searching chemical space for bioactive compounds is a particularly important application: novel bioactive compounds are constantly needed for drug discovery, and it has been estimated that there are ~10^{60} possible, largely organic, drug-like compounds[10]. Given this large chemical space—made larger by including additional inorganic compounds—applying AI and ML models to efficiently search it has been a longstanding goal. Early work in this area focused on applying simple ML models, including support vector machines and decision trees, to identify novel bioactive compounds in virtual screening libraries[11–14]. More recent work has applied deep learning (DL) methods, including neural networks and variational autoencoders, to discover bioactive compounds more efficiently in virtual libraries and to generate compounds de novo[1,2,7,15–33].

DL approaches have accurately modeled compounds, with increased true discovery rates, and resulted in the discovery of drug candidates including antibiotics[7,17–22], senolytics[23] and anticancer and antiviral combinations[24,25]. However, a major limitation to DL approaches is that they are typically black box in nature or unable to provide reasoning behind model predictions. Explainable DL is an emerging field that aims to open up the black box by providing this reasoning[7,34–37]. Explainable DL contrasts with interpretable DL, which aims to reveal patterns of decision-making steps the models perform to arrive at their predictions[38] (see Box 2 for further comparison). As applied to molecular discovery, explainable DL identifies patterns of chemical atoms and bonds—chemical substructures—that have positive predictive value for a

BOX 1

A brief introduction to ML and DL

While the terms are often used interchangeably, ML is a general subfield of AI that includes DL, which uses neural network models to perform ML tasks. The foundations of ML and DL emerged in the 1950s, when computer scientists aimed to perform human-like reasoning with computers using rule-based approaches[79] and mimic the neural circuits of the human brain[80]. Early work in ML introduced basic algorithms that were used to extract patterns from data and make predictions on unseen data. However, ML models have become substantially more powerful in recent years due to innovations in DL. Starting from the development of backpropagation[81], which allowed complex neural networks to be trained, progress in DL has been driven by the widespread adoption of graphics processing units for training and the proliferation of large-scale datasets[8]. These advances were underscored in 2012 by the development of large convolutional neural networks for accurate image recognition[82], and DL models are now state-of-the-art for tasks including natural language processing and speech recognition[8].

While ML as a field has increased in popularity in recent years, additional DL models, particularly generative models such as transformers[83], are currently being developed and refined. Further research has also focused on leveraging ML and DL models for scientific and mathematical reasoning applications[84] and improving the modeling framework by increasing explainability and interpretability (Box 2). As demonstrated by this protocol, we anticipate that next-generation DL models—including explainable and interpretable models—will continue to provide valuable insights into scientific problems such as molecular discovery.

BOX 2

Explainability and interpretability in ML

DL approaches are generally black box in nature or unable to provide reasoning as to how model predictions were made. Explainable DL and interpretable DL address this issue in different ways. Explainable DL aims to understand and articulate why a model makes specific predictions, while interpretable DL aims to reveal the specific patterns of decision-making steps the models perform to arrive at their predictions[38]. Explainability and interpretability both underlie shifts in research toward increased comprehension, trustworthiness, and accountability of AI and ML models.

Explainability provides insights into the input features that are important for model predictions, helping to make model predictions more transparent and understandable. Two ways in which models can be made explainable are by using feature importance analyses and individual instance explanations. For instance, Shapley Additive Explanations (SHAP) values numerically quantify the impact of any feature on predicted activity[85]. In molecular discovery, SHAP analyses have been used to identify specific chemical substructures that influence a compound's predicted activity[36,37]; these approaches are similar to the MCTS approach described in this protocol, and the extent to which they would produce similar or meaningfully different results requires further investigation. Additionally, Local Interpretable Model-agnostic Explanations is an explanation technique and a special form of SHAP that provides local explanations for individual predictions by approximating complex models with simpler ones[86].

Interpretability aims for deconstructive understanding of how models operate and the relationships between input features and outputs. Two ways in which models can be made interpretable are by approximating their predictions with the predictions of logical or sparse models[38] (for example, using decision trees) and leveraging visualizations of the input to guide predictions (for example, using *k*-nearest neighbors).

property of interest. The ability to move beyond predictions for lone compounds to predictions for classes of compounds defined by shared substructures enables substantial generalization. By enabling models to predict structural classes directly, explainable DL can produce specific chemical insights, efficiently narrow chemical spaces and lead researchers toward effective chemical scaffolds.

In this protocol, we present an explainable DL platform for molecular discovery (Fig. 1). Although we focus on applying the platform to antibiotic discovery, the platform is general-purpose and can be used to discover molecules with other biological or chemical properties of interest. The platform uses graph neural networks (GNNs) as a DL model architecture. GNNs model chemical structures as collections of nodes (chemical atoms) and edges (chemical bonds) and perform computations that pool together information from neighboring atoms and bonds[16,39–42]. We provide a practical introduction to Chemprop[16,43–45], a software package implementing GNNs, and guide users through the process of experimental data generation, model implementation, model explainability and evaluation and experimental evaluation. This protocol includes both computational and experimental components, and it does not require coding proficiency or specialized hardware. Starting from data generation and ending in the testing of model predictions, this protocol can be executed in as little as 1–2 weeks, depending on dataset size and the time required for experiments.

Development of the protocol

AI and ML have been increasingly applied to molecular discovery. Pioneering studies in the 2010s used simple ML models (such as support vector machines) to identify functional chemical compounds. Since then, advances in DL and computing have enabled the application of highly predictive neural network-based model architectures to drug discovery[16,45]. GNNs are a type of neural network in which convolution steps are performed on the basis of the two-dimensional structure of graphs. They differ from other neural network models in that they directly operate on the graph structure, as opposed to other computed elements (such as physicochemical parameters). Various types of GNNs have been proposed, but they can be unified under the framework of message passing GNNs (MPNNs)[39]. Message passing refers to the convolutions GNNs perform on numerical values (messages) associated with each node (chemical atom) or edge (chemical bond). MPNNs were first developed and applied to infer quantum properties of compounds in 2017, and a technical description of GNNs and MPNNs is provided in Box 3.

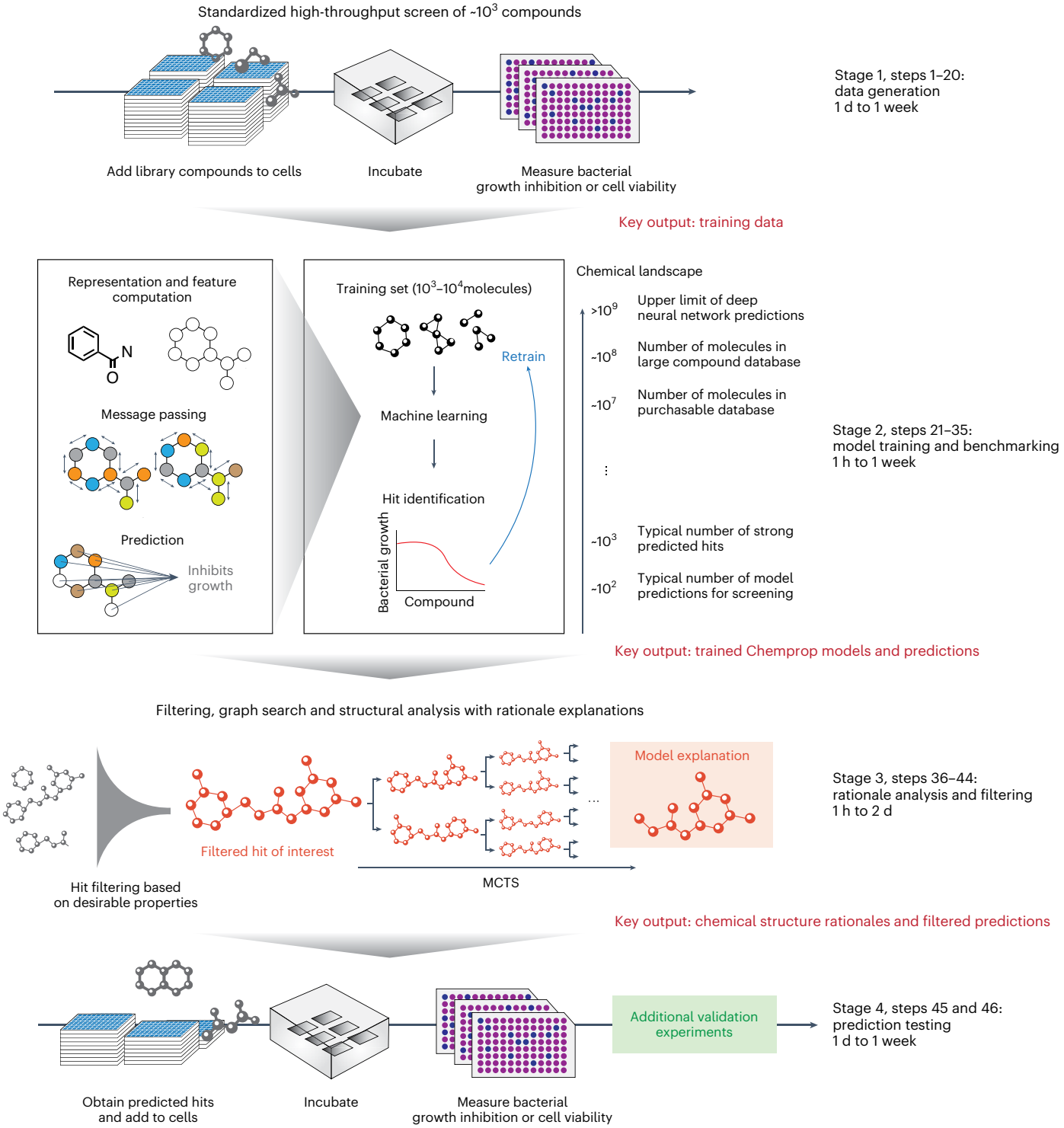


Fig. 1 | Overview of the protocol. The protocol stages involving experimentation and computation are illustrated. This protocol includes the generation of training data from experimental high-throughput screens of ~2,500 compounds for antibacterial activity and cytotoxicity. These data are used to train GNNs to predict the antibacterial activity and cytotoxicity of up to >10^9 molecules in silico, contrasting with expensive and time-consuming experimental screening of large chemical libraries. Shortlisted compounds ('hits') are filtered for desirable chemical properties and analyzed using a MCTS-based approach that identifies substructure rationales as explanations of model predictions. These rationales lead to predictions of structural classes of hits, and all predicted hits are experimentally tested to confirm activity. This approach can be iterated and the model can be retrained to generate new predictions of structural classes and associated hits. The figure is adapted from ref. 7, Springer Nature Ltd.

Chemprop is a software package implementing a variant of MPNNs, in which messages are associated with directed edges in contrast to nodes[16,45]. Chemprop has been developed, benchmarked and applied in several publications[16–19,23,24,43–45]. Key features of Chemprop are that it is user friendly, possesses diverse functionalities and customization options and, since its development in 2019, has been constantly maintained. Its application to antibiotic discovery, resulting in the identification of halicin as an antibacterial compound[17], was reported by our laboratory in 2020. Since then, Chemprop has also been used to discover small-molecule senolytics[23] and antiviral drug combinations[24], and these applications exemplify Chemprop-based approaches and provide additional references on best practices.

We recently showed that Chemprop models can be made explainable and that model explainability enables the discovery of structural classes of antibiotics[7,44]. Explainability is now built into Chemprop, as detailed in the 'Procedure' section below, and relies on an algorithm called Monte Carlo tree search (MCTS; Box 4). MCTS is a nondeterministic search algorithm that can efficiently explore large search spaces[44,46] and is notable for having been used by AlphaGo to identify Go moves[3]. As applied in Chemprop, MCTS takes any compound with

BOX 3

A technical primer on GNNs

Artificial neural networks (ANNs) are computational frameworks that make predictions on input data by passing the input through layers of connected nodes or 'neurons'. These nodes can communicate with each other using mathematical functions that can be specified or learned from the training data. ANNs make high-level predictions (such as antibiotic activity of a molecule) by performing successive convolutions at each node that extract features from input data as the data are passed through the network. By iteratively adjusting the relative contributions—the weights—of each convolutional step to the final output in a way that depends on the training data, ANNs have been shown to be able to make accurate predictions for a variety of inputs, including text and images.

GNNs are a type of ANN designed to operate on input data represented as graphs. Graphs are mathematical objects that consist of nodes and edges. For molecular discovery, these nodes and edges represent chemical atoms and bonds, respectively. In this way, graphs provide mathematical encodings of two-dimensional chemical structures.

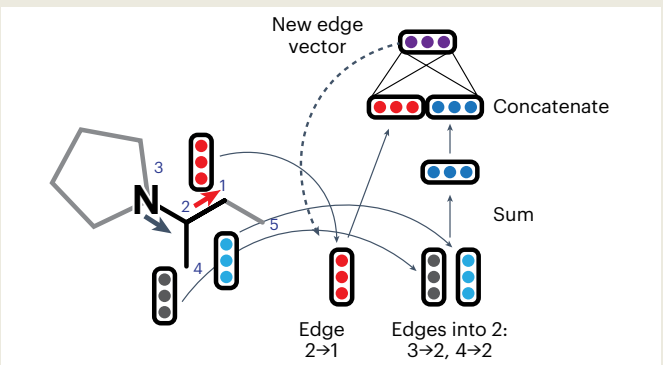
Unlike feed-forward ANNs, which pass data linearly from input to output layers, GNNs perform convolutions simultaneously based on all the nodes and edges of the input. To incorporate local connectivity information, message passing—a step where embeddings from a node or edge's neighbors are pooled with its preexisting embeddings—is often incorporated into a GNN layer's update function, resulting in an architecture called MPNNs. Given an undirected graph G with node features x_v and edge features e_{vw}, at step t of the message passing, hidden states h_v^t and messages m_v^t associated with each node v are updated using a message function M_t and vertex update function U_t as follows:

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}), h_v^{t+1} = U_t\left(h_v^t, m_v^{t+1}\right).$$

Here, $N(v)$ is the set of neighbors of v in G and h_v^0 is a function of the initial atom features, x_v. The number of message passing steps (n) per layer determines how far information from a node or edge propagates: a given graph component only receives messages from components n steps away. At the end of message passing, the final set of hidden states, $\{h_v^T\}$, is passed into a feedforward network or other readout function to make a prediction for the entire graph.

As implemented by Chemprop, directed MPNNs are a variant of MPNNs in which messages are associated with the directed edges of a graph, in contrast to its nodes. As shown in the image below representing message passing (adapted with permission from ref. 16), messages from the grey directed edge 3 → 2 (and any other edge into node 2, such as 4 → 2) are used to update the hidden state associated with the red directed edge 2 → 1, and this process is repeated across all edges. Directed MPNNs were first introduced as structure2vec in ref. 87 and were further developed in ref. 16.

In general, GNNs have been shown to be state-of-the-art for many tasks and additional applications—including their use in transfer learning[88], dynamical modeling[89] and integration with transformers and other generative models[90]—have recently been studied to improve their applicability to other problems and domains.



BOX 4

The MCTS

For each molecule with a suitably high prediction score ('hit'), we aim to determine the smallest substructure resulting in the molecule being classified as active (the molecule's 'rationale'). As detailed in ref. 7, a rationale should satisfy three properties: its maximum size must be no more than a set number of atoms, it must be a connected structure and its predicted activity value, when the rationale is provided as an input to Chemprop in its own right, must be greater than a threshold value.

The MCTS algorithm solves the discrete optimization problem of finding the rationale of a molecule. MCTS is a search process in which the root of the search tree is the full molecule, and each state in the search tree is a substructure (subgraph) resulting from a sequence of bond or ring deletions. We ensure that the substructure is chemically valid by requiring it to always be connected throughout the process.

