Advancing Drug-Target Interactions Prediction: Leveraging a Large-Scale Dataset with a Rapid and Robust Chemogenomic Algorithm

Gwenn Guichaoua,[*,†,‡,¶] Philippe Pinel,[†,‡,¶,§] Brice Hoffmann,[§] Chloé-Agathe Azencott,[†,‡,¶] and Véronique Stoven[†,‡,¶]

[†]*Center for Computational Biology (CBIO), Mines Paris-PSL, 75006 Paris, France*

[‡]*Institut Curie, Université PSL, 75005 Paris, France*

[¶]*INSERM U900, 75005 Paris, France*

[§]*Iktos SAS, 75017 Paris, France*

E-mail: gwenn.guichaoua@minesparis.psl.eu

Abstract

Predicting drug-target interactions (DTIs) is crucial for drug discovery, and heavily relies on supervised learning techniques. In the context of DTI prediction, supervised learning algorithms use known DTIs to learn associations between molecule and protein features, allowing for the prediction of new interactions based on learned patterns. In this paper, we present a novel approach addressing two key challenges in DTI prediction: the availability of large, high-quality training datasets and the scalability of prediction methods. First, we introduce LCIdb, a curated, large-sized dataset of DTIs, offering extensive coverage of both the molecule and druggable protein spaces. Notably, LCIdbcontains a much higher number of molecules, expanding coverage of the molecule space compared to traditional benchmarks. Second, we propose Komet (Kronecker Optimized METhod), a DTI prediction pipeline designed for scalability without compromising performance. Komet leverages a three-step framework, incorporating efficient computation choices tailored for large datasets and involving the Nyström approximation. Specifically, Komet employs a Kronecker interaction module for (molecule, protein) pairs, which is sufficiently expressive and whose structure allows for reduced computational complexity. Our method is implemented in open-source software, leveraging GPU parallel computation for efficiency. We demonstrate the efficiency of our approach on various datasets, showing that Komet displays superior scalability and prediction performance compared to state-of-the-art deep-learning approaches. Additionally, we illustrate the generalization properties of Komet by showing its ability to solve challenging scaffold-hopping problems gathered in the publicly available $\mathcal{LH}$ benchmark. Komet is available open source at `https://komet.readthedocs.io`.

1 Introduction

Most marketed drugs are small molecules that interact with a protein, modulating its function to prevent the progression of a disease. Therefore, the development of computational methods for the prediction of drug-target interactions (DTIs) has been an active field of research in the last decades, intending to reduce the number of wet-lab experiments to be performed for solving various problems related to drug discovery.

Among current computational approaches, we focus on chemogenomic DTI prediction methods, i.e. methods that predict whether a (molecule, protein) pair interacts or not, based on known DTIs in a reference database of interactions. In the present paper, we formulate DTI prediction as a classification problem: (molecule, protein) pairs are classified as interacting (i.e. positive examples, labelled $+1$) or not interacting (i.e. negative examples, labelled -1). Chemogenomic methods offer a global framework to predict drugs' protein interaction profiles, or proteins' drug interaction profiles, at large scales both in the molecule and protein spaces, which cannot be performed by other methods (mainly QSAR and docking approaches) directly. Therefore, chemogenomic methods allow for tackling important problems in drug design. In particular, the prediction of drugs' protein profiles allows the prediction of deleterious off-targets responsible for unwanted side-effects and potentially leading to drug withdrawal or beneficial off-targets that may be of interest to treat other diseases thus offering drug repositioning opportunities. Moreover, the prediction of proteins' drug interaction profiles is an interesting tool to solve scaffold hoping problems in the context of drug design.[1]

Enhancing the performance of DTI predictions requires to use of ever-larger training datasets and the development of Machine-Learning (ML) algorithms capable of scaling to these dataset sizes. In this paper, we tackle these challenges by presenting a curated large-sized dataset LCIdb and Komet, a GPU-friendly DTI prediction pipeline. These two components complement each other, resulting in state-of-the-art performance.

2 State-of-the-art in chemogenomic approaches

Most chemogenomic DTI prediction methods rely on the global framework comprising three main steps and presented in Figure 1. Therefore, we present a short review of state-of-the-art approaches used in these three steps.

2.1 Step 1: Feature representations for proteins and molecules

Various methods[2] have been designed to compute feature representations for proteins and molecules. For molecules, several types of features are considered, as discussed in recent papers.[3,4] They can globally be classified into: (1) string-based formats such as the Simplified Molecular-Input Line-Entry System[5] (SMILES), or the International Chemical Identifier[6] (InChI); (2) table-based formats that represent the chemical graph of the molecule such as the sdf format;[7] (3) feature-based formats that consist in vectors whose elements encode various molecular characteristics. They include Morgan fingerprints, Extended-connectivity

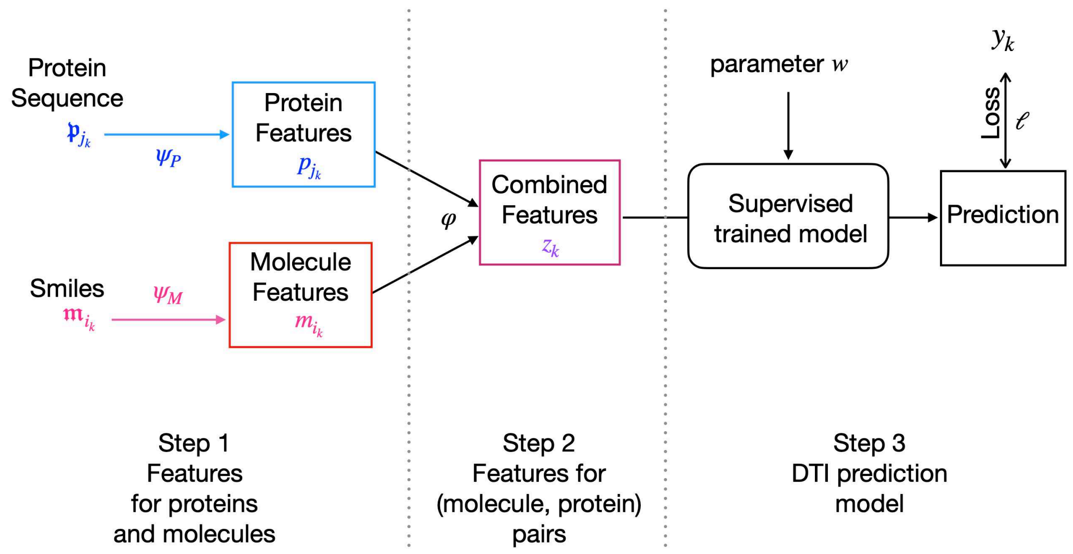

Figure 1: Global framework for DTI prediction in 3 key steps

fingerprints[8] (ECFP), or 2D and 3D pharmacophore fingerprints as described in the RD-Kit toolbox;[9] (4) computer-learned representations that are derived by neural networks and used to encode molecules in deep-learning approaches. These representations can be learned from recurrent neural networks (RNNs) or convolutional neural networks (CNNs) that use SMILES representations as input, as seen in Lee et al and Zhao et al.[10,11] Graph convolution networks have also been applied to 2D molecular graphs to learn small molecule representations,[12,13] and strategies to pre-trained Graph Neural Networks have been studied by Hu et al.[14] to compute molecule embeddings. Similar to natural language models, Mol2vec[15] and SMILES2vec[16] adapt the principles of the "word2vec" method[17] to learn embeddings for molecular structures. Additionally, transformer-based models like MolTrans[18] have emerged in this domain. Finally, other learned representation methods such as X-Mol[19] or MolGNet[20] use AutoEncoder (AE) techniques for molecular representation.

Similarly, proteins can globally be described by: (1) string-based representations corresponding to their primary sequence of amino-acids; (2) vector-based feature representations, where the elements of the vector are calculated according to various characteristics, as reviewed in Zhu et al.[21]. Such representations include composition, transition, and distribution (CTD) descriptors that are classically used;[22] (3) computer-learned representations derived by neural networks in deep-learning approaches. In this context, protein features can be acquired by a variety of deep learning architectures, including recurrent neural networks (RNNs) or convolutional neural networks (CNNs),[10,11] as well as transformer models.[18] As in natural language models, protein embeddings can also be learned from pre-trained transformer-based models on external tasks such as ESM2,[23] or auto-encoder models such as ProtBert[24] and ProtT5XLUniref50.[24]

2.2 Step 2: Features for (molecule, protein) pairs

The second step of many DTI prediction pipelines consists of defining a representation for (molecule, protein) pairs, thus defining a latent space for pairs. The method that is used to define this latent space has a critical impact on the prediction performance, and a key aspect is that the features representing the (molecule, protein) pair should capture information about the interaction, which is not fully achieved by simple concatenation between molecule and protein features[25] . Therefore, step 2 usually consists of a non-linear mixing of the protein and molecule embeddings, to better encode information about interaction determinants. One common approach is to use the tensor product, which is equivalent to a Kronecker kernel.[26,27] Alternatively, in deep-learning methods, the features for pairs can be learned from an interaction module that consists of fully connected multi-layer perceptrons (MLPs).[10,28–30] Attention mechanisms applied to molecule and protein features constitute another option.[11,18,31] Then, the last layer of the network can be used to define features for the (molecule, protein) pairs.

2.3 Step 3: DTI prediction model

The third step consists of a supervised classifier that is trained in the latent space of (molecule, protein) pairs, using a training dataset of positive and negative DTIs. These classifiers include tree-based methods,[32] and network-based inference approaches.[33] In linear models, step 3 consists of the optimization of the weights applied to the pair features calculated in step 2, according to a logistic loss or a hinge loss for Support Vector Machines (SVM).[34] For example, the various articles[27,35–37] rely on a linear model on a latent representation of pairs. In deep-learning chemogenomic algorithms, step 3 relies on the pair features determined by the last layer of the neural network in step 2. The features' weights are optimized based on a loss function, typically binary cross-entropy, as the input progresses through the network in a feed-forward manner. For instance, this approach is used in several recent papers[10,11,18,28–31] proceed in this way.

2.4 Issues in chemogenomic studies

Although different chemogenomic approaches have been proposed, as briefly reviewed above, all require a training dataset of positive and negative (molecule, protein) pairs. Recent ML chemogenomic algorithms have often been trained on small to medium-sized benchmarks that present various biases. Indeed, most classical benchmark datasets are extracted from a single biological database, and often favour drug and target families that have been more widely studied, and for which many known DTIs have been recorded[38,39] . Additionally, Bagherian et al.[40] highlights that most datasets use negative DTIs randomly chosen among pairs with unknown interaction status, and may therefore include false negative DTIs. One suggestion to overcome this problem is to derive training datasets from interaction databases that compile continuous values for binding affinities and choose stringent activity thresholds to derive confident positive and negative pairs, as suggested by Wang et al.[41].

In addition, training chemogenomic models that are broadly applicable and can generalize to many different families of proteins and drugs requires training on very large, high-quality, verified and well-established DTI datasets. This appears to be an important bottleneck since publicly available training datasets that meet these criteria are seldom.

However, training ML algorithms on very large datasets, potentially comprising hundreds of thousands of molecules and DTIs, leads to challenges in terms of computation times and memory requirements. In particular, the choice of the interaction module in step 2 has significant implications for computation time and memory resources in large-sized datasets. In the case of deep-learning approaches, the complexity of neural network architectures, and the size of parameter spaces, may also contribute to the computational expense. Learning the interaction module requires iteratively adjusting the model parameters, leading to time-consuming training phases.

Overall, there is a critical need for chemogenomic approaches that can scale to very large datasets.

3 Contributions

In the present paper, we tackle two important issues mentioned above:

- in Section 4.2, we propose the Large Consensus Interaction dataset, called LCIdb hereafter, a new very large and high-quality dataset of DTIs that was designed to train chemogenomic ML algorithms for DTI prediction at large scale in the protein and molecule spaces. In particular, our dataset comprises a much larger number of molecules than commonly used datasets, offering a better coverage of the chemical space. Additionally, we paid attention to limiting potential bias among negative DTIs.
- in Sections 4.3 and 4.4, we propose Komet (Kronecker Optimized METhod), a simple yet efficient DTI prediction method that lies within the global pipeline presented in Figure 1. This method incorporates specific computation choices that provide scalability for very large training datasets, without compromising prediction performance.

We show that Komet competes with or outperforms state-of-the-art deep-learning approaches for DTI prediction on medium-sized datasets, but that it scales much better to very large datasets in terms of prediction performances, computation time, and memory requirements (see Section 5.4).

Finally, we illustrate the performance of Komet trained on LCIdb using DrugBank as an external dataset for DTI prediction, and on a publicly available benchmark[42] designed to evaluate the performance of prediction algorithms in solving difficult scaffold hoping problems.[1]

Komet adopts the global three-step framework shown in Figure 1, which aligns with recent computational pipelines, such as in Huang et al.[28]. However, Komet includes specific choices whose principles are presented below, while mathematical details are provided in Materials and Methods.

In step 1, molecule (resp. protein) features ψ_M (resp. ψ_P) are computed based on the distances of the considered molecule (resp. protein) to molecules in the training set, thus leveraging ideas from kernel methods. More precisely, from a small set of reference landmark molecules extracted for the training dataset, we use the Nyström approximation and dimensionality reduction to efficiently compute embeddings ψ_M and ψ_P that approximate the features derived from the chosen kernels. The parameters of the method are the numbers m_M (resp. m_P) of molecule (resp. protein) landmarks, and the dimension d_M (resp d_P) of the molecule (resp. protein) embeddings. The impact of these parameters is studied in Section 5.2.

