Appendix

A Topological Data Analysis (TDA)

A.1 Single Parameter Persistent Homology

Here, we give further details on single parameter persistent homology. To sum up, PH machinery is a 3-step process. The first step is the *filtration* step, where one can integrate the domain information to the process. The second step is the *persistence diagrams*, where the machinery records the evolution of topological features (birth/death times) in the filtration, sequence of the simplicial complexes. The final step is the *vectorization* (fingerprinting) where one can convert these records to a function or vector to be used in suitable ML models.

Constructing Filtrations: As PH is basically the machinery to keep track of the evolution of topological features in a sequence, the most important step is the construction of the sequence $\widehat{\mathcal{G}}_1 \subseteq \ldots \subseteq \widehat{\mathcal{G}}_N$. This is the key step where one can inject the valuable domain information to the PH process by using important domain functions (e.g., atomic mass, partial charge). While there are various filtration techniques used for PH machinery on graphs [5, 41], we will focus on two most common methods: *Sublevel/superlevel filtration* and *Vietoris-Rips (VR) filtration*.

For a given unweighted graph (compound) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{v_1, \ldots, v_m\}$ the set of nodes (atoms) and $\mathcal{E} = \{e_{rs}\}$ the set of edges (bonds), the most common technique is to use a filtering function $f : \mathcal{V} \to \mathbb{R}$ with a choice of thresholds $\mathcal{I} = \{\alpha_i\}$ where $\alpha_1 = \min_{v \in \mathcal{V}} f(v) < \alpha_2 < \ldots < \alpha_N = \max_{v \in \mathcal{V}} f(v)$. For $\alpha_i \in \mathcal{I}$, let $\mathcal{V}_i = \{v_r \in \mathcal{V} \mid f(v_r) \leq \alpha_i\}$ (sublevel sets for f). Here, in VS problem, this filtering function f can be atomic mass, partial charge, bond type, electron affinity, ionization energy or another important function representing chemical properties of the atoms. One can also use the natural graph induced functions like node degree, betweenness, etc. Let $\mathcal{G}_i$ be the induced subgraph of $\mathcal{G}$ by $\mathcal{V}_i$, i.e., $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ where $\mathcal{E}_i = \{e_{rs} \in \mathcal{E} \mid v_r, v_s \in \mathcal{V}_i\}$. This process yields a nested sequence of subgraphs $\mathcal{G}_1 \subset \mathcal{G}_2 \subset \ldots \subset \mathcal{G}_N = \mathcal{G}$. To obtain a filtration, next step is to assign a simplicial complex $\widehat{\mathcal{G}}_i$ to the subgraph $\mathcal{G}_i$. One of the most common techniques is the clique complexes [5]. The clique complex $\widehat{\mathcal{G}}$ is a simplicial complex obtained from $\mathcal{G}$ by assigning (filling with) a k-simplex to each complete $(k+1)$-complete subgraph in $\mathcal{G}$, e.g., a 3-clique, a complete 3-subgraph, in $\mathcal{G}$ will be filled with a 2-simplex (triangle). This technique is generally known as *sublevel filtration* with clique complexes. As $f(v_i) \leq \alpha_i$ condition in the construction gives sublevel filtration, one can similarly use $f(v_i) \geq \alpha_i$ condition to define *superlevel filtration*. Similarly, for a weighted graph (bond strength), sublevel filtration on edge weights provides corresponding filtration reflecting the domain information stored in the edge weights [5].