Formally, during the search process, each state (S) in the search tree contains the following statistics[7]:

- $N(S)$ is the number of times state S has been visited during the search process and is a quantity used for the exploration–exploitation tradeoff in the MCTS algorithm
- $W(S)$ is the total long-term reward, which indicates how likely state S will eventually lead to a valid rationale
- $R(S)$ is the predicted activity score of S, viewed as an input to Chemprop in its own right, which indicates the immediate reward from choosing this state

Guided by these statistics, the MCTS algorithm searches for rationales through an iterative process. Each iteration consists of two phases:

1. Forward pass: the MCTS algorithm selects a path from the root (the starting compound) to a leaf state, S_{leaf} (a candidate rationale). At each intermediate state S, a deletion action is selected on the basis of the mean action value:

$$S' = \text{argmax}_{s \in \text{child}(S)} \frac{W(s) + c_s R(s)}{1 + N(s)},$$

where the parameter c_s controls the tradeoff between the long-term reward, $W(s)$, and immediate reward, $R(s)$. This parameter is set according to the well-known predictor upper confidence bound applied to trees equation[48].

2. Backward pass: the state statistics are updated for each visited state in the selected path: $N(S) \leftarrow N(S) + 1$; $W(S) \leftarrow W(S) + R(S_{leaf})$.

Based on the backward pass update, $W(S)$ represents the sum of the predicted activity of all valid rationales (leaf nodes) derived from state S. Different from the immediate reward $R(S)$, $W(S)$ measures long-term reward because it focuses on the predicted activity of the leaf nodes. The intuition is that the immediate reward is useful for filtering poor choices: states are unlikely to contain a rationale if $R(S)$ is low. Among states with similar $R(S)$ values, $W(S)$ aids in selecting those with higher long-term reward.

To illustrate the MCTS algorithm, we provide an example in Fig. 5. Here, the starting molecule is a molecule that was predicted to be active (antibacterial activity score >0.2) using ensembles of 20 Chemprop models trained on 39,312 compounds, as detailed in ref. 7. The starting molecule and its output rationale are shown in Fig. 4h.

In Chemprop, MCTS is implemented as part of the built-in 'interpret' function, which outputs rationales for input hit molecules. These rationales contain scaffolds that describe chemical motifs underlying putative structural classes of active compounds. However, the computed rationales can differ from other, structurally similar rationales by a small (typically <3) number of atoms due to the stochastic nature of the MCTS. Thus, calculating the chemical scaffolds shared between similar rationales, for instance, by using RDKit's FindMCS function, provides 'rationale groups' that most generally describe the models' predictions of structural classes.

high predicted activity score as an input and produces a chemical substructure (or 'rationale') contributing substantively to the compound's prediction score as an output. In contrast to other substructures, such as the maximal common substructures shared between compounds with high predicted activity, these rationales are guaranteed by construction to result in at least a baseline prediction score when inputted into their models. In this way, the models have 'explained' their predictions.

Overview of the protocol

This protocol includes both computational and experimental components, combining experimental data generation with Chemprop to guide users through the process of discovering compounds with antibacterial activity (Fig. 1). The protocol broadly consists of four stages:

Stage 1, data generation (Steps 1–20)

Data quality is paramount to ML models, as nonstandardized datasets make it difficult to consistently infer a property of interest and incorrectly labeled inputs hamper the ability of ML models to generalize. The protocol, therefore, starts by outlining the experimental procedures needed to perform a standardized screen of a library of compounds for antibacterial activity. To focus on compounds with selective antibacterial activity, we also outline experimental

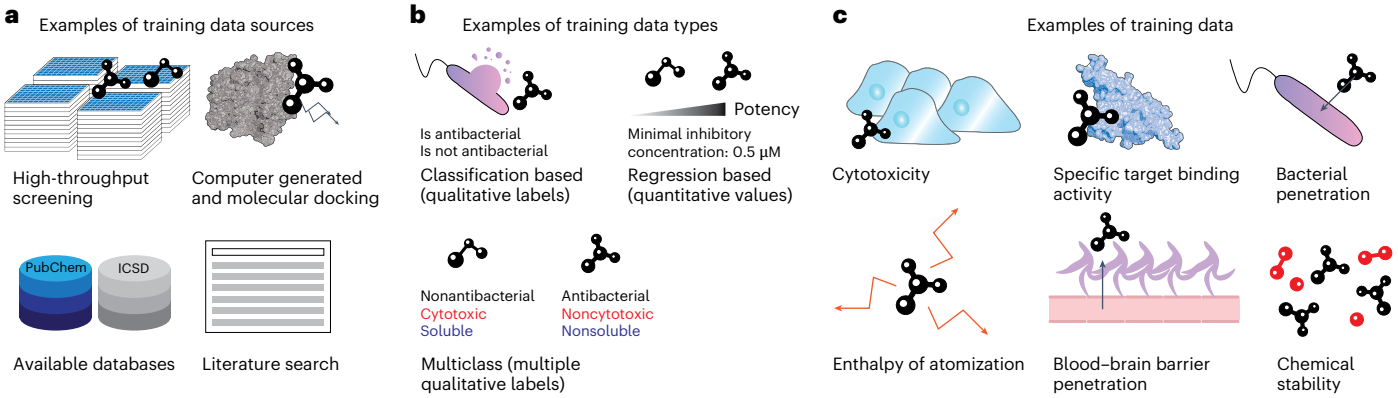


Fig. 2 | Examples of training data. a, Examples of training data sources. This protocol outlines steps needed to perform a high-throughput screen of ~2,500 compounds for antibacterial activity against *S. aureus* and cytotoxicity against three human cell types. However, training data can be compiled from a diverse variety of sources and do not necessarily need to be experimentally acquired. **b**, Examples of training data types. This protocol leverages classification-based Chemprop models trained on data corresponding to binarized antibacterial activity and cytotoxicity values. Regression-based models and multiclass models can similarly be developed using Chemprop. **c**, Examples of training data. Chemprop accommodates any property related to the chemical structure of a molecule.

procedures for generating data in a counter-screen of human cell cytotoxicity. The result of this stage is a table of compound structures and their associated activity values. The compound structures can be represented as text strings in a simplified molecular input line-entry system (SMILES) format, and the associated activity value can be represented as discrete values (for example, binary '0' or '1' values for binary classification models) or continuous values (for example, any positive number for regression models) depending on the type of prediction performed in Stage 2. In general, different sources and types of data can also be used (Fig. 2).

Stage 2, model training and benchmarking (Steps 21–35)

This protocol trains Chemprop models using the datasets generated in Stage 1. Several parameters of the models must be specified, including the type of model (for example, classification or regression), the number of models to train, model hyperparameters and additional optional inputs (for example, how to split the data for validation, which additional features and loss function to use for training and which metrics to use for benchmarking). Depending on dataset size and compute power, training times may vary ('Timing' section). For the datasets described in this protocol and a typical computer, model training can typically be completed within 1–2 d. Trained models should then be benchmarked, a process which entails applying the models to withheld subsets of the training data, comparing the model predictions to the known ground-truth values and quantitatively evaluating the results using a suitable metric such as the area under the precision–recall curve (AUPRC) or the area under the operating characteristic curve (AUROC).

Stage 3, rationale analysis and filtering (Steps 36–44)

After the user is satisfied with model performance, Chemprop models can be trained using the full training dataset as described in Stage 2 and applied to other chemical spaces. Compounds from these chemical spaces that are predicted to be active ('hits') can be shortlisted for further validation. To make these predictions explainable, we describe Chemprop's 'interpret' function, which performs MCTS to identify rationales from hits. Due to the large number of possible MCTS steps needed to arrive at each rationale, it is possible that the MCTS may not converge for certain hits. However, when rationales are successfully generated, studying frequently occurring rationales can provide chemical substructure insights into what Chemprop models view as indicative of activity. Additionally, these rationales provide an efficient means of down-sampling compounds to validate in Stage 4. At any stage of this process, compounds of interest may be computationally filtered to remove compounds with undesirable properties, for instance, unfavorable physicochemical parameters or problematic chemical substructures.

Stage 4, prediction testing (Steps 45 and 46)

Hits should be validated using the same experimental protocol as in Stage 1. However, if rationales were generated or filtering steps were applied in Stage 3, then these hits can be down sampled. This allows for fewer—but typically more structurally interesting—compounds to be curated and tested, enabling more efficient exploration of large chemical spaces. It is also important to point out that, in many cases, hits with validated activity in Stage 4 require additional study to support their promise for further development. For antibiotics and other therapeutic areas, these additional studies might include mechanistic characterization and further in vitro and in vivo testing.

To guide readers, we illustrate the process of modeling and the expected inputs and outputs at each stage in Fig. 3, as well as example outputs of the protocol in Fig. 4.

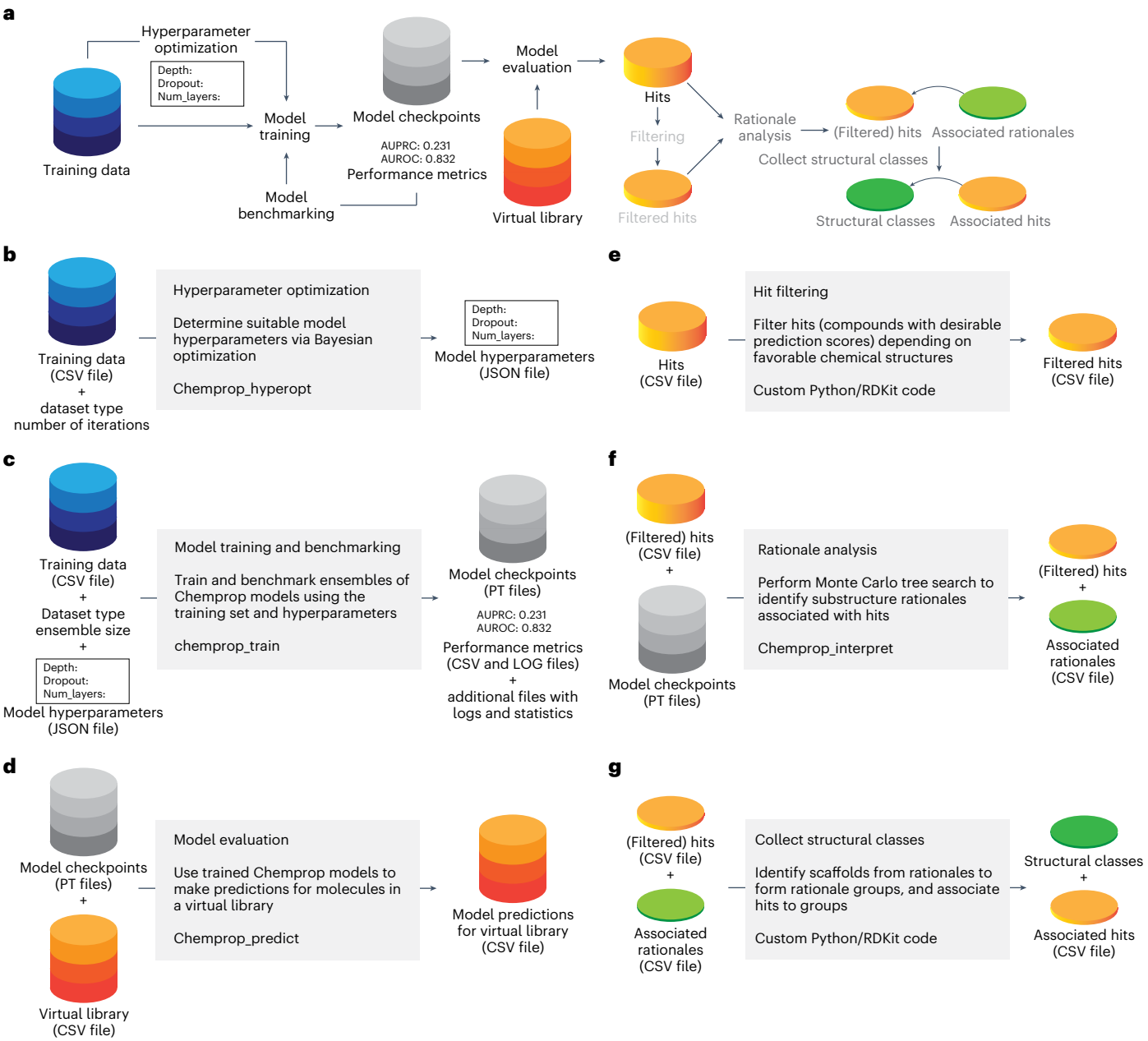


Fig. 3 | The process of modeling and expected inputs and outputs at each modeling step. a, Overview of model building using Chemprop. **b**–**g**, Steps involving hyperparameter optimization (**b**), model training and benchmarking (**c**), model evaluation (**d**), hit filtering (**e**), rationale analysis (**f**) and identification of structural classes (**g**), as described in Stages 2 and 3, are shown. PT files are model checkpoint files that end in '.pt'.

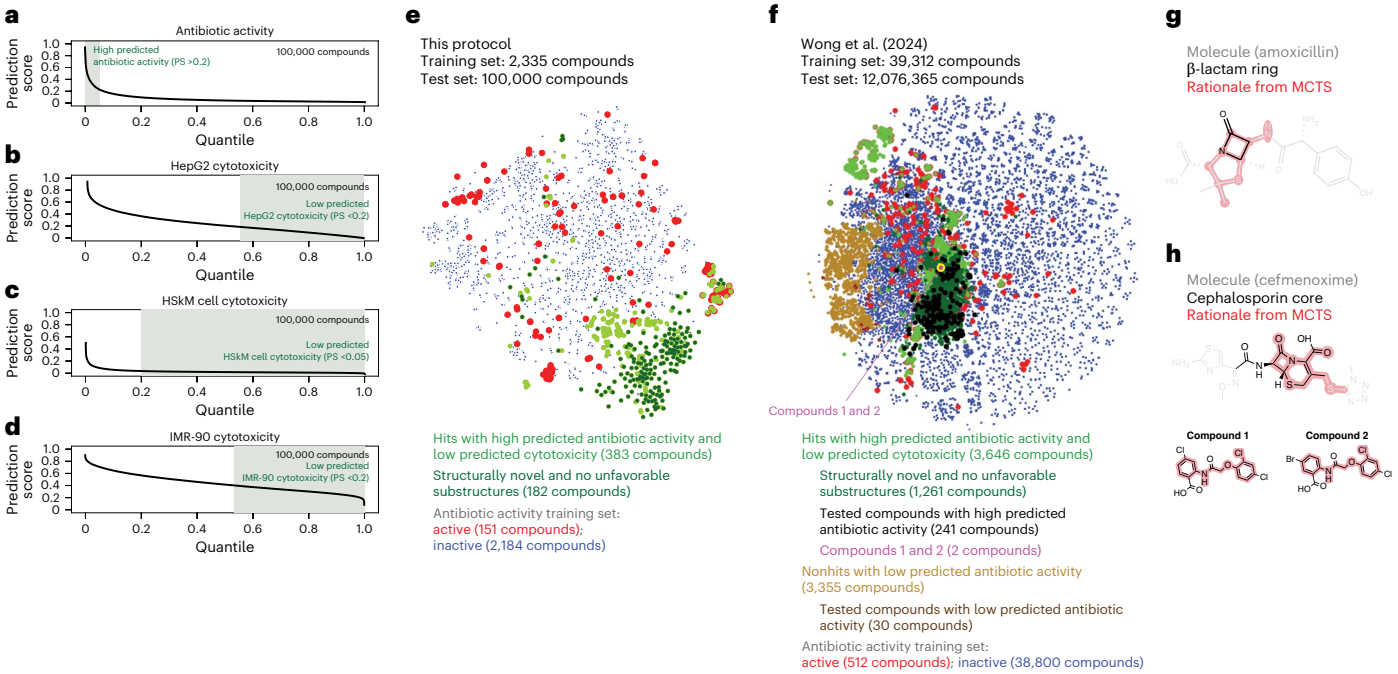


Fig. 4 | Example results. a–**d**, Rank-ordered antibiotic activity (**a**), HepG2 cytotoxicity (**b**), HSkM cell cytotoxicity (**c**) and IMR-90 cytotoxicity (**d**) prediction scores of a test set of 100,000 compounds, as determined following this protocol. PS, prediction score. **e**, A *t*-SNE plot of compounds with high antibiotic prediction scores, in addition to compounds in the training set, as determined following this protocol. **f**, A *t*-SNE plot of compounds with high and low antibiotic prediction scores, in addition to compounds in the training set and empirically tested compounds, for the larger training and test sets used in ref. 7. **g**, A rationale (red) determined using a MCTS for amoxicillin, an example hit compound, using the Chemprop models created in this protocol. The rationale overlaps with the β-lactam ring, a substructure known to confer antibacterial activity. **h**, Rationales (red) determined using a MCTS for cefmenoxime (top) and two compounds, compounds **1** and **2** (bottom), using the Chemprop models created in ref. 7. For cefmenoxime, the rationale overlaps with the cephalosporin core, a substructure known to confer antibacterial activity. Panels **f** and **h** were reproduced from ref. 7, Springer Nature Ltd.

