PHARMACOMATCH: EFFICIENT 3D PHARMACOPHORE SCREENING VIA NEURAL SUBGRAPH MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

The increasing size of screening libraries poses a significant challenge for the development of virtual screening methods for drug discovery, necessitating a re-evaluation of traditional approaches in the era of big data. Although 3D pharmacophore screening remains a prevalent technique, its application to very large datasets is limited by the computational cost associated with matching query pharmacophores to database ligands. In this study, we introduce PharmacoMatch, a novel contrastive learning approach based on neural subgraph matching. Our method reinterprets pharmacophore screening as an approximate subgraph matching problem and enables efficient querying of conformational databases by encoding query-target relationships in the embedding space. We conduct comprehensive evaluations of the learned representations and benchmark our method on virtual screening datasets in a zero-shot setting. Our findings demonstrate significantly shorter runtimes for pharmacophore matching, offering a promising speed-up for screening very large datasets.

1 INTRODUCTION

A challenging task in the early stages of drug discovery campaigns is the identification of hit molecules that effectively bind to a protein target of interest. Due to the vastness of the chemical space, estimated to encompass more than 10^{60} small organic molecules (Virshup et al., 2013), identifying molecules with desirable drug-like properties is often compared to finding a needle in a haystack. Virtual screening methods have therefore become an essential component of the computer-aided drug discovery toolkit, aiding medicinal chemists in filtering molecular databases to efficiently explore the search space for potential hit compounds (Sliwoski et al., 2014).

A pharmacophore represents non-bonding interactions of chemical features that are essential for binding to a specific protein target (Wermuth et al., 1998). A pharmacophore query can, for example, be generated from the interaction profile of a ligand-receptor complex and used to identify potential hit compounds from databases by searching for molecules with similar pharmacophoric patterns (Wolber & Langer, 2005). The process involves a positional alignment of the pharmacophore model with the three-dimensional conformations of molecules in the database, which are ranked based on their agreement with the pharmacophore query (Wolber et al., 2006). Since pharmacophore screening focuses on abstract interaction patterns, rather than specific molecular structures, it allows for the identification of structurally diverse hit compounds (Seidel et al., 2017).

Virtual screening of make-on-demand libraries like Enamine REAL (Shivanyuk et al., 2007) is of growing interest because these libraries contain compounds that can be synthesized through reliable synthetic routes within a short period, making them readily commercially available. These libraries encompass billions of molecules and continue to expand due to advances in synthetic accessibility (Llanos et al., 2019). While screening larger compound libraries enhances the likelihood of identifying hit compounds, it also extends screening times, thereby necessitating the scaling up of virtual screening methods (Sadybekov et al., 2022). However, scaling up 3D pharmacophore screening to accommodate billions of molecules presents significant challenges (Warr et al., 2022). Although various filtering techniques have been developed (Seidel et al., 2010), molecules that pass these methods must still undergo alignment algorithms, which ultimately determine the speed of the process. Despite substantial efforts to optimize these algorithms (Wolber et al., 2008; Permann et al., 2021), the overall screening procedure remains time-intensive.

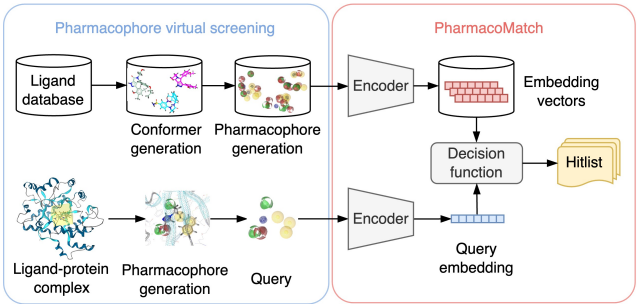

Figure 1: Overview of the *PharmacoMatch* workflow: Conformer and pharmacophore generation from ligands and query creation, for example from a ligand-protein complex, precede pharmacophore screening. The encoder model converts the screening database into embedding vectors, stored for later use. A hitlist is generated by comparing the query embedding with the database embeddings.

In this work, we propose using self-supervised learning to create meaningful 3D pharmacophore representations for efficient virtual screening. Our *PharmacoMatch* model employs a graph neural network (GNN) encoder, trained with a contrastive learning objective, to map 3D pharmacophores into an order embedding space (Ying et al., 2020), thereby enabling pharmacophore matching through vector comparisons. The embedding vectors for the screening database are computed once and then used to quickly generate a hitlist based on the query embedding. An overview of the workflow is presented in Figure 1.

Our key contributions are:

- We develop a *GNN encoder model* that generates meaningful vector representations from 3D pharmacophores. The model is trained in a self-supervised manner on unlabeled data, employing a contrastive loss objective to capture the relationships between queries and targets based on their partial ordering in the learned embedding space.
- We use the learned representation for fast *virtual screening in the embedding space* and evaluate the performance of our method through experiments on virtual screening benchmark datasets.

2 Related Work

Pharmacophore alignment algorithms Alignment algorithms compute a rigid-body transformation, the *pharmacophore alignment*, to match a query's pharmacophoric pattern to database ligands. A *scoring function* then evaluates the *pharmacophore matching* by considering both the number of matched features and their spatial proximity. The alignment is typically preceded by fast filtering methods that prune the search space based on pharmacophoric types, pharmacophoric point counts, and quick distance checks. Only molecules that pass these filters undergo the final, computationally expensive 3D alignment step, which is usually performed by minimizing the root mean square deviation (RMSD) between pairs of pharmacophoric points (Seidel et al., 2010; Dixon et al., 2006). The algorithm by Wolber et al. (2006) creates smoothed histograms from the neighborhoods of pharmacophoric points for pair assignment using the Hungarian algorithm, followed by alignment with Kabsch's method (Kabsch, 1976). A recent implementation by Permann et al. (2021) improves on runtime and accuracy by using a search strategy that maximizes pairs of matching pharmacophoric points. Alternatively, shape-matching algorithms like ROCS (Hawkins et al., 2007) and Pharao (Taminau et al., 2008) model pharmacophoric points with Gaussian volumes, optimizing for volume overlap.

Machine learning for virtual screening A common approach to using machine learning for virtual screening is to train models on measured bioactivity values. However, these models are constrained by the scarcity of experimental data, which is both costly and challenging to obtain (Li et al., 2021). Unsupervised training of target-agnostic models for virtual screening avoids dependence on

labeled data, but remains relatively unexplored. DrugClip (Gao et al., 2023) approaches virtual screening as a similarity matching problem between protein pockets and molecules, using a multi-modal learning approach where a protein and a molecule encoder create a shared embedding space for virtual screening. Sellner et al. (2023) used the Schrödinger pharmacophore shape-screening score to train a transformer model on pharmacophore similarity, which is a different objective than pharmacophore matching. PharmacoNet (Seo & Kim, 2023) uses instance segmentation for pharmacophore generation in protein binding sites and a graph-matching algorithm for binding pose estimation, employing deep learning for pharmacophore modeling, but not for the alignment nor matching.

Contrastive representation learning A common approach for the extraction of vector embeddings is the use of contrastive learning frameworks. These frameworks make use of a Siamese network architecture and a contrastive loss function, where an embedding space is learned by comparing positive and negative examples (Bengio et al., 2013). In the last years, the computer vision community reported great improvements in the use of self-supervised learning (SSL) frameworks, which can be seen as a special case of contrastive learning. Instead of labels, these frameworks use augmentations to create positive and negative examples during training, which allows to train models on large datasets of unlabeled data. SSL is often used for model pretraining, followed by fine-tuning through supervised learning, which is especially useful when data is limited; however, the learned representations can also be utilized without fine-tuning if no labeled data is available (Balestriero et al., 2023).

3 PRELIMINARIES

Pharmacophore representation In this work, we treat 3D pharmacophores as attributed point clouds (Mahé et al., 2006; Kriege & Mutzel, 2012). A pharmacophore P can be represented by a set of pharmacophoric points $P = \{(\mathbf{r}_i, l_i) \in \mathbb{R}^3 \times \mathcal{L}\}_i$ with the Cartesian coordinates $\mathbf{r}_i$ and the label l_i of the pharmacophoric point p_i. The label set $\mathcal{L}$ contains the following *pharmacophoric descriptors*: hydrogen bond donors (HBD) and acceptors (HBA), halogen bond donors (XBD), positive (PI) and negative electrostatic interaction sites (NI), hydrophobic interaction sites (H), and aromatic moieties (AR). Directed descriptors like HBD and HBA can be associated with a vector component, but for simplicity, we will omit this information in our study.

The pharmacophore P can be represented as a complete graph $G(P) = (V_P, E_P, \lambda_P)$, where $V_P = \{v_1, ..., v_{|P|}\}$ denotes the set of nodes with node attributes $\lambda_P(v_i) = l_i$, and $E_P = V_P \times V_P$ denotes the set of edges, where λ_P represents a labelling function $\lambda : V \cup E \to \mathcal{L}$ that assigns a label to the corresponding vertex v or edge e. The edges are undirected, edge e_{uv} can be identified with edge e_{vu}, and the label of e_{uv} is the pair-wise Euclidean distance $\lambda_P(e_{uv}) = \|\mathbf{r}_u - \mathbf{r}_v\|_2$ between the positions of nodes u and v. This representation is invariant to translation and rotation.

Subgraph matching Two graphs $G_1 = (V_1, E_1, \lambda_1)$ and $G_2 = (V_2, E_2, \lambda_2)$ are *isomorphic*, denoted by $G_1 \simeq G_2$, if there exists an edge-preserving bijection $f : V_1 \to V_2$ such that $\forall (u, v) \in E_1 : (f(u), f(v)) \in E_2$. Additionally, we require the preservation of node and edge labels, such that $\forall v \in V_1 : \lambda_1(v) = \lambda_2(f(v))$, and $\forall (u, v) \in E_1 : \lambda_1((u, v)) = \lambda_2((f(u), f(v)))$. Let $G_Q = (V_Q, E_Q, \lambda_Q)$ be a query graph, $G_T = (V_T, E_T, \lambda_T)$ a larger target graph, and $G_H = (V_H, E_H, \lambda_H)$ a subgraph of G_T such that $V_H \subseteq V_T$, and $E_H \subseteq E_T$. The objective of *subgraph matching* is to decide, whether G_Q is *subgraph isomorphic* to G_T, denoted by $G_Q \lesssim G_T$, which requires the existence of a non-empty set of subgraphs $\mathcal{H} = \{G_H \mid G_H \simeq G_Q\}$ that are isomorphic to G_Q.

Pharmacophore matching In its most general setting, *pharmacophore matching* seeks to match all pharmacophoric points of a query pharmacophore P_Q with the corresponding pharmacophoric points of a larger target pharmacophore P_T.