While sublevel/superlevel filtration with clique complexes is computationally cheaper and more common in practise, in this paper, we will essentially use a distance-based filtration technique called *Vietoris-Rips (VR) filtration* where the pairwise distances between the nodes play key role. This technique is computationally more expensive, but gives much finer information about the graph's intrinsic properties [2]. For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define the distance between $d(v_r, v_s) = d_{rs}$ where d_{rs} is the smallest number of edges required to get from v_r to v_s in $\mathcal{G}$. Then, let $\Gamma_n = (\mathcal{V}, \mathcal{E}_n)$ be the graph where $\mathcal{E}_n = \{e_{rs} \mid d_{rs} \leq n\}$, i.e. $\mathcal{E}_0 = \emptyset$ and $\mathcal{E}_1 = \mathcal{E}$ with $\Gamma_0 = \mathcal{V}$ and $\Gamma_1 = \mathcal{G}$. In other words, we start with the nodes of $\mathcal{G}$, then for any pair of vertices v_r, v_s with distance $d_{rs} \leq n$ in $\mathcal{G}$, we add an edge e_{rs} to the graph Γ_n. Then, define the simplicial complex $\Delta_n = \widehat{\Gamma}_n$, the clique complex of Γ_n. This defines a filtration $\Delta_0 \subset \Delta_1 \subset \cdots \subset \Delta_K$ where $K = \max d_{rs}$, i.e. the distance between farthest two nodes in the graph $\mathcal{G}$. Hence, for $n \geq K$, $\Delta_n = \Delta_K$ which is a $(m-1)$-simplex as Γ_K is complete m-graph where $|\mathcal{V}| = m$. In particular, in this setting, we consider the vertex set $\mathcal{V}$ as a point cloud where the distances between the points induced from the graph $\mathcal{G}$. In graph setting, VR-filtration is also known as *power filtration* as the graph Γ_n is also called $\mathcal{G}^n$, n^{th} power of $\mathcal{G}$.

Persistence Diagrams: The second step in PH process is to obtain persistence diagrams (PD) for the filtration $\Delta_0 \subset \Delta_1 \subset \cdots \subset \Delta_K$. As explained in Section 3.1, PDs are collection of 2-tuples, marking the birth and death times of the topological features appearing in the filtration, i.e. $\mathrm{PD_k}(\mathcal{G}) = \{(b_\sigma, d_\sigma) \mid \sigma \in H_k(\Delta_i) \text{ for } b_\sigma \leq i < d_\sigma\}$. This step is pretty standard and there are various software libraries for this task [74].

Vectorizations (Fingerprinting): While PH extracts hidden shape patterns from data as persistence diagrams (PD), PDs being collection of points in $\mathbb{R}^2$ by itself are not very practical for statistical and ML purposes. Instead, the common techniques are by faithfully representing PDs as kernels [54] or vectorizations [39]. One can consider this step as converting PDs into a useful format to be used in ML process as fingerprints of the dataset. This provides a practical way to use the outputs of PH in real life applications. *Single Persistence Vectorizations* transform obtained PH information (PDs) into a function or a feature vector form which are much more suitable for ML tools than PDs. Common single persistence (SP) vectorization methods are Persistence Images [3], Persistence Landscapes [13], Silhouettes [21], Betti Curves and various Persistence Curves [25]. These vectorizations define a single variable or multivariable functions out of PDs, which can be used as fixed size $1D$ or $2D$ vectors in applications, i.e $1 \times n$ vectors or $m \times n$ vectors. For example, a Betti curve for a PD with n thresholds can also be expressed as $1 \times n$ size vectors. Similarly, Persistence Images is an example of $2D$ vectors with the chosen resolution (grid) size. See the examples given in Section B.4 for further details.

B Multiparameter Persistence (MP) Fingerprints

B.1 Stability of MP Fingerprints

Stability of Single Persistence Vectorizations: A given PD vectorization φ can be considered as a map from space of persistence diagrams to space of functions, and the stability intuitively represents the continuity of this operator. In other words, stability question is whether a small perturbation in PD cause a big change in the vectorization or not. To make this question meaningful, one needs to define what "small perturbation" means in this context, i.e., a metric in the space of persistence diagrams. The most common such metric is called *Wasserstein distance* (or matching distance) which is defined as follows.

Let $PD(\mathcal{X}^+)$ and $PD(\mathcal{X}^-)$ be persistence diagrams two datasets $\mathcal{X}^+$ and $\mathcal{X}^-$ (We omit the dimensions in PDs). Let $PD(\mathcal{X}^+) = \{q_j^+\} \cup \Delta^+$ and $PD(\mathcal{X}^-) = \{q_l^-\} \cup \Delta^-$ where $\Delta^\pm$ represents the diagonal (representing trivial cycles) with infinite multiplicity. Here, $q_j^+ = (b_j^+, d_j^+) \in PD(\mathcal{X}^+)$ represents the birth and death times of a topological feature σ_j in $\mathcal{X}^+$. Let $\phi : PD(\mathcal{X}^+) \to PD(\mathcal{X}^-)$ represent a bijection (matching). With the existence of the diagonal $\Delta^\pm$ in both sides, we make sure the existence of these bijections even if the cardinalities $|\{q_j^+\}|$ and $|\{q_l^-\}|$ are different.