Users may adapt this protocol in various ways, including applying the protocol to the discovery of molecules with other properties, forgoing the experimental generation of training data in favor of assembling training data based on published literature or deploying trained Chemprop models without making them explainable or without any additional filtering steps. In the 'Procedure' section, we provide practical guidelines that are helpful to consider when adapting this protocol.

Applications of the method

Making DL models, including Chemprop models, explainable enables researchers to focus on salient chemical scaffolds and directly predict structural classes of compounds with activity. Automating the identification of structural motifs can be valuable, especially when applied to large databases, such as the ZINC22 (ref. 47). This protocol provides a source of chemical novelty that can suggest specific chemical spaces to explore and augment discovery pipelines. As mentioned above, model explainability is also pragmatic in that it allows large chemical spaces to be efficiently down sampled. This capability is helpful because many available compound libraries are redundant: compounds often sample the same or similar scaffolds, and the chemical diversity in these libraries is limited[48,49]. By focusing only on key scaffolds of interest and testing small numbers of unique compounds representing these scaffolds, researchers can sample this chemical diversity while keeping costs low.

Notable applications of Chemprop include the discovery of halicin[17] and the discovery of small-molecule senolytics[23] and antiviral drug combinations[24]. Chemprop has also been used to predict absorption, distribution, metabolism, excretion and toxicity properties[50], chemical spectra[51] and chemical reaction properties[43]. As used in this protocol, Chemprop and its explainability features have enabled the discovery of a structural class of antibiotics[7]. This protocol

builds on ref. 7, but uses more minimal data files and examples to illustrate specific concepts. We anticipate that this protocol can be readily adapted for use in different applications, such as the discovery of bioactive compounds—including other anti-infective, anticancer and antiaging compounds—and the discovery of inorganic compounds with desired material properties[42,45].

For biological applications, prior studies[7,17–19,23] have focused on using phenotypic screening data to train Chemprop and similar models. In contrast to modeling based on target-specific activity, modeling based on phenotypic activity allows for chemical structure information across more active chemotypes to be integrated[52–54]. Nevertheless, developing models that are trained to predict target-specific activity is possible, and further work is needed to determine how predictive such approaches can be given the potentially more limited space of active chemotypes.

Comparison with other methods

Several points of comparison can be made with respect to other methods:

Protocol aims

As Chemprop and other chemotype-based approaches learn and generalize based on the patterns of molecular atoms and bonds in chemical structures, these methods cannot be replaced by traditional quantitative structure–activity relationship analyses. Quantitative structure–activity relationship analyses build on known scaffolds and do not identify novel chemotypes[55]. In general, Chemprop and other chemotype-based approaches are also target independent and different from target-specific approaches, such as molecular docking[56,57].

This protocol focuses on the problem of inference. As such, this protocol does not generate new molecules (but could be leveraged to score molecules generated by other models). This protocol also focuses on explainability, which contrasts with black box or interpretable models. Black box models have been thought to be more predictive than explainable models[58], but in our approach, Chemprop can be used with or without explainability features. Explainability is different from interpretability, which aims to identify the model's decision-making steps (Box 2).

Protocol performance and other ML architectures

As mentioned above, GNNs have been benchmarked and shown to be state of the art for numerous prediction tasks, and additional benchmarking details and statistics can be found in these references[16,45]. Additional ML architectures that have been used to predict activity from chemical structure include support vector machines, random forest models and vector-based convolutional neural networks[11–14]. Although these models typically underperform relative to Chemprop, most of the stages of this protocol (including training data generation, model explainability and hit filtering) can be generalized to different ML models.

'Full-stack' versus computational pipelines

Different from fully computational protocols, this protocol guides users through the 'full-stack' process of experimental data generation, model implementation, model explainability and evaluation and experimental validation. The experimental sections of this protocol are important because they determine the quality of the training data. Model predictions should always be experimentally tested because false positive rates can be expected to be high (typically, >50% in the case of antibiotic discovery[7,17–19]). High false positive rates reflect the difficulty of molecular discovery, and this protocol should only be expected to increase working true discovery rates (and not perfectly predict activity). The experimental parts of this protocol include standard high-throughput screening of a sizeable (>1,000) number of compounds for antibiotic activity, and we do not expect minor methodological variations of this screen—for instance, those conforming to Clinical and Laboratory Standards Institute guidelines[59]—to produce substantively different results. As this protocol includes both computation and experimentation, we anticipate that it will be useful to both dry-laboratory and wet-laboratory researchers.

Required expertise

This protocol requires basic knowledge of how to run command line commands using Linux/Unix syntax. Although familiarity with Python and RDKit is helpful, this is not needed for most of

the protocol, and parts of the protocol that require this (for example, for filtering or visualizing model prediction results) are optional. Access to computational resources (for example, Google Cloud, Oracle Research credits or supercomputing services housed at several universities and institutions) may be helpful for advanced analyses and provide the best experience, but this is not required. For the experimental sections of this protocol, familiarity with performing general wet-laboratory experiments is recommended. Experience with or knowledge of high-throughput screening procedures, including familiarity with liquid-handling robots, may be helpful but is not required.

Limitations

As with ML approaches in general, the quality of the results generated by this protocol depends on the quality of the datasets used for training. This protocol, therefore, includes experimental procedures relevant to antibiotic discovery, and any wet-laboratory procedures for different applications should generate similarly consistent and standardized data. Here, we focus on bacterial growth inhibition and human cell cytotoxicity for antibiotic discovery. These two types of activity constitute starting points for identifying antibiotic candidates, but compounds validated to have selective antibacterial activity should be further characterized, for instance, by considering protein binding in serum, chemical stability, absorption, distribution, metabolism and excretion properties and efficacy in treating bacterial infections in vivo.

Chemprop is currently only able to handle the inference of scalar quantities. This entails that higher-dimensional data describing molecular properties—for instance, imaging data, high-throughput 'omics datasets or 3D geometric structures—should be mapped to scalar quantities. Functionality wise, Chemprop's focus on inference means that it cannot generate new molecules; rather, it scores user-provided chemical structures. This protocol is, therefore, best applied to mine preexisting virtual chemical libraries, but these libraries can differ in terms of the chemical diversity represented.

Finally, although model explainability provides generalizable and human-understandable reasoning for predictions, it has limitations. Explainability is often considered post hoc to rationalize model predictions[34–37,44]. Interpretability is needed to understand how models arrive at specific predictions[38]. Explainable and interpretable ML are developing subfields of AI, and how both concepts may apply to other application areas of AI and ML remain to be further studied.

Experimental design

Suitable types of activity

Although this protocol can be used to predict any type of activity that can be inferred from chemical structure, we recommend focusing on well-defined activity that can be measured using high-throughput screens. This ensures that standardized and well-controlled training datasets can be generated and that multiple (>100) hits can be experimentally tested. There are various types of high-throughput screens and, for scale and cost efficiency, we typically recommend simple screens that provide numerical scalar quantities as a measure of activity. Examples of such screens include antibacterial growth inhibition screens for different bacterial species measuring optical density, cytotoxicity screens for different cell types measuring cell viability, viral replication screens for different viral species measuring viral titer, protein inhibition screens for different proteins measuring catalytic activity and aqueous solubility screens for different compounds measuring compound concentration in solution. High-content imaging-based screens can in principle be used, but imaging data should be mapped to scalar quantities, as Chemprop does not handle two-dimensional data.

Choice of screening library

Initial screening libraries should contain at least ~1,000 compounds. In our experience, this ballpark number of compounds is typically needed to ensure that there are enough active and inactive compounds from which to learn. While models can be trained on as few as 45 active compounds[23], having a larger number (at least hundreds) of inactive compounds is important for models to sufficiently learn from negative examples. For screening drug-like compounds,

commercial vendors, including MicroSource Discovery Systems, MedChemExpress, ChemDiv, Cayman Chemical, Selleck Chemicals and others, sell ready-to-use libraries that can be customized depending on the application (for example, contain compounds known for having kinase-inhibiting activity). There are fewer available libraries for screening inorganic compounds, but several databases contain large datasets that might be acceptable to use as training data (Table 1). Where applicable, compounds that are known to have activity should be included in experimental screen as controls, and these compounds may also be used to augment the training data.

Table 1 | Overview of organic and inorganic compound libraries available for in silico screening

Database	Number of molecules	Description	Reference
Organic drug-like databases			
Readily purchasable chemical spaces			
ZINC22	750 million purchasable	Database of commercially accessible small molecules derived from multibillion-scale make-on-demand libraries. Includes catalogs from Enamine (REAL), WuXi (GalaXi) and Mcule (Ultimate)[1]	https://doi.org/10.1021/acs.jcim.2c01253 https://cartblanche22.docking.org
PubChem	117 million (117,473,493)	Chemical structures extracted from PubChem Substance records (313,651,772 total chemical entities)	https://pubchem.ncbi.nlm.nih.gov/docs/statistics
Mcule	41 million (41,766,991)	Mcule is a drug discovery platform integrating purchasable compound databases	https://mcule.com/database/
MedChemExpress	20 million (20,211,464)	MedChemExpress (MCE) provides curated compound libraries[2] containing structurally diverse compounds	https://www.medchemexpress.com/screening-libraries.html
MolPort	7.6 million (7,602,279)	MolPort is an online compound sourcing platform to purchase in-stock small molecules from over 25 chemical vendors[3]	https://www.molport.com/shop/screening-compounds
ChEMBL	2 million (2,399,743)	ChEMBL (version 33) is a manually curated database of bioactive molecules with drug-like properties	https://www.ebi.ac.uk/chembl/
ChemDiv	>1.6 million	Collection of low molecular weight organic compounds including screening libraries centered around chemical diversity, solubility, permeability and other criteria	https://www.chemdiv.com/catalog/screening-libraries/
ChemBridge	>1.3 million	ChemBridge provides small molecule, lead-like and drug-like screening compounds that are readily available in-stock	https://chembridge.com/screening-compounds/lead-like-drug-like-compounds/
Broad Institute	800,000	In-house compounds curated and maintained by the Center for the Development of Therapeutics at the Broad Institute	https://www.broadinstitute.org/center-development-therapeutics/compound-management-and-repurposing-hub
Chemical Entities of Biological Interest (ChEBI)	61,189 fully annotated	A freely available dictionary of molecular entities focused on 'small' chemical compounds	https://www.ebi.ac.uk/chebi/
DrugBank	16,581	A biomedical knowledgebase comprising relevant information on drugs and drug targets. DrugBank Online (version 5.1.11) includes 2,769 approved small molecule drugs	https://go.drugbank.com
Drug Repurposing Hub	7,934	Curated and annotated collection of US Food and Drug Administration (FDA)-approved drugs, clinical trial drugs, and pre-clinical compounds from the Broad Institute	https://repo-hub.broadinstitute.org/repurposing
Pharmakon and Natural Product libraries	2,560	Pharmakon-1760 combines the 1360 drugs in Microsource Discovery System's US Drug Collection (FDA approved) with the 400 drugs from the International Drug Collection. The NatProd Collection contains 800 natural products and their derivatives	http://www.msdiscovery.com/pharmakon.html http://www.msdiscovery.com/natprod.html
Prestwick	1,520	Collection of diverse small molecules, 95% of which are drugs approved by FDA, European Medicines Agency (EMA) and other agencies	https://www.prestwickchemical.com/screening-libraries/prestwick-chemical-library/
On-demand ultralarge chemical spaces			
eMolecules eXplore	5 trillion	Large virtual chemical space containing over >4.9 trillion compounds, built using 45 chemical reactions	https://www.emolecules.com/explore
Ambinter AMBrosia	110 billion (110,496,278,572)	AMBrosia is an ultralarge chemical space gathering virtual and accessible molecules based on 53,694 in-house building blocks and 32 chemical reactions	https://ambinter.com/ambrosia

Table 1 (continued) | Overview of organic and inorganic compound libraries available for in silico screening

Database	Number of molecules	Description	Reference
Organic drug-like databases			
Readily purchasable chemical spaces			
Enamine REAL Space	48 billion (48,078,672,450)	The REAL Space is assembled via more than 167 parallel synthesis protocols with ~143,000 qualified reagents and building blocks	https://enamine.net/compound-collections/real-compounds/real-space-navigator
WuXi GalaXi	16 billion	GalaXi leverages WuXi LabNetwork's building blocks and >30 different reaction types	https://wuxibiology.com/drug-discovery-services/hit-finding-and-screening-services/virtual-screening/
OTAVAchemicals CHEMriya	11.8 billion	The first release of the CHEMriya Space contains ~12 billion accessible on-demand molecules, based on 30,000 building blocks and 44 in-house reactions	https://www.otavachemicals.com/products/chemriya
Chemspace Freedom Space	5 billion (4,999,512,444)	Collection of ~5 billion molecules, a chemical space created using ML-based models, trained on reaction success data from Enamine	https://chem-space.com/compounds/freedom-space
ChemBridge VAST	700 million	The VAST structural database is a virtual database of synthetically accessible small molecules for hit generation and library expansion	
Inorganic compound-containing databases			
CAS Registry	219 million	The CAS registry has a curated list of organic substances, alloys, coordination compounds, minerals, mixtures, polymers and salts disclosed in publications since the early 1800s	https://www.cas.org/cas-data/cas-registry
ChemSpider	129 million	ChemSpider is a free chemical structure database providing access to over 58 million chemical structures, properties, and associated information	https://www.chemspider.com
Gmelin	1,500,000	A database of inorganic and organometallic compounds that is closed access	https://www.library.illinois.edu/chx/reaxys/
Atom Work Adv	379,736	Database that contains data on crystal structure, X-ray diffraction, properties and state diagrams of inorganic materials extracted from scientific literature	https://atomwork-adv.nims.go.jp/en/service.html
Inorganic Crystal Structure Database	>291,000 structures	A list of known inorganic crystal structures published since 1913, including their atomic coordinates. Includes experimental inorganic structures, experimental metal-organic structures and theoretical inorganic structures	https://icsd.products.fiz-karlsruhe.de/en
Materials Project	154,718 materials	Comprehensive database of computed information about inorganic, crystalline materials and molecules. Includes 4,351 intercalation electrodes and 172,874 inorganic small molecules	https://next-gen.materialsproject.org
The Electrolyte Genome	26,000	A database of liquid organic electrolytes used for calculating key properties for beyond lithium-ion batteries	https://www.jcesr.org/scientific-tools/materials-project-and-electrolyte-genome/
Fragment libraries			
GDB-17	166 billion (166,443,860,262)	Enumerated molecules of up to 17 atoms of C, N, O, S and halogens, covering a size range containing many drugs and typical for lead compounds	https://doi.org/10.1021/ci300415d
GDB-13	970 million (977,468,314)	All enumerated small organic molecules containing up to 13 atoms of C, N, O, S and Cl following simple chemical stability and synthetic feasibility rules	https://doi.org/10.1021/ja902302h
GDB-11	26.4 million (26,434,571)	All possible molecules of up to 11 atoms of C, N, O and F following simple valency, chemical stability and synthetic feasibility rules	https://doi.org/10.1021/ci600423u
Enamine REAL Fragments	259,380	Fragment collection based on over 300,000 building blocks. Contains fragments bearing covalent warheads and photolabels, to be grown into lead compounds	https://enamine.net/compound-collections/fragment-collection
Life Chemicals	60,000	Fragment collection available in stock and including a selection of fragment subsets with unique properties	https://lifechemicals.com/screening-libraries/fragment-libraries
MayBridge Fragments	30,000	The fragment collection contains small molecules with molecular weight (≤350 Da) and is selected for maximum diversity and drug-like properties	https://www.thermofisher.com/us/en/home/industrial/pharma-biopharma/drug-discovery-development/screening-compounds-libraries-hit-identification/maybridge-fragment-libraries.html

Table 1 (continued) | Overview of organic and inorganic compound libraries available for in silico screening

Database	Number of molecules	Description	Reference
Organic drug-like databases			
Readily purchasable chemical spaces			
MCE Fragment Library	22,419	'Rule-of-three' fragment library comprising over 20,000 fragments	https://www.medchemexpress.com/screening/Fragment_Library.html
FBDD (WuXi)	3,100	Diverse set of fragments for fast and cost-efficient hit identification using fragment-based screening	https://wuxibiology.com/drug-discovery-services/hit-finding-and-screening-services/fragment-screening-fbdd/
Prestwick Fragments	1,456	Collection of small fragments (molecular weight <300 Da) arising from the fragmentation of approved drugs (1500 drugs approved up to the year 2016)	https://www.prestwickchemical.com/screening-libraries/prestwick-drug-fragment-library/

[1]ZINC-22 currently uses five source catalogs: Enamine REAL Database (5 billion), Enamine REAL Space (29 billion), WuXi (2.5 billion), Mcule (128 million) and ZINC20 in stock (4 million). The database contains over 37 billion enumerated, searchable, commercially available compounds in two dimensions (2D). [2]MedChemExpress comprises over 15 highly-curated compound libraries (see Table 2 for a full list). [3]MolPort enables compound sourcing from over 30 different chemical suppliers including ChemDiv, ChemBridge, VitasM, WuXi LabNetwork, MayBridge and MedChemExpress (see Table 3 for a full list).