Let $P_H \subseteq P_T$ denote a subset of the pharmacophoric points of P_T. Then P_Q matches P_T *after alignment* if there exists a bijection $g : P_Q \to P_H$ such that $\forall i \in P_Q : l_i = l_{g(i)}$ and $\|\mathbf{r}_i - \mathbf{r}_{g(i)}\|_2 < r_T$, where r_T is the radius of a tolerance sphere. It is thereby sufficient that query pharmacophoric points are mapped into the tolerance sphere of their target counterpart. For simplicity, we assume the same tolerance radii among all pharmacophoric points. The ultimate goal of pharmacophore

matching is to retrieve molecules from a database. A matching pharmacophore is always linked to a corresponding ligand molecule *via* a look-up table.

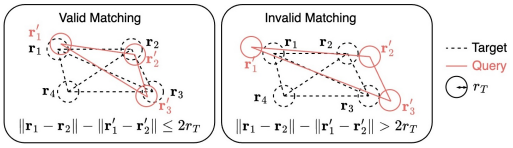

Figure 2: Illustration of the pharmacophore matching objective: The aim is to match the pharmacophoric points of a query with the corresponding points of a target pharmacophore such that the query points fall within the tolerance sphere of the target points, with a tolerance radius r_T.

When represented as graphs $G_Q = G(P_Q)$, $G_H = G(P_H)$, and $G_T = G(P_T)$, this task boils down to the node-induced subgraph matching of a query pharmacophore graph G_Q to a target pharmacophore graph G_T. The tolerance sphere, however, weakens the requirement on edge label matching. An approximate matching $\lambda_Q((u,v)) \approx \lambda_H((f(u), f(v)))$ is sufficient if the difference between $\lambda_Q((u,v))$ and $\lambda_H((f(v), f(u)))$ is less than $2r_T$, where r_T represents the tolerance radius of each pharmacophoric point. This ensures that the query points fall within the tolerance spheres of the target points (compare Figure 2). Our problem formulation of pharmacophore matching relies on relative distances instead of the absolute positioning of pharmacophoric features and is therefore *independent of prior alignment*.

4 Methodology

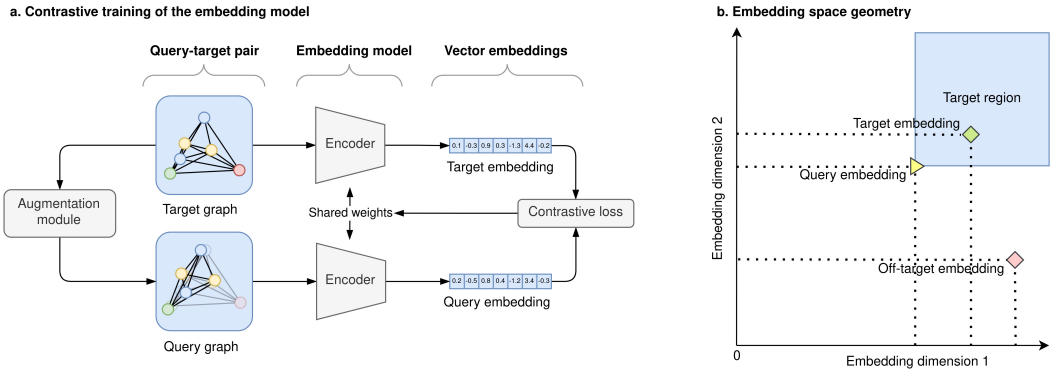

Figure 3: (a.) The embedding model learns an order embedding space by comparing augmented pharmacophores. (b) Illustration of the embedding space, where pharmacophores matching a query are positioned to the upper right.

Overview In the following we introduce PharmacoMatch, a novel contrastive learning framework with the aim to encode *query-target relationships* of 3D pharmacophores into an embedding space. We propose to train a GNN encoder model in a *self-supervised* fashion, as illustrated in Figure 3. Our model is trained on approximately 1.2 million *unlabeled small molecules* from the ChEMBL database (Davies et al., 2015; Zdrazil et al., 2023) and learns pharmacophore matching solely from *augmented examples*, comparing positive and negative pairs of query and target pharmacophore graphs, while optimizing an *order embedding loss* to extract relevant matching patterns.

Unlabeled data for contrastive training To span the *pharmaceutical compound space*, we download a set of drug-like molecules sourced from the ChEMBL (2024) website in the form of Simplified Molecular Input Line Entry System (SMILES) strings (Weininger, 1988) and curate an unlabeled dataset using the open-source Chemical Data Processing Toolkit (CDPKit) (Seidel, 2024) (see Appendix A.1 for details). After an initial data clean-up, which includes the removal of solvents and counter ions, adjustment of protonation states to a physiological pH, and elimination of duplicate structures, the dataset contains approximately 1.2 million small molecules. To ensure a *zero-shot*

setting in our validation experiments, we remove all molecules from the training data that also appear in the test sets. Finally, we generate a low-energy 3D conformation and the corresponding pharmacophore for each ligand.

Model input We represent the node labels $\{\lambda_P(v_1), ..., \lambda_P(v_{|P|})\}$ of a given pharmacophore graph $G(P) = (V_P, E_P, \lambda_P)$ as *one-hot-encoded* (OHE) feature vectors $\mathbf{h} = (\mathbf{h}_1, ..., \mathbf{h}_{|P|})$. We employ a *distance encodin*g to represent pair-wise distances, which was inspired by the SchNet architecture (Schütt et al., 2018). The edge attributes of edge e_{uv} are derived from the edge label $\lambda_P(e_{uv})$ and represented by a radial basis function $\mathbf{e}_k(\mathbf{r}_u - \mathbf{r}_v) = \exp(-\beta(\|\mathbf{r}_u - \mathbf{r}_v\|_2 - \mu_k)^2)$, where centers μ_k were taken from a uniform grid of K points between zero and the distance cutoff at 10 Å, and the smoothing factor β represents a hyperparameter. To this end, the pharmacophore P is represented by a data point $\mathbf{x} = [\mathbf{h}, \mathbf{e}]$ which is a tuple of the feature matrix $\mathbf{h} \in \mathbb{R}^{|P| \times L}$ and the distance-encodings $\mathbf{e} \in \mathbb{R}^{(|P| \times |P|) \times K}$.

GNN encoder architecture The encoder input is the pharmacophore graph representation $\mathbf{x} = [\mathbf{h}, \mathbf{e}]$, with the feature matrix $\mathbf{h}$ and the edge attributes $\mathbf{e}$. Node feature embeddings are generated by initially passing the OHE feature matrix through a single dense layer without an activation function. We then update the node representations through *message passing* using the *edge-conditioned convolution operator* (NNConv) by Gilmer et al. (2017); Simonovsky & Komodakis (2017), which was originally designed for representation learning on point clouds and 3D molecules, to aggregate distance information into the learned node representations (see Appendix A.3 for details). We connect successive convolutional layers using DenseNet-style skip connections (Huang et al., 2017). Graph-level read-out is achieved by *additive pooling* of the updated feature matrix $\mathbf{h} \in \mathbb{R}^{|P| \times m}$ into a graph representation $\mathbf{q} \in \mathbb{R}^m$, which is then projected to the final output embedding $\mathbf{z} \in \mathbb{R}^D_+$ by a multi-layer perceptron. The employed loss function requires to map the final representation to the *non-negative real number space*. We accomplish this by using the absolute values of the learnable weights for the last linear transformation immediately after the final ReLU unit (see Appendix A.4 for details).

Loss function In order to encode query-targets relationship of pharmacophores into the embedding space, we employ the loss function by Ying et al. (2020). The key insight is that subgraph relationships can be effectively encoded in the geometry of an order embedding space through a partial ordering of the corresponding vector embeddings. Let $\mathbf{z}_Q$ the embedding of graph G_Q, $\mathbf{z}_T$ the embedding of graph G_T, and $f_\Theta : \mathcal{G} \rightarrow \mathbb{R}^D_+$ a GNN encoder to map pharmacophore graphs $\mathcal{G}$ to embedding vectors $\mathbf{z} \in \mathbb{R}^D_+$. The partial ordering $\mathbf{z}_Q \preceq \mathbf{z}_T$ reflects, whether G_Q is subgraph isomorphic to G_T:

$$\mathbf{z}_Q[i] \leq \mathbf{z}_T[i],\ \forall i \in \{1, ..., D\} \text{ iff } G_Q \lesssim G_T \tag{1}$$

The following max-margin objective can be used to train the GNN encoder f_Θ on this relation:

$$\mathcal{L}(\mathbf{z}_Q, \mathbf{z}_T) = \sum_{(\mathbf{z}_Q, \mathbf{z}_T) \in Pos} E(\mathbf{z}_Q, \mathbf{z}_T) + \sum_{(\mathbf{z}_Q, \mathbf{z}_T) \in Neg} \max\{0, \alpha - E(\mathbf{z}_Q, \mathbf{z}_T)\} \tag{2}$$

The penalty function $E : \mathbb{R}^D_+ \times \mathbb{R}^D_+ \rightarrow \mathbb{R}_+$ reflects violation of the partial ordering on the embedding vector pair:

$$E(\mathbf{z}_Q, \mathbf{z}_T) = \|\max\{\mathbf{0}, \mathbf{z}_Q - \mathbf{z}_T\}\|_2^2 \tag{3}$$

Pos is the set of positive pairs per batch, these are pairs of query $\mathbf{z}_Q$ and target graph embedding $\mathbf{z}_T$ with a subgraph-supergraph relationship, and Neg is the set of negative examples, these are pairs of query and target embedding vectors that violate this relationship. The positive and negative pairs are generated on-the-fly *via* augmentation during training.

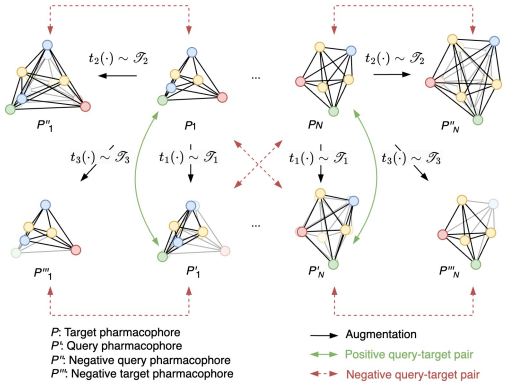

Figure 4: Augmentation strategies for model training involve generating positive and negative query-target pairs on-the-fly by combining node deletion with varying degrees of node displacement. Negative pairs are also created by shuffling the batch, mapping query pharmacophores to random target pharmacophores.

Augmentation module The PharmacoMatch model correlates the matching of a query and a target pharmacophore with the partial ordering of their vector representations. Positive pairs represent successful matchings, while negative pairs serve as counter examples. In order to create these pairs from unlabeled training data, we define three families of augmentations $\mathcal{T}$, which are composed of *random point deletions* and *positional point displacements*.

For positive pairs, valid queries are created by randomly deleting some nodes from a pharmacophore P, leaving at least three, and displacing the remaining nodes within a tolerance sphere of radius r_T. This augmentation, denoted as $t_1(\cdot) \sim \mathcal{T}_1$, produces the positive pair $(t_1(P), P)$.