Definition B.1 *Let $PD(\mathcal{X}^\pm)$ be persistence diagrams of the datasets $\mathcal{X}^\pm$, and $\mathbf{M} = \{\phi\}$ represent the space of matchings as described above. Then, the p^{th} Wasserstein distance $\mathcal{W}_p$ defined as*

$$\mathcal{W}_p(PD(\mathcal{X}^+), PD(\mathcal{X}^-)) = \min_{\phi \in \mathbf{M}} \left(\sum_j \|q_j^+ - \phi(q_j^+)\|_\infty^p \right)^{\frac{1}{p}}, \quad p \in \mathbb{Z}^+.$$

Now, we define stability of vectorizations. A vectorization can be considered as an operator from space of persistence diagrams $\mathbf{P}$ to space of functions (or vectors) $\mathbf{Y}$, e.g., $\Psi : \mathbf{P} \to \mathbf{Y}$. In particular, when Ψ is persistence landscape, $\mathbf{Y} = \mathcal{C}([0, K], \mathbb{R})$ and when Ψ is Betti summary, then $\mathbf{Y} = \mathbb{R}^m$ (See MP Examples in Section B.4) Stability of vectorization Ψ basically corresponds to the continuity of Ψ as an operator. Let $\mathrm{d}(.,.)$ be a suitable metric on the space of vectorizations used. Then, we define the stability of Ψ as follows:

Definition B.2 *Let $\Psi : \mathbf{P} \to \mathbf{Y}$ be a vectorization for single persistence diagrams. Let $\mathcal{W}_p, \mathrm{d}$ be the metrics on $\mathbf{P}$ and $\mathbf{Y}$ respectively as described above. Let $\psi^\pm = \Psi(PD(\mathcal{X}^\pm)) \in \mathbf{Y}$. Then, Ψ is called stable if*

$$\mathrm{d}(\psi^+, \psi^-) \leq C \cdot \mathcal{W}_{p_\Psi}(PD(\mathcal{X}^+), PD(\mathcal{X}^-))$$

Here, the constant $C > 0$ is independent of $\mathcal{X}^\pm$. This stability inequality interprets as the changes in the vectorizations are bounded by the changes in PDs. Two nearby persistence diagrams are represented by nearby vectorizations. If a given vectorization φ holds such a stability inequality for some d and $\mathcal{W}_p$, we call φ a *stable vectorization* [7]. Persistence Landscapes [13], Persistence Images [3], Stabilized Betti Curves [48] and several Persistence curves [25] are among well-known examples of stable vectorizations.

Now, we are ready to prove the stability of MP Fingerprints given in Section 4.1

Let $\mathcal{G}^+ = (\mathcal{V}^+, \mathcal{E}^+)$ and $\mathcal{G}^- = (\mathcal{V}^-, \mathcal{E}^-)$ be two graphs. Let φ be a stable SP vectorization with the stability equation

$$\mathrm{d}(\varphi(PD(\mathcal{G}^+)), \varphi(PD(\mathcal{G}^-))) \leq C_\varphi \cdot \mathcal{W}_{p_\varphi}(PD(\mathcal{G}^+), PD(\mathcal{G}^-)) \tag{2}$$

for some $1 \leq p_\varphi \leq \infty$. Here, $\varphi(\mathcal{G}^\pm)$ represent the corresponding vectorizations for $PD(\mathcal{G}^\pm)$ and $\mathcal{W}_p$ represents Wasserstein-p distance as defined in Section B.1.