Design of experimental screen

As training data for sizeable numbers (>1,000) of compounds is needed for the best results, experimental screens should be designed to ensure high reproducibility and consistency. One way to assess reproducibility and consistency is to perform experimental replicates of the same screen using the same compounds and calculate the Pearson correlation coefficient (R) between the values from each replicate. We have previously done so for a screen of 39,312 compounds for antibacterial activity against *Staphylococcus aureus*[7], finding $R = 0.8$. In general, screens for which $R > 0.7$ can be regarded as reproducible and consistent. Other ways of assessing screening results and effect sizes, for instance using the Z factor[60], are possible. Experimental considerations for increasing reproducibility and consistency include standardization of all reagents, use of robotic or liquid-handling systems and proper maintenance of compound libraries.

It may be difficult to perform screens in replicate, particularly if resources are limited. As Chemprop models learn from chemotypes in the training data, we generally prefer screening more compounds without replicates to screening fewer compounds with replicates, as this provides a greater number of chemotypes from which to learn. If the screening will be performed without replicates, we recommend repeating the screen for a small and randomly chosen subset of all compounds and validating the subset of active compounds. There are typically few active compounds compared with inactive compounds and, for this reason, false positives can be expected to have a stronger effect on model accuracy than false negatives.

Design of Chemprop models

Properly defining the models—ranging from the type of model to additional optional inputs to Chemprop—will depend on the quality of the training data and the type of activity predicted. If the training data are sufficient but not large (for example, thousands of compounds screened and dozens to hundreds of active compounds), we recommend classification models, which coarse grain the data and can be expected to perform better than regression models when information is limited. We recommend ensembling with a small number (<20) of models to average out prediction score noise arising from the stochasticity in model training. We also generally recommend performing the optional step of Bayesian hyperparameter optimization as a principled way to determine initial sets of model hyperparameters. For initial tests, other Chemprop parameters may best be set to default given that similar settings have been applied in previous work[7,16–19,23,24,45].

Suitable libraries for mining

This protocol can be broadly applied to diverse chemical spaces, and the choice of which libraries to mine in silico should be informed by chemical diversity and the feasibility of experimentally testing predicted hits. For the discovery of drug-like molecules, public databases, including PubChem[61], ChEMBL[62], ChEBI[63], DrugBank[64], EPA CompTox[65] and ZINC22 (ref. 47), along with commercial databases, offer possible starting points (Table 1). For the discovery of inorganic compounds, databases including the Inorganic Crystal Structure Database[66,67] and newer text-

mined datasets[68] may be relevant. Additionally, databases containing organic compounds, including PubChem, ZINC22, CAS and ChemSpider also contain many inorganic compounds (Table 1). For both drug-like and inorganic compounds, chemical fragment libraries also offer a large source of chemical structures to mine (Table 1). Exploring large virtual libraries (such as the ZINC22) provides attractive opportunities to mine highly chemically diverse spaces, but a limitation is that most compounds in these libraries are not commercially available. For instance, the MolPort and MedChemExpress databases of commercially available, mostly organic compounds contains between ~8 and ~20 million compounds, which need to be combined from multiple suppliers (Tables 2 and 3). It may be possible to synthesize commercially unavailable compounds, and several commercial databases (such as the Enamine REAL space[69]) contain compounds that are expected to be synthetically feasible. Nevertheless, accurately predicting which compounds can be synthesized remains a challenge[70] and, in our experience, most chemically possible drug-like compounds and fragments—such as those found in the generated databases (GDBs)[71–74]—have low synthetic accessibility.

The need for counterscreening and filtering

This protocol includes counterscreening for cytotoxicity against human cells and training Chemprop models that predict cytotoxicity, as we require discovered antibacterial compounds

Table 2 | List of chemical libraries and purchasable numbers of molecules through MedChemExpress

Library	Name	Number of molecules (as of March 2024)	Description
HY-L001V	MCE Bioactive Compound Library	22,275	A collection of bioactive and structurally diverse compounds. The bioactivity and safety was confirmed by preclinical research and clinical trials. Some have been approved by the FDA
HY-L0091V	Chemspace Lead-Like Compound Library	981,244	The Chemspace Lead-Like Compound Library contains in-stock lead-like compounds with favorable physicochemical profiles and high quantitative estimation of drug-likeness
HY-L0093V	Chemspace Scaffold derived set	10,119	Includes 3,373 scaffolds, with three compounds sampling each
HY-L0094V	Chinese National Compound Library	1,398,968	~1.4 million compounds possessing diverse structures
HY-L0101V	FCH Group Screening Library	2,244,487	The FCH Group Screening Library is a library containing diverse screening compounds
HY-L0105V	InterBioScreen Synthetic Compounds Library	485,000	The InterBioScreen Synthetic Compounds is another library containing diverse screening compounds
HY-L0088V	Life Chemicals 50K Diversity Library	50,240	The Life Chemicals 50K Diversity Library is another library containing diverse screening compounds
HY-L0087V	Life Chemicals HTS Compound Collection	494,471	The Life Chemicals HTS library is another library containing diverse screening compounds
HY-L0107V	Life Chemicals Natural Product-like Compound Library	13,236	This library contains in-stock natural product-like compounds. Designed by two-dimensional fingerprint similarity filtering and chemical descriptor-based and natural-likeness-based scoring
HY-L912V	MegaUni 10M Virtual Diversity Library	10,000,000	This library contains synthetically accessible screening compounds for lead identification. The compounds are easy to synthesize via standard one- to two-step procedures
HY-L910V	MegaUni 50K Virtual Diversity Library	50,000	A collection of synthetically accessible, lead-like compounds with structural diversity. The compounds are easy to synthesize via standard one- to two-step procedures
HY-L0095V	OTAVAchemicals Screening Collection	270,000	The OTAVAchemicals Screening Collection is another library containing diverse screening compounds
HY-L0106V	Protein-Protein Interaction Modulators Library	2,906	Designed for the discovery of protein–protein interaction modulators
HY-L0086V	Specs HTS Compounds Library	208,518	The Specs HTS library is another library containing diverse screening compounds
HY-L0104V	UORSY New Generation Screening Library	1,900,000	The UORSY New Generation Screening is another library containing diverse screening compounds
HY-L0103V	UORSY Screening Library	680,000	The UORSY Screening Library contains compounds designed on the basis of a polymerization synthesis method
HY-L0096V	Vitas-M Screening Compounds Library	1,400,000	The Vitas-M Screening Compounds Library is another library containing diverse screening compounds

Table 3 | List of chemical vendors and purchasable numbers of molecules through MolPort

Supplier	Number of molecules (as of February 2024)
ChemDiv	1,670,550
ChemBridge Corporation	1,654,084
Vitas M Chemical Limited	1,413,838
TimTec, LLC (4 weeks)	590,948
INTERBIOSCREEN DOO BAR	554,762
Life Chemicals	548,824
Otava	201,449
Specs	200,729
TimTec	141,606
WuXi LabNetwork	121,838
BIONET—Key Organics	104,760
Otava (2 weeks)	101,656
Eximed	57,951
Maybridge	53,336
AnalytiCon Discovery, GmbH	43,758
HTS Biochemie Innovationen	36,702
MedChemExpress Europe	28,772
TargetMol Chemicals	14,147
Cayman Europe	13,976
Molnova	10,350
ChemFaces	7,285
Selleck Chemicals	6,490
Menai Organics Limited	5,007
Apexbio Technology	4,858
BioTechne Ireland Limited	3,938
ChemNorm	3,289
Axon Medchem	2,382
Chengdu Biopurify Phytochemicals	2,306
Fulfilled by MolPort	1,687
Excenen Pharmatech Co	1001

to be selective. Where applicable, we recommend performing appropriate counterscreens that inform compound selectivity for the main activity of interest. These counterscreens should be performed when experimentally validating active compounds or model-predicted hits, and they may also be performed for training data generation. Examples of such counterscreens may include screens measuring cytotoxicity against different cell types and species, screens measuring off-target binding and screens measuring compound color. When the counterscreening dataset is large enough to train additional Chemprop models that predict unspecific or undesired activity (according to the experimental design considerations discussed above), doing so facilitates the filtering of model-predicted hits, as detailed in the 'Procedure' section.

Materials

Biological materials

- *Staphylococcus aureus* RN4220 (BEI Resources cat. no. NR-45946)
  ▲ **CAUTION** *S. aureus* is a biosafety level 2 organism. Ensure appropriate use and handling of pathogenic microorganisms, including wearing gloves and appropriate personal protective equipment.

- HepG2 cells (American Type Culture Collection (ATCC), cat. no. HB-8065, RRID:CVCL_0027, https://scicrunch.org/resolver/RRID:CVCL_0027)
- Human primary skeletal muscle cells (HSkM cells; ATCC, cat. no. PCS-950-010)
- IMR-90 cells (ATCC cat. no. CCL-186, RRID:CVCL_0347, https://scicrunch.org/resolver/RRID:CVCL_0347)

▲ **CAUTION** The cell lines used should be regularly checked to ensure they are authentic and are not infected with mycoplasma.

Reagents

Bacterial culture

- Liquid LB medium (Becton Dickinson, cat. no. 244620)
- LB media containing 1.5% (wt/vol) Difco agar (Becton Dickinson, cat. no. 244520)

Mammalian cell culture

- Dulbecco's modified Eagle medium (DMEM; Corning, cat. no. 10-013-CV)
- Fetal bovine serum (FBS; ThermoFisher, cat. no. 16140071)
- Penicillin–streptomycin (PS; ThermoFisher, cat. no. 15070063)
- Mesenchymal stem cell basal medium for adipose, umbilical and bone marrow-derived mesenchymal stem cells (ATCC, cat. no. PCS-500-030)
- Primary skeletal muscle growth kit (ATCC, cat. no. PCS-950-040)
- Eagle's minimum essential medium (EMEM; ATCC cat. no. 30-2003)
- Trypsin–EDTA, 0.05% (wt/vol) (Corning cat. no. 25-052-CI)
- Resazurin (Millipore Sigma, cat. no. R7017)

Screening

- Pharmakon Collection (MicroSource Discovery Systems cat. no. Pharmakon-1760)
- Natural Products Collection (MicroSource Discovery Systems cat. no. NatProd Collection)
- Optional: dimethylsulfoxide (DMSO; Millipore Sigma, cat. no. D5879)

Equipment

Equipment for wet laboratory experiments

- Sterile pipette tips (Rainin cat. no. 17005872 and 17005874)
- Optional: liquid-handling robot (for example, Agilent Bravo cat. no. G5498B#041 or similar)
- Sterile microbiological inoculation loops (Beckton Dickinson cat. no. 220215)
- Sterile 14 mL culture tubes (Corning cat. no. 352059)
- Sterile 15 mL centrifuge tubes (Corning cat. no. 352096)
- Sterile untreated clear flat-bottom 96-well plates (Corning cat. no. 3370)
- Sterile tissue-treated black flat-bottom 96-well plates (Corning cat. no. 3904)
- Sterile untreated petri dishes, 100 mm × 15 mm (Corning cat. no. 351029)
- Sterile tissue culture flasks, T-175 (Corning cat. no. 353028)
- Sterile reagent reservoirs (Corning cat. no. 4870)
- Sterile filter, 0.45 µm (Corning cat. no. 431225)
- Microbiological humidity-controlled incubator set to 37 °C and 0% CO_2 (vol/vol), with shaking option (Thermo Fisher cat. no. SHKE6000)
- Tissue culture humidity-controlled incubator set to 37 °C and 5% CO_2 (vol/vol) (Thermo Fisher cat. no. 50162969)
- Inverted tissue culture microscope with a low-magnification (for example, 4×) objective (Leica cat. no. 11526208)
- Plate reader capable of reading absorbance and fluorescence (for example, Molecular Devices SpectraMax plate reader cat. no. M3 or similar)
- Centrifuge with rotors for tubes and plates (for example, Eppendorf cat. no. 5920R or similar)
- Optional: liquid dispenser with sterile cassette (for example, Agilent BioTeK MultiFlo dispenser cat. no. MFP or similar)

Hardware requirements

- Internet connection is required to download the software in the 'Equipment setup' section
- A personal computer will suffice for typical workflows, and for the best experience, we recommend using a computer with at least 16 GB of memory and 10 GB of disk space

Software requirements

- A command line or terminal shell is needed to execute Chemprop commands. If using RDKit (optional), we recommend running RDKit commands in a Python notebook (for example, Jupyter) in a modern web browser (for example, Google Chrome) for the best experience.

Reagent setup

For the LB medium and LB agar, dissolve LB medium powder and LB agar powder in Milli-Q water and autoclave to sterilize. LB liquid medium is stable for >1 year when stored at room temperature (20 °C) and kept sterile. LB agar should be poured onto sterile untreated Petri dishes (20 mL per dish). LB agar plates are stable for >1 year when stored at 4 °C under high humidity to slow evaporation and kept sterile.

For the complete human cell culture media, add FBS and PS to DMEM and EMEM to make complete growth media, each containing final concentrations of 10% (vol/vol) FBS and 1% (vol/vol) PS for HepG2 and IMR-90 cells, respectively. Add all components of the primary skeletal muscle growth kit to the mesenchymal stem cell basal medium for adipose, umbilical and bone marrow-derived mesenchymal stem cells together with PS (1% (vol/vol) final concentration) to make complete growth media for HSkM cells. Complete growth media is stable for at least 1 month when stored at 4 °C and kept away from light.

For the chemical libraries, if needed to facilitate accurate pipetting, chemical libraries can be diluted in DMSO to an appropriate stock concentration. We recommend choosing the stock concentration such that the final concentration of DMSO in samples is ≤0.5% (vol/vol), as higher DMSO concentrations of ~1% (vol/vol) will noticeably affect human cell viability and higher DMSO concentrations of ~4–8% (vol/vol) will affect bacterial growth. Chemical libraries should be stored at −20 °C or lower when not in use and thawed completely before use. We also recommend centrifuging the chemical libraries before use to consolidate the available amounts of solution.