Negative pairs are used to show the model examples of unsuccessful matching, employing three strategies that illustrate different types of undesired outcomes. Our first strategy provides the model with examples of positional mismatch by placing the pharmacophoric points of P on the surface of the tolerance sphere without any point deletions. To implement this, we calculate the mean of the positions of the pharmacophoric points, $\mu = \frac{1}{n}\sum_{i=1}^{|P|} \mathbf{r}_i$, and displace each point p_i by r_T in the direction $\mathbf{r}_i - \mu$. Positional displacement away from the center ensures that the displacement directions do not cancel each other. This augmentation, denoted by $t_2(\cdot) \sim \mathcal{T}_2$, is used to generate the negative query-target pair $(t_2(P), P)$.

Our second strategy teaches the model that every pharmacophoric point in the query should correspond to a point in the target. This is achieved by deleting some target nodes, using an augmentation operator $t_3(\cdot) \sim \mathcal{T}_3$, where $\mathcal{T}_3$ involves node deletion without displacement. As a result, the query in the pair $(t_1(P), t_3(P))$ only partially matches its target.

With the third strategy, we train the model to avoid matching queries with targets that are significantly different. This approach involves randomly mapping queries $t_1(P_i)$ to the incorrect targets P_j, where $i \neq j$ (for more details, see Appendix A.2).

Curriculum learning We design a curriculum learning strategy for training on pharmacophore graphs. We start training with pharmacophores containing four nodes. If the loss does not decrease significantly within 10 epochs, we add pharmacophores with one additional node to the training data. This approach allows the model to start with very simple examples, gradually increasing the difficulty of the matching task.

Model Training Our GNN encoder model is implemented with three convolutional layers with an output dimension of 64. The MLP has a depth of three dense layers with a hidden dimension of 1024 and an output dimension of 512. The final model was trained for 500 epochs using an Adam optimizer with a learning rate of 10^{-3}. The margin of the best performing model was set to $\alpha = 100$. The tolerance radius r_T for node displacement was set to 1.5 Å, which is the default value in the pharmacophore screening functionalities of the CDPKit (see Appendix A.5 for more details).

Ablation studies To evaluate the importance of various model parameters, we conduct a series of ablation studies using the best-performing model. In these experiments, we systematically alter one parameter at a time and assess its impact on classification performance using the validation hold-out set. The complete details of these experiments are provided in the Appendix A.5 (Table 4). Our findings reveal several key insights. The embedding dimension of the learned representations can be reduced to 128 without loss in performance. The encoder requires at least 32 dimensions to remain effective. Skip-connections are critical for model performance, with DenseNet-style con-

nections slightly outperforming ResNet-style (He et al., 2015) alternatives. Interestingly, the choice of message-passing layer, whether NNConv, graph isomorphism operator (GINE) (Hu et al., 2020), graph attention operator (GAT) (Brody et al., 2022), or continuous-filter convolutional layers (CF-Conv) (Schütt et al., 2018), has minimal impact on performance. The depth of the projector and encoder also does not significantly affect results. In contrast, the margin value plays a significant role in model performance. While larger values enhance performance, excessively high margins can lead to training instability. A margin of 100 provides an effective balance between these factors. The displacement radius for augmentations in creating positive pairs is most effective at 1.5 Å.

Decision function for model inference We are using the trained GNN encoder f_Θ to precompute vector embeddings $\mathbf{z}_T$ of the database pharmacophores. These are queried with the pharmacophore embedding $\mathbf{z}_Q$ by verification of the partial ordering constraint (3), which shall not be violated by more than a threshold t. This leads to the decision function $g : \mathbb{R}_+^D \times \mathbb{R}_+^D \to \{0, 1\}$:

$$g(\mathbf{z}_Q, \mathbf{z}_T) = \begin{cases} 1 & \text{iff} E(\mathbf{z}_Q, \mathbf{z}_T) < t \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

which evaluates to 1 if the partial ordering on $\mathbf{z}_Q$ and $\mathbf{z}_T$ reflects a pharmacophore matching, and 0 otherwise. In the following, we will refer to equation (4) as matching decision function. In practice, we recommend a decision threshold of $t = 6500$, which was determined during our benchmark experiments.

5 Experiments

We designed our embeddings to reflect the type and relative positioning of pharmacophoric points. Comparison of embedding vectors *via* the matching prediction function should emulate the matching of the underlying pharmacophores. To get a better understanding of the encoder's latent space, we investigate these properties as follows:

1. **Pharmacophoric point perception**: We investigate the learned embedding space quantitatively through dimensionality reduction.
2. **Positional perception**: We investigate the influence of positional changes on the output of the matching decision function.
3. **Virtual screening performance**: The performance of our model is evaluated using ten DUD-E targets, and the produced hitlists are compared with the performance and runtime of the CDPKit (Seidel, 2024) alignment algorithm.

DUD-E benchmark dataset We perform our experiments on the DUD-E benchmark dataset (Mysinger et al., 2012), which is commonly used to evaluate the performance of molecular docking and structure-based screening. The complete benchmark contains 102 protein targets, each accompanied by active and decoy ligands in the form of SMILES strings (Weininger, 1988) and the PDB template (Burley et al., 2017) of the ligand-receptor complex. We randomly select ten different protein targets for the evaluation of our model. Ligands in these datasets are processed according to the data curation pipeline outlined in the Methodology section, except that we sample up to 25 conformations per compound. The ligand-receptor complex is used to generate a structure-based query with 5-7 pharmacophoric points (see Appendix A.6 for more details).

5.1 Pharmacophoric point perception

We conduct a quantitative analysis through dimensionality reduction to gain a first intuition for the properties of the learned embedding space.

The partial ordering of graph representations in the embedding space, based on the number of nodes per graph, is essential for encoding query-target relationships. This ordering property of the embedding space can be visualized using principal component analysis (PCA). Figure 5a displays the first two principal component axes of the learned representations, with the representations labeled

according to the number of pharmacophoric points of the corresponding pharmacophore. This visualization demonstrates how the embedding vectors are systematically ordered relative to the number of nodes in each pharmacophore graph.

Similarly, the Uniform Manifold Approximation and Projection (UMAP) algorithm (McInnes et al., 2020), a dimensionality reduction technique that preserves the local neighborhood structure of high-dimensional data, was employed. Figure 5b shows the UMAP representation of the embeddings, labeled by the number of pharmacophoric points of a specific type. This visualization suggests that pharmacophores with a similar set of points are mapped proximally within the order embedding space.

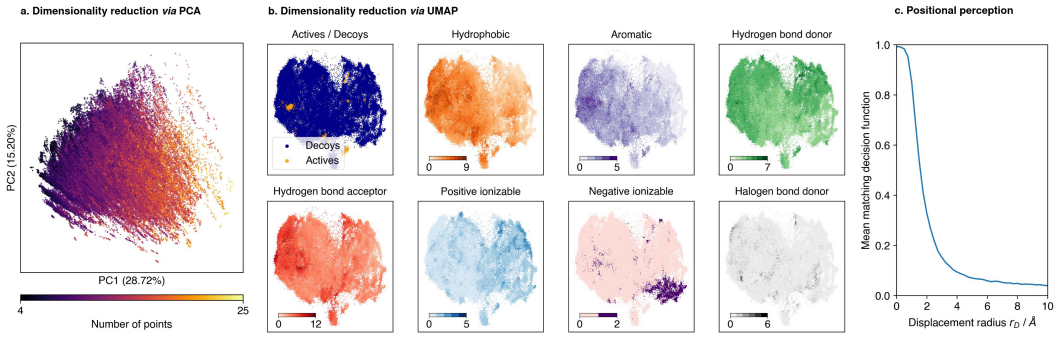

Figure 5: (a.) Dimensionality reduction of the ADA target's embedding space via PCA, with embeddings labeled by pharmacophoric point count. (b.) Dimensionality reduction via UMAP, with embeddings labeled by pharmacophoric point type. (c.) Experimental validation of the model's perception of 3D point positions, showing the mean matching decision function versus the displacement radius r_D of the augmentation, with a decision threshold set to $t = 6500$.

5.2 Positional Perception

We define a family of augmentations $\mathcal{T}_{r_D}$ to randomly delete nodes from a pharmacophore P and displace the remaining nodes by a radius r_D. We sample augmentations $t_{r_D}(\cdot) \sim \mathcal{T}_{r_D}$ with increasing radius r_D taken from a uniform grid of m distances between 0 and 10 Å.

For a given batch of pharmacophores $\{P_1, ..., P_n\}$, we generate the query-target pairs $\{(t_{r_D}(P_1), P_1), ..., (t_{r_D}(P_n), P_n)\}$. We then evaluate the decision function $g(\cdot, \cdot)$ (Equation 4) on the corresponding vector representations and calculate the mean of the decision function across all pairs against an increasing radius r_D, which is illustrated in Figure 5c.

Without node displacement, the mean matching decision function is close to 1, indicating that the model recognizes pharmacophores with reduced node sets as valid queries. With a displacement of approximately 1.5 Å, the mean matching decision value drops to 50%, demonstrating the model's consideration of the chosen tolerance radius. Beyond a displacement of 1.5 Å, the decision function further decreases, approaching a plateau at approximately 6 Å. The results show that our model integrates 3D-positional information of pharmacophoric points into the learned representations.

5.3 Virtual Screening

Each benchmark set is comprised of a pharmacophore query P_Q and a set of ligands $\mathcal{L} = \{L_1, ..., L_n\}$, where each ligand L_i is associated with a set of pharmacophores $\{P_1, ..., P_{k_i}\}_i$ and a label y_i, which indicates whether the ligand is active or decoy. The task is to rank the database ligands *w.r.t.* the query, based on a scoring function $F : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_+$. The ranking score ψ_i of ligand L_i is calculated through aggregation of the pharmacophore scores $\bigoplus(\{F(P_Q, P_1), ..., F(P_Q, P_{k_i})\}_i)$, where $\bigoplus$ is an aggregation operator. PharmacoMatch transforms the query $G(P_Q) \mapsto \mathbf{z}_Q$ and the set of pharmacophores $\{G(P_1), ..., G(P_{k_i})\}_i \mapsto \{\mathbf{z}_1, ..., \mathbf{z}_{k_i}\}_i$ *via* encoder model $f_\Theta : \mathcal{G} \to \mathbb{R}_+^D$ and evaluates the penalty function $E : \mathbb{R}_+^D \times \mathbb{R}_+^D \to \mathbb{R}_+$. A low penalty corresponds to a high ranking. The ranking score of database ligand L_i is calculated as $\psi_i = \min(\{E(\mathbf{z}_Q, \mathbf{z}_1), ..., E(\mathbf{z}_Q, \mathbf{z}_{k_i})\}_i)$.