In step 2, the interaction module consists of the tensor product between the protein and molecule spaces. One of the motivations for using the tensor product is that it offers a systematic way to encode correlations between molecules and protein features, independently from the choice of these features. A potential issue with this approach, however, is that the size of the resulting vector representation for the (molecule, protein) pair equals $d_M \times d_P$, and may be prohibitively large for computation time and memory. However, a classical property of tensor products is their factorization between inner products between the two tensor product vectors of molecules and proteins, called the Kronecker product. This avoids the explicit calculation of the interaction embedding, thus addressing the challenges posed by large datasets. Overall, as shown in Section 5, we found that this tensor product representation efficiently captured information about features interactions that govern the (molecule, protein) binding.

In step 3, Komet uses a simple SVM loss together with a BFGS optimization algorithm. This allows to leverage the Kronecker factorization of pairs' features, leading to a significant speedup of the training. It is important to note that, in the proposed approach, steps 2 and 3 are executed simultaneously. This is made possible by avoiding the implicit calculation of pairs' features, thanks to the Kronecker interaction module.

Our method is implemented in an open source software, leveraging parallel computation on GPU through a PyTorch[43] interface, and is available at `https://github.com/Guichaoua/komet`.

4 Materials and Methods

We first recall known and publicly available medium-sized DTI datasets that are used in the present paper (Section 4.1), and describe the building of our large-sized DTI dataset LCIdb (Section 4.2). Then, we detail our computational approach for large-sized DTI prediction with Komet (Sections 4.3 and 4.4), and present the methodology used to compare the performance of Komet to those of a few state-of-the-art deep-learning algorithms (Section 4.5). Finally, we introduce ($\mathcal{LH}$), a publicly available benchmark dataset to assess the performances of computational methods to solve scaffold hoping problems and used in the present study.

4.1 Medium-scale datasets

We first use medium-scale datasets to compare the performance of Komet to those of state-of-the-art algorithms: BIOSNAP, BIOSNAP_Unseen_drugs, BIOSNAP_Unseen_proteins, BindingDB, and DrugBank. The four first of these datasets are publicly available and were established in Huang et al.[18]. They are used in various recent studies.[28,44] We also used the DrugBank-derived dataset established in Najm et al.[45], from which we built an additional so-called DrugBank (Ext) to be used as an external dataset, as detailed below.

The authors from Huang et al.[18] and Singh et al.[44] proposed to train and compare the performance of various DTI prediction algorithms based on splitting the datasets in training (Train), validation (Val), and test (Test) sets according to a 7:1:2 ratio. Therefore, to make fair comparisons, we followed these schemes. The number of drugs, targets, and interactions for all datasets used in the present study is given in Table 1. In addition, the number of positive and negative interactions across the training (Train), validation (Val), and test (Test) sets for all datasets used in the present paper is detailed in Table 2.

BIOSNAP in its three prediction scenarios The ChGMiner dataset from BIOSNAP[46] contains exclusively positive DTIs. Negative DTIs are generated by randomly selecting an equal number of positive DTIs, assuming that a randomly chosen (molecule, protein) pair is unlikely to interact. We considered three scenarios, as proposed in Huang et al.[18]. The first scenario is referred to as "BIOSNAP", and corresponds to random splitting of the DTIs. In the BIOSNAP_Unseen_targets scenario, the Train and Test sets do not share any protein. The BIOSNAP_Unseen_drugs dataset follows a similar process for molecules. The two last scenarios allow us to evaluate the generalization properties of the algorithm on proteins or molecules that were not seen during training.

BindingDB-derived dataset The BindingDB database, referenced in[47], stores (molecule, protein) pairs with measured bioactivity data. We used a dataset derived from BindingDB and introduced by Huang et al.[18], where BindingDB is filtered to include only pairs with known dissociation constants (Kd). Pairs with Kd $<$ 30 nM are considered positive DTIs, while those with Kd $>$ 30 nM values are considered negative. This leads to a much larger number of negative DTIs than positive DTIs. Although the resulting dataset does not include the whole BindingDB database, for the sake of simplicity, it will be called BindingDB hereafter.

DrugBank-derived datasets We used the dataset provided in Najm et al.[45]. This dataset was built by filtering drug-like molecules and human protein targets in the Drug-Bank database, adding an equal number of negative DTIs through balanced sampling. More precisely, to avoid bias towards well-studied proteins for which many interactions are known, negative examples are randomly chosen among unlabeled DTIs in such a way as to ensure that each protein and each drug appear an equal number of times in positive and negative interactions, using the greedy algorithm from Najm et al.[45]. This dataset will be referred to as DrugBank in the following, for the sake of simplicity, and corresponds to the dataset called DrugBank (S1) in the original paper.

We created another dataset called DrugBank (Ext), derived from the above dataset, and used it as an external validation to compare the prediction performances of the considered algorithms when trained on BindingBD or on LCIdb. Positive interactions from DrugBank were selected, excluding those present in BindingDB and LCIdb, to gather a set of positive DTIs absent from the BindingDB and LCIdb datasets. All other DTIs in DrugBank are kept in DrugBank (Ext). As above, balanced negative interactions were added in DrugBank (Ext), as proposed in Najm et al.[45].

4.2 Building the new large scale dataset LCIdb

To build a large-sized dataset of DTIs, we started from the database described by Isigkeit et al.[48], as it combines and curates data from prominent databases including ChEMBL[49] , PubChem[50] , IUPHAR/BPS[51] , BindingDB[52] , and Probes & Drugs.[53] We filtered the DTIs in this database according to 4 filters, as detailed below.

Filtering positive DTIs : (1) Chemical structure quality filter: for DTIs present in several of the source databases, we only retained those for which the SMILES representation of the molecule was identical in all sources, to exclude potential erroneous (molecule, protein) pairs. We only kept molecules with molecular weights between 100 and 800 g.mol^{-1}, which is a standard choice for selecting drug-like molecules. Among these molecules, we selected those that target at least one human protein. These filters were used because the goal was to build a training dataset of DTIs that are relevant in the context of drug discovery projects.

(2) Bioactivity filter: we retained only DTIs for which inhibition constant Ki, dissociation constant Kd, or half maximal inhibitory concentration IC50 measurements were available in at least one of the source databases.

(3) Quantitative bioactivities filter: for DTIs with bioactivity measurements present in multiple source databases, we only retained those whose bioactivities were within one log unit from one another.

(4) Binary labelling of DTIs: Bioactivity measurements were converted into binary interactions based on a threshold. If the bioactivity value was less than 100 nM (10^{-7}M), the interaction was classified as positive DTI (binding). If the bioactivity value (Ki, Kd or IC50) was greater than 100μM (10^{-4}M), the interaction was classified as negative DTI (non-binding). When the bioactivity value was in the margin, i.e. between 100 nM and 100μM, DTIs were classified as known non-conclusive.

This scheme leads to the selection of 274 515 molecules, 2 069 proteins, 402 538 positive interactions and 8 296 negative interactions. We then added negative interactions to build a balanced dataset.

Completion of a balanced negative DTI dataset: We randomly split the dataset into training (Train), validation (Val), and testing (Test) sets in a 7:1:2 ratio. We use unlabeled DTIs to include negative interactions to these three sets, assuming most unknown DTIs are negative. For the Train set the selection of additional negative interactions should be designed with care to tackle two classical issues: (1) reduce the number of false negative DTIs present in the training set; (2) correct potential statistical bias in the database towards highly studied molecules or proteins. To take into account the former, we excluded known non-conclusive interactions, and for the latter, we applied the algorithm by Najm et al.[45]

for selecting additional negative DTIs. In the Val and Test sets, remaining negative and randomly chosen unknown interactions are added. These sets form LCIdb, mirroring the DrugBank dataset scenario discussed in section 4.1.

Different prediction scenarios: To evaluate performance in different prediction scenarios, we also derive different datasets from to LCIdbbased on specific splits of the Train, Val, and Test sets, as proposed in Huang et al.[18] and Singh et al.[44]. Datasets are built to correspond to LCIdb, LCIdb_Unseen_drug, LCIdb_Unseen_protein, and LCIdb_Orphan (unseen molecule and protein) scenarios. We added the Orphan case, which presents the greater difficulty for prediction tasks.

More precisely: (1) LCIdb is balanced in positive and negative pairs chosen at random; (2) LCIdb_Unseen_drugsis built so that (molecule, protein) pairs in one of the Train/Val/Test sets only contain molecules that are absent from the two other sets; (3) LCIdb_Unseen_targets is built so that (molecule, protein) pairs in one of the Train/Val/Test sets only contain proteins that are absent from the two other sets; (4) LCIdb_Orphan is built so that (molecule, protein) pairs in one of the Train/Val/Test sets only contain proteins and molecules that are absent from the two other sets. The number of drugs, targets, and interactions in these four datasets is given in Table 1. Table 2 provides the number of positive and negative interactions across the Train, Val, and Test sets in these four datasets.

4.3 Features for proteins and molecules in Komet

The initial step of our DTI prediction framework consists of computing simple and fixed features for molecules and proteins.

Nyström-based molecule and protein features ψ_M and ψ_P in Komet: In Komet, we encode molecules and proteins leveraging the Nyström approximation[54,55] and dimensionality reduction. For a molecule $\mathfrak{m}$ (for instance, represented as a SMILES string), let us explain how we compute its embedding $\psi_M(\mathfrak{m})$ in $\mathbb{R}^{d_M}$. The same computation applies for the protein embedding $\psi_P(\mathfrak{p}) \in \mathbb{R}^{d_P}$ (where $\mathfrak{p}$ is for instance a FASTA string). ψ_M is built from a small set of landmark molecules $\{\hat{\mathfrak{m}}_\ell\}_{\ell=1}^{m_M}$ with $m_M \geq d_M$ that are randomly chosen in the training dataset. The other ingredient in Komet is a kernel $k_M(\mathfrak{m}, \mathfrak{m}')$ that can be viewed as a similarity measure between two molecules, and that is used to define molecule features (the choice of this kernel is discussed below). We first compute the small kernel matrix $\hat{K}_M \in \mathbb{R}^{m_M \times m_M}$ where $(\hat{K}_M)_{i,j} \coloneqq k_M(\hat{\mathfrak{m}}_i, \hat{\mathfrak{m}}_j)$. Then, we define the extrapolation matrix $E \in \mathbb{R}^{m_M \times d_M}$ from the Singular Value Decomposition of $\hat{K}_M = U \operatorname{diag}(\sigma) U^\top$ as $E \coloneqq U[:, : d_M] \operatorname{diag}(\sigma_s^{-1/2})_{s=1}^{d_M}$. The molecule embedding is then

$$\psi_M(\mathfrak{m}) \coloneqq \left(\sum_{\ell=1}^{m_M} E_{\ell,s}\, k_M(\hat{\mathfrak{m}}_\ell, \mathfrak{m}) \right)_{s=1}^{d_M} \in \mathbb{R}^{d_M}.$$

Note that when no dimensionality reduction is performed ($d_M = m_M$), this embedding satisfies the relation $k_M(\hat{\mathfrak{m}}_i, \hat{\mathfrak{m}}_j) = \langle \psi_M(\hat{\mathfrak{m}}_i), \psi_M(\hat{\mathfrak{m}}_j) \rangle$ (see Appendix C for details). In addition, for any molecule $\mathfrak{m}$ that is not in the landmark set, $k_M(\mathfrak{m}, \hat{\mathfrak{m}}_i) \approx \langle \psi_M(\mathfrak{m}), \psi_M(\hat{\mathfrak{m}}_i) \rangle$ through a Nyström approximation (see Appendix C for details). Hence, E allows us to "extrapolate" the embedding ψ_M, which is the underlying kernel map of k_M, from the landmarks to new molecules. Finally, we mean-center and normalize the features:

$$\psi_M(\mathfrak{m}) \leftarrow \frac{\psi_M(\mathfrak{m}) - \bar{m}}{\|\psi_M(\mathfrak{m}) - \bar{m}\|} \quad \text{where} \quad \bar{m} \coloneqq \frac{1}{m_M} \sum_{\ell=1}^{m_M} \psi_M(\mathfrak{m}_\ell).$$

We adopt a similar approach to build ψ_P but use all proteins from the data set as landmarks, as their number is much smaller. Again, because the number of proteins is small enough, we do not apply dimensionality reduction: $d_P = m_P = n_P$.

Choice of molecule and protein kernels: The embeddings ψ_M and ψ_P depend on the choice of molecule and protein kernels. We follow the choices made in Playe et al.[37] and adopt the Tanimoto kernel k_M for molecules. For each molecule $\mathfrak{m}$ represented in SMILES format, we calculate Morgan fingerprints with a radius of 2, generating a 1024-bit binary vector using the RDKit package[9] . Values of the Tanimoto kernel between two molecules are then computed as the Jaccard index between their fingerprints. The Tanimoto kernel measures the similarity between two molecules based on the substructures they share, based on fingerprints. For each protein represented as a sequence $\mathfrak{p}$ of amino acids, we opt for the Local Alignment kernel (LAkernel)[56] . This kernel k_P detects remote homology by aggregating contributions from all potential local alignments with gaps in the sequences, thereby extending the Smith–Waterman score.[57] The hyperparameters were adjusted by cross-validation[37] .