Now, let $f : \mathcal{V}^\pm \to \mathbb{R}$ be a filtering function with threshold set $\{\alpha_i\}_{i=1}^m$. Then, define the sublevel vertex sets $\mathcal{V}_i^\pm = \{v_r \in \mathcal{V}^\pm \mid f(v_r) \leq \alpha_i\}$. For each $\mathcal{V}_i^\pm$, construct the induced VR-filtration $\Delta_{i0}^\pm \subset \Delta_{i1}^\pm \subset \cdots \subset \Delta_{iK}^\pm$ as before. For each $1 \leq i_0 \leq m$, we will have persistence diagram $PD(\mathcal{V}_{i_0}^\pm)$ of the filtration $\{\Delta_{i_0 k}^\pm\}$.

We define the induced matching distance between the multiple persistence diagrams as

$$\mathbf{D}_{p,f}(\mathcal{G}^+, \mathcal{G}^-) = \sum_{i=1}^m \mathcal{W}_p(PD(\mathcal{V}_i^+), PD(\mathcal{V}_i^-)). \tag{3}$$

Now, we define the distance between induced MP Fingerprints as

$$\mathfrak{D}_f(\mathbf{M}_\varphi(\mathcal{G}^+), \mathbf{M}_\varphi(\mathcal{G}^-)) = \sum_{i=1}^m \mathrm{d}(\varphi(PD(\mathcal{V}_i^+)), \varphi(PD(\mathcal{V}_i^-))) \tag{4}$$

Theorem B.1 *Let φ be a stable SP vectorization. Then, the induced MP Fingerprint $\mathbf{M}_\varphi$ is also stable, i.e., with the notation above, there exists $\widehat{C}_\varphi > 0$ such that for any pair of graphs $\mathcal{G}^+$ and $\mathcal{G}^-$, we have the following inequality.*

$$\mathfrak{D}(\mathbf{M}_\varphi(\mathcal{G}^+), \mathbf{M}_\varphi(\mathcal{G}^-)) \leq \widehat{C}_\varphi \cdot \mathbf{D}_{p_\varphi}(\{PD(\mathcal{G}^+)\}, \{PD(\mathcal{G}^-)\})$$

Proof: As φ is a stable SP vectorization, by Equation 2, for any $1 \leq i \leq m$, we have $\mathrm{d}(\varphi(PD(\mathcal{V}_i^+)), \varphi(PD(\mathcal{V}_i^+))) \leq C_\varphi \cdot \mathcal{W}_{p_\varphi}(PD(\mathcal{V}_i^+), PD(\mathcal{V}_i^-))$ for some $C_\varphi > 0$, where $\mathcal{W}_{p_\varphi}$ is Wasserstein-p distance. Notice that the constant $C_\varphi > 0$ is independent of i. Hence,

$$\begin{aligned}
\mathfrak{D}(\mathbf{M}_\varphi(\mathcal{G}^+), \mathbf{M}_\varphi(\mathcal{G}^-)) &= \sum_{i=1}^m \mathrm{d}(\varphi(PD(\mathcal{V}_i^+)), \varphi(PD(\mathcal{V}_i^-))) \\
&\leq \sum_{i=1}^m C_\varphi \cdot \mathcal{W}_{p_\varphi}(PD(\mathcal{V}_i^+), PD(\mathcal{V}_i^-)) \\
&= C_\varphi \sum_{i=1}^m \mathcal{W}_{p_\varphi}(PD(\mathcal{V}_i^+), PD(\mathcal{V}_i^-)) \\
&= C_\varphi \cdot \mathbf{D}_{p_\varphi}(\mathcal{G}^+, \mathcal{G}^-)
\end{aligned}$$

where the first and last equalities are due to Equation 3 and Equation 4, while the inequality follows from Equation 2 which is true for any i. This concludes the proof of the theorem. □

B.2 MP Fingerprint for Other Types of Data

So far, to keep the exposition focused on VS setting, we described our construction only in the graph setup. However, our framework is suitable for various types of data. Let $\mathcal{X}$ be a an image data or a point cloud. Let $f : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{X} \to \mathbb{R}$ be two filtering functions on $\mathcal{X}$. e.g., grayscale function for image data, or density function on point cloud data.