Baseline algorithm The baseline for our comparison is the alignment algorithm implemented in the open-source software CDPKit (Seidel, 2024), which utilizes clique-detection followed by Kabsch alignment (Kabsch, 1976). The alignment of a query P_Q and a target P_T is evaluated with an *alignment score* $S : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_+$, which takes into account the number of matched features and their geometric fit (further details are provided in the Appendix A.6). The ligand ranking score is calculated as $\psi_i = \max(\{S(P_Q, P_1), ..., S(P_Q, P_{k_i})\}_i)$, the highest alignment score represents the score for the database ligand. Analogous to equation (4), we can also define a matching decision function ϕ based on the alignment score, where $t = |P_Q|$:

$$\phi(P_Q, P_T) = \begin{cases} 1 & \text{iff} S(P_Q, P_T) \geq t \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Evaluation Both algorithms rank database ligands to produce a *hitlist*. We assess the performance of PharmacoMatch on the benchmark using two approaches. First, we demonstrate that the PharmacoMatch penalty $E(\cdot, \cdot)$ correlates with the matching decision function $\phi(\cdot, \cdot)$ of the alignment algorithm. We evaluate both functions against all pharmacophores in a dataset *w.r.t.* query P_Q. The outputs are compared by generating the corresponding receiver operating characteristic (ROC) curves, and the performance is quantified using the area under the ROC curve (AUROC) metric.

Second, we compare the virtual screening performance of our model and the alignment algorithm using the ligand ranking score ψ. The primary objective of virtual screening is to find active compounds amongst decoys. We evaluate this using two different metrics. The AUROC metric is used to evaluate the overall classification performance *w.r.t.* activity label y_i. A drawback of this metric is that it does not reflect the early enrichment of active compounds in the hitlist, which is of significant interest in virtual screening. Early enrichment is assessed using the Boltzmann-enhanced discrimination of ROC (BEDROC) metric (Truchon & Bayly, 2007), which assigns higher weights to better-ranked samples. Note that these performance metrics are entirely dependent on the chosen query. Rather than aiming to maximize those metrics, our goal is to achieve comparable values between our model and the alignment algorithm.

Screening performance Our results, comparing PharmacoMatch with the alignment algorithm across all ten targets, are summarized in Table 1 (ROC plots are provided in the Appendix A.6). We observe a robust correlation between the hitlists generated by the two algorithms, demonstrating the effectiveness of our approach. This correlation varies by target, reflecting the sensitivity of virtual screening to the chosen query. Although the alignment algorithm achieves generally higher AUROC scores and early enrichment, our method consistently produces hitlists with competitive performance across several targets.

Table 1: Method comparison and screening performance of the PharmacoMatch algorithm and the CDPKit alignment algorithm on ten different DUD-E protein targets (see Appendix A.6 for details). BEDROC values are calculated with $\alpha = 20$, as recommended by Truchon & Bayly (2007), AUROC and BEDROC are reported in percent. Confidence intervals are calculated using bootstrapping (Efron, 1979), with standard deviations reported based on 1,000 resampled datasets.

Protein target	Relative performance	Absolute screening performance									
		PharmacoMatch					CDPKit				
	AUROC	AUROC	BEDROC	$EF_{1\%}$	$EF_{5\%}$	$EF_{10\%}$	AUROC	BEDROC	$EF_{1\%}$	$EF_{5\%}$	$EF_{10\%}$
ACES	90.7 ± 0.2	57 ± 2	19 ± 2	10 ± 1	3.9 ± 0.4	2.3 ± 0.2	55 ± 1	15 ± 1	4 ± 1	3.4 ± 0.3	2.3 ± 0.2
ADA	97.7 ± 0.3	80 ± 3	39 ± 4	10 ± 4	9 ± 1	5.5 ± 0.5	93 ± 2	80 ± 4	53 ± 4	16.2 ± 0.9	8.6 ± 0.4
ANDR	98.0 ± 0.2	78 ± 2	32 ± 2	15 ± 2	5.8 ± 0.5	4.3 ± 0.3	72 ± 2	26 ± 2	13 ± 2	4.3 ± 0.5	3.7 ± 0.3
EGFR	90.4 ± 0.5	59 ± 1	10 ± 1	2.7 ± 0.7	1.8 ± 0.2	1.5 ± 0.2	72 ± 1	22 ± 1	11 ± 1	4.0 ± 0.4	3.1 ± 0.2
FA10	84.2 ± 0.1	48 ± 1	2.1 ± 0.4	0.2 ± 0.2	0.3 ± 0.1	0.4 ± 0.1	55 ± 1	5.8 ± 0.6	-	0.7 ± 0.2	1.2 ± 0.1
KIT	80.2 ± 0.1	52 ± 2	2.3 ± 0.6	-	0.3 ± 0.2	0.3 ± 0.1	58 ± 2	7 ± 1	1.2 ± 0.9	0.9 ± 0.3	1.2 ± 0.3
PLK1	79.1 ± 0.5	61 ± 3	9 ± 2	1.4 ± 1.2	0.8 ± 0.4	1.8 ± 0.4	75 ± 3	39 ± 3	6 ± 2	10 ± 1	5.4 ± 0.5
SRC	95.6 ± 0.1	77 ± 1	22 ± 1	3.9 ± 0.8	3.9 ± 0.3	4.0 ± 0.2	77 ± 1	24 ± 1	5 ± 1	4.7 ± 0.4	3.3 ± 0.2
THRB	86.4 ± 0.4	73 ± 1	26 ± 2	6 ± 1	5.5 ± 0.4	3.8 ± 0.2	81 ± 1	40 ± 2	16 ± 2	8.4 ± 0.4	5.3 ± 0.2
UROK	83.3 ± 0.2	59 ± 2	3 ± 1	0.6 ± 0.6	0.4 ± 0.2	0.4 ± 0.2	91 ± 1	52 ± 3	25 ± 3	10.6 ± 0.7	5.5 ± 0.4

Runtime comparison In terms of runtime, PharmacoMatch significantly outperforms the alignment algorithm. We compare the time required for alignment, embedding, and vector matching per pharmacophore. Alignment is performed in parallel on an AMD EPYC 7713 64-Core Processor with 128 threads, while pharmacophore embedding and matching are run on an NVIDIA GeForce RTX 3090, with both devices having comparable purchase prices and release dates. Creating vector embeddings from pharmacophore graphs takes 92 ± 12 μs per pharmacophore, which takes longer than aligning a query to a target with 13 ± 7 μs. However, the embedding process only needs to be performed once. Subsequently, the preprocessed vector data can be used for vector matching, which takes 0.30 ± 0.09 μs, being approximately two orders of magnitude faster than the alignment. Additionally, vector comparison is independent of the query size, an advantage not shared by the alignment algorithm. Although executed on different hardware, this comparison highlights the speed-gain of our algorithm.

Practical considerations There are two options for integrating our model into a virtual screening pipeline. First, the PharmacoMatch model can be used in place of the alignment algorithm to generate a hitlist of ligands, which is suitable for quickly producing a compound list for experimental testing. Alternatively, our method can serve as an efficient prefiltering tool for very large databases, reducing the number of molecules from billions to millions, after which the slower alignment algorithm can be applied to this filtered subset. Note that alignment will still be necessary if visual inspection of aligned pharmacophores and corresponding ligands is desired.

6 Conclusion

We have presented PharmacoMatch, a contrastive learning framework that creates meaningful pharmacophore representations for virtual screening. The proposed method tackles the matching of 3D pharmacophores through vector comparison in an order embedding space, thereby offering a valuable method for significant speed-up of virtual screening campaigns. PharmacoMatch is the first machine-learning based solution that approaches pharmacophore virtual screening *via* an approximate neural subgraph matching algorithm. We are confident that our method will help to improve on existing virtual screening workflows and contribute to the assistance of medicinal chemist in the complex task of drug discovery.

Reproducibility statement

The source code of this project can be found under the following link: `https://anonymous.4open.science/r/PharmacoMatch-34A0/README.md`. Please follow the instructions in the repository for reproduction of our results. Training and test data can be downloaded here: `https://figshare.com/s/24757b89ea7f0932bf3c?file=49290172`.

References

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *arxiv preprint arxiv:2304.12210*, 2023.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arxiv preprint arxiv:2105.14491*, 2022.

Stephen K Burley, Helen M Berman, Gerard J Kleywegt, John L Markley, Haruki Nakamura, and Sameer Velankar. Protein data bank (pdb): the single global macromolecular structure archive. *Protein crystallography: methods and protocols*, pp. 627–641, 2017.

ChEMBL. Chembl database - embl-ebi. `https://www.ebi.ac.uk/chembl/`, 2024. Accessed: 2024-01-06.

Mark Davies, Michał Nowotka, George Papadatos, Nathan Dedman, Anna Gaulton, Francis Atkinson, Louisa Bellis, and John P. Overington. ChEMBL web services: streamlining access to drug discovery data and utilities. *Nucleic Acids Res.*, 43(W1):W612–W620, 2015.

Steven L Dixon, Alexander M Smondyrev, Eric H Knoll, Shashidhar N Rao, David E Shaw, and Richard A Friesner. Phase: a new engine for pharmacophore perception, 3d qsar model development, and 3d database screening: 1. methodology and preliminary results. *J. Comput. Aided Mol. Des.*, 20:647–671, 2006.

B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1 – 26, 1979.

William Falcon and The PyTorch Lightning team. PyTorch Lightning, March 2019. URL `https://github.com/Lightning-AI/lightning`.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arxiv preprint arxiv:1903.02428*, 2019.

Bowen Gao, Bo Qiang, Haichuan Tan, Yinjun Jia, Minsi Ren, Minsi Lu, Jingjing Liu, Wei-Ying Ma, and Yanyan Lan. Drugclip: Contrastive protein-molecule representation learning for virtual screening. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44595–44614. Curran Associates, Inc., 2023.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Paul CD Hawkins, A Geoffrey Skillman, and Anthony Nicholls. Comparison of shape-matching and docking as virtual screening tools. *J. Med. Chem.*, 50(1):74–82, 2007.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arxiv preprint arxiv:1512.03385*, 2015.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arxiv preprint arxiv:1905.12265*, 2020.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32 (5):922–923, 1976.

Nils Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 291–298, 2012.

Hongjian Li, Kam-Heung Sze, Gang Lu, and Pedro J Ballester. Machine-learning scoring functions for structure-based virtual screening. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 11(1):e1478, 2021.

Christopher A. Lipinski, Franco Lombardo, Beryl W. Dominy, and Paul J. Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*, 23(1):3–25, 1997.

Eugenio J Llanos, Wilmer Leal, Duc H Luu, Jürgen Jost, Peter F Stadler, and Guillermo Restrepo. Exploration of the chemical space and its three historical regimes. *Proceedings of the National Academy of Sciences*, 116(26):12660–12665, 2019.

Pierre Mahé, Liva Ralaivola, Véronique Stoven, and Jean-Philippe Vert. The pharmacophore kernel for virtual screening with support vector machines. *J. Chem. Inf. Model*, 46(5):2003–2014, 2006.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arxiv preprint arxiv:1802.03426*, 2020.