Equipment setup

Download Miniconda. Miniconda (which contains Python) can be downloaded and installed from https://conda.io/miniconda.html.

Download Chemprop (v1). Chemprop can be downloaded and installed from https://github.com/chemprop/chemprop. Run the following commands to install Chemprop from source:

```
git clone https://github.com/chemprop/chemprop.git
cd chemprop
conda env create -f environment.yml
conda activate chemprop
pip install -e .
```

Additionally, Chemprop can be installed using the Python Package Index (PyPi) by running the following commands:

```
conda create -n chemprop python=3.8
conda activate chemprop
pip install chemprop
```

▲ **CRITICAL** It is important to note that Chemprop is in the middle of a major update (from v1 to v2). As key functionality (for example, producing rationales) is only available in Chemprop v1, this protocol focuses on using this version.

(Optional) Download example datasets. Example datasets can be downloaded from https://github.com/felixjwong/antibioticsai. Smaller example datasets corresponding to the Pharmakon Collection and Natural Products Collection libraries described in this protocol are available as Supplementary Datasets 1–4. These smaller examples are subsets of the 39,312 compound datasets described in previous work from our laboratory[7].

(Optional) Download RDKit. RDKit can be downloaded and installed from https://www.rdkit.org/docs/Install.html. Additionally, RDKit can be installed using PyPi by running the following command:

```
pip install rdkit
```

Procedure

Stage 1, data generation

● **TIMING** 1 d to 1 week

▲ **CRITICAL** This stage can be customized (or skipped) if one is adapting the protocol or applying the protocol to cases in which the process of generating training data is different (or difficult; Fig. 2). In such cases, consult the 'Experimental design' section for training data guidelines.

Bacterial growth inhibition screening

1. Streak 1 µL of a stock culture of *S. aureus* RN4220 on one LB agar plate using a sterile inoculation loop. Incubate at 37 °C overnight (16–24 h) in a stationary incubator.
2. Using a new sterile inoculation loop, pick a single colony of *S. aureus* RN4220 from the plate and inoculate 2 mL of liquid LB in a 14 mL Falcon tube. Incubate at 37 °C overnight (16–24 h) with shaking at 300 rpm.
3. Calculate the required total working volume needed for the screen. Here, we use a 100 µL working volume per well and a total of 64 96-well plates for the screen (performed in biological duplicate) because the source library, the combination of the Pharmakon Collection and the Natural Products Collection from MicroSource Discovery Systems, can be supplied as a total of 32 96-well plates (for example, 96 wells per plate × 64 plates = 6,144 wells. 6,144 wells × 100 µL per well = 614,400 µL total working volume = 614.4 mL total working volume).
   ▲ **CRITICAL STEP** We have previously used 384-well plates for initial screens, and using plates with any number of wells is possible. However, this protocol will focus on using 96-well plates to make pipetting easier. We recommend at least 80 µL working volume per well for 96-well plates.
   ▲ **CRITICAL STEP** Note that it is common for plated libraries to not use outer wells in to avoid evaporation and edge effects, and in these cases, fewer than 96 compounds can be screened per plate.
4. Dilute the overnight culture 1:10,000 in liquid LB to make a working bacterial culture volume greater than that needed for the screen. The final concentration of bacteria should be ~10^5 colony forming units per milliliter.
   ▲ **CRITICAL STEP** We recommend making at least 1.1× the needed volume to accommodate pipetting and measurement errors.
   ▲ **CRITICAL STEP** After being diluted with fresh LB, bacteria should be used as soon as possible to avoid the confounding effects of initial bacterial growth on the experiment. We recommend adding all compounds (Step 6) and starting incubation (Step 7) within 15 min of dilution. If it is not possible to add all compounds in this timeframe, smaller volumes of the working bacterial culture can be made in batches.
5. Plate the working bacterial culture into 64 sterile untreated clear flat-bottom 96-well plates. This can be done by adding bacterial culture to a sterile reagent reservoir and using a manual or electronic multichannel pipette, but for the best experience and to improve

reproducibility, we recommend using a liquid dispenser or liquid-handling robot. In all cases, bacteria should be added aseptically.

6. Add 0.5 µL of each 10 mM stock compound from the Pharmakon Collection and Natural Products Collection libraries (or a library of your choice) to each well of the 96-well plates. Pipetted amounts can vary depending on the working volumes and stock compound concentrations, and here, our screen is at a final compound concentration of 50 µM. If the library needs to be diluted in DMSO to facilitate pipetting, we recommend keeping final DMSO concentrations below or equal to 2% (vol/vol) for bacteria.
   ▲ **CRITICAL STEP** For the best experience and to improve reproducibility, we recommend using a liquid-handling robot.
7. Incubate all plates at 37 °C overnight (16–24 h) in a stationary incubator.
8. Remove plates from incubation and read the absorbance at 600 nm using a plate reader. One reading should be taken for each of the 64 plates, which contain biological duplicates.
   ■ **PAUSE POINT** This concludes the initial screening for *S. aureus* RN4220 growth inhibitory activity. Ensure that the data generated are of reasonable quality before proceeding. For instance, the Pearson correlation coefficient between the two biological duplicate absorbance readings across all compounds may be expected to be at least 0.5–0.7 to ensure high reproducibility.

Human cell cytotoxicity screening

9. Upon receiving the vials from ATCC, thaw and seed HepG2 cells, IMR-90 cells and HSkM cells in T-75 flasks containing complete growth media. HepG2 cells should be cultured in complete DMEM, IMR-90 cells should be cultured in complete EMEM and HSkM cells should be cultured in complete mesenchymal stem cell medium. Incubate at 37 °C in 5% (vol/vol) CO_2 overnight before changing media the following day to remove residual cryoprotectant. Expand the cells in T-175 flasks incubated at 37 °C in 5% (vol/vol) CO_2 until there are ~60 million cells of each cell type. If possible, ensure that passage numbers are low (<10) and that cells are not overconfluent (that is, cells are at most 80–90% confluent).
10. As described in Step 3, calculate the required total working volume needed for the screen. Here, we again use a 100 µL working volume per well and a total of 64 96-well plates for the screen (performed in biological duplicate) for each cell type.
11. Detach cells using trypsin by aspirating the growth media and adding 10 mL of 0.05% trypsin–EDTA solution to each T-175 flask. Flasks can be incubated for 5–30 min at 37 °C in 5% (vol/vol) CO_2 to facilitate detachment. Confirm that cells are detached by examining cells under a tissue culture microscope. Transfer the detached cells into a 15 mL centrifuge tube, and centrifuge gently at room temperature at 120*g* for 5 min to pellet cells. Aspirate and discard the trypsin, being careful not to disturb the cell pellets.
12. Resuspend cells in total working volume of each complete media for each screen.
13. Plate cells at a final concentration of ~10,000 cells per well. We again recommend using a liquid dispenser or liquid-handling robot. If these are used, it is important to check that the machine's shearing forces are small enough to ensure human cell viability. In all cases, human cells should be added aseptically.
14. Allow cells to attach by incubating all plates at 37 °C and in 5% (vol/vol) CO_2 overnight (16–24 h).
15. Add 0.5 µL of each 2 mM stock compound, for final concentrations of 10 µM. If the library needs to be diluted in DMSO to facilitate pipetting, we recommend keeping final DMSO concentrations below or equal to 0.5% for human cells. We again recommend using a liquid-handling robot.
16. Incubate all plates at 37 °C and in 5% (vol/vol) CO_2 for 2–3 d.
17. To prepare a working solution of resazurin, a cell viability indicator, first make a stock solution of 100 mM resazurin in Milli-Q water and sterile-filter the solution. Make the working solution by diluting the stock solution 1:110 in complete media.
18. Add 20 µL of working resazurin solution to each well of each plate. The final concentration of resazurin is ~0.15 mM.
19. Incubate all plates at 37 °C and in 5% (vol/vol) CO_2 for an additional 4–24 h.

20. Remove plates from incubation and read the fluorescence intensity at 550/590 nm (excitation/emission) using a plate reader. One reading should be taken for each of the 64 plates, which contain biological duplicates.
    ■ **PAUSE POINT** This concludes the initial screening for human cell cytotoxicity. As before, ensure that the data generated are of reasonable quality before proceeding. For instance, for any cell type of interest, the Pearson correlation coefficient between the two biological duplicate fluorescence intensity readings across all compounds may be expected to be at least 0.5–0.7 to ensure high reproducibility.
    ◆ **TROUBLESHOOTING**

Stage 2, model training and benchmarking

● **TIMING 1 h to 1 week**

Data processing

21. Normalize the experimental data. We recommend normalizing absorbance or fluorescence intensity values by the interquartile mean of each plate to account for variation across plates. The resulting values will measure relative bacterial growth or relative human cell viability, respectively.
22. Bin the data. As ~2,500 compounds are screened in biological duplicate in this protocol, we will use binary classification. Accordingly, using a typical growth inhibition cutoff of 0.2, assign compounds resulting in relative mean bacterial growth <0.2 the active or positive label ('1'), and compounds resulting in relative mean bacterial growth ≥0.2 the inactive or negative label ('0'). Similarly, assign compounds resulting in relative mean human cell viability <0.9 the cytotoxic or positive label ('1'), and compounds resulting in relative mean human cell viability ≥0.9 the noncytotoxic or negative label ('0').
23. Prepare the training dataset. The SMILES string for each compound, provided by MicroSource Discovery Systems to accompany the Pharmakon Collection and the Natural Products Collection, should correspond to one value for each type of activity measured. Tabularize this information in .csv format. It may also be useful, although not strictly needed, to deduplicate compounds that are replicated in the screen according to their SMILES strings. For this protocol, there are four .csv files that will be needed, one each for *S. aureus* antibacterial activity, HepG2 cytotoxicity, IMR-90 cytotoxicity and HSkM cell cytotoxicity. Sample deduplicated output datasets from Stage 1 for each of these activities are provided as part of this protocol as Supplementary Datasets 1–4. Each of the four .csv files will be used to train separate Chemprop models and should resemble the following, with one SMILES string and its corresponding binarized activity value per line:

```
SMILES,ACTIVITY
Oc1ccnc(S)n1,0
O=[N+]([O-])c1ccc(O)c([N+](=O)[O-])c1,1
…
```

24. In a command line or terminal shell, navigate to the directory of interest and activate Chemprop by running the following command:

```
conda activate chemprop
```

    ◆ **TROUBLESHOOTING**
25. Define Chemprop model parameters. Chemprop provides many customizable parameters, several of which are summarized in Table 4. For this protocol, we will use ten iterations of Bayesian hyperparameter optimization to determine suitable GNN architecture parameters for the dataset named 'train.csv'. To do so, run the following command:

```
chemprop_hyperopt --data_path train.csv --dataset_type classification
--num_iters 10 --config_save_path hyperparameters.json
```

Table 4 | Commonly used Chemprop model parameters

Parameter	Description	Options and recommended usage
Directory parameters		
data_path	A directory path to the .csv file containing the training dataset	Specify a .csv file that is of the proper format, containing SMILES strings and activity values (Box 5)
dataset_type	A specification of type of Chemprop model to train	Options include 'classification', 'regression', 'multiclass' and 'spectra'; this protocol recommends 'classification'
save_dir	A directory path to the folder that will contain saved Chemprop models	Specify an accessible folder. If the folder does not exist, Chemprop will create the folder
Hyperparameter optimization parameters[1]		
num_iters	Number of hyperparameter trial configurations to try for Bayesian hyperparameter optimization	Can be any positive integer. This protocol recommends '10'
config_save_path	Directory path to the .json that will contain saved hyperparameters	Specify a suitable filename for the .json file in an accessible directory
Training parameters[2]		
config_path	A directory path to a .json file specifying model hyperparameters	Specify a .json file that is of the proper format, such as that resulting from chemprop_hyperopt
split_type	A type of dataset split to be used during model training	Options include unspecified, 'scaffold_balanced', 'cv', 'cv-no-test', and 'random_with_repeated_smiles'. This protocol recommends unspecified, which will result in the data being randomly split into train, validation and test sets
num_folds	Number of cross-fold validation folds (specified only when split_type is 'cv')	Can be any positive integer or unspecified. This protocol recommends unspecified
loss_function	A loss function to use for training	Options include unspecified; 'binary_cross_entropy', 'mcc' and 'dirichlet' for classification; 'mse', 'bounded_mse', 'mve', 'evidential' and 'quantile_interval' for regression; 'cross_entropy' and 'mcc' for multiclass; and 'sid' and 'wasserstein' for spectra. This protocol recommends unspecified, which for classification defaults to 'binary_cross_entropy'
dropout	A dropout fraction	Can be any float between 0 and 1 or unspecified. This protocol recommends unspecified
metric	A metric used to assess model against test and validation sets	Options include unspecified; 'auc', 'prc-auc', 'accuracy', 'binary_cross_entropy', 'f1', 'mcc', 'recall', 'precision' and 'balanced_accuracy' for classification; 'rmse', 'mae', 'mse', 'r2', 'bounded_rmse', 'bounded_mae', 'bounded_mse' and 'quantile' for regression; 'cross_entropy', 'accuracy', 'f1' and 'mcc' for multiclass; and 'sid' and 'wasserstein' for spectra. This protocol recommends unspecified, which for classification defaults to 'auc'
ensemble_size	Number of models in the ensemble	Can be any positive integer or unspecified. This protocol recommends '1'. Specifying a value greater than 1 will result in multiple models being trained, validated and tested on the same splits of the data
features_generator	Additional features to use in the model	Options include 'rdkit_2d_normalized –no_features_scaling', 'rdkit_2d_normalized', 'morgan', 'morgan_count' and 'rdkit_2d'. This protocol recommends unspecified or 'rdkit_2d_normalized –no_features_scaling'
Predicting parameters[3]		
test_path	A directory path to the .csv file containing the test dataset	Specify a .csv file that is of the proper format, containing SMILES strings (Box 5)
checkpoint_dir	A directory path to the folder that contains all saved Chemprop models, or to any specific Chemprop model (single .pt file)	Specify the same directory that was used for 'save_dir' above or for a subset of models, any folder containing a subset of .pt files
preds_path	A directory path to the .csv file containing the model predictions for the test dataset	Specify a suitable filename for the .csv file in an accessible directory
Explainability parameters[4]		
property_id	Which property to generate model explanations for	Can be '1' for binary classification models or any positive integer less than or equal to the number of properties learned for multiclass models. This protocol recommends '1'
min_atoms	Minimum number of atoms in a rationale	Can be any positive integer or unspecified. This protocol recommends unspecified
max_atoms	Maximum number of atoms in a rationale	Can be any positive integer greater than 'min_atoms' or unspecified. This protocol recommends unspecified

[1]Typical usage: `chemprop_hyperopt --data_path <data_path> --dataset_type <type> --num_iters <int> --config_save_path <config_path>`. [2]Typical usage: `chemprop_train --data_path <path> --dataset_type <type> --config_path <path> --ensemble_size <n> --save_dir <dir> --split_type <type> --num_folds <k> --loss_function <function> --dropout <float> --metric <metric>`. [3]Typical usage: `chemprop_predict --test_path <test_path> --checkpoint_dir <dir> --preds_path <preds_path>` [4]Typical usage: `chemprop_interpret --data_path <path> --checkpoint_dir <dir> --property_id <int>`.