4.4 Large-scale chemogenomic framework with Komet

We address DTI prediction as a supervised binary classification problem, incorporating established steps, as outlined in Sections 2.2 and 2.3.

Features for molecule-protein pairs: Let us consider a DTI dataset containing molecules and proteins $(\mathfrak{m}_i)_{i=1}^{n_M}$ and $(\mathfrak{p}_j)_{j=1}^{n_P}$, where n_M and n_P are respectively the number of molecules and proteins in the dataset. To alleviate notations, in what follows, we denote by $m \coloneqq \psi_M(\mathfrak{m})$ the embedding of a molecule $\mathfrak{m}$ and by $p \coloneqq \psi_P(\mathfrak{p})$ the embedding of a protein $\mathfrak{p}$.

The training dataset consists of a set of n_Z (molecule, protein) pairs with indices $(i_k, j_k)_{k=1}^{n_Z}$ and their associated labels $y_k \in \{-1, 1\}$. If $y_k = 1$ (resp. -1), molecule $\mathfrak{m}_{i_k}$ and protein $\mathfrak{p}_{j_k}$ interact (resp. don't interact). The classification is performed in the space of pairs, which we define as the tensor product of the space of molecules and the space of proteins. Hence, the embedding for pairs is given by $\varphi(\mathfrak{m}, \mathfrak{p}) \coloneqq (m[s] \times p[t])_{1 \le s \le d_M, 1 \le t \le d_P} \in \mathbb{R}^{d_Z}$, where $m[s]$ is the s-th coordinate of m and $p[t]$ is the t-th coordinate of p.

Thus, the space of pairs has dimension $d_Z = d_M \times d_P$. This embedding corresponds to the use of a Kronecker kernel, already shown to be efficient in several publications.[27,37,45] Using a Kronecker kernel is crucial in our approach, not only because it is a state-of-the-art method, but also due to its favourable mathematical properties, which we will detail below.

It is worth noting that our approach avoids explicitly calculating the embedding φ, which mitigates the computational burden associated with the large value of d_Z.

SVM classification: Our classification approach follows previous work (see Section 2.3), relying on a linear model with weight vector $w \in \mathbb{R}^{d_Z}$ and bias term $b \in \mathbb{R}$. The class decision for a pair feature vector $z \in \mathbb{R}^{d_Z}$ is determined by $\operatorname{sign}(\langle w, z \rangle + b) \in \{-1, 1\}$. The parameters w and b are obtained by minimizing a penalized empirical risk:

$$\min_{w \in \mathbb{R}^{d_Z}} \sum_{k=1}^{n_Z} \ell(\langle w, z_k \rangle + b, y_k) + \frac{\lambda}{2} \|w\|^2. \qquad (1)$$

In Komet, we employ a Support Vector Machine (SVM) classification where $\ell(y', y) = \max(0, 1 - yy')$.

The minimization of Equation (1) is computationally demanding, particularly when n_Z and d_Z are large. A conventional Stochastic Gradient Descent (SGD)[58] can result in slow convergence. Therefore, we use an alternative approach that leverages the specific structure of our embedding φ, as was previously done in Airola and Pahikkala[59] . Specifically, we exploit: (1) the tensor product nature of φ and (2) the fact that the sizes n_M and n_P of the input databases are much smaller than the number n_Z of interactions.

Efficient computation The core ingredient leading to a significant improvement in computational efficiency on a large-sized dataset is the efficient computation of the gradient by bypassing the evaluation of φ. Indeed, the function to be minimized in Equation (1) has the form $L(Zw + b) + \frac{\lambda}{2}\|w\|^2$, where the rows of $Z \in \mathbb{R}^{n_Z \times d_Z}$ are the vectors $z_k^\top$, and L takes into account ℓ and y. The main computational burden for evaluating this function and its gradient is the computation of Zw. A naive implementation would require $n_Z d_Z$ operations just to compute Z, which would be unavoidable if one used a generic φ, such as a deep neural network. However, we bypass this bottleneck by directly computing Zw. This relies on the following identity:

$$(Zw)_k = \langle w, z_k \rangle_{\mathbb{R}^{d_Z}} \overset{(a)}{=} \langle m_{i_k}, W p_{j_k} \rangle_{\mathbb{R}^{d_M}} \overset{(b)}{=} \langle m_{i_k}, q_{j_k} \rangle_{\mathbb{R}^{d_M}}, \tag{2}$$

where $W \in \mathbb{R}^{d_M \times d_P}$ is such that it has w as flattened representation in $\mathbb{R}^{d_Z}$ and $q_j := W p_j$.

Equality (a) exploits the tensor product structure of φ. Please refer to the Appendix D for a detailed proof.

Equality (b) is interesting because all the $(q_j)_{j=1}^{n_P}$ can be computed in only $n_P d_Z$ operations. Once this has been computed, evaluating all n_Z values of $(Zw)_k = \langle m_{i_k}, q_{j_k} \rangle_{\mathbb{R}^{d_M}}$ require $n_Z d_M$ operations. We then minimize Equation (1) using a full batch method, which enables the use of efficient quasi-Newton methods. In practice, we use the BFGS method with limited memory[60] (refer to chapter 6 of this book). The complexity of our algorithm is then $\mathcal{O}(n_P d_Z + n_Z d_M)$ where $\mathcal{O}(.)$ takes into account the number of iterations of the BFGS algorithm to reach a fixed accuracy. This number is quite small (10 to 50) in our numerical experiments. Note that we can exchange the role of the protein embeddings and the molecule embeddings in this calculation, resulting in a complexity of $O(n_M d_Z + n_Z d_P)$. In our setting $n_P \ll n_M$ so we prefer the initial formulation of Equation (2).

From classification to probability estimation Once the weight w has been computed, Platt scaling[61] computes a probability using the formula

$$p_k := \sigma(-y_k(s\langle z_k, w\rangle + t)) \text{ where } \sigma(u) = \frac{e^u}{1+e^u},$$

where the scale s (level of confidence in some sense) and offset t needs to be optimized. They are found by minimizing the same energy as the logistic classification.

$$\min_{s,t} E(s,t) := \sum_k \ell(-y_k(s\langle z_k, w\rangle + t)) = L(-\operatorname{diag}(y)(sm + t)),$$

where $\ell(u) := \log(1 + e^u)$ and $m := Zw$. We use the BFGS method to estimate s and t.

4.5 Evaluation of prediction performance

Comparing the prediction performances of various algorithms requires defining the evaluation strategies and the metrics used.

Metrics: We formulate the DTI prediction problem as a classification task, therefore, we use AUPR (area under the precision–recall curve), ROC-AUC (area under the ROC curve) and prediction accuracy, as metrics to compare prediction performances.

Evaluation strategies: There is only one hyperparameter in our model, as shown in Equation (1). We select the best $\lambda \in \{10^{-11}, 10^{-10}, ..., 10, 100\}$ based on AUPR performance from the validation (Val) set. This value is used to train the parameters of the model 5 times on the training set, each time with new landmark molecules and approximated molecule features, and we calculate the mean prediction probability. The final computed model is then evaluated on the Test set.

Implementation details: We use a server with 2 CPUs and 1 NVIDIA A40 GPU with 48 GB of memory. We provide a Python implementation of Komet and the code used to build LCIdb at `https://Komet.readthedocs.io`.

4.6 Application to the scaffold hopping problem

To assess computational methods for solving large-step scaffold hopping problems, Pinel et al.[42] built a high-quality benchmark called Large-Hops ($\mathcal{LH}$) comprising 143 pairs of highly dissimilar molecules that are active against diverse protein targets. In $\mathcal{LH}$, one active molecule is considered as known, and the second active molecule must be retrieved among 499 decoys carefully selected to avoid statistical bias.

For each case, the considered algorithms were trained with one molecule of the pair considered as the only known active for the query protein. If the known interaction was absent from the training dataset, it was added to it, and all other interactions involving the query protein potentially present in the database were removed. After training, the algorithms rank the unknown active and the 499 decoy molecules, according to the predicted binding probabilities of the (molecule, query protein) pairs. The lower the rank of the unknown active, the better the prediction performance.

Three criteria are employed to compare prediction algorithms: (1) Cumulative Histogram Curves (CHC) are drawn to represent the number of cases where a method ranks the unknown active below a given rank, with better-performing methods having curves above others; (2) Area Under the Curve (AUC) of CHC curves provide a global quantitative assessment of the methods; (3) the proportion of cases where the unknown active was retrieved in the top 1% and 5% best-ranked molecules.[62]

5 Results

In the following, we first present the new LCIdb DTI dataset, analyze its coverage of the molecule and protein spaces, and compare it to other available and widely used datasets. Next, we explore different parameters within the Komet pipeline, to find a balance between speed and prediction performance. We then show that Komet displays state-of-the-art DTI prediction performance capabilities on the considered medium- and large-sized datasets, and on the DrugBank (Ext) DTI dataset used as an external dataset (see Section 4.1 for a description of this dataset). Finally, we highlight the efficiency of our approach on the publicly available ($\mathcal{LH}$) benchmark dataset designed to address challenging scaffold hopping problems.

5.1 Coverage of the protein and molecule spaces in the LCIdb dataset

Different reviews introduce numerous biological databases that can be used to derive large-sized training datasets,[2,40] to best cover the protein and molecule spaces. Following Isigkeit et al.[48], we combine and filter curated data from prominent databases including ChEMBL[49] PubChem,[50] IUPHAR/BPS,[51] BindingDB[52] , and Probes & Drugs[53] , and built LCIdb, a large-sized high-quality DTI database, as detailed in Section 4.2. Table 1 provides the numbers of molecules, proteins, and interactions in all the DTI training datasets considered in the present study.

Table 1: Numbers of molecules, proteins, and positive/negative DTIs in the considered datasets. "random" indicates that negative DTIs were randomly chosen among unlabeled DTIs. "balanced" indicates that negative DTIs were randomly chosen among unlabeled DTIS, but in such a way that each protein and each drug appears in the same number of positive and negative DTIs.

Datasets	Molecules	Proteins	Positive DTIs	Negative DTIs
BIOSNAP	4,510	2,181	13,836	(13,647 random)
Unseen_drugs			13,836	(13,647 random)
Unseen_targets			13,836	(13,647 random)
BindingDB	7,161	1,254	9,166	23,435
DrugBank	4,813	2,507	13,715	(13,715 balanced)
DrugBank (Ext)	4,257	1,216	10,838	(10,838 balanced)
LCIdb	274,515	2,069	402,538	8,296 (+ 394,242 balanced)
Unseen_drugs	274,515	2,069	402,538	8,296 (+ 394,242 balanced)
Unseen_targets	232,018	2,069	431,011	8,296 (+ 422,715 balanced)
Orphan	143,255	2,069	151,690	8,296 (+ 143,394 balanced)

Table 1 reveals that DrugBank- or BIOSNAP-derived datasets and BindingDB share a few characteristics: their numbers of proteins are similar (in the range of one to two thousand), their numbers of molecules are modest (in the range of a few thousand), their number of known positive DTIs are similar (in the range of thousands). BindingDB contains true negative DTIs, while the DrugBank- or BIOSNAP-derived datasets use DTIs of unknown status as negative DTIs, randomly chosen for BIOSNAP-derived datasets, and randomly chosen in such a way that all molecules and proteins appear in the same number of positive and negative DTIs (labelled "balanced" in Table 1) for the Drugbank-derived datasets. Overall, these observations underline the need for a larger dataset, as required for chemogenomic studies. As shown in Table 1, LCIdb includes 40 times more molecules and 30 times more positive DTIs than the other considered datasets, the number of human proteins being in the same order of magnitude.

However, it is important to evaluate whether this larger number of molecules corresponds to better coverage of the chemical space and whether the different datasets are comparable in terms of biological space coverage. Indeed, the chemical space is estimated to be extremely large,[63] and efficient sampling of this space by the training dataset is expected to have a great impact on the generalization properties of the prediction models.

We use the t-SNE algorithm[64] on the molecule features ψ_M derived from the Tanimoto kernel, as defined in Section 4.3, to visualize the resulting high-dimensional molecular space in a two-dimensional space, thus facilitating analysis. Figure 2 shows that the LCIdb dataset not only contains a much larger number of molecules but that these molecules display more diversity concerning the t-SNE features than the BIOSNAP, DrugBank, and BindingDB datasets. While the representation in Figure 2 does not embrace the entire vast and unknown chemical space, LCIdb seems to provide a better overall sampling for t-SNE features. In addition, it shows that LCIdb also covers the chemical space more uniformly than the other datasets. Interestingly, Figure 2 highlights that the BIOSNAP dataset originates from DrugBank, displaying similar patterns of red clusters of molecules.

We also run the t-SNE algorithm based on Tanimoto features computed using an alternative set of molecule landmarks, and based on other molecule features. In all cases, results visualization confirmed the above conclusions that LCIdb presents a wider and more uniform coverage of the chemical space, underscoring their robustness. The corresponding results are shown in Figure 2 of the Appendix A.