Michael M Mysinger, Michael Carchia, John J Irwin, and Brian K Shoichet. Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *J. Med. Chem.*, 55 (14):6582–6594, 2012.

Christian Permann, Thomas Seidel, and Thierry Langer. Greedy 3-point search (g3ps)—a novel algorithm for pharmacophore alignment. *Molecules*, 26(23):7201, 2021.

Arman A Sadybekov, Anastasiia V Sadybekov, Yongfeng Liu, Christos Iliopoulos-Tsoutsouvas, Xi-Ping Huang, Julie Pickett, Blake Houser, Nilkanth Patel, Ngan K Tran, Fei Tong, et al. Synthon-based ligand discovery in virtual libraries of over 11 billion compounds. *Nature*, 601(7893): 452–459, 2022.

Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet–a deep learning architecture for molecules and materials. *J. Chem. Phys.*, 148(24), 2018.

Thomas Seidel. Chemical data processing toolkit source code repository. `https://github.com/molinfo-vienna/CDPKit`, 2024. Accessed: 2024-01-06.

Thomas Seidel, Gökhan Ibis, Fabian Bendix, and Gerhard Wolber. Strategies for 3d pharmacophore-based virtual screening. *Drug Discov. Today Technol.*, 7(4):e221–e228, 2010.

Thomas Seidel, Sharon D. Bryant, Gökhan Ibis, Giulio Poli, and Thierry Langer. *3D Pharmacophore Modeling Techniques in Computer-Aided Molecular Design Using LigandScout*, chapter 20, pp. 279–309. John Wiley & Sons, Ltd, 2017. ISBN 9781119161110.

Thomas Seidel, Christian Permann, Oliver Wieder, Stefan M Kohlbacher, and Thierry Langer. High-quality conformer generation with conforge: algorithm and performance assessment. *J. Chem. Inf. Model*, 63(17):5549–5570, 2023.

Manuel S. Sellner, Amr H. Mahmoud, and Markus A. Lill. Enhancing ligand-based virtual screening with 3d shape similarity via a distance-aware transformer model. *bioRxiv preprint bioRxiv:2023.11.17.567506*, 2023.

Seonghwan Seo and Woo Youn Kim. Pharmaconet: Accelerating large-scale virtual screening by deep pharmacophore modeling. *arxiv preprint arxiv:2310.00681*, 2023.

Alexander N Shivanyuk, Sergey V Ryabukhin, A Tolmachev, AV Bogolyubsky, DM Mykytenko, AA Chupryna, W Heilman, and AN Kostyuk. Enamine real database: Making chemical diversity real. *Chemistry today*, 25(6):58–59, 2007.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3693–3702, 2017.

Gregory Sliwoski, Sandeepkumar Kothiwale, Jens Meiler, and Edward W Lowe. Computational methods in drug discovery. *Pharmacological reviews*, 66(1):334–395, 2014.

Jonatan Taminau, Gert Thijs, and Hans De Winter. Pharao: pharmacophore alignment and optimization. *Journal of Molecular Graphics and Modelling*, 27(2):161–169, 2008.

Jean-François Truchon and Christopher I Bayly. Evaluating virtual screening methods: good and bad metrics for the "early recognition" problem. *J. Chem. Inf. Model*, 47(2):488–508, 2007.

Aaron M Virshup, Julia Contreras-García, Peter Wipf, Weitao Yang, and David N Beratan. Stochastic voyages into uncharted chemical space produce a representative library of all possible drug-like compounds. *J. Am. Chem. Soc.*, 135(19):7296–7303, 2013.

Wendy A Warr, Marc C Nicklaus, Christos A Nicolaou, and Matthias Rarey. Exploration of ultralarge compound collections for drug discovery. *J. Chem. Inf. Model*, 62(9):2021–2034, 2022.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput.*, 28(1):31–36, 1988.

Camille-Georges Wermuth, CR Ganellin, Per Lindberg, and LA Mitscher. Glossary of terms used in medicinal chemistry (iupac recommendations 1998). *Pure Appl. Chem.*, 70(5):1129–1143, 1998.

Gerhard Wolber and Thierry Langer. Ligandscout: 3-d pharmacophores derived from protein-bound ligands and their use as virtual screening filters. *J. Chem. Inf. Model*, 45(1):160–169, 2005.

Gerhard Wolber, Alois A Dornhofer, and Thierry Langer. Efficient overlay of small organic molecules using 3d pharmacophores. *J. Comput. Aided Mol. Des.*, 20(12):773–788, 2006.

Gerhard Wolber, Thomas Seidel, Fabian Bendix, and Thierry Langer. Molecule-pharmacophore superpositioning and pattern matching in computational drug design. *Drug Discov. Today*, 13 (1-2):23–29, 2008.

Rex Ying, Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, and Jure Leskovec. Neural subgraph matching. *arxiv preprint arxiv:2007.03092*, 2020.

Barbara Zdrazil, Eloy Felix, Fiona Hunter, Emma J Manners, James Blackshaw, Sybilla Corbett, Marleen de Veij, Harris Ioannidis, David Mendez Lopez, Juan F Mosquera, Maria Paula Magarinos, Nicolas Bosc, Ricardo Arcila, Tevfik Kizilören, Anna Gaulton, A Patrícia Bento, Melissa F Adasme, Peter Monecke, Gregory A Landrum, and Andrew R Leach. The ChEMBL Database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods. *Nucleic Acids Res.*, 52(D1):D1180–D1192, 2023.

A APPENDIX

A.1 DATASET CURATION & STATISTICS

Unlabeled training data was downloaded from the ChEMBL database to represent small molecules with drug-like properties. At the time of data download, the ChEMBL database contained 2,399,743 unique compounds. We constrained the compound category to "small molecules" and enforced adherence to the Lipinsky rule of five (Lipinski et al., 1997), specifically setting violations to "0," resulting in a refined set of 1,348,115 compounds available for download. The molecules were acquired in the form of Simplified Molecular Input Line Entry System (SMILES) (Weininger, 1988) strings. Subsequent to data retrieval, we conducted preprocessing using the database cleaning functionalities of the Chemical Data Processing Toolkit (CDPKit) (Seidel, 2024). This process involved the removal of solvents and counter ions, adjustment of protonation states to a physiological pH value, and elimination of duplicate structures, where compounds differing only in their stereo configuration were regarded as duplicates. To prevent data leakage, we carefully removed all structures from the training data that would occur in one of the test sets we used for our benchmark experiments. The final set was comprised of 1,221,098 compounds. For each compound within the dataset, a 3D conformation was generated using the CONFORGE (Seidel et al., 2023) conformer generator from the CDPKit, which was successful for 1,220,104 compounds. To enhance batch diversity, we generated only one conformation per compound for contrastive training. Subsequently, 3D pharmacophores were computed for each conformation, with removal of pharmacophores containing less than four pharmacophoric points. The ultimate dataset comprised 1,217,361 distinct pharmacophores.

Figure 6 shows the frequency of pharmacophores with a specific pharmacophoric point count in the training data. On average, a pharmacophore consists of 13 pharmacophoric points, with the largest pharmacophore in the dataset containing 32 points. Pharmacophores with fewer than four points were omitted during data clean-up. Hydrophobic pharmacophoric points and hydrogen bond acceptors are the most prominent, while hydrogen bond donors and aromatics occur less frequently. Ionizable pharmacophoric points and halogen bond donors are comparatively rare.

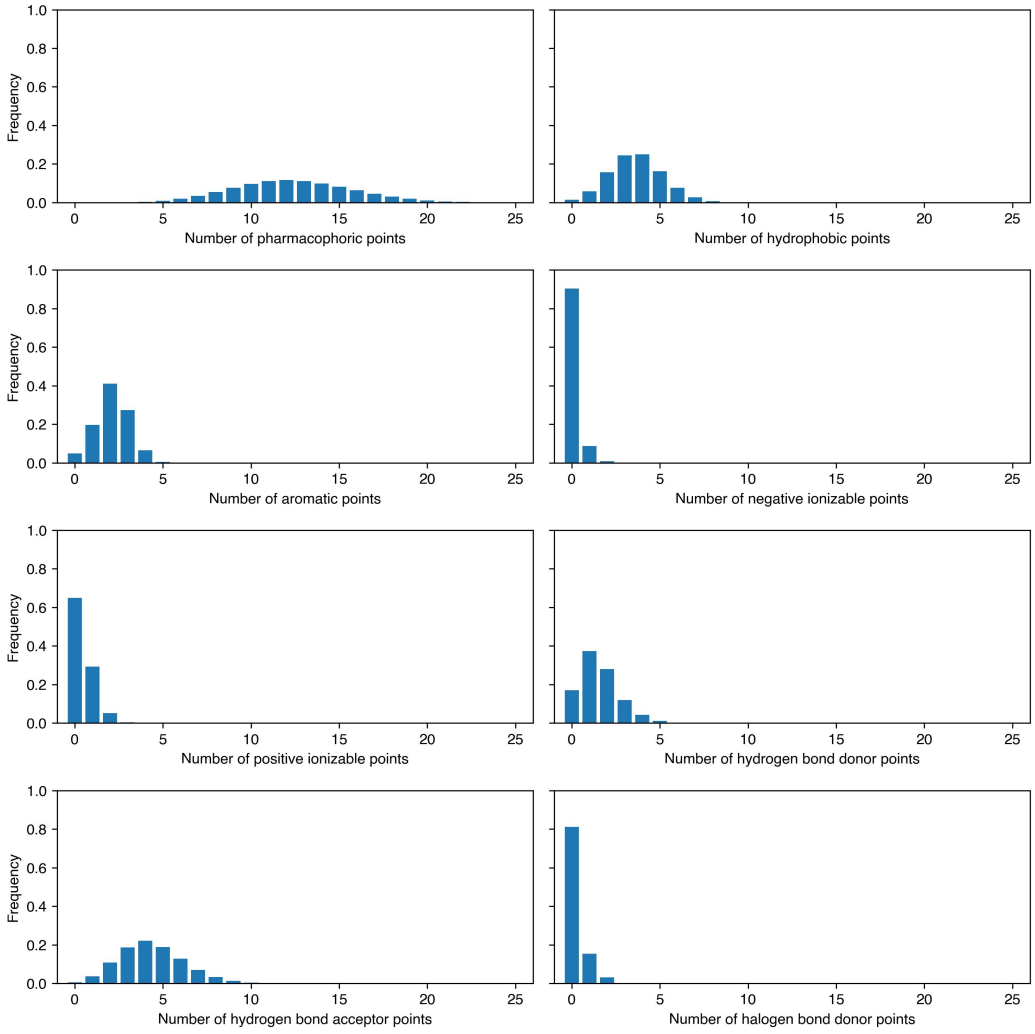

Figure 6: Pharmacophoric point statistics of the training data. The respective histograms display the total number of pharmacophoric points and the number of points of specific types per pharmacophore in the training data. The complete training dataset contains 1,217,361 distinct pharmacophores.