When completed, the output is saved to the 'hyperparameters.json' file, which contains text in the following format:

```
{
    "depth": 5,
    "dropout": 0.15,
    "ffn_hidden_size": 1600,
    "ffn_num_layers": 1,
    "hidden_size": 1600
}
```

■ **PAUSE POINT** Depending on the size of the training data, hyperparameter optimization can take days. For the ~2,500 compound dataset generated in this protocol, hyperparameter optimization should take ~1 h to 1 d on a standard personal computer.
▲ **CRITICAL STEP** Note that, to run this and other commands from a Jupyter notebook instead of a command line, append '!' in front of the command and specify exact paths to files.
◆ **TROUBLESHOOTING**

26. Train an ensemble of Chemprop models to predict activity. Here, we use the hyperparameters generated in the previous step and train ten models that are saved to the 'checkpoints' folder. To do so, run the following command:

```
for i in $(seq 0 9); do chemprop_train --data_path train.csv
--dataset_type classification --config_path hyperparameters.json
--save_dir checkpoints/${i} --ensemble_size 1 --seed ${i} & done
```

To supplement training, we may optionally use prenormalized RDKit features. This can be done by appending the following flags to the 'chemprop_train' command above:

```
--features_generator rdkit_2d_normalized --no_features_scaling
```

In the above step, each model is trained, validated and tested on a random 80–10–10% split of the training data, according to Chemprop's default parameters. Each model will be trained on the basis of a different data split due to the different random seed being provided.
■ **PAUSE POINT** Depending on the size of the training data and training parameters, model training can take days. For the ~2,500 compound dataset generated in this protocol, model training should take ~1 h to 1 d on a standard personal computer.
▲ **CRITICAL STEP** Training ten models by setting the '–ensemble_size' flag to 10, instead of running the command above, will result in splits of the training data being handled differently. If –ensemble_size is 10, then each trained model within the ensemble will share data splits. In contrast, the trained models from the chemprop_train command above do not share data splits.
◆ **TROUBLESHOOTING**

27. Check model training results. Check that each directory in 'checkpoints' contains multiple folders each containing a '.pt' file; these .pt files are the model checkpoints that store the trained models. Additionally, each directory in 'checkpoints' should include four other files: 'quiet.log', 'verbose.log', 'test_scores.csv' and 'args.json'. Detailed descriptions of each of these files are provided in Box 5.
28. Benchmark the model by repeating Steps 26 and 27, but for training, use a subset of all available training data. This differs from benchmarking using the training metrics provided in the 'quiet.log' files in Step 27; these metrics indicate the performance of specific models when they are trained, validated and tested on random splits of 'train.csv', but altogether, the ten models have seen all the data in 'train.csv' above. Here, we aim

BOX 5

Data files used for Chemprop

Detailed descriptions and sample file formats of the data files used for Chemprop are as follows:

train.csv

User-provided comma-separated values (CSV) file containing training data.

Example:

```
SMILES,ACTIVITY
C=CC(C1=CC(=O)C(OC)=CC1=O)c1ccc(O)cc1,0
C=C1CCC=C(C)CCC2C1CC2(C)C.O,0
O=[N+]([O-])c1ccc(O)c([N+](=O)[O-])c1,1
…
```

hyperparameters.json

User-provided or Chemprop-generated JavaScript object notation (JSON) file containing hyperparameter specifications.

Example:

```
{
   "depth": 5,
   "dropout": 0.15,
   "ffn_hidden_size": 1600,
   "ffn_num_layers": 1,
   "hidden_size": 1600
}
```

checkpoints/0/quiet.csv

Chemprop-generated LOG file containing training statistics.

Example:

```
Fold 0
Model 0 best validation auc = 0.778898 on epoch 7
Model 0 test auc = 0.694481
…
```

checkpoints/0/verbose.log

Chemprop-generated LOG file containing training statistics.

Example:

```
Command line
python /Users/felix/anaconda3/envs/chemprop/bin/chemprop_train --data_path
train.csv --dataset_type classification --config_path hyperparameters.json
--save_dir checkpoints/0 --ensemble_size 1 --seed 0
Args
{'activation': 'ReLU',…
```

checkpoints/0/test_scores.csv and checkpoints/0/fold_0/test_scores.json

Chemprop-generated CSV and JSON files containing test scores of the final trained models.

Example:

```
Task,Mean auc,Standard deviation auc,Fold 0 auc
ACTIVITY,0.6944805194805194,0.0,0.6944805194805194
```

(continued from previous page)

checkpoints/0/args.json
Chemprop-generated JSON file containing all parameters of the final trained models.
Example:

```
{
   "activation": "ReLU",
   "adding_bond_types": true,
   "adding_h": false,
   "aggregation": "mean",
…
```

checkpoints/0/fold_0/model_0/model.pt
Chemprop-generated binary file containing all parameters needed to specify one Chemprop model (a model checkpoint).
Example:

```
504b 0304 0000 0808 0000 0000 0000 0000
0000 0000 0000 0000 0000 0e00 1400 6d6f
…
```

test.csv (or withheld.csv)
User-provided CSV file containing SMILES strings of test compounds.
Example:

```
SMILES
Cc1ccn(n1)CN1CCN(Cn2ccc(n2)C)C1=S
C1OC(OC[C@H]2O[C@@H](O[C@@H]12)c1ccccc1)c1ccccc1
…
```

test_results.csv (or withheld_results.csv)
Chemprop-generated CSV file containing SMILES strings and model prediction values of test compounds.
Example:

```
SMILES,ACTIVITY
Cc1ccn(n1)CN1CCN(Cn2ccc(n2)C)C1=S,0.04471180747987091
C1OC(OC[C@H]2O[C@@H](O[C@@H]12)c1ccccc1)c1ccccc1,0.044345813874315354
…
```

hits.csv
User-provided CSV file containing SMILES strings of hit compounds.
Example:

```
SMILES
CC3(C)S[C@@H]2[C@H](NC(=O)[C@H](N)C1=CC=C(C=C1)O)C(=O)N2[C@H]3C(=O)O
```

hits_rationales.csv
Chemprop-generated CSV file containing SMILES strings and model prediction values of test compounds.
Example:

```
smiles,score,rationale,rationale_score
['CC3(C)S[C@@H]2[C@H](NC(=O)[C@H](N)C1=CC=C(C=C1)O)C(=O)N2[C@H]3C(=O)O'],
0.863,C[CH:1]1S[C@@H]2[C@H]([NH2:1])[CH2:1]N2[CH2:1]1,0.748
Elapsed time = 0:24:15
```

to benchmark a similar set of ten models by manually withholding part of 'train.csv' and repeating the training process. We recommend withholding ~20% of the training data, which should contain representative numbers of active and inactive compounds and using the remaining ~80% of the training data to train Chemprop models as above. The withheld ~20% of the training data can be used for model benchmarking by comparing the model predictions against the known ground-truth values. Both subsets of the training data can be manually saved to two different .csv files, one for training and one for testing.

29. Using the Chemprop models trained on ~80% of the training data, as stored in 'checkpoints', run predictions on the withheld ~20% of the training data using the following command:

```
chemprop_predict --test_path withheld.csv --checkpoint_dir checkpoints
--preds_path withheld_results.csv
```

Here, 'withheld.csv' is a user-generated file containing a column of SMILES strings, with each SMILES string corresponding to a compound in the withheld set. If RDKit features were used to supplement training, remember to add the same flags during predictions to the command above:

```
--features_generator rdkit_2d_normalized --no_features_scaling
```

For a withheld ~500-compound dataset, the prediction step should take less than or ~1 h on a standard personal computer.

◆ **TROUBLESHOOTING**

30. View the prediction results in 'withheld_results.csv'. As we have used binary classification models, each prediction score is a scalar value between 0 and 1 that represents the probability of the corresponding compound being active (antibacterial or cytotoxic), given as an average value across the trained models.
31. Calculate model performance metrics for the withheld set. There are several standard methods to assess performance. Two common metrics are the AUPRC and the AUROC, and these can be readily computed using scikit-learn. For instance, the AUPRC can be calculated as follows:

```
import csv
from sklearn.metrics import precision_recall_curve, auc
y_scores = []
myReader = csv.reader(open("./sa_models_benchmarking/withheld_results.
csv", 'r'),delimiter=',')
for row in myReader:
 try:
  y_scores.append(float(row[1]))
 except:
  pass
y_true = []
myReader = csv.reader(open("./sa_models_benchmarking/withheld.csv",
'r'),delimiter=',')
for row in myReader:
 try:
  y_true.append(float(row[1]))
 except:
  pass

precision, recall, _ = precision_recall_curve(y_true, y_scores)
print(auc(recall, precision))
```

Here, the second columns of 'withheld.csv' and 'withheld_results.csv' are expected to be the ground-truth activity values ('0' or '1') and prediction scores for the compounds in the withheld test set, respectively. The SMILES strings of these compounds are expected in the first column, and each row should correspond to the same compound in both .csv files.
If a numerical prediction score threshold is specified for declaring compounds as predicted to be active and inactive, then metrics that depend on this prediction score threshold can be calculated. These metrics are based on the confusion matrix and include accuracy, sensitivity and positive predictive value.

32. The metrics calculated in Step 31 represent an estimate of the performance that can be expected for the model trained on the full training dataset. If this performance is favorable, then proceed to Step 33; otherwise, repeat Steps 25–31 with different Chemprop model parameters to determine if performance can be improved. Generally, an AUPRC value greater than the baseline fraction of active compounds in the training data and an AUROC value greater than 0.5 indicate that the models perform better than random.
33. Using the Chemprop models trained on all the training data, run predictions on a larger test database using the following command:

```
chemprop_predict --test_path test.csv --checkpoint_dir checkpoints
--preds_path test_results.csv
```

As above in Step 29, 'test.csv' contains a user-provided column of SMILES strings, with each SMILES string corresponding to a compound in the test set. If RDKit features were used to supplement training, remember to add the same flags during predictions to the command above:

```
--features_generator rdkit_2d_normalized --no_features_scaling
```

■ **PAUSE POINT** Depending on the size of the test database, making predictions can take days. For the 100,000-compound test database provided as Supplementary Dataset 5 in this protocol, predictions should take ~1 h to 1 d on a standard personal computer. The prediction results are saved in the file 'test_results.csv'.
■ **PAUSE POINT** This step concludes the training and testing of one ensemble of Chemprop models. Subsequent models can be developed independently at your own pace.
◆ **TROUBLESHOOTING**

34. Repeat Steps 23–33 for each of the remaining .csv training dataset files to generate a total of four ensembles of Chemprop models, where each ensemble predicts *S. aureus* antibacterial activity, HepG2 cytotoxicity, IMR-90 cytotoxicity or HSkM cell cytotoxicity.
35. Define predicted hits as compounds with prediction scores above (or below) a user-specified numerical threshold. The choice of threshold depends on the models, and we recommend choosing a threshold resulting in $\lesssim$1% of all test database compounds being predicted as hits. For a quantitatively informed choice, the prediction score threshold may also be selected to correspond to specific points on the precision–recall or receiver operating characteristic curves generated in Step 31.
For this protocol, antibiotic activity prediction scores >0.5 are suitable to define active antibacterial compounds, and cytotoxicity prediction scores <0.2, <0.05 and <0.2 are suitable to define noncytotoxicity compounds across all three human cell types (Fig. 4a–d). Note that the models more strongly predict compounds with higher prediction scores to be active, and the prediction scores can be used to quantitatively rank predicted hits.

Stage 3, rationale analysis and filtering

● **TIMING** 1 h to 1 week

▲ **CRITICAL** This stage is optional and can be skipped if filtering or model explainability is not required. If model explainability is not required, the model predictions made in Stage 2 should be viewed as black box predictions that can be experimentally validated in Stage 4.

36. Filter the predicted hits. For the discovery of drug candidates, various rule-based guidelines have been proposed to help researchers narrow down compounds that are drug-like. This may include filtering out compounds with pan-assay interference substructures (PAINS) or Brenk substructures, which refer to chemical substructures that may be promiscuous or toxic[75,76]. To do so, pass a .csv file of SMILES strings corresponding to shortlisted hits ('hits.csv') into an RDKit filter using Python as follows:

```
import csv
from rdkit import Chem
from rdkit.Chem.FilterCatalog import FilterCatalog,
FilterCatalogParams
hits = []
myReader = csv.reader(open("./hits.csv", 'r'),delimiter=',')
for row in myReader:
  mol = Chem.MolFromSmiles(row[0])
  if mol is not None:
    hits.append(mol)

params = FilterCatalogParams()
params.AddCatalog(FilterCatalogParams.FilterCatalogs.PAINS)
params.AddCatalog(FilterCatalogParams.FilterCatalogs.BRENK)
catalog = FilterCatalog(params)
filtered_hits = []
for i in hits:
  entry = catalog.GetFirstMatch(i)
  if entry is None:
    # Collect hits without PAINS or Brenk substructures
    filtered_hits.append(i)
```

Calculations for additional properties, such as conformity to Lipinski's rule of five[77], can be similarly implemented using Python and RDKit.

37. (Optional) Prioritize compounds that are structurally dissimilar from the training set. This enables researchers to discover new chemotypes instead of known chemotypes that are readily inferred on the basis of the training data, and users may find this step useful if the number of filtered hits remaining from Step 36 is large. To filter out compounds that are structurally similar to compounds in the training set, one may generate Morgan fingerprints for all compounds and calculate the maximum Tanimoto similarity score between each hit and all compounds in the training set. If the training set consists of the SMILES strings in the first column of 'train.csv', then this is performing by running the following Python and RDKit code:

```
from rdkit import DataStructs
from rdkit.Chem import AllChem
train = []
myReader = csv.reader(open("./train.csv", 'r'),delimiter=',')
for row in myReader:
  mol = Chem.MolFromSmiles(row[0])
  if mol is not None:
    fingerprint = AllChem.GetMorganFingerprintAsBitVect(mol,2,nBi
ts=2048)
  train.append(fingerprint)

tanimoto = []
for mol in filtered_hits:
  fingerprint = AllChem.GetMorganFingerprintAsBitVect(mol,2,nBits=2048)
```

```
    max_similarity = 0
    for train_fingerprint in train:
      tanimoto_similarity = DataStructs.FingerprintSimilarity(fingerprint,
train_fingerprint)
      if tanimoto_similarity > max_similarity:
        max_similarity = tanimoto_similarity
    tanimoto.append(max_similarity)

filtered_hits_tanimoto=[filtered_hits[i] for i in range(len(tanimoto))
if tanimoto[i] < 0.5]
```

Here, the 'tanimoto' array is a list of maximum similarity scores corresponding to the list of compounds in 'filtered_hits'. To begin, we recommend removing compounds with similarity scores ≥0.5 as shown.

38. (Optional) Visualize chemical space. Chemical space visualization can be performed using various methods—including *t*-stochastic neighborhood embedding (*t*-SNE), principal components analysis (PCA) or uniform manifold approximation and projection on molecular fingerprints—and may help users identify structurally novel compounds if desired. Here, we use *t*-SNE on a Morgan fingerprint representation of the compounds as one possible approach, by running the following command:

```
import numpy as np
import pandas as pd
from sklearn.manifold import TSNE
import seaborn as sns
import matplotlib.pyplot as plt

# Read and classify molecules
def read_molecules(file_path):
  train_pos, train_neg = [], []
  with open(file_path, 'r') as f:
    reader = csv.reader(f, delimiter=',')
    for row in reader:
      mol = Chem.MolFromSmiles(row[0])
      if mol:
        if float(row[1]) == 1:
          train_pos.append(mol)
        else:
          train_neg.append(mol)
  return train_pos, train_neg

# Calculate fingerprint array
def calc_fp_arr(mols):
  fplist = []
  for mol in mols:
    arr = np.zeros((1,))
    fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)
    DataStructs.ConvertToNumpyArray(fp, arr)
    fplist.append(arr)
  return np.asarray(fplist)

# Load data
train_pos, train_neg = read_molecules("./sa_models/train.csv")
```

```
# Combine all data
alldata = [train_neg, train_pos, hits, filtered_hits_tanimoto]
flatalldata = [item for sublist in alldata for item in sublist]
alllengths = [len(data) for data in alldata]

# Calculate fingerprints and apply TSNE
res = calc_fp_arr(flatalldata)
model = TSNE(n_components=2, init="pca", perplexity=30,
metric='jaccard')
x = model.fit_transform(res)
df1 = pd.DataFrame(x, columns=['x1', 'y1'])

# Split data
lengths = np.cumsum(alllengths)
lengths = np.insert(lengths, 0, 0)
dfs = [df1[lengths[i]:lengths[i+1]] for i in range(len(lengths)-1)]

# Plot data
colors = ['blue', 'red', 'yellowgreen', 'darkgreen']
sizes = [20, 80, 40, 20]

plt.figure(figsize=(10, 10))
for df, color, size in zip(dfs, colors, sizes):
 plt.scatter(df['x1'], df['y1'], Color=color, s=size)
plt.show()
```

Key parameters—such as the fingerprint radius (2), fingerprint number of bits (2,048), *t*-SNE perplexity (30) and choice of initialization (PCA)—will affect the visualization, but the default values for these parameters shown are recommended. For further details regarding these parameters, consult the accompanying sklearn documentation.
As an example of this generated output, Fig. 4e,f shows visualizations of the chemical space covered by different training sets, which includes the chemical library screened as part of Stage 1 and compounds having high model prediction scores from Stage 2.