In Isigkeit et al.[48], the authors analyzed the space formed by the five databases from which LCIdb originates. Specifically, they examined distributions of common drug-like fea-

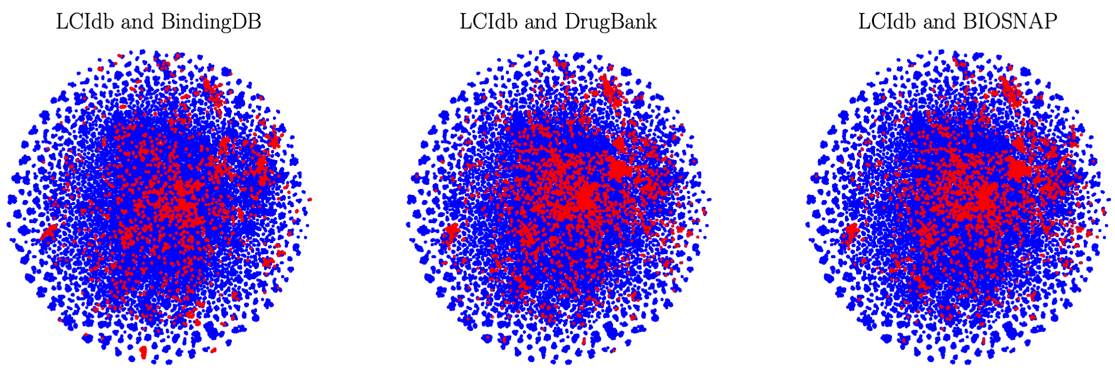

Figure 2: 2D representation of the molecular space with the t-SNE algorithm based on molecule features. In blue: the large-sized LCIdb dataset, in red: the medium-sized DrugBank, BIOSNAP, and BindingDB datasets.

tures such as molecular weight, the number of aromatic bonds, the number of rotatable bonds, and predicted octanol-water partition coefficients. The authors observed that these distributions are similar across all sources. In Appendix A, we present plots illustrating the distribution of drugs in our LCIdb dataset, based on the five databases from which they originate.

By contrast, the number of human proteins is comparable across all considered datasets, although not identical (see Figure 3). We also used t-SNE plots based on protein features defined in Section 4.3 to explore the coverage of the protein space by LCIdb . As shown in the resulting 2D representation presented in Figure 4, the protein space covered by LCIdb contains clusters that align with functional families of proteins. This was expected when using features calculated using the LAkernel (see Section 4.3), since proteins that share high sequence similarity usually belong to the same protein family. Thus, we can leverage this representation to discuss the diversity of proteins in our datasets. As shown in Figure 5, although LCIdb contains slightly fewer proteins than the DrugBank dataset, their coverage of the biological space is similar. BIOSNAP appears to have a lower coverage of a few protein clusters (such as protein kinases), while BindingDB focuses more on a few clusters corresponding to specific protein families.

As detailed in Section 4.1, for BIOSNAP and LCIdb, additional datasets are derived,

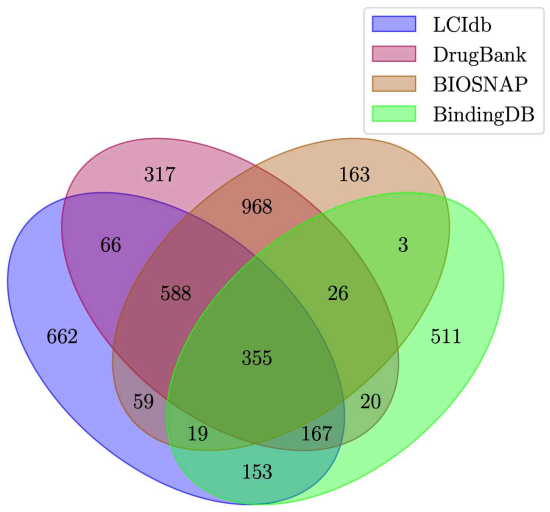

Figure 3: Overlap between LCIdb , DrugBank, BIOSNAP, and BindingDB datasets in terms of proteins.

as suggested in various studies,[10,11,27,37,65] as well as in Huang et al.[18] and Singh et al.[44], two papers that respectively introduced the MolTrans and ConPLex algorithms. They correspond to scenarios of varying difficulties encountered in real-life situations in drug discovery projects: (1) the Unseen_drugs case is typical of new drugs identified in phenotypic screen and for targets are searched to elucidate the drug's mechanism of action; (2) the Unseen_targets case is typical of newly identified therapeutic targets against for which repositioning opportunities if known drugs are searched; (3) The Orphan case is typical of a new therapeutic target has been identified, and against which ligands (inhibitors or activators) are searched at large scale in the molecule space.

The composition of the corresponding datasets is provided in Table 1. In Huang et al.[18] and Singh et al.[44], only the Unseen_drugs and Unseen_targets were considered, but we added the Orphan case for LCIdb.

Finally, following Huang et al.[18] and Singh et al.[44], in all the prediction experiments reported in the Results, the prediction performances of all considered algorithms are computed based on the Test set, after optimization of the parameters on the Train/Val sets built from the considered DTI datasets. Details about the Train/Val/Test sets are given in Section

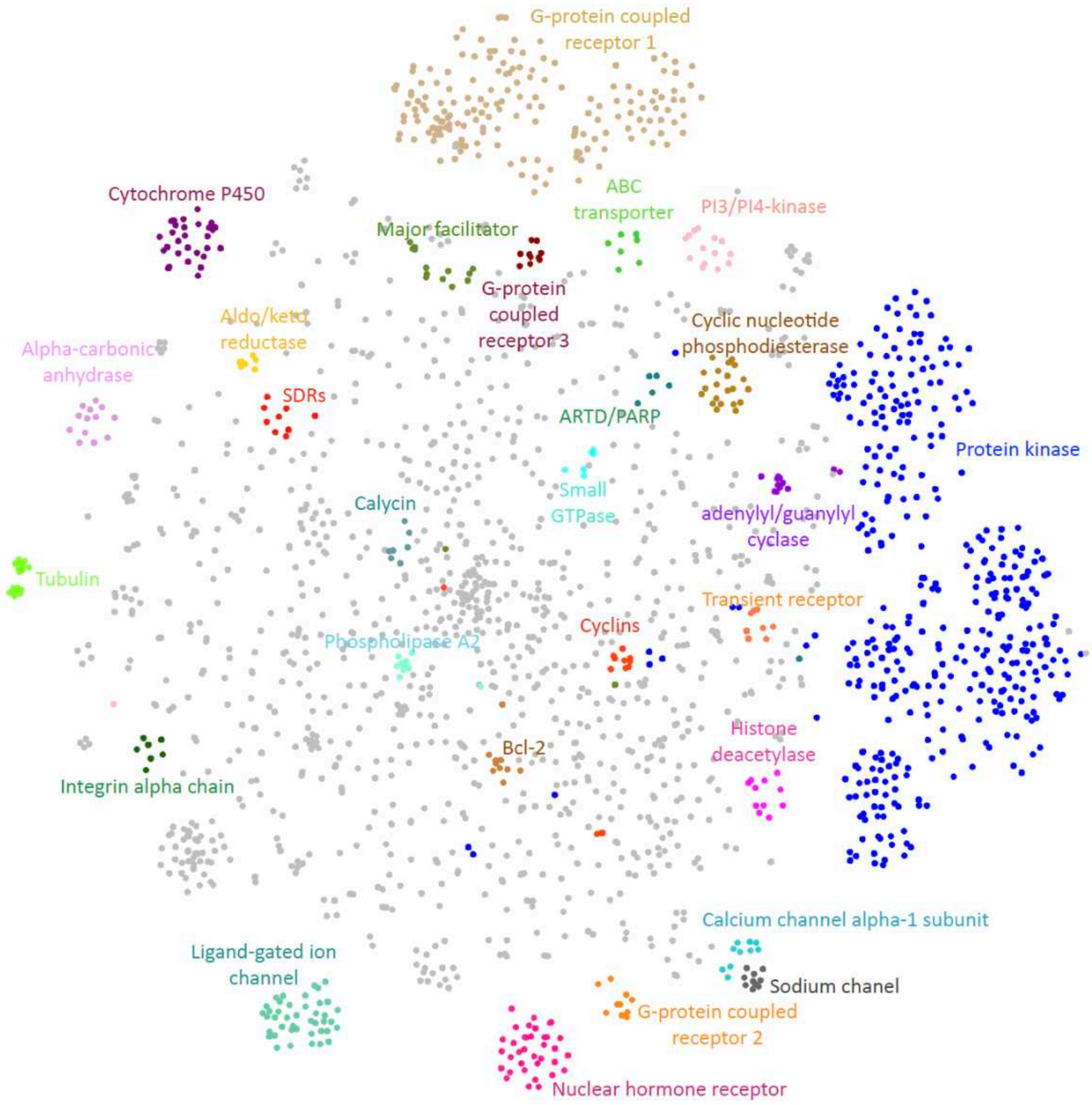

Figure 4: Representation of the protein space in LCIdb according to the t-SNE algorithm based on protein features derived from the LAkernel. A few protein families are labelled and coloured.

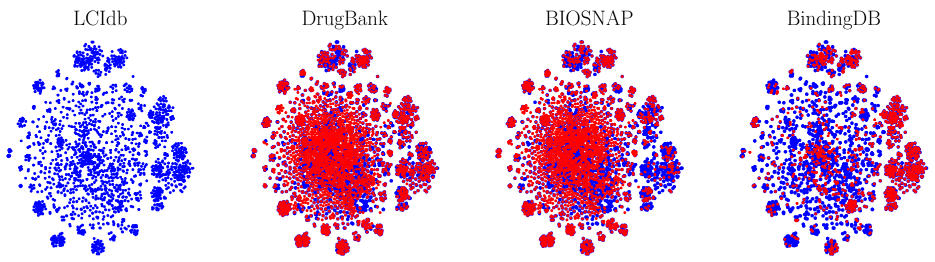

Figure 5: Representation of the protein space according to the t-SNE algorithm based on protein features derived from the LAkernel. In blue: LCIdb, in red: DrugBank, BIOSNAP, and BindingDB.

4.1). The number of molecules, proteins and interactions in these sets are provided in Table 2.

Table 2: Full specification of the Train/Val/Test sets for all datasets. DrugBank (Ext) is only used as an external validation dataset when algorithms are trained on BindingDB or LCIdb (see Section 5.4.3). Therefore, no Train, Val, or Test sets were built for DrugBank (Ext)

Datasets	#Train	#Val	#Test
BIOSNAP	9,670/9,568	1,396/1,352	2,770/2,727
Unseen_drugs	9,535/9,616	1,383/1,353	2,918/2,675
Unseen_targets	9,876/9,499	1,382/1,386	2,578/2,762
BindingDB	6,334/6,334	927/5,717	1,905/11,384
DrugBank	10,972/10,972	1,098/1,098	1,645/1,645
DrugBank (Ext)	-	-	10,838/10,838
LCIdb	161,015/161,015	32,204/32,204	48,304/48,304
Unseen_drugs	156,942/156,942	32,326/32,326	56,328/56,328
Unseen_targets	154,683/161,015	32,349/32,349	60,822/60,822
Orphan	59,132/59,132	10,145/10,145	22,503/22,503

5.2 Parameters set-up of the model

Due to the vast number of molecules in LCIdb (see Table 1), our Komet algorithm incorporates the Nyström approximation to calculate molecular features as well as a dimension reduction, which involved parameters m_M (number of landmark molecules) and d_M (dimension of molecular features). By contrast, for proteins, we retain all the proteins in the Train set as protein landmarks ($n_P = m_P = d_P$). It is therefore crucial to evaluate the potential impact of the m_M and d_M parameters on the prediction performance of Komet, the resulting gain in calculation time, and to study whether good default values can be determined. This study was performed on LCIdb_Orphan and BindingDB, respectively large- and medium-sized datasets. LCIdb_Orphan was chosen as the large dataset for exploring the impact of m_M and d_M because it corresponds to the most difficult dataset, on which it is critical not to degrade the prediction performances. Figure 6 shows that both for datasets, we can significantly reduce the number of landmark molecules (m_M) and the dimension (d_M) of molecular features without losing performance, while saving time and computational resources. In particular, results on BindingDB illustrate that reducing m_M from the total number of molecules (7 161) to 5 000 or 3 000 does not significantly affect the AUPR. In addition, for the large-sized datasets like LCIdb_Orphan, reducing m_M from 10 000 to 5 000 or 3 000 does not degrade the prediction performance.

Moreover, the AUPR curves reach a plateau for d_M values between 1 000 and 2 000, suggesting that we can limit the number of molecular features without a loss in performance. This observation is confirmed with the medium-size dataset BindingDB, for which a plateau is also reached for similar values of d_M, particularly the green curve for which no approximation was made ($n_M = m_M = 7\,161$). This suggests that d_M values in the range of 1 000-2 000 could be good default values for the number of features used in molecular representations. In addition, Figure 6 illustrates that, as expected, reducing m_M and d_M significantly reduces computational time and GPU memory usage. Consequently, we choose $d_M = 1\,000$ and $m_M = 3\,000$ as a good compromise to design a rapid and less resource-intensive algorithm, without majorly compromising performance.