A.2 AUGMENTATION MODULE

The augmentation module receives the initial pharmacophore $\mathbf{x_0} = [\mathbf{h_0}, \mathbf{r_0}]$, with the initial OHE feature matrix $\mathbf{h_0}$ and the Cartesian coordinates $\mathbf{r_0}$. Edge attributes of the complete graph were calculated from the pair-wise distances between nodes after modifying the input according to the augmentation strategy, which combines random node deletion and random node displacement. The module outputs the modified tuple $\mathbf{x} = [\mathbf{h}, \mathbf{e}]$ with the feature matrix $\mathbf{h}$ and the edge attributes $\mathbf{e}$.

Node deletion Random node deletion involved removing at least one node, with the upper bound determined by the cardinality of the set of nodes V_i of graph G_i. To ensure the output graph retained at least three nodes, the maximum number of deletable nodes was $|V_i| - 3$. The number of nodes to delete was drawn uniformly at random.

Node displacement There are two modes for the displacement of pharmacophoric points, displacement within the tolerance sphere, and displacement onto the surface of the tolerance sphere. For simplicity, we assumed the same tolerance sphere radius r_T across different pharmacophoric types. For displacement within the tolerance sphere, we calculated the coordinate displacement $(\Delta x, \Delta y, \Delta z)$ from spherical coordinates $\phi \sim \mathcal{U}(0, 2\pi)$ and $\cos\theta \sim \mathcal{U}(-1, 1)$, which were drawn at random from a uniform distribution:

$$\Delta x = \Delta r \sin\theta \cos\phi,\ \Delta y = \Delta r \sin\theta \sin\phi,\ \Delta z = \Delta r \cos\theta \tag{6}$$

where $\Delta r = r_T \sqrt[3]{u}$ and $u \sim \mathcal{U}(0, 1)$. Displacement of the nodes onto the tolerance sphere surface was achieved by calculating the mean of the positions of the pharmacophoric points, $\mu = \frac{1}{n} \sum_{i=1}^{|P|} \mathbf{r}_i$, and displacing each point p_i by r_T in the direction $\mathbf{r}_i - \mu$. The displacement away from the center ensures that displacement directions do not cancel each other.

A.3 MESSAGE PASSING NEURAL NETWORK

Convolution on irregular domains like graphs is formulated as message passing, which can generally be described as:

$$\mathbf{h}_i^{(k)} = \gamma^{(k)}(\mathbf{h}_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} \phi^{(k)}(\mathbf{h}_i^{(k-1)}, \mathbf{h}_j^{(k-1)}, \mathbf{e}_{ij})) \tag{7}$$

where $\mathbf{h}_i^{(k)} \in \mathbb{R}^{F'}$ denotes the node features of node i at layer k, $\mathbf{h}_i^{(k-1)} \in \mathbb{R}^F$ denotes the node features of node i at layer $k-1$, $\mathbf{e}_{ij} \in \mathbb{R}^D$ the edge features of the edge from node i to node j, $\gamma^{(k)}$ and $\phi^{(k)}$ are parameterized, differentiable functions, and $\bigoplus$ is an aggregation operator like, *e. g.*, the summation operator (Fey & Lenssen, 2019). In our encoder architecture, we employed the following edge-conditioned convolution operator, which was proposed both by Gilmer et al. (2017); Simonovsky & Komodakis (2017):

$$\mathbf{h}_i^{(k)} = \Theta \mathbf{h}_i^{(k-1)} + \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^{(k-1)} \cdot \psi_\Theta(\mathbf{e}_{ij}) \tag{8}$$

where $\Theta \in \mathbb{R}^{F \times F'}$ denotes learnable weights and $\psi_\Theta(\cdot) : \mathbb{R}^D \to \mathbb{R}^{F \times F'}$ denotes a neural network, in our case an MLP with one hidden layer. These transformations map node features $\mathbf{h}$ into a latent representation that combines pharmacophoric types with distance encodings.

A.4 ENCODER IMPLEMENTATION

The encoder was implemented as a GNN $f_\Theta : \mathcal{G} \to \mathbb{R}_+^D$ that maps a given graph G to the abstract representation vector $\mathbf{z} \in \mathbb{R}_+^D$. The architecture is comprised of an initial embedding block, three subsequent convolution blocks, followed by a pooling layer, and a projection block.

Embedding block The embedding block receives the pharmacophore graph G_i as the tuple $\mathbf{x}_i = [\mathbf{h}_i, \mathbf{e}_i]$, with the OHE feature matrix $\mathbf{h}_i$ and the edge attributes $\mathbf{e}_i$. Initial node feature embeddings are created from the OHE features with a fully-connected (FC) dense layer with learnable weights $\mathbf{W}$ and bias $\mathbf{b}$:

$$\mathbf{h}_i \leftarrow \mathbf{W}\mathbf{h}_i + \mathbf{b} \tag{9}$$

Convolution block The convolution block consists of a graph convolution layer, which is implemented as edge-conditioned convolution operator (NNConv), the update rule is described in Section A.3. The network further consists of batch normalization layers (BN), GELU activation functions, and dropout layers. The hidden representation $\mathbf{h}_i^l$ of graph G_i is updated at block l as follows:

$$[\mathbf{h}_i^l, \mathbf{e}_i] \rightarrow \{\text{NNConv} \rightarrow \text{BN} \rightarrow \text{GELU} \rightarrow \text{concat}(\mathbf{h}_i^{l'}, \mathbf{h}_i^l) \rightarrow \text{dropout}\} \rightarrow \mathbf{h}_i^{l+1} \tag{10}$$

where $\mathbf{h}_i^{l'}$ represents the latent representation after activation. Updating the feature matrix l times yields the final node representations of the pharmacophoric points.

Pooling layer We employed additive pooling for graph-level read-out $\mathbf{r}_i$, which aggregates the set of $|V|$ node representations $\{\mathbf{h}_1, ..., \mathbf{h}_{|V|}\}_i$ of a Graph G_i by element-wise summation:

$$\mathbf{q}_i = \sum_{k=1}^{|V|} \mathbf{h}_k \tag{11}$$

Projection block The projection block maps the graph-level read-out to the positive real number space and is implemented as a multi-layer perceptron $\text{MLP} : \mathbb{R}^d \rightarrow \mathbb{R}_+^D$, where d is the dimension of the vector representation before and D the dimension after the projection. The block consists of k sequential layers of FC layers, BN, ReLU activation, and dropout:

$$\mathbf{q}_i^k \rightarrow \{\text{FC} \rightarrow \text{BN} \rightarrow \text{ReLU} \rightarrow \text{Dropout}\} \rightarrow \mathbf{q}_i^{k+1} \tag{12}$$

The final layer is a FC layer without bias and with positive weights, only:

$$\mathbf{z}_i \leftarrow \text{abs}(\mathbf{W})\mathbf{q}_i \tag{13}$$

Matrix multiplication of the positive learnable weights $\mathbf{W}$ and the output of the last ReLU activation function produces the final representation $\mathbf{z}_i \in \mathbb{R}_+^D$.

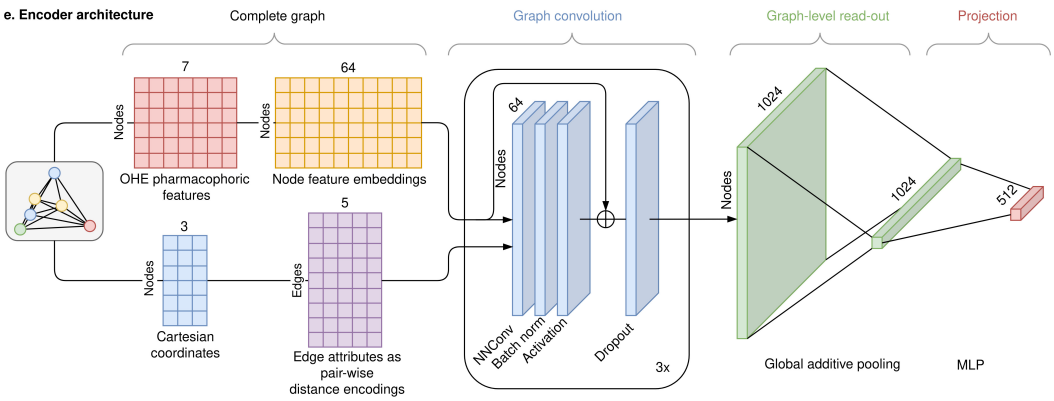

Figure 7: Architecture of the GNN encoder model.

A.5 Model implementation and training

Implementation dependencies The GNN was implemented in Python 3.10 with PyTorch (v2.0.1) and the PyTorch Geometric library (v2.3.1) (Fey & Lenssen, 2019). Both, model and dataset, were implemented within the PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019) framework (v2.1.0). Model training was monitored with Tensorboard (v2.13.0). CDPKit (v1.1.1) was employed for chemical data processing. Software was installed and executed on a Rocky Linux (v9.4) system with x86-64 architecture.

Model training Training was performed on a single NVIDIA GeForce 3090 RTX graphics unit with 24 GB GDDR6X. Training runs were performed for a maximum of 500 epochs with a batch size of 256 pharmacophore graphs. Curriculum learning was applied by gradual enrichment of the dataset with increasingly larger pharmacophore graphs. At training start, only pharmacophore graphs with 4 nodes were considered. After 10 subsequent epochs without considerable minimization of the loss function, pharmacophore graphs with one additional node were added to the training data. The loss function was minimized with the Adam optimizer, we further applied gradient clipping. A training run on the full dataset took approximately 48 hours with the above hardware specifications.

Hyperparameter tuning & model selection Hyperparameters were optimized through random parameter selection, the tested ranges are summarized in Table 3. Unlabeled data was split into training and validation data with a 98:2 ratio. Training runs were compared using the AUROC value on the validation data. This was calculated by treating the positive and negative pairs as binary labels, and the predictions were based on their respective order embedding penalty, which was calculated with Equation (3). Hyperparameter optimization was performed on a reduced dataset with 100,000 graphs, which took approximately 5 hours per run. The best performing models were retrained on the full dataset. The hyperparameters of the final encoder model are summarized in Table 2. After model selection, the final model performance was tested on virtual screening datasets.

Table 2: Hyperparameters of the best performing encoder model

Hyperparameter	
batch size	256
dropout convolution block	0.2
dropout projection block	0.2
max. epochs	500
hidden dimension convolution block	64
hidden dimension projection block	1024
output dimension convolution block	1024
output dimension projection block	512
learning rate optimizer	0.001
margin for negative pairs	100.0
number of convolution blocks	3
depth of the projector MLP	3
edge attributes dimension	5
sampling sphere radius positive pairs	1.5
sampling surface radius negative pairs	1.5

Table 3: Tested hyperparameter ranges for model training.