39. Run the 'interpret' function, which uses the MCTS algorithm to identify rationales, as follows:

```
chemprop_interpret --data_path hits.csv --checkpoint_dir sa_models/
checkpoints --property_id 1 --prop_delta 0.5 > hits_rationales.csv
```

The '–prop_delta 0.5' flag is optional but provides the syntax for adjusting the 'prop_delta' parameter, which specifies the minimum Chemprop score that any rationale must have. Here, 'hits.csv' contains a column of SMILES strings, with each SMILES string corresponding to a compound in the (filtered) set of hits resulting from Step 36 or 37. As we are calculating model rationales for high antibacterial activity prediction scores, 'checkpoints' should point to the models predicting antibacterial activity. If RDKit features were used to supplement training, remember to add the same flags to the command above:

```
--features_generator rdkit_2d_normalized --no_features_scaling
```

In the above command, '–property_id 1' tells Chemprop to explain the first property (which, for the models developed in this protocol, is antibacterial activity). This flag should be specified because Chemprop also can perform multi-task training, which is not covered in this protocol. The implementation of 'chemprop_interpret' outputs results directly to the command line, and we specify these results to be saved in 'hits_rationales.csv'.

▲ **CRITICAL STEP** Due to the MCTS algorithm (Box 4), 'chemprop_interpret' runs can be slow to converge and require substantial (>20 GB) computer memory. The command also processes molecules sequentially, which can be problematic if the MCTS is not converging. In these cases, we recommend splitting up the filtered hits to as few as one SMILES string per file and running the above command individually for each of the files. Doing so will typically require vastly more computational resources than that supported by a common personal computer. For the best experience, we recommend running these large batch runs on supercomputing platforms such as Google Cloud or those offered by universities and institutions. Runs that do not converge within several days of computation might indicate that it is difficult to find a suitable rationale, and these runs can be manually killed.

■ **PAUSE POINT** For each molecule, running 'chemprop_interpret' might take over 10 h on a typical laptop computer and use over 10 GB of memory. Thus, for users with larger sets of molecules or more limited computational resources, we recommend running the command on a supercomputing platform such as Google Cloud.

◆ **TROUBLESHOOTING**

40. (Optional) Advanced users may adjust additional parameters of Chemprop's 'interpret' function, which requires Chemprop to be rebuilt ('Troubleshooting' section). This optional step is helpful if Step 39 consistently fails to produce outputs. Chemprop does not currently allow all parameters to be modified via command line, so open the file 'chemprop/args.py' in a text editor and scroll to the following section:

```
class InterpretArgs(CommonArgs):
 """:class:`InterpretArgs` includes :class:`CommonArgs` along with
additional arguments used for interpreting a trained Chemprop model."""

 data_path: str
 """Path to data CSV file."""
 batch_size: int = 500
 """Batch size."""
 property_id: int = 1
 """Index of the property of interest in the trained model."""
 rollout: int = 20
 """Number of rollout steps."""
 c_puct: float = 10.0
 """Constant factor in MCTS."""
 max_atoms: int = 20
 """Maximum number of atoms in rationale."""
 min_atoms: int = 8
 """Minimum number of atoms in rationale."""
 prop_delta: float = 0.5
 """Minimum score to count as positive."""
```

The most important variable to consider for this protocol is 'prop_delta', which can also be changed from the command line as in Step 39 above. As the default setting of 0.5 (also used above in Step 39) indicates that any rationale must result in a Chemprop score of at least 0.5, prop_delta may be too large if our prediction score threshold is 0.2, and users may change the 0.5 to 0.1 (or another suitably low number) instead. Additionally, the number of rollout steps can optionally be decreased from 20 to 10 to decrease runtime and the exhaustiveness of the MCTS. For different test sets and applications, verify that the values for 'min_atoms' and 'max_atoms', which may also be specified as inputs on the command line (Table 4), are suitable. These values provide the range for the number of atoms that any rationale must have. We do not recommend altering the other parameters to begin.

▲ **CRITICAL STEP** Despite the name, Chemprop's 'interpret' function does not make models interpretable in the sense described above and in Box 2. Instead, it aids in making models explainable by identifying chemical substructure rationales of interest.

41. Visualize results of chemprop_interpret. The MCTS algorithm allows for the rationales of structurally similar hits to vary, often slightly. Finding the scaffolds that are common to rationales allows us to define key conserved substructures of interest. To do so, one can visually inspect the rationales as follows, assuming that the rationale SMILES outputs are now in the first column of the file 'rationales.csv':

```
from rdkit.Chem import Draw
rationales = []
myReader = csv.reader(open("./rationales.csv", 'r'),delimiter=',')
for row in myReader:
  rationale = Chem.MolFromSmiles(row[0])
  if rationale is not None:
    rationales.append(rationale)
Draw.MolsToGridImage(rationales)
```

Example rationales are visualized in Fig. 4g,h.

42. (Optional) Systematically identify common scaffolds to delineate 'rationale groups', which may be helpful when the rationales exhibit minor variations due to the stochasticity of the MCTS search process. One way to identify common scaffolds is with RDKit's rdFMCS function, which is a flexible maximal common substructure (MCS) algorithm. Running the following command on two rationales, 'rationale_1' and 'rationale_2', will output the maximal common substructure where bonds are equivalent if and only if they have the same bond type and complete rings only are compared (if an atom is part of the MCS and the atom is in a ring of the entire molecule, then that atom is also in a ring of the MCS):

```
from rdkit.Chem import rdFMCS
this_MCS = Chem.MolFromSmarts(rdFMCS.FindMCS([rationale_1,rationale_2],
bondCompare=rdFMCS.BondCompare.CompareOrderExact,completeRingsOnly=True).
smartsString)
```

If desired, this code can be replicated across all pairs of rationales.

43. (Optional) Assign compounds to the rationale groups. This can be done visually as above or computationally using the HasSubstructMatch function in RDKit as follows:

```
associated_mols = []
for mol in filtered_hits:
if mol.HasSubstructMatch(test_MCS):
  associated_mols.append(mol)
```

Here, test_MCS can be any MCS from Step 42, and filtered hits are contained as RDKit molecules in the 'filtered_hits' array.

44. Compile the list of hits and associated rationale groups, if applicable. If the same rationale group contains multiple hits, then these groups define predicted structural classes of active compounds.

PAUSE POINT This concludes the computational component of this protocol. Subsequent experimental testing in Stage 4 can be done at your own pace.

Stage 4, prediction testing

TIMING 1 d to 1 week

45. Procure or synthesize shortlisted compounds. We recommend searching for previously-deposited organic compounds using their SMILES strings on PubChem (https://pubchem.ncbi.nlm.nih.gov/), ChemSpider (https://www.chemspider.com/) or CAS's SciFinder^n (https://scifinder-n.cas.org/), as each database provides commercial vendor information (if available).

46. Repeat Steps 1–20 to experimentally test the compounds for the desired activity. For organic compounds that are supplied in powder form, we recommend dissolving the compound in DMSO to make stock solutions of at least 10 mM.

Troubleshooting

Troubleshooting advice can be found in Table 5. To guide troubleshooting, a working example of the files provided as inputs and created as outputs of this protocol is available at https://github.com/felixjwong/protocol. This repository also contains a Python notebook that walks users through all analyses downstream of Chemprop, as described in Stages 2–4 of this protocol.

Installation and execution of Chemprop:
As mentioned above, we have focused on using Chemprop v1 in this protocol. Chemprop can be downloaded and installed from GitHub or using the PyPi, and reports of installation problems are few. Should issues during installation or execution of Chemprop arise, refer to Table 5.

Table 5 | Troubleshooting table

Step	Problem	Possible reason	Possible solution
20	Inadequate or biased spread of resazurin fluorescence values	Incubation time is too short, results in uneven heating across the wells of a plate	Incubate longer with resazurin for up to 2–3 d. Avoid stacking plates in the incubator and ensure 100% humidity in the incubator to mitigate evaporation
24	Cannot activate Chemprop	Chemprop is not installed	Install Chemprop according to the requirements of this protocol
24	Error in installing Chemprop	Python and Miniconda dependencies are not available. Unsuitable hardware setup or no access to the Internet	Verify that the operating system supports Python. Download and install Miniconda as instructed in the 'Equipment setup' section. Check for proper hardware (as detailed in the 'Hardware requirements' section) and ensure that the computer is connected to the Internet
25, 26, 29, 33, 39	Error in Chemprop runs	The input files are not in the correct format; invalid flags specified	Check that all SMILES strings as well as the input file are correct. Consider using Supplementary Datasets 1–5 to start
25, 26, 29, 33, 39	Chemprop producing different results on different computers, even with the same models and datasets	The versions of Chemprop are different	Check the version and Github commit number of the Chemprop software being used. Ensure that the Chemprop setups are consistent by downloading and installing a specific commit number of Chemprop, if needed
29, 33, 39	'ValueError: If scaling of the additional features was done during training, the same must be done during prediction.' Message	The model was trained using the '–features_generator rdkit_2d_normalized –no_features_scaling' flags, which were not specified during prediction, or vice-versa	Include the '–features_generator rdkit_2d_normalized –no_features_scaling' flags as needed
39	The chemprop_interpret command hangs	An unsuitable choice of MCTS parameters or nonconvergence of the MCTS	Ensure that the parameters in chemprop/args.py are suitable and that the edited file is being used in the Chemprop installation. Continue running the command or skip the problematic compound
39	The computer crashes or unexpectedly shuts down while executing chemprop_interpret	Insufficient memory resources	If the error is caused by insufficient memory resources, run the command on a server with more computational resources, such as Google Cloud
39	'Can't keukulize mol.' Message	Ambiguity in aromaticity of compounds as part of the MCTS	These messages can be ignored
39	'OSError: [Errno 24] Too many open files' error	Too many files are being generated by chemprop_interpret than can be handled by the operating system	Run 'ulimit -n 2048' on the command line, then rerun the chemprop_interpret command
39	The values specified for MCTS are not being reflected in chemprop_interpret runs	Chemprop is not updated with the latest values in args.py	Rerun the chemprop_interpret command from the Chemprop directory but replace 'chemprop_interpret' with 'python interpret.py'. Alternatively, install Chemprop from source and rebuild the Chemprop environment by running 'conda env remove -n chemprop' on the command line, followed by 'conda env create -f environment.yml' from the Chemprop directory, then 'conda activate chemprop', then 'pip install -e .'

Timing

Stage 1, data generation (Steps 1–20): 1 d to 1 week
Stage 2, model training and benchmarking (Steps 21–35): 1 h to 1 week
Stage 3, rationale analysis and filtering (Steps 36–44): 1 h to 1 week
Stage 4, prediction testing (Steps 45 and 46): 1 day to 1 week
These times reflect estimates for a drug discovery program centering on the identification of antibacterial compounds, assuming typical computational resources, training libraries on the order of 10^2–10^5 compounds, test libraries on the order of 10^4–10^8 compounds, and that compounds of interest are experimentally tested. For larger libraries and the discovery of compounds that require different experimental procedures, the required time may be longer.

Anticipated results

Expected inputs and outputs at each stage of the protocol are summarized in Fig. 3, example outputs of the protocol are illustrated in Fig. 4, and an illustration of the MCTS is provided in Fig. 5. The data files used for Chemprop are detailed in Box 5. Specifically, this protocol results in the generation of the following files at each stage:

Stage 1, data generation.
A list of chemical structures and associated activity values (for example, in a .csv file).

Stage 2, model training and benchmarking.
Model checkpoints (.pt files) and benchmarking results (for example, numerical AUPRC or AUROC values). Additional files generated as part of Chemprop model training may include directories, .log files (quiet.log and verbose.log), .json files (opt.json and args.json) and a .csv file (test_scores.csv).

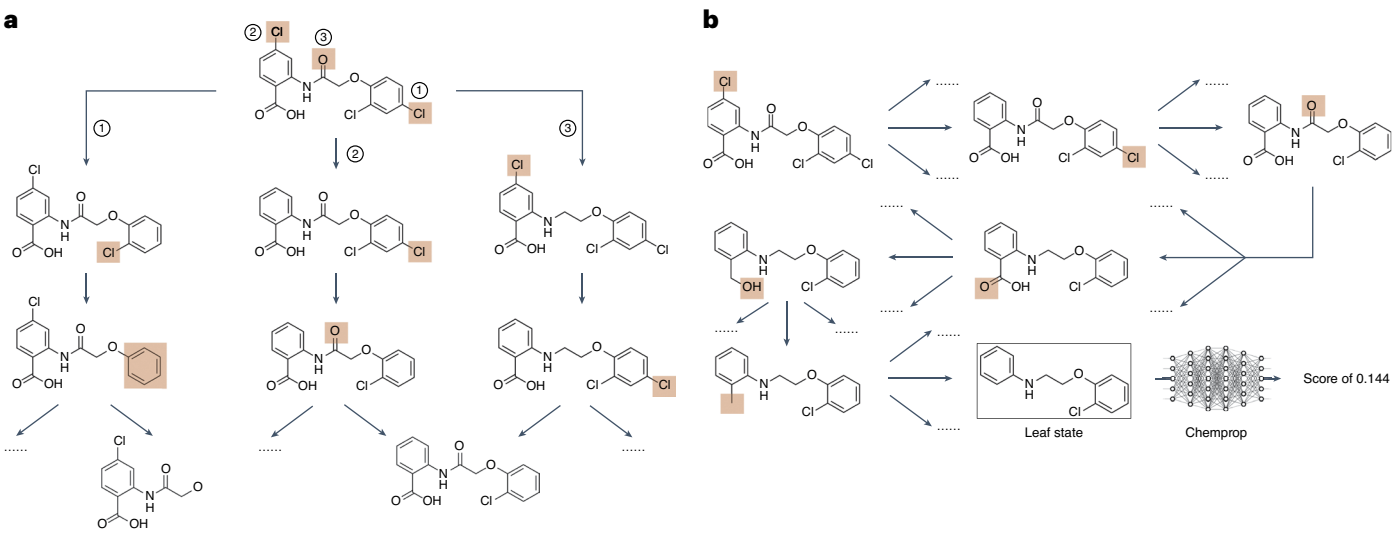


Fig. 5 | Illustration of the MCTS. a, Illustration of the MCTS forward pass, using compound **1** as shown in Fig. 4h. The figure shows three possible search paths from compound **1** by deleting correspondingly labeled peripheral bonds or rings (highlighted in orange). Due to space limitations, only three steps from the root are shown. **b**, Illustration of a complete MCTS search path from compound **1** to a rationale. Chemprop is used to predict the activity of each molecular substructure, and these predictions are used to make updates to the statistics of each intermediate state in the backward pass. The figure was reproduced from ref. 7, Springer Nature Ltd.

Stage 3, rationale analysis and filtering.
Prediction rationales and shortlisted compounds (that is, lists of SMILES strings tabulated in a .csv file).

Stage 4, prediction testing.
Experimental activity values for compounds of interest (for example, in a .csv file).

At the end of Stage 4, users will ideally have validated novel compounds with the activity of interest (here, selective antibacterial activity), which may represent structural classes of compounds according to the rationales generated from Stage 3. These compounds represent starting points for further development and should be validated using secondary assays and additional tests. If users are unable to validate novel compounds with the activity of interest, we recommend testing additional predicted hits and retraining the models using additional data, which can be done by repeating Stages 1–4 as needed.