5.3 Impact of different molecule and protein features on Komet prediction performances

We explored the impact of molecule and protein features on the prediction performances of Komet. For molecule features, we consider the features extracted from the Tanimoto

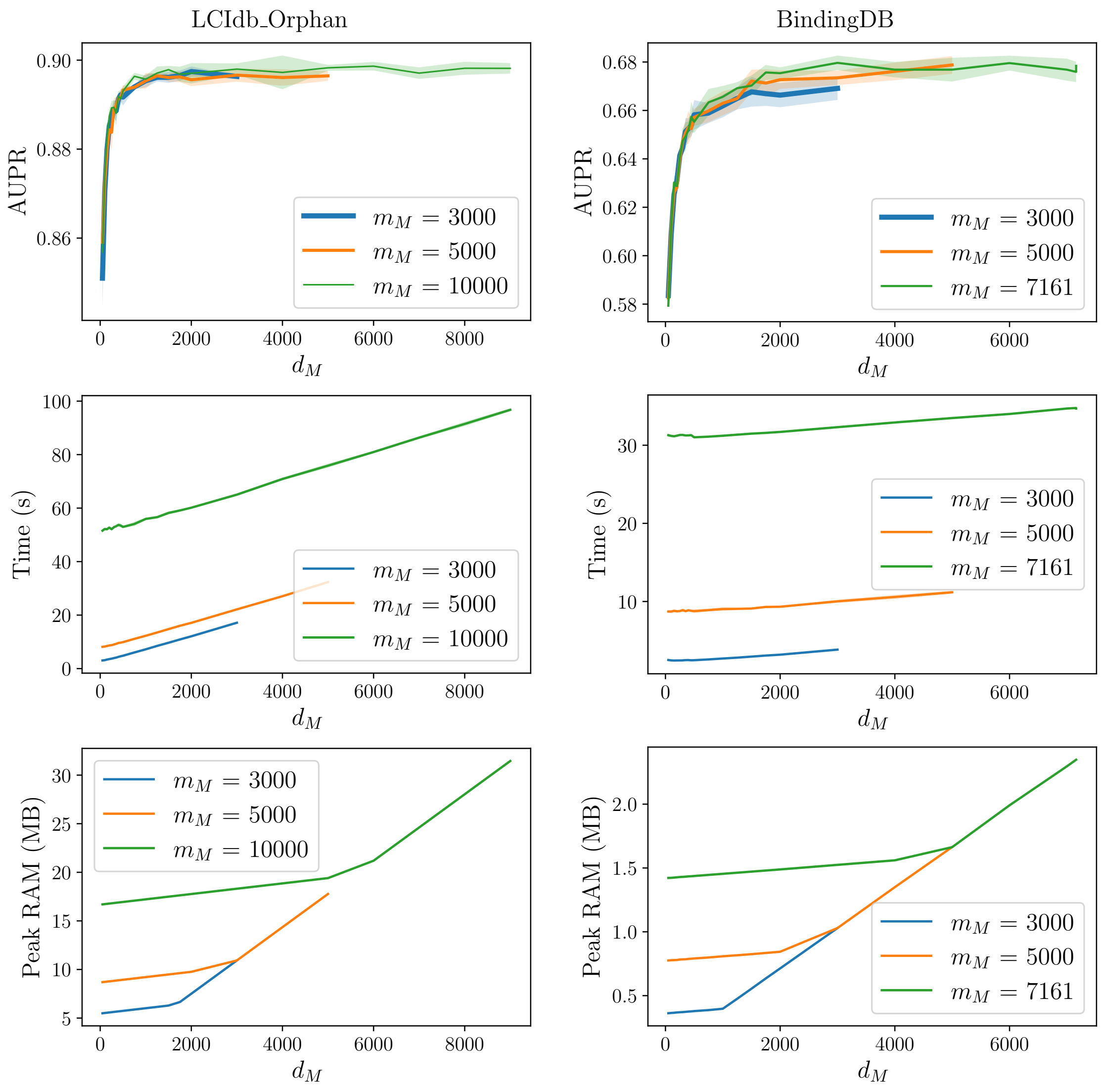

Figure 6: Influence of m_M and d_M on AUPR on the validation set of LCIdb_Orphan, computation time (in seconds) and usage and peak GPU RAM (in Gb). In each graph, the three curves correspond to three values of m_M, i.e. the number of random molecules used by the Nyström approximation of the molecular kernel. Error bars correspond to the choice of different landmark molecules. Graphs on the left refer to the large-sized dataset (LCIdb_Orphan) and on the right to the medium-sized dataset (BindingDB).

kernel between ECFP4 fingerprints, as described in Section 4.3, with the ECFP4 fingerprints themselves. This is equivalent to using the dot product between ECFP4 fingerprints, rather than the Tanimoto kernel, and no approximation (neither through the choice of a reduced set of landmark molecules nor through dimensionality reduction). Previous studies have shown that ECFP4 fingerprints perform as well as state-of-the-art fingerprint-based 3D models,[66] and are not significantly outperformed by embeddings learned from deep learning methods[67] . Therefore, we also considered pre-trained Graph Neural Networks (GNNs) for the generation of molecule features. Specifically, the work by Hu et al.[14] outlines several pre-training strategies for GNNs using a dataset of two million molecules. These strategies include supervised learning for molecular property prediction and semi-supervised learning methods such as context prediction, mutual information maximization between local and global graph representations, encouraging similarity in representations of adjacent nodes while differentiating distant nodes, and predicting masked node and edge attributes. We use the trained models adapted by Li et al.[68] to calculate the molecular embeddings and we present in 3 only the features giving the best results, which are supervised learning for molecular property prediction and semi-supervised learning on context prediction.

For proteins, we compare features extracted from the LAkernel, as described in Section 4.3, with features computed similarly, but using the 20 605 proteins of the UniProt human proteome[69] as landmark proteins, with a dimension reduction step ($d_P = 1\,200$). In addition, we used three embeddings from deep-learning models: ESM2[23] which is based on transformers, and ProtBert[24] and ProtT5XLUniref50[24] which are based on variational autoencoders trained on very large data sets of proteins.

Results are displayed in Table 3 for LCIdb_Orphan, the most challenging large-sized dataset. They show that the features proposed for Komet in the present study lead to the best prediction performance. However, replacing the molecular embeddings built from the Tanimoto kernel between ECFP4 fingerprints with the ECFP4 fingerprints themselves barely degrades the performance. This could indicate that the molecular information lost

Table 3: AUPR of Komet using different molecule and protein features on the LCIdb_Orphan dataset. "Tanimoto" features are built from the Tanimoto kernel between ECFP4 fingerprints as described in Section 4.3. "LAkernel" features are built from the Local Alignment kernel between proteins as described in Section 4.3. "UniProt LAkernel" features are built in the same way, but considering all human proteins from UniProt as landmarks proteins and using dimensionality reduction.

		Protein embedding				
		LAkernel	UniProt LAkernel	ProtBert	ProtT5XLUniref50	ESM2
Molecule embedding	Tanimoto	0.897	0.873	0.834	0.632	0.864
	ECFP4	0.893	0.861	0.829	0.630	0.866
	dgl-lifesci (GNN supervised contextpred)	0.887	0.857	0.834	0.618	0.858

by approximations (using a subset of landmark molecules and performing dimensionality reduction) is compensated by the Tanimoto kernel being a more appropriate kernel than the dot product. The protein embedding derived from the LAkernel on the 2 069 druggable proteins,[69]i.e. human proteins for which at least one drug-like ligand is known, leads to the best prediction performances. One explanation could be that the human druggable proteins present some sequence and family bias, and do not span the whole human proteome space. As a consequence, generic embeddings learned in deep learning approaches on very large sets of proteins from multiple species (ProtBert, ProtT5XLUniref50, ESM2), may be less appropriate for the specific problem DTI prediction in the context of drug-like molecules and human druggable proteins. This may also explain why features derived from the LAkernel computed on 20 605 human proteins also degrade the prediction performance. For this latter case, using the whole human proteome comes with the necessity of dimensionality reduction ($d_P = 1\,200$), which may also contribute to reducing the prediction performance.

As a consequence, the molecule features derived from the Tanimoto kernel on and the ECFP4 fingerprints and the protein features derived from the LAkernel on the 2 069 druggable proteins are used in all the following prediction experiments performed with Komet. However, one should note that except for the ProtT5XLUniref50 protein features, the prediction performances of Komet remain relatively stable to molecule and protein features.

5.4 Comparison of the prediction performances between Komet and deep-learning algorithms

Because LCIdb is large, deep-learning methods are expected to perform well on it.[70] Therefore, we compare Komet to the recently proposed ConPLex[44] algorithm, a deep-learning approach that was shown to achieve state-of-the-art performance on medium-sized datasets.

ConPLex uses molecules using Morgan fingerprints and proteins using the pre-trained Protein Language Model ProtBert[24] as input. The latent space for (molecule, protein) pairs is learned through a non-linear transformation into a shared latent space. This learning phase combines a binary DTI classification phase with a contrastive divergence phase, in which the DUD-E database[71] , comprising 102 proteins together with ligands and non-binding decoys, is used to compute a loss that minimizes the target-ligand distances (corresponding to positive DTIs) and maximizes the target-decoy distances (corresponding to negative DTIs).

We also compared Komet to MolTrans, another recent and state-of-the-art deep-learning framework[18] . MolTrans uses a representation of molecules (resp. proteins) based on frequent subsequences of the SMILES (resp. amino acid) strings, combined through a transformer module.

5.4.1 DTI prediction performances on medium-sized datasets

We first compare the performance of Komet to those of ConPLex and MolTrans on the medium-sized datasets BIOSNAP, BindingDB and DrugBank introduced in Section 4.1. We only use the AUPR score because most negative interactions in the considered datasets are unknown interactions. The results are presented in Table 4. Note that the performance of a random predictor would correspond to an AUPR score of 0.5 (except for BindingDB in which the number of negative DTIs is much larger than the number of positive DTIs, and for which the performance of a random predictor would be equal to 0.4). We report the average and standard deviation of the area under the precision-recall curve (AUPR) for 5 random initializations of each model. Interestingly, in all cases, Komet's AUPR performances (with $d_M = 1\,000$ and $m_M = 3\,000$) are similar to or higher than those of the two deep-learning methods. This is consistent with the expectation that deep-learning methods only outperform shallow learning methods when training data are abundant, due to their larger number of parameters to fit.

Table 4: AUPR performances of Komet, ConPLex, and MolTrans on medium-sized datasets BIOSNAP, BindingDB, and DrugBank. The ConPLex and MolTrans algorithms were re-run on these three datasets, and the resulting AUPR are very close (in fact slightly better) to those in the original paper.

Dataset	Komet	ConPLex	MolTrans
BIOSNAP	0.940 ± 0.001	0.921 ± 0.002	0.893 ± 0.001
Unseen_drugs	0.914 ± 0.001	0.899 ± 0.011	0.871± 0.002
Unseen_targets	0.891 ± 0.001	0.863 ± 0.005	0.683 ± 0.005
BindingDB	0.667 ± 0.005	0.669 ± 0.003	0.611 ± 0.004
Drugbank	0.939 ± 0.001	0.935 ± 0.002	0.809 ± 0.004

In the Unseen_drugs and Unseen_targets scenarios on BIOSNAP, as expected, the AUPR performances decrease for all algorithms but remain high, except for MolTrans which overall tends to display lower performances than the two other algorithms.

5.4.2 DTI prediction performances on large-sized datasets

Then, we trained Komet, ConPlex, and MolTrans on the four large-sized LCIdb-derived datasets. The results demonstrate that Komet achieves state-of-the-art AUPR prediction performance in all cases (see Table 5) at a much lower cost in terms of training time (see Table 6).

Table 5: Comparison of AUPR prediction performance on large-sized datasets

	Komet	ConPLex	MolTrans
LCIdb	0.990 ± 0.001	0.969 ± 0.002	0.967 ± 0.001
Unseen_drugs	0.994 ± 0.0003	0.978 ± 0.003	0.968 ± 0.002
Unseen_targets	0.915 ± 0.001	0.894 ± 0.031	0.591 ± 0.007
Orphan	0.896 ± 0.0008	0.846 ± 0.003	0.552± 0.013

Overall, the performance of Komet is consistently high, with AUPR scores above 0.9 in most cases. Because the number of molecules is still very large in the LCIdb Unseen_drugs

Table 6: Comparison of training time for the considered algorithms

	Komet	ConPLex	MolTrans
LCIdb	15s	907.3s	69838s
Unseen_drugs	15s	1734s	68400s
Unseen_proteins	15s	888s	64800s
Orphan	8s	1329s	25200s

dataset, thus covering a broad chemical space, the performance remains excellent, although molecules in the Test set are absent in the Train set. In LCIdb Unseen_targets and LCIdb_Orphan, where the proteins in the Test set are absent in the Train set, the performances are slightly lower but remain high. The ConPLex algorithm also displays high performances (although lower than those of Komet) in all cases, while MolTrans appears to be less stable.

We conducted a comparison using various performance measures, and the outcomes consistently align with the above results. For these additional insights, please refer to the Appendix B.

5.4.3 Validation on Drugbank (Ext) as external dataset

In the above sections, the performances of the algorithms are compared based on Train/Val /Test splits on all the considered datasets. To better assess the generalization properties of the algorithms, we used as an external dataset the DrugBank (Ext) introduced in Section 4.1.