Hyperparameter	Parameter range
dropout	[0.2, 0.3, 0.4, 0.5]
margin for negative pairs	[0.1, 0.5, 1, 2, 5, 10, 100, 1000]
output dimension projection block	[64, 128, 256, 512, 1024]
displacement sphere radius r_T of positive pairs	[0.25, 0.5, 1.0, 1.5]

Table 4: Tested hyperparameter ranges for ablation studies.

Parameter	Parameter range	Validation AUROC
Tolerance radius	[0.0, 0.5, 1.0, 1.5, 2.0]	[0.91, 0.93, 0.94, 0.94, 0.93]
Encoder dimension	[8, 16, 32, 64, 96]	[0.92, 0.93, 0.94, 0.94, 0.94]
Embedding dimension	[32, 64, 128, 256, 512, 1024]	[0.91, 0.93, 0.94, 0.94, 0.94, 0.93]
Skip-connection	[dense, res, none]	[0.94, 0.93, 0.74]
GNN Layer	[NNConv, GINE, CFConv, GAT]	[0.94, 0.94, 0.94, 0.93]
Projector layers	[1, 2, 3]	[0.94, 0.94, 0.94]
Convolution layers	[1, 2, 3]	[0.94, 0.94, 0.94]
Margin	[0.01, 0.1, 1, 2, 5, 10, 100, 1000]	[0.88, 0.90, 0.92, 0.92, 0.93, 0.93, 0.94, 0.94]

A.6 VIRTUAL SCREENING

DUD-E dataset details General information about the DUD-E targets is summarized in Table 5. For each target we downloaded the receptor structure from the PDB and created the corresponding interaction pharmacophore with the CDPKit. Vector features were converted into undirected pharmacophoric points with LigandScout (Wolber & Langer, 2005). The resulting pharmacophore queries (Figure 8) were used in our virtual screening experiments.

Table 5: DUD-E targets that were selected for bechmarking experiments in this study.

Target	PDB code	Ligand ID	Active Ligands	Active Conformations	Decoy Ligands	Decoy Conformations	Query Points
ACES	1e66	HUX	451	10048	26198	567122	6
ADA	2e1w	FR6	90	2166	5448	125035	7
ANDR	2am9	TES	269	3039	14333	211968	6
EGFR	2rgp	HYZ	541	12468	35001	755017	7
FA10	3kl6	443	537	13343	28149	638831	5
KIT	3g0e	B49	166	3703	10438	224364	5
PLK1	2owb	626	107	2531	6794	152999	6
SRC	3el8	PD5	523	11868	34407	737864	6
THRB	1ype	UIP	461	11494	26894	626722	7
UROK	1sqt	UI3	162	3450	9837	199204	6

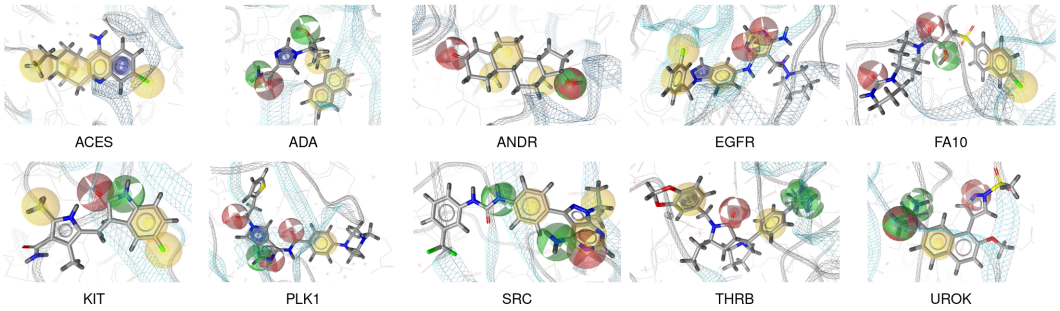

Figure 8: Structure-based pharmacophore queries of ten targets of the DUD-E benchmark dataset.

CDPKit alignment scoring function The CDPKit implements alignment as a clique-detection algorithm and computes a rigid-body transformation *via* Kabsch's algorithm to align the pharmacophore query P_Q to the pharmacophore target P_T. The goodness of fit is evaluated with a geometric scoring function $S : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}_+$:

$$S(P_Q, P_T) = S_{MFP}(P_Q, P_T) + S_{Geom}(P_Q, P_T) \tag{14}$$

where $S_{MFP} : \mathcal{P} \times \mathcal{P} \to \mathbb{Z}_+$ counts the number of matched feature pairs and $S_{Geom} : \mathcal{P} \times \mathcal{P} \to [0, 1)$ evaluates their geometric fit.

Runtime measurement We measured alignment runtimes using the psdscreen tool from the CDPKit with 128 threads on an AMD EPYC 7713 64-Core Processor, while embedding and matching runtimes with PharmacoMatch were recorded using an NVIDIA GeForce RTX 3090 GPU with 24 GB GDDR6X. Runtime per pharmacophore was estimated by dividing the total runtime by the number of pharmacophores in each dataset, with the final estimate taken as the mean of ten runs. The results report the mean and standard deviation of these estimates across all ten datasets.

ROC curves The performance metrics of our virtual screening experiments are derived from the ROC curves presented in Figures 9 and Figure 10.

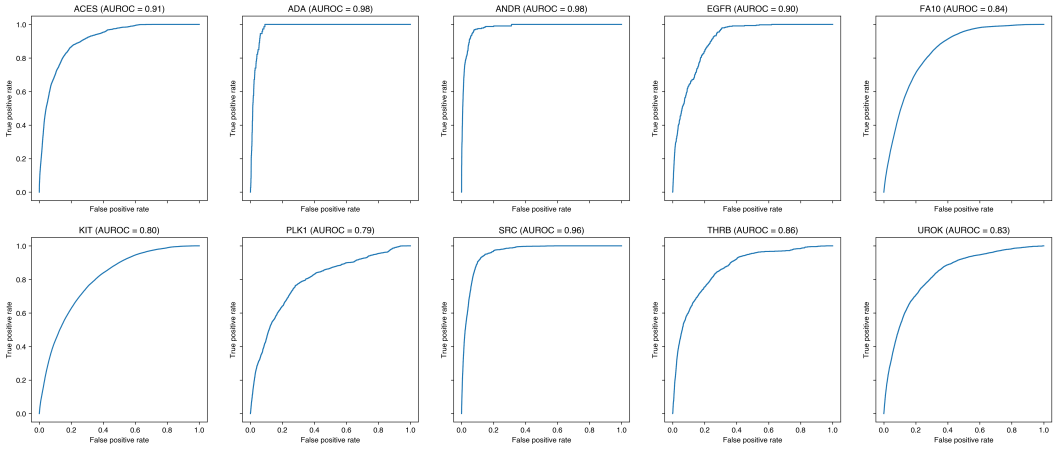

Figure 9: Performance comparison of PharmacoMatch and the alignment algorithm. The ROC-curves display the agreement of the hitlist ranking of the two algorithms for ten targets of the DUD-E benchmark dataset.

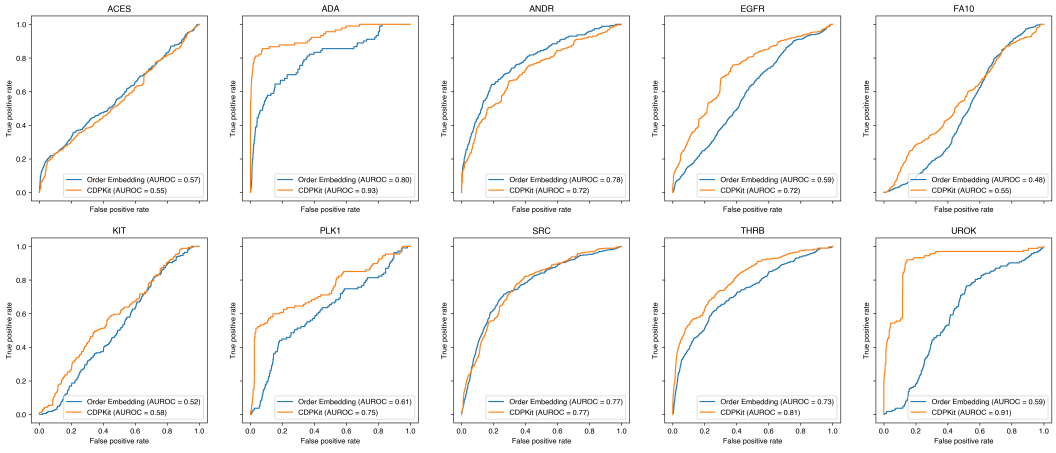

Figure 10: Absolute screening performance of PharmacoMatch and the alignment algorithm performance for ten targets of the DUD-E benchmark dataset. The pharmacophore queries were generated from the respective PDB ligand-receptor structures.

A.7 EMBEDDING SPACE VISUALIZATION

UMAP visualization UMAP embeddings for visualization plots were calculated with the *UMAP* Python library. The 'metric' parameter was set to Manhattan distance, all other parameters are the default settings of the implementation. We tested a range of hyperparameters to ensure that the visualization results are not sensitive to parameter selection.

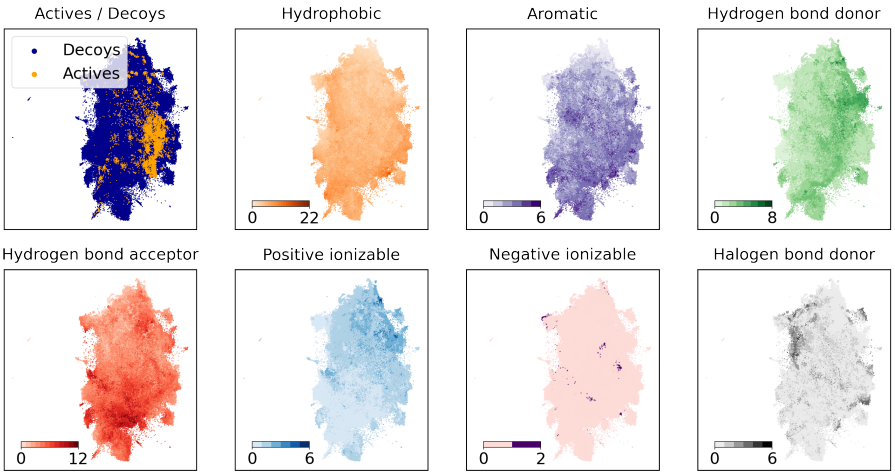

Figure 11: UMAP visualization of the vector embeddings of the ACES target.