Data availability

Example datasets are available as Supplementary Information. The main datasets used in this protocol are subsets of data from a previously published study (ref. 7) identifying a structural class of antibiotics using explainable DL.

Code availability

Chemprop is available at https://github.com/chemprop/chemprop. A working example of the files provided as inputs and created as outputs of this protocol is available at https://github.com/felixjwong/protocol. Additional code from a previously published study, which includes Chemprop checkpoints for models trained on larger datasets, are available at https://github.com/felixjwong/antibioticsai and https://zenodo.org/records/10095879 (ref. 78). The code in this protocol has been peer reviewed.

Received: 11 March 2024; Accepted: 26 September 2024;
Published online: 9 December 2024

References

1. Wong, F. et al. Leveraging artificial intelligence in the fight against infectious diseases. *Science* **381**, 164–170 (2023).
2. Wan, F., Wong, F., Collins, J. J. & de la Fuente-Nunez, C. Machine learning for antimicrobial peptide identification and design. *Nat. Rev. Bioeng.* **2**, 392–407 (2024).
3. Silver, D. et al. Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).
4. Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).
5. Baek, M. et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science* **373**, 871–876 (2021).
6. Lin, Z. et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
7. Wong, F. et al. Discovery of a structural class of antibiotics with explainable deep learning. *Nature* **626**, 177–185 (2024).
8. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
9. Bengio, Y., Lodi, A. & Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *Eur. J. Oper. Res.* **290**, 405–421 (2021).
10. Bohacek, R. S., McMartin, C. & Guida, W. C. The art and practice of structure-based drug design: a molecular modeling perspective. *Med. Res. Rev.* **16**, 3–50 (1996).
11. Yang, X., Wang, Y., Byrne, R., Schneider, G. & Yang, S. Concepts of artificial intelligence for computer-assisted drug discovery. *Chem. Rev.* **119**, 10520–10594 (2019).
12. Burbidge, R., Trotter, M., Buxton, B. & Holden, S. Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Comput. Chem.* **26**, 5–14 (2001).
13. Warmuth, M. K. et al. Active learning with support vector machines in the drug discovery process. *J. Chem. Inf. Comput. Sci.* **43**, 667–673 (2003).
14. Zernov, V. V., Balakin, K. V., Ivaschenko, A. A., Savchuk, N. P. & Pletnev, I. V. Drug discovery using support vector machines. The case studies of drug-likeness, agrochemical-likeness, and enzyme inhibition predictions. *J. Chem. Inf. Comput. Sci.* **43**, 2048–2056 (2003).
15. Sadybekov, A. V. & Katritch, V. Computational approaches streamlining drug discovery. *Nature* **616**, 673–685 (2023).
16. Yang, K. et al. Analyzing learned molecular representations for property prediction. *J. Chem. Inf. Model.* **59**, 3370–3388 (2019).
17. Stokes, J. M. et al. A deep learning approach to antibiotic discovery. *Cell* **180**, 688–702 (2020).
18. Liu, G. et al. Deep learning-guided discovery of an antibiotic targeting *Acinetobacter baumannii*. *Nat. Chem. Biol.* **19**, 1342–1350 (2023).
19. Zheng, E. J. et al. Discovery of antibiotics that selectively kill metabolically dormant bacteria. *Cell. Chem. Biol.* **31**, 712–728.e9 (2024).
20. Melo, M. C. R., Maasch, J. R. M. A. & de la Fuente-Nunez, C. Accelerating antibiotic discovery through artificial intelligence. *Commun. Biol.* **4**, 1050 (2021).
21. Cesaro, A., Bagheri, M., Torres, M., Wan, F. & de la Fuente-Nunez, C. Deep learning tools to accelerate antibiotic discovery. *Expert Opin. Drug Discov.* **18**, 1245–1257 (2023).
22. Krishnan, S. R. et al. De novo design of anti-tuberculosis agents using a structure-based deep learning method. *J. Mol. Graph. Model.* **118**, 108361 (2023).
23. Wong, F. et al. Discovering small-molecule senolytics with deep neural networks. *Nat. Aging* **3**, 734–750 (2023).
24. Jin, W. et al. Deep learning identifies synergistic drug combinations for treating COVID-19. *Proc. Natl Acad. Sci. USA* **118**, e2105070118 (2021).
25. Preuer, K. et al. DeepSynergy: predicting anti-cancer drug synergy with deep learning. *Bioinformatics* **34**, 1538–1546 (2018).
26. Wan, F., Kontogiorgos-Heintz, D. & de la Fuente-Nunez, C. Deep generative models for peptide design. *Digit. Discov.* **1**, 195–208 (2022).
27. De Cao, N. & Kipf, T. MolGAN: an implicit generative model for small molecular graphs. In *ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models* (2018).
28. Gómez-Bombarelli, R. et al. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.* **4**, 268–276 (2018).

29. Jin, W., Barzilay, R. & Jaakkola, T. In *Proc. 35th International Conference on Machine Learning* 2323–2332 (2018).
30. Blaschke, T. et al. REINVENT 2.0: an AI tool for de novo drug design. *J. Chem. Inf. Model.* **60**, 5918–5922 (2020).
31. Zhou, Z., Kearnes, S., Li, L., Zare, R. N. & Riley, P. Optimization of molecules via deep reinforcement learning. *Sci. Rep.* **9**, 10752 (2019).
32. Zeng, X. et al. Deep generative molecular design reshapes drug discovery. *Cell Rep. Med.* **3**, 100794 (2022).
33. Zhavoronkov, A. et al. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. *Nat. Biotechnol.* **37**, 1038–1040 (2019).
34. Ying, R., Bourgeois, D., You, J., Zitnik, M. & Leskovic, J. GNNExplainer: generating explanations for graph neural networks. *Adv. Neural Inf. Process. Syst.* **32**, 9240–9251 (2019).
35. Jiménez-Luna, J., Grisoni, F. & Schneider, G. Drug discovery with explainable artificial intelligence. *Nat. Mach. Intell.* **2**, 573–584 (2020).
36. Yuan, H., Yu, H., Gui, S. & Ji, S. Explainability in graph neural networks: a taxonomic survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **45**, 5782–5799 (2023).
37. Yuan, H., Yu., H., Wang, J., Li, K. & Ji, S. In *Proc. 38th International Conference on Machine Learning* 12241–12252 (2021).
38. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **1**, 206–215 (2019).
39. Gilmer, J. et al. In *Proc. 34th International Conference on Machine Learning* 1263–1272 (2017).
40. Wu, Z. et al. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **32**, 2162–2388 (2021).
41. Zhou, J. et al. Graph neural networks: a review of methods and applications. *AI Open* **1**, 57–81 (2020).
42. Reiser, P. et al. Graph neural networks for materials science and chemistry. *Commun. Mater.* **3**, 93 (2022).
43. Heid, E. & Green, W. H. Machine learning of reaction properties via learned representations of the condensed graph of reaction. *J. Chem. Inf. Model.* **62**, 2101–2110 (2022).
44. Jin, W., Barzilay, R. & Jaakkola, T. In *Proc. 37th International Conference on Machine Learning* 4849–4859 (2020).
45. Heid, E. et al. Chemprop: a machine learning package for chemical property prediction. *J. Chem. Inf. Model.* **64**, 9–17 (2024).
46. Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo Tree Search. In *Computers and Games (CG 2006). Lecture Notes in Computer Science* (eds van den Herik, H. J. et al.) **4630**, 72–83 (Springer, 2007)
47. Tingle, B. I. et al. ZINC-22—a free multi-billion-scale database of tangible compounds for ligand discovery. *J. Chem. Inf. Model.* **63**, 1166–1176 (2023).
48. Verheij, H. J. Leadlikeness and structural diversity of synthetic screening libraries. *Mol. Divers.* **10**, 377–388 (2006).
49. Krier, M., Bret, G. & Rognan, D. Assessing the scaffold diversity of screening libraries. *J. Chem. Inf. Model.* **46**, 512–524 (2006).
50. Swanson, K. et al. ADMET-AI: a machine learning ADMET platform for evaluation of large-scale chemical libraries. *Bioinformatics* **40**, btae416 (2024).
51. McGill, C., Forsuelo, M., Guan, Y. & Green, W. H. Predicting infrared spectra with message passing neural networks. *J. Chem. Inf. Model.* **61**, 2594–2609 (2021).
52. Swinney, D. C. & Anthony, J. How were new medicines discovered. *Nat. Rev. Drug Discov.* **10**, 507–519 (2011).
53. Swinney, D. C. Phenotypic vs. target-based drug discovery for first-in-class medicines. *Clin. Pharmacol. Ther.* **93**, 299–301 (2013).
54. Moffat, J. G., Vincent, F., Lee, J. A., Eder, J. & Prunotto, M. Opportunities and challenges in phenotypic drug discovery: an industry perspective. *Nat. Rev. Drug Discov.* **16**, 531–543 (2017).
55. Muratov, E. N. et al. QSAR without borders. *Chem. Soc. Rev.* **49**, 3525–3564 (2020).
56. Wong, F. et al. Benchmarking AlphaFold-enabled molecular docking predictions for antibiotic discovery. *Mol. Syst. Biol.* **18**, e11081 (2022).
57. Bender, B. J. et al. A practical guide to large-scale docking. *Nat. Protoc.* **16**, 4799–4832 (2021).
58. Loyola-González, O. Black-box vs. white-box: understanding their advantages and weaknesses from a practical point of view. *IEEE Access* **7**, 154096–154113 (2019).
59. Clinical and Laboratory Standards Institute. *M100: Performance Standards for Antimicrobial Susceptibility Testing* (2021).
60. Zhang, J. H., Chung, T. D. & Oldenburg, K. R. A simple statistical parameter for use in evaluation and validation of high throughput screening assays. *J. Biomol. Screen.* **4**, 67–73 (1999).
61. Kim, S. et al. PubChem substance and compound databases. *Nucleic Acids Res.* **44**, D1202–D1213 (2016).
62. Gaulton, A. et al. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Res.* **40**, D1100–D1107 (2012).
63. Degtyarenko, K. et al. ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Res.* **36**, D344–D350 (2008).
64. Wishart, D. S. et al. DrugBank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Res.* **34**, D668–D672 (2006).
65. Williams, A. J. et al. The CompTox Chemistry Dashboard: a community data resource for environmental chemistry. *J. Cheminform.* **9**, 61 (2017).
66. Bergerhoff, G., Hundt, R., Sievers, R. & Brown, I. D. The inorganic crystal structure data base. *J. Chem. Inf. Comput. Sci.* **23**, 66–69 (1983).
67. Belsky, A., Hellenbrandt, M., Karen, V. L. & Luksch, P. New developments in the Inorganic Crystal Structure Database (ICSD): accessibility in support of materials research and design. *Acta Cryst.* **58**, 364–369 (2022).
68. Kononova, O. et al. Text-mined dataset of inorganic materials synthesis recipes. *Sci. Data* **6**, 203 (2019).
69. Shivanyuk, A., Ryabukhin, S. V., Bogolubsky, A. V. & Tolmachev, A. Enamine REAL database: making chemical diversity real. *Chem. Today* **25**, 58–59 (2007).
70. Coley, C. W., Green, W. H. & Jensen, K. F. Machine learning in computer-aided synthesis planning. *Acc. Chem. Res.* **51**, 1281–1289 (2018).
71. Fink, T., Bruggesser, H. & Reymond, J.-L. Virtual exploration of the small-molecule chemical universe below 160 Daltons. *Angew. Chem. Int. Ed.* **44**, 1504–1508 (2005).
72. Fink, T. & Reymond, J.-L. Virtual exploration of the chemical universe up to 11 atoms of C, N, O, F: assembly of 26.4 million structures (110.9 million stereoisomers) and analysis for new ring systems, stereochemistry, physicochemical properties, compound classes, and drug discovery. *J. Chem. Inf. Model.* **47**, 342–353 (2007).
73. Blum, L. C. & Reymond, J.-L. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.* **131**, 8732–8733 (2009).
74. Ruddigkeit, L., van Deursen, R., Blum, L. C. & Reymond, J.-L. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *J. Chem. Inf. Model.* **52**, 2864–2875 (2012).
75. Baell, J. B. & Holloway, G. A. New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. *J. Med. Chem.* **53**, 2719–2740 (2010).
76. Brenk, R. et al. Lessons learnt from assembling screening libraries for drug discovery for neglected diseases. *ChemMedChem* **3**, 435–444 (2008).
77. Lipinski, C. A., Lombardo, F., Dominy, B. W. & Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug. Dis. Rev.* **23**, 3–25 (1997).
78. Wong, F. et al. Supporting code for: discovery of a structural class of antibiotics with explainable deep learning. *Zenodo* https://doi.org/10.5281/zenodo.10095879 (2023).
79. Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM J.* **3**, 211–229 (1959).
80. Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65**, 386–408 (1958).
81. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
82. Krizhensky, A., Sutskever, I. & Hinton, G. E. In *Advances in Neural Information Processing Systems* 1106–1114 (2012).
83. Vaswani, A. et al. In *Advances in Neural Information Processing Systems* (2017).
84. Trinh, T. H., Wu, Y., Le, Q. V., He, H. & Luong, T. Solving olympiad geometry without human demonstrations. *Nature* **625**, 476–482 (2024).
85. Lundberg, S. M. and Lee, S.-I. In *Proc. 31st International Conference on Neural Information Processing Systems* 4768–4777 (2017).
86. Ribeiro, M. T., Singh, S. & Guestrin, C. In *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 1135–1144 (2016).
87. Dai, H., Dai, B. & Song, L. In *Proc. 33rd International Conference on Machine Learning* 2702–2711 (2016).
88. Buterez, D., Janet, J. P., Kiddle, S. J., Oglic, D. & Lió, P. Transfer learning with graph neural networks for improved molecular property prediction in the multi-fidelity setting. *Nat. Commun.* **15**, 1517 (2024).
89. Xie, T., France-Lanord, A., Wang, Y., Shao-Horn, Y. & Grossman, J. Y. Graph dynamical networks for unsupervised learning of atomic scale dynamics in materials. *Nat. Commun.* **10**, 2667 (2019).
90. Yun, S., Jeong, M., Kim, R. Kang, J. & Kim, H. J. In *33rd Conference on Neural Information Processing Systems* 11983–11993 (2019).

Acknowledgements

F.W. was supported by the National Institute of Allergy and Infectious Diseases of the National Institutes of Health under award number K25AI168451. A.K. was supported by the Swiss National Science Foundation under grant number SNSF_203071. J.J.C. was supported by the Defense Threat Reduction Agency (grant numbers HDTRA12210032 and HDTRA12210010), the National Institutes of Health (grant number R01-AI146194) and the Broad Institute of MIT and Harvard. This work is part of the Antibiotics-AI Project, which is directed by J.J.C. and supported by the Audacious Project, Flu Lab, LLC, the Sea Grape Foundation, Rosamund Zander and Hansjorg Wyss for the Wyss Foundation and an anonymous donor.

Author contributions

F.W. prepared the manuscript and supervised research. S.O., A.L., A.K., R.S.L., J.R. and M.Z.W. contributed to writing and validating the protocol steps. J.J.C. supervised research. All authors assisted with manuscript editing.

Competing interests

J.J.C. is an academic cofounder and Scientific Advisory Board chair of EnBiotix, an antibiotic drug discovery company and Phare Bio, a nonprofit venture focused on antibiotic drug development. J.J.C. is also an academic cofounder and board member of Cellarity and the founding Scientific Advisory Board chair of Integrated Biosciences. F.W. and M.Z.W. are cofounders of Integrated Biosciences. S.O., A.L. and R.S.L. contributed to this work as employees of Integrated Biosciences, and S.O. and R.S.L. may have equity interest in Integrated Biosciences. The remaining authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41596-024-01084-x.

Correspondence and requests for materials should be addressed to James J. Collins.

Peer review information *Nature Protocols* thanks Octavio Franco and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.