The prediction performance of the three considered algorithms on DrugBank (Ext), when trained on BindingDB or on LCIdb, are reported in Table 7, from which two conclusions can be drawn. First, all ML algorithms perform better when trained on LCIdb compared to BindingDB. This improvement is attributed to LCIdb's more large coverage of both chemical and protein spaces. Indeed, according to Figure 2, the molecule space covered by LCIdb globally includes that covered by DrugBank, but this does not appear to be the case for the BindingDB dataset. Similarly, according to Figure 4, the protein space of LCIdb globally covers that of DrugBank, whereas the protein space of BindingDB does not seem to cover that of DrugBank.

Second, Komet always outperforms the two deep-learning algorithms. Overall, Komet trained on LCIdb displays the best generalization performances on DrugBank (Ext).

Table 7: AUPR performance for considered algorithms trained on BibndingDB and LCIdb

Training \ Algorithm	Komet	ConPLex	MolTrans
LCIdb	0.848	0.822	0.558
BindingDB	0.659	0.611	0.503

5.5 Case Study: solving scaffold hopping problems

Finally, we evaluate the pipeline that leads to the best performance, i.e. Komet trained on the LCIdb dataset based on its ability to solve scaffold hopping problems, which requires highly demanding generalization properties, and which corresponds to an important challenge in drug discovery.[1] When a hit molecule has been identified against a therapeutic target, it may not be a proper drug candidate because of poor selectivity or ADME profile, unacceptable toxicity, or expensive synthesis route. The hit molecular scaffold may also be protected by patents, which restrains its downstream development. To circumvent these limitations other active molecules with different molecular scaffolds are searched. The difficulty of the problem posed by this search depends on the degree of "dissimilarity" that is required for the new active molecule concerning the known hit. Although various examples of solving scaffold hopping cases have been reported, these types of problems are known to be difficult to solve using in silico approaches (AJOUTER QQ REFS).

Pinel et al.[42] proposed the $\mathcal{LH}$ benchmark to assess the performance of computational methods to solve scaffold hopping problems. They focused on the most difficult case, i.e. the "large-step" scaffold hopping scenario, where one ligand molecule for a given target is known, and another ligand molecule of a highly dissimilar structure is searched for the same target. The $\mathcal{LH}$ benchmark comprises 143 pairs of highly dissimilar molecules that are active against diverse protein targets. Computational methods are evaluated as follows: for each pair, one active molecule is considered as known, and the second active has to be retrieved among decoys that were carefully selected to avoid statistical bias. Since either molecule of the pair can be chosen as the known active, this leads to 286 scaffold hopping cases to solve. More precisely, given one molecule of the pair, the objective is to rank the other (considered as unknown active) among a pool of decoy molecules. The lower the rank of the unknown active, the better the prediction performance.

In Figure 7, we compare the performance of Komet and ConPLex prediction algorithms trained on LCIdbor BindingDB, using Cumulative Histogram Curves (CHC). This criterion illustrates the frequency of cases where the method ranked the unknown active molecule below a specific rank. Table 8 supplements this evaluation by providing the Area Under the Curve (AUC) of CHC curves, offering a quantitative comparison of methods, along with the proportion of cases where the unknown active was retrieved within the top 1% and 5% of best-ranked molecules. These metrics serve as indicators of the success rate of the methods. We also re-computed the results obtained by the Kronecker kernel with an SVM calculated with the scikit-learn toolbox, using the same kernels as in Komet, and trained on the DrugBank dataset. These results align with those of the original paper by Pinel et al.[42].

As shown in Figure 7 and Table 8, Komet trained on LCIdb leads to the best performances on all criteria. The ConPLex deep-learning algorithm trained on LCIdb (and fine-tuned with DUD-E) performs better on all criteria than when trained on BindingDB (and fine-tuned with DUDE-E), while the Kernel SVM trained on DrugBank of the original paper displays performances that are intermediates with those of ConPlex on the two considered training datasets. The fact that ConPLex does not outperform Komet specifically on the $\mathcal{LH}$ benchmark is somewhat puzzling. Indeed, one of the reasons why we chose ConPLex is that it incorporates a contrastive learning step based on DUD-E, which should help separate the unknown positive from the decoys in $\mathcal{LH}$. One explanation may reside in the fact that DUD-E presents a hidden bias that was shown to mislead the performance of deep learning algorithms.[72] The use of an unbiased database for contrastive learning may improve the performance of ConPLex on the $\mathcal{LH}$ benchmark.

Table 8: Prediction performances on the $\mathcal{LH}$ benchmark.

Dataset	Komet on LCIdb	Kernel SVM on DrugBank	ConPLex on BindingDB and contrastive on DUD-E	ConPLex on LCIdb and contrastive on DUD-E
Roc-AUC	0.85	0.77	0.70	0.75
Top 1%	32%	22%	12%	24%
Top 5 %	52%	36%	26%	43%

Notably, in 50% of cases, our pipeline involving Komet trained on LCIdb successfully ranks the unknown active in the top 5%. This performance surpasses those of all ligand-based methods tested in the original paper by Pinel et al.[42], the best of which, involving 3D pharmacophore descriptors, ranked the unknown active in the top 5% in 20% of cases.

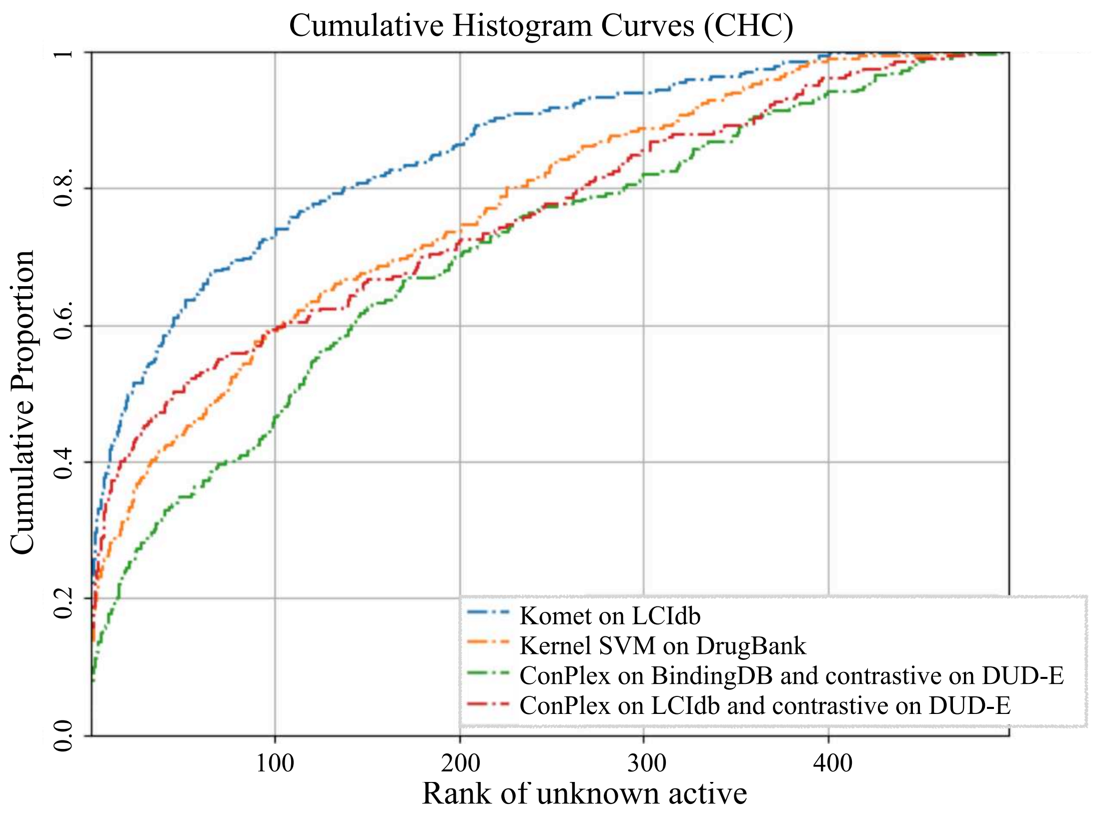

Figure 7: Cumulative Histogram Curves of the considered algorithm, measuring the cumulative proportion of cases the unknown active is retrieved below a given rank.

The fact that Komet trained on LCIdb outperforms ConPLex train on the same dataset may again be explained by more expressive features for the (molecule, protein) pairs in

Komet. In addition, the facts that (1) the performances of ConPLex are improved when trained on LCIdb over those obtained with BindingDB, and that (2) the performances of Komet trained on LCIdb over than those obtained with Kernel SVM trained on DrugBank, may be explained by a better coverage of the active molecules space in $\mathcal{LH}$ by LCIdb than by BindingDB and DrugBank. Indeed, we used the t-SNE algorithm to visualize the molecule space coverage of the LCIdb , DrugBank, BindingDB and superposed with the space of active molecules in $\mathcal{LH}$. As shown in Figure 8, LCIdb uniformly spans the entire space of active molecules in $\mathcal{LH}$, which is not the case for the DrugBank and the BindingDB datasets.

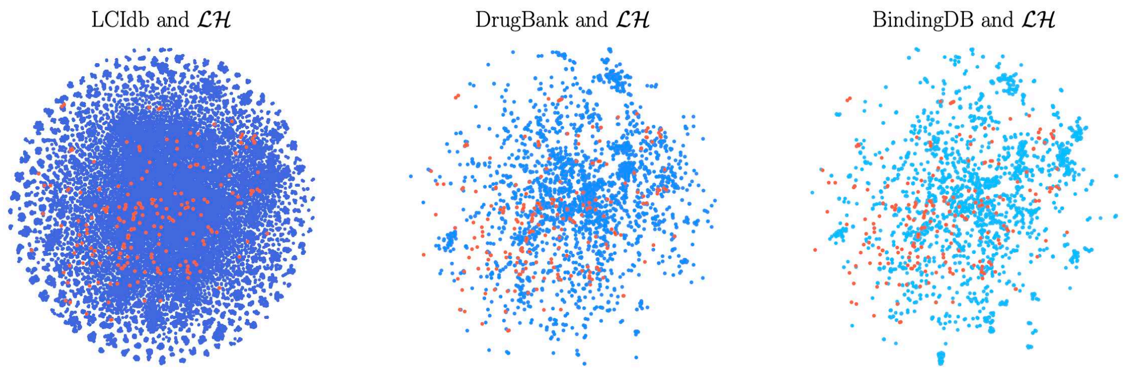

Figure 8: t-SNE on molecule features. In blue and from left to right: LCIdb, DrugBank and BindingDB, in orange: active molecules of $\mathcal{LH}$.

6 Discussion

An important contribution of the present work resides in providing the LCIdb DTI dataset which appears much larger than most public datasets used in the recent literature. A key feature of this dataset is a wider and more uniform coverage of the molecular space. A recurrent problem when building DTI datasets for training ML algorithms is that negative interactions are usually not reported. One way to circumvent this problem is to use reference databases that provide quantitative bioactivity measurements and choose threshold values to define positive and negative interactions. In previous studies,[18,44] other authors chose a common and rather low threshold value of 30 nM for both types of DTIs, leading to a modest number of positive (9 166) and three times more negative DTIs(23 435), as shown in Table 1. The notion of positive and negative DTIs is not absolute, because bioactivities are continuous, and threshold values are somewhat arbitrary. In the present paper, we chose distinct thresholds for positive and negative interactions, respectively under 100 nM (10^{-7}M) and above 100μM (10^{-4}M). This leads to a limited number of known negative DTIs in the dataset (8 296) compared to known positives (402 538). Overall, our goal was to limit the potential false negative DTIs and the bias towards well-studied molecules and proteins. Therefore, true negative DTIs were completed by randomly chosen DTIs according to the algorithm in Najm et al.[45], while excluding all DTIs with activities falling in the 100 nM 10^{-7}M-100μM margin. However, we are aware that using a lower threshold value for the negative DTIs in LCIdb would have allowed us to select a high number of DTIs considered as known negatives.

Another important contribution is the proposal of the Komet pipeline, a DTI prediction algorithm designed to learn on very large training datasets such as LCIdb . This algorithm contains two parameters, m_M (number of landmark molecules) and d_M (dimension of molecular features). We were able to define good default values for these parameters ($d_M = 1\,000$ and $m_M = 3\,000$), significantly reducing the computational time and memory requirements. Interestingly, computational resources will not increase drastically if the size of the Train set increases (if new DTIs are added), as can be judged from Figure 6.

We also showed that the performance of the algorithm was robust for the choice of the landmark molecules and the molecule and protein features, although learned features tended to decrease the performance, as shown in Table 3.

Importantly, Komet belongs to the family of shallow ML algorithms and proved to outperform ConPLex and MolTrans, two recently proposed deed-learning algorithms, at a much lower computational cost. One explanation for the good performance of Komet could be that features for the (molecule, protein) pairs derived by Komet in Step 2, simply based on the Kronecker product, may better capture determinants of the interaction than the combined learned features in the considered deep-learning algorithms. The Kronecker product strategy to combine molecule/protein features to encode interactions seems more important than the choice of features for (molecule, protein) pairs since different molecule features did not significantly impact the performances (see Table 3), and since ConPlex does not reach the performance of Komet when both are trained on LCIdb (see Table 7). In addition, the proposed architectures in ConPLex and MolTrans may not yet be fully optimized for the DTI prediction problem. Furthermore, our study focuses on DTI prediction in the human drugable space of proteins, because our goal is to propose a valuable tool to use in the context of drug discovery projects. The dimension of this space is modest, as illustrated by the number of proteins in LCIdb (2 069), with respect to that of the human proteome (above 20 000, but expected to be in the order of 90 000 when including splicing variants). Therefore, the druggable human proteins may present some sequence bias, and the protein features used in ConPLex and MolTrans and learned based on a much wider space of proteins may not be optimal for the DTI prediction problem at hand. This is consistent with the results in Table 3, showing that learned features did not improve the performances of Komet.