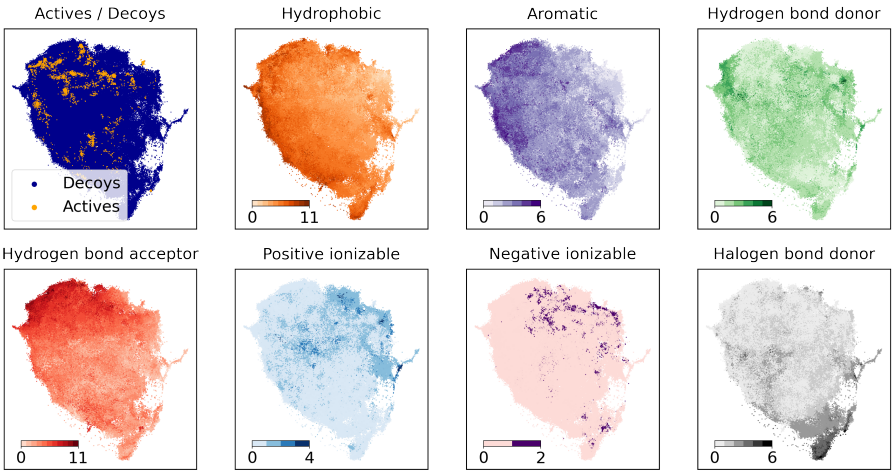

Figure 12: UMAP visualization of the vector embeddings of the ANDR target.

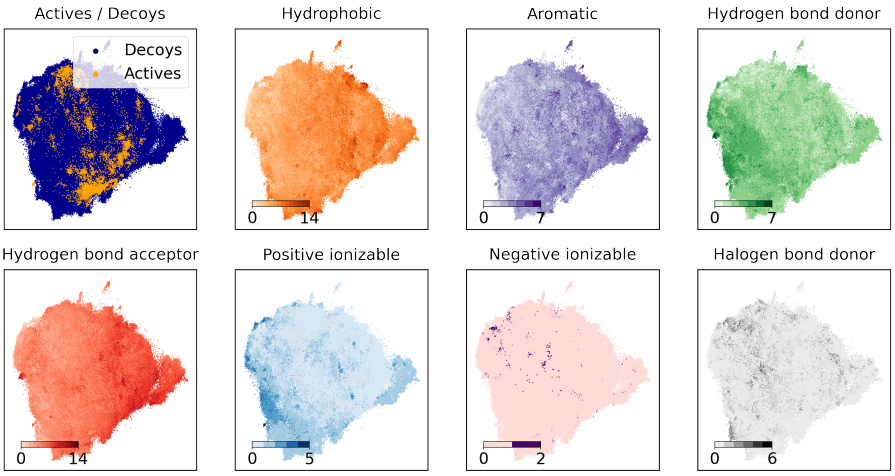

Figure 13: UMAP visualization of the vector embeddings of the EGFR target.

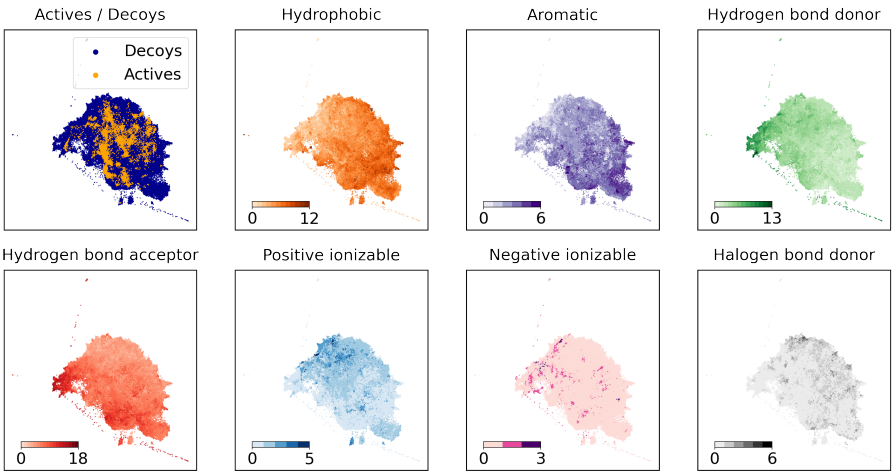

Figure 14: UMAP visualization of the vector embeddings of the FA10 target.

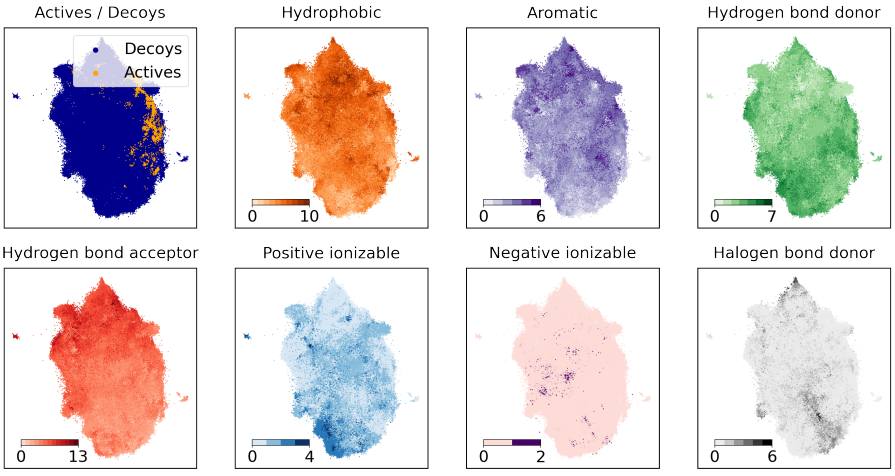

Figure 15: UMAP visualization of the vector embeddings of the KIT target.

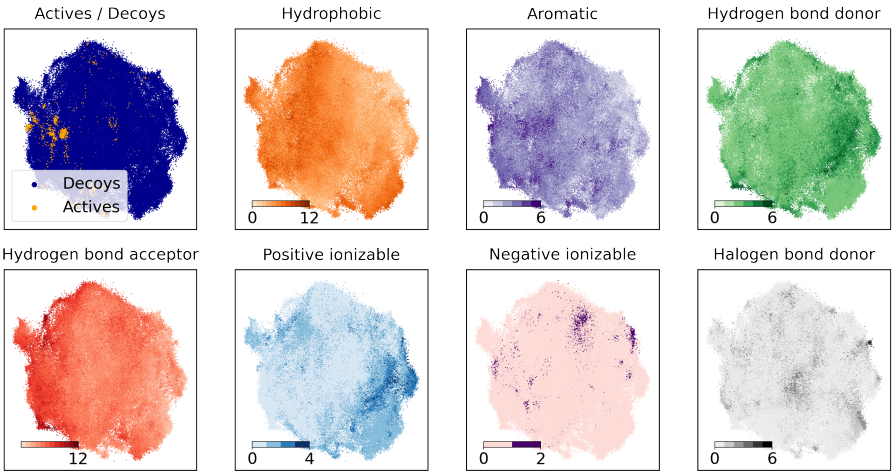

Figure 16: UMAP visualization of the vector embeddings of the PLK1 target.

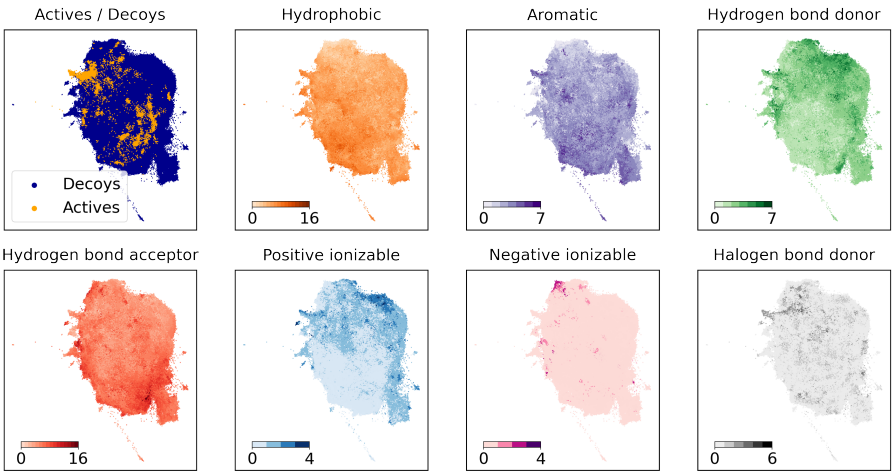

Figure 17: UMAP visualization of the vector embeddings of the SRC target.

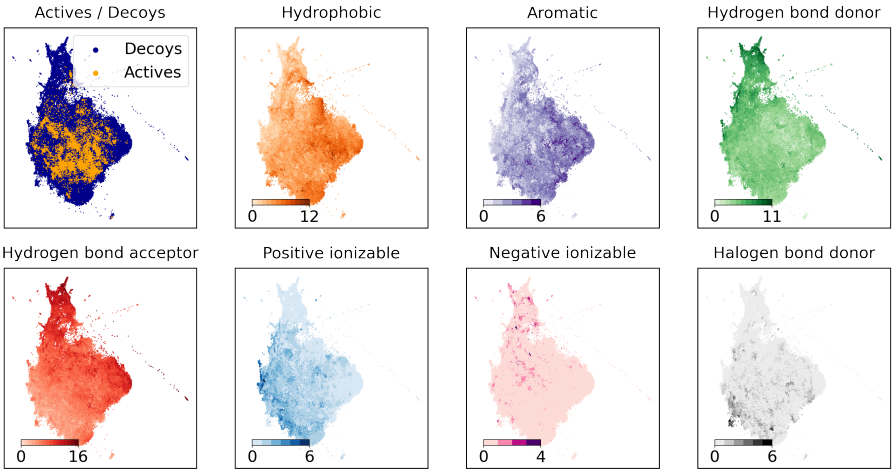

Figure 18: UMAP visualization of the vector embeddings of the THRB target.

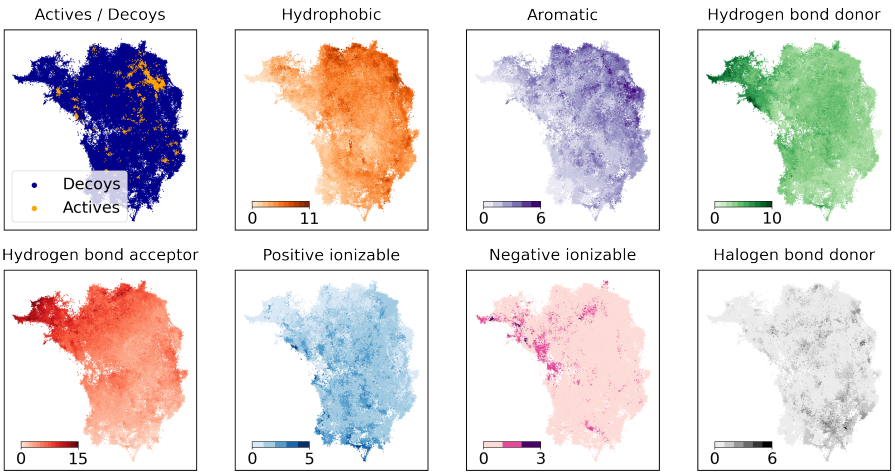

Figure 19: UMAP visualization of the vector embeddings of the UROK target.