Komet proved to display state-of-the-art performances on various prediction scenarios, including the most difficult problems. In particular, it proved to be efficient in solving scaffold hopping cases. Although it was not designed and tuned for this specific scenario, it appears as an interesting tool to guide medicinal chemists in solving such problems. One possible future improvement would be to use other molecule kernels. Indeed, the Tanimoto molecule kernel used in Komet is a measure of structure similarity between molecules, which is a priori not well suited to the scaffold hopping problem. Other molecule kernels based on pharmacophore features may improve the prediction performances of Komet on the specific problem of scaffold hoping.

References

(1) Schneider, G.; Neidhart, W.; Giller, T.; Schmid, G. "Scaffold-hopping" by topological pharmacophore search: a contribution to virtual screening. *Angewandte Chemie International Edition* **1999**, *38*, 2894–2896.

(2) Lim, S.; Lu, Y.; Cho, C. Y.; Sung, I.; Kim, J.; Kim, Y.; Park, S.; Kim, S. A review on compound-protein interaction prediction methods: data, format, representation and model. *Computational and Structural Biotechnology Journal* **2021**, *19*, 1541–1556.

(3) Wigh, D. S.; Goodman, J. M.; Lapkin, A. A. A review of molecular representation in the age of machine learning. *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2022**, *12*, e1603.

(4) Kim, J.; Park, S.; Min, D.; Kim, W. Comprehensive survey of recent drug discovery using deep learning. *International Journal of Molecular Sciences* **2021**, *22*, 9983.

(5) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* **1988**, *28*, 31–36.

(6) Heller, S. R.; McNaught, A.; Pletnev, I.; Stein, S.; Tchekhovskoi, D. InChI, the IUPAC international chemical identifier. *Journal of cheminformatics* **2015**, *7*, 1–34.

(7) Dalby, A.; Nourse, J. G.; Hounshell, W. D.; Gushurst, A. K.; Grier, D. L.; Leland, B. A.; Laufer, J. Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. *Journal of chemical information and computer sciences* **1992**, *32*, 244–255.

(8) Rogers, D.; Hahn, M. Extended-connectivity fingerprints. *Journal of chemical information and modeling* **2010**, *50*, 742–754.

(9) Landrum, G. et al. rdkit/rdkit: Release_2023.09.5. 2024; `https://doi.org/10.5281/zenodo.10633624`.

(10) Lee, I.; Keum, J.; Nam, H. DeepConv-DTI: Prediction of drug-target interactions via deep learning with convolution on protein sequences. *PLoS computational biology* **2019**, *15*, e1007129.

(11) Zhao, Q.; Zhao, H.; Zheng, K.; Wang, J. HyperAttentionDTI: improving drug–protein interaction prediction by sequence-based deep learning with attention mechanism. *Bioinformatics* **2022**, *38*, 655–662.

(12) Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems* **2015**, *28*.

(13) Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V.; Riley, P. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design* **2016**, *30*, 595–608.

(14) Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; Leskovec, J. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265* **2019**,

(15) Jaeger, S.; Fulle, S.; Turk, S. Mol2vec: unsupervised machine learning approach with chemical intuition. *Journal of chemical information and modeling* **2018**, *58*, 27–35.

(16) Goh, G. B.; Hodas, N.; Siegel, C.; Vishnu, A. Smiles2vec: Predicting chemical properties from text representations. **2018**,

(17) Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* **2013**, *26*.

(18) Huang, K.; Xiao, C.; Glass, L. M.; Sun, J. MolTrans: molecular interaction transformer for drug–target interaction prediction. *Bioinformatics* **2021**, *37*, 830–836.

(19) Xue, D.; Zhang, H.; Xiao, D.; Gong, Y.; Chuai, G.; Sun, Y.; Tian, H.; Wu, H.; Li, Y.; Liu, Q. X-MOL: large-scale pre-training for molecular understanding and diverse molecular analysis. *bioRxiv* **2020**, 2020–12.

(20) Li, P.; Wang, J.; Qiao, Y.; Chen, H.; Yu, Y.; Yao, X.; Gao, P.; Xie, G.; Song, S. An effective self-supervised framework for learning expressive molecular global representations to drug discovery. *Briefings in Bioinformatics* **2021**, *22*, bbab109.

(21) Zhu, L.; Davari, M. D.; Li, W. Recent advances in the prediction of protein structural classes: Feature descriptors and machine learning algorithms. *Crystals* **2021**, *11*, 324.

(22) Dubchak, I.; Muchnik, I.; Holbrook, S. R.; Kim, S.-H. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences* **1995**, *92*, 8700–8704.

(23) Rives, A.; Meier, J.; Sercu, T.; Goyal, S.; Lin, Z.; Liu, J.; Guo, D.; Ott, M.; Zitnick, C. L.; Ma, J.; others Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences* **2021**, *118*, e2016239118.

(24) Elnaggar, A.; Heinzinger, M.; Dallago, C.; Rehawi, G.; Wang, Y.; Jones, L.; Gibbs, T.; Feher, T.; Angerer, C.; Steinegger, M.; others Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* **2021**, *44*, 7112–7127.

(25) Nguyen, N.-Q.; Jang, G.; Kim, H.; Kang, J. Perceiver CPI: a nested cross-attention network for compound–protein interaction prediction. *Bioinformatics* **2023**, *39*, btac731.

(26) Jacob, L.; Hoffmann, B.; Stoven, V.; Vert, J.-P. Virtual screening of GPCRs: an in silico chemogenomics approach. *BMC bioinformatics* **2008**, *9*, 1–16.

(27) Pahikkala, T.; Airola, A.; Pietilä, S.; Shakyawar, S.; Szwajda, A.; Tang, J.; Aittokallio, T. Toward more realistic drug–target interaction predictions. *Briefings in bioinformatics* **2015**, *16*, 325–337.

(28) Huang, K.; Fu, T.; Glass, L. M.; Zitnik, M.; Xiao, C.; Sun, J. DeepPurpose: a deep learning library for drug–target interaction prediction. *Bioinformatics* **2020**, *36*, 5545–5547.

(29) Öztürk, H.; Özgür, A.; Ozkirimli, E. DeepDTA: deep drug–target binding affinity prediction. *Bioinformatics* **2018**, *34*, i821–i829.

(30) Sledzieski, S.; Singh, R.; Cowen, L.; Berger, B. Adapting protein language models for rapid DTI prediction. *bioRxiv* **2022**, 2022–11.

(31) Tsubaki, M.; Tomii, K.; Sese, J. Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. *Bioinformatics* **2019**, *35*, 309–318.

(32) Shi, H.; Liu, S.; Chen, J.; Li, X.; Ma, Q.; Yu, B. Predicting drug-target interactions using Lasso with random forest based on evolutionary information and chemical structure. *Genomics* **2019**, *111*, 1839–1852.

(33) Cheng, F.; Liu, C.; Jiang, J.; Lu, W.; Li, W.; Liu, G.; Zhou, W.; Huang, J.; Tang, Y. Prediction of drug-target interactions and drug repositioning via network-based inference. *PLoS computational biology* **2012**, *8*, e1002503.

(34) Rosasco, L.; De Vito, E.; Caponnetto, A.; Piana, M.; Verri, A. Are loss functions all the same? *Neural computation* **2004**, *16*, 1063–1076.

(35) Nagamine, N.; Sakakibara, Y. Statistical prediction of protein–chemical interactions based on chemical structure and mass spectrometry data. *Bioinformatics* **2007**, *23*, 2004–2012.

(36) Jacob, L.; Vert, J.-P. Protein-ligand interaction prediction: an improved chemogenomics approach. *bioinformatics* **2008**, *24*, 2149–2156.

(37) Playe, B.; Azencott, C.-A.; Stoven, V. Efficient multi-task chemogenomics for drug specificity prediction. *PloS one* **2018**, *13*, e0204999.

(38) Sieg, J.; Flachsenberg, F.; Rarey, M. In need of bias control: evaluating chemical data for machine learning in structure-based virtual screening. *Journal of chemical information and modeling* **2019**, *59*, 947–961.

(39) Chen, L.; Tan, X.; Wang, D.; Zhong, F.; Liu, X.; Yang, T.; Luo, X.; Chen, K.; Jiang, H.; Zheng, M. TransformerCPI: improving compound–protein interaction prediction by sequence-based deep learning with self-attention mechanism and label reversal experiments. *Bioinformatics* **2020**, *36*, 4406–4414.

(40) Bagherian, M.; Sabeti, E.; Wang, K.; Sartor, M. A.; Nikolovska-Coleska, Z.; Najarian, K. Machine learning approaches and databases for prediction of drug–target interaction: a survey paper. *Briefings in bioinformatics* **2021**, *22*, 247–269.

(41) Wang, Z.; Liang, L.; Yin, Z.; Lin, J. Improving chemical similarity ensemble approach in target prediction. *Journal of cheminformatics* **2016**, *8*, 1–10.

(42) Pinel, P.; Guichaoua, G.; Najm, M.; Labouille, S.; Drizard, N.; Gaston-Mathé, Y.; Hoffmann, B.; Stoven, V. Exploring isofunctional molecules: Design of a benchmark and evaluation of prediction performance. *Molecular Informatics* **2023**, *42*, 2200216.

(43) Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; others Pytorch: An imperative style, high-

performance deep learning library. *Advances in neural information processing systems* **2019**, *32*.

(44) Singh, R.; Sledzieski, S.; Bryson, B.; Cowen, L.; Berger, B. Contrastive learning in protein language space predicts interactions between drugs and protein targets. *Proceedings of the National Academy of Sciences* **2023**, *120*, e2220778120.

(45) Najm, M.; Azencott, C.-A.; Playe, B.; Stoven, V. Drug Target Identification with Machine Learning: How to Choose Negative Examples. *International journal of molecular sciences* **2021**, *22*, 5118.

(46) Zitnik, M.; Sosic, R.; Leskovec, J. BioSNAP Datasets: Stanford biomedical network dataset collection. *Note: http://snap. stanford. edu/biodata Cited by* **2018**, *5*.

(47) Liu, T.; Lin, Y.; Wen, X.; Jorissen, R. N.; Gilson, M. K. BindingDB: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic acids research* **2007**, *35*, D198–D201.

(48) Isigkeit, L.; Chaikuad, A.; Merk, D. A consensus compound/bioactivity dataset for data-driven drug design and chemogenomics. *Molecules* **2022**, *27*, 2513.

(49) Mendez, D.; Gaulton, A.; Bento, A. P.; Chambers, J.; De Veij, M.; Félix, E.; Magariños, M. P.; Mosquera, J. F.; Mutowo, P.; Nowotka, M.; others ChEMBL: towards direct deposition of bioassay data. *Nucleic acids research* **2019**, *47*, D930–D940.

(50) Kim, S.; Chen, J.; Cheng, T.; Gindulyte, A.; He, J.; He, S.; Li, Q.; Shoemaker, B. A.; Thiessen, P. A.; Yu, B.; others PubChem in 2021: new data content and improved web interfaces. *Nucleic acids research* **2021**, *49*, D1388–D1395.

(51) Harding, S. D.; Armstrong, J. F.; Faccenda, E.; Southan, C.; Alexander, S. P.; Davenport, A. P.; Pawson, A. J.; Spedding, M.; Davies, J. A.; NC-IUPHAR The IUPHAR/BPS guide to PHARMACOLOGY in 2022: curating pharmacology for

COVID-19, malaria and antibacterials. *Nucleic Acids Research* **2022**, *50*, D1282–D1294.

(52) Gilson, M. K.; Liu, T.; Baitaluk, M.; Nicola, G.; Hwang, L.; Chong, J. BindingDB in 2015: a public database for medicinal chemistry, computational chemistry and systems pharmacology. *Nucleic acids research* **2016**, *44*, D1045–D1053.

(53) Škuta, C.; Southan, C.; Bartůněk, P. Will the chemical probes please stand up? *RSC medicinal chemistry* **2021**, *12*, 1428–1441.

(54) Scholkopf, B.; Mika, S.; Burges, C. J.; Knirsch, P.; Muller, K.-R.; Ratsch, G.; Smola, A. J. Input space versus feature space in kernel-based methods. *IEEE transactions on neural networks* **1999**, *10*, 1000–1017.

(55) Williams, C.; Seeger, M. Using the Nyström method to speed up kernel machines. *Advances in neural information processing systems* **2000**, *13*.

(56) Saigo, H.; Vert, J.-P.; Ueda, N.; Akutsu, T. Protein homology detection using string alignment kernels. *Bioinformatics* **2004**, *20*, 1682–1689.

(57) Smith, T. F.; Waterman, M. S. Identification of common molecular subsequences. *Journal of molecular biology* **1981**, *147*, 195–197.

(58) Bottou, L. Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers. 2010; pp 177–186.

(59) Airola, A.; Pahikkala, T. Fast Kronecker product kernel methods via generalized vec trick. *IEEE transactions on neural networks and learning systems* **2017**, *29*, 3374–3387.

(60) Nocedal, J.; Wright, S. J. *Numerical optimization*; Springer, 1999.

(61) Platt, J.; others Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* **1999**, *10*, 61–74.

(62) Grisoni, F.; Merk, D.; Byrne, R.; Schneider, G. Scaffold-hopping from synthetic drugs by holistic molecular representation. *Scientific reports* **2018**, *8*, 16469.

(63) 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. *Journal of chemical information and modeling* **2012**, *52*, 2864–2875.

(64) Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *Journal of machine learning research* **2008**, *9*.

(65) Kim, Q.; Ko, J.-H.; Kim, S.; Park, N.; Jhe, W. Bayesian neural network with pre-trained protein embedding enhances prediction accuracy of drug-protein interaction. *Bioinformatics* **2021**, *37*, 3428–3435.

(66) Gao, K.; Nguyen, D. D.; Sresht, V.; Mathiowetz, A. M.; Tu, M.; Wei, G.-W. Are 2D fingerprints still valuable for drug discovery? *Physical chemistry chemical physics* **2020**, *22*, 8373–8390.

(67) Sabando, M. V.; Ponzoni, I.; Milios, E. E.; Soto, A. J. Using molecular embeddings in QSAR modeling: does it make a difference? *Briefings in bioinformatics* **2022**, *23*, bbab365.

(68) Li, M.; Zhou, J.; Hu, J.; Fan, W.; Zhang, Y.; Gu, Y.; Karypis, G. DGL-LifeSci: An Open-Source Toolkit for Deep Learning on Graphs in Life Science. *ACS Omega* **2021**,

(69) Boutet, E.; Lieberherr, D.; Tognolli, M.; Schneider, M.; Bansal, P.; Bridge, A. J.; Poux, S.; Bougueleret, L.; Xenarios, I. UniProtKB/Swiss-Prot, the manually annotated section of the UniProt KnowledgeBase: how to use the entry view. *Plant bioinformatics: methods and protocols* **2016**, 23–54.

(70) Playe, B.; Stoven, V. Evaluation of deep and shallow learning methods in chemogenomics for the prediction of drugs specificity. *Journal of cheminformatics* **2020**, *12*, 11.

(71) Mysinger, M. M.; Carchia, M.; Irwin, J. J.; Shoichet, B. K. Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry* **2012**, *55*, 6582–6594.

(72) Chen, L.; Cruz, A.; Ramsey, S.; Dickson, C. J.; Duca, J. S.; Hornak, V.; Koes, D. R.; Kurtzman, T. Hidden bias in the DUD-E dataset leads to misleading performance of deep learning in structure-based virtual screening. *PloS one* **2019**, *14*, e0220113.

A Molecule space coverage of various datasets

This section shows cases of a 2D visualization of the chemical space covered by various datasets considered in the paper, using the t-SNE algorithm on various molecule features.

Figure 9 shows the drug distribution in LCIdb across the five databases from which the initial dataset[48] is extracted. It highlights a significant contribution from the CHemBL and PubChem databases, enhanced mainly by data from Probes&Drugs.

Figure 10 shows the t-SNE visualizations of the molecular space for various considered datasets, based on Tanimoto features (as in Figure 2) for one choice of 3000 landmark molecules, for another choice of 3000 landmark molecules, and for ECFP4 features. It confirms that LCIdb offers broader and more uniform coverage of the chemical space than BindingDB, DrugBank, or BIOSNAP.

B Several metrics to compare prediction performances

Table 9 presents various metrics for comparing prediction performances on the four LCIdb-datasets. While ConPlex has better accuracy in two cases, overall, Komet outperforms the

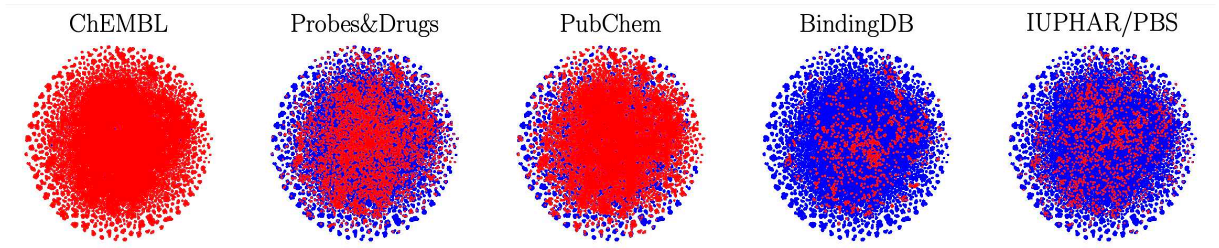

Figure 9: t-SNE on molecule features. In blue: large-sized benchmark LCIdb, in red: 5 databases from which the initial dataset[48] is extracted.

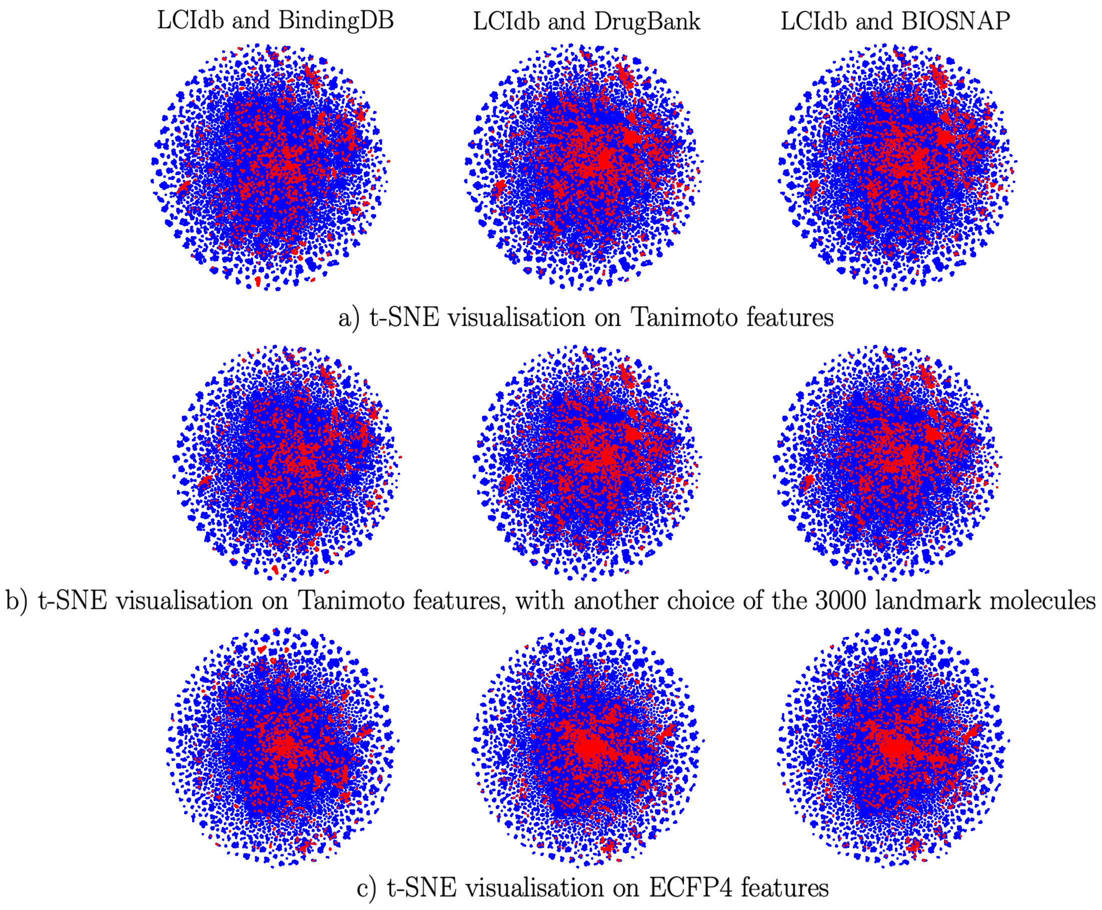

Figure 10: 2D representation of the molecular space, based on the t-SNE algorithm on molecule features. In blue: large-sized LCIdb dataset, and in red: medium-scale DrugBank, BIOSNAP, and BindingDB datasets.

other algorithms in most cases according to AUPR, ROC-AUC and Accuracy prediction performances, supporting the main conclusions in the paper.

Table 9: AUPR, ROC-AUC and Accuracy prediction performances

	Komet			ConPLex			MolTrans	
	AUPR	ROC-AUC	Accuracy	AUPR	ROC-AUC	Accuracy	AUPR	ROC-AUC
LCIdb	0.990	0.990	0.966	0.970	0.971	0.917	0.967	0.970
Unseen_drugs	0.994	0.994	0.976	0.980	0.977	0.934	0.968	0.969
Unseen_targets	0.915	0.896	0.714	0.893	0.874	0.763	0.591	0.584
Orphan	0.896	0.879	0.682	0.845	0.834	0.689	0.552	0.536

C Nyström approximation

In Komet, we encode molecules leveraging the Nyström approximation.[54,55] In the following, we present the mathematical details of Section 4.3.

Let us consider a set of landmark molecules $\{\hat{\mathfrak{m}}_1, \dots \hat{\mathfrak{m}}_{m_M}\}$, a new molecule $\mathfrak{m}$, and a kernel k_M over molecules. The kernel matrix $K \in \mathbb{R}^{(m_M+1)\times(m_M+1)}$ over these m_{M+1} molecules can be written as $K = \begin{bmatrix} \hat{K}_M & \kappa^\top \\ \kappa & k_M(\mathfrak{m},\mathfrak{m}) \end{bmatrix}$ with $\hat{K}_M \in \mathbb{R}^{m_M \times m_M}$ being the kernel matrix over the landmark molecules and $\kappa = (k_M(\mathfrak{m}, \hat{\mathfrak{m}}_1), \dots, k_M(\mathfrak{m}, \hat{\mathfrak{m}}_{m_M})) \in \mathbb{R}^{m_M}$ the vector of kernel values between $\mathfrak{m}$ and the landmark molecules.

The Nyström's approximation consists in approximating K as $K \approx C\hat{K}_M^{-1}C^\top = \begin{bmatrix} \hat{K}_M & \kappa^\top \\ \kappa & \kappa\hat{K}_M^{-1}\kappa^\top \end{bmatrix}$ with $C = \begin{bmatrix} \hat{K}_M \\ \kappa \end{bmatrix} \in \mathbb{R}^{(m_M+1)\times m_M}$.

Writing the Single Value Decomposition of $\hat{K}_M$ as $\hat{K}_M = U\,\mathrm{diag}(\sigma)U^\top$, the approximation of K can be rewritten as $K \approx \Phi\Phi^\top$ with $\Phi = CU\,\mathrm{diag}(\sigma)^{-1/2} \approx CE$. When no dimensionality reduction is performed ($d_M = m_M$), $E = U\,\mathrm{diag}(\sigma)^{-1/2}$ and $\Phi = CE$.

The last line of matrix Φ is $\Phi_{m_M+1} = (\sum_{l=1}^{m_M} C_{m_M+1,l}E_{ls})_{s=1}^{m_M} = \psi_M(\mathfrak{m})$. Similarly, its m_M first lines are $\psi_M(\hat{\mathfrak{m}}_1)$, ..., $\psi_M(\hat{\mathfrak{m}}_{m_M})$. Hence $k_M(\mathfrak{m}, \hat{\mathfrak{m}}_i) \approx \langle\psi_M(\mathfrak{m}), \psi_M(\hat{\mathfrak{m}}_i)\rangle$ for any molecule $\mathfrak{m}$ (including one of the landmark molecules), which justifies our proposition of ψ_M.

Furthermore, if we do not use dimensionality reduction, because the Nyström approximation is an equality on the upper-left block $\hat{K}_M$, $k_M(\hat{\mathfrak{m}}_i, \hat{\mathfrak{m}}_j) = \langle \psi_M(\hat{\mathfrak{m}}_i), \psi_M(\hat{\mathfrak{m}}_j) \rangle$ for any pair of landmark molecules.

D Efficient computation

We explicit here the details for equality (a) of Eq (2) in paragraph 4.4.

$$(Zw)_k = \langle w, z_k \rangle_{\mathbb{R}^{d_Z}} \overset{(a)}{=} \langle m_{i_k}, W p_{j_k} \rangle_{\mathbb{R}^{d_M}} \overset{(b)}{=} \langle m_{i_k}, q_{j_k} \rangle_{\mathbb{R}^{d_M}}.$$

We use the matrix representation $W \in \mathbb{R}^{d_M \times d_P}$ instead of $w \in \mathbb{R}^{d_Z}$ in a way that w is the flattened representation of W.

$$\begin{aligned}
\forall k = 1..n_Z,\ (Zw)_k &= \langle w, z_k \rangle_{\mathbb{R}^{d_Z}} \\
&= \langle W, m_{i_k} p_{j_k}^\top \rangle_{\mathbb{R}^{d_M \times d_P}} \\
&= \operatorname{tr}\left(W (m_{i_k} p_{j_k}^\top)^\top \right) = \operatorname{tr}\left(W p_{j_k} m_{i_k}^\top \right) = \langle W p_{j_k}, m_{i_k} \rangle_{\mathbb{R}^{d_M}}
\end{aligned}$$