Let $f : \mathcal{X} \to \mathbb{R}$ be the filtering function with threshold set $\{\alpha_i\}_1^m$. Let $\mathcal{X}_i = f^{-1}((-\infty, \alpha_i])$. Then, we get a filtering of $\mathcal{X}$ as nested subspaces $\mathcal{X}_1 \subset \mathcal{X}_2 \subset \cdots \subset \mathcal{X}_m = \mathcal{X}$. By using the second filtering function, we obtain finer filtrations for each subspace $\mathcal{X}_i$ where $1 \leq i \leq m$. In particular, fix $1 \leq i_0 \leq m$ and let $\{\beta_j\}_{j=1}^n$ be the threshold set for the second filtering function g. Then, by restricting g to $\mathcal{X}_{i_0}$, we get a filtering function on X_{i_0}, i.e., $g : \mathcal{X}_{i_0} \to \mathbb{R}$ which produces filtering $\mathcal{X}_{i_0 1} \subset \mathcal{X}_{i_0 2} \subset \cdots \subset \mathcal{X}_{i_0 n} = \mathcal{X}_{i_0}$. By inducing a simplicial complex $\widehat{\mathcal{X}}_{i_0 j}$ for each $\mathcal{X}_{i_0 j}$, we

get a filtration $\widehat{\mathcal{X}}_{i_0 1} \subset \widehat{\mathcal{X}}_{i_0 2} \subset \cdots \subset \widehat{\mathcal{X}}_{i_0 n} = \widehat{\mathcal{X}}_{i_0}$. This filtration results in a persistence diagram (PD) $PD(\mathcal{X}_{i_0}, g)$. For each $1 \leq i \leq m$, we obtain $PD(\mathcal{X}_i, g)$. Note that after getting $\{\mathcal{X}_i\}_{i=1}^m$ via f, instead of using second filtering function g, one can apply Vietoris-Rips construction based on distance for each X_{i_0} in order to get a different filtration $\widehat{\mathcal{X}}_{i_0 1} \subset \widehat{\mathcal{X}}_{i_0 2} \subset \cdots \subset \widehat{\mathcal{X}}_{i_0 n} = \widehat{\mathcal{X}}_{i_0}$.

By using m PDs, we follow a similar route to define our MP Fingerprints. Let φ be a single persistence vectorization. By applying the chosen SP vectorization φ to each PD, we obtain a function $\varphi_i = \varphi(PD(\mathcal{X}_i, g))$ on the threshold domain $[\beta_1, \beta_n]$, which can be expresses as a $1D$ (or $2D$) vector in most cases (Section B.4). Let $\vec{\varphi}_i$ be the corresponding $1 \times k$ vector for the function φ_i. Define the corresponding MP Fingerprint $\mathbf{M}_\varphi$ as $\mathbf{M}_\varphi^i = \vec{\varphi}_i$ where $\mathbf{M}_\varphi^i$ is the i^{th} row of $\mathbf{M}_\varphi$. In particular, $\mathbf{M}_\varphi$ is a $2D$-vector (a matrix) of size $m \times k$ where m is the number of thresholds for the first filtering function f, and k is the length of the vector $\vec{\varphi}$.

B.3 $3D$ or higher dimensional MP Fingerprints:

If one wants to use two filtering functions and get $3D$-vectors as the topological fingerprint of the process, the idea is similar. Let $f, g : \mathcal{V} \to \mathbb{R}$ be two filtering functions with threshold sets $\{\alpha_i\}_{i=1}^m$ and $\{\beta_j\}_{j=1}^n$ respectively. Let $\mathcal{V}_{ij} = \{v_r \in \mathcal{V} \mid f(v_r) \leq \alpha_i \text{ and } g(v_r) \leq \beta_j\}$. Again, compute all the pairwise distances $d(v_r, v_s) = m_{rs}$ in $\mathcal{G}$ before defining simplicial complexes. Then, for each i_0, j_0, obtain a VR-filtration $\Delta_{i_0 j_0 0} \subseteq \Delta_{i_0 j_0 1} \ldots \subseteq \Delta_{i_0 j_0 K}$ for the vertex set $\mathcal{V}_{i_0 j_0}$ with distances $d(v_r, v_s) = m_{rs}$ in $\mathcal{G}$. Compute the persistence diagram $PD(\mathcal{V}_{i_0 j_0})$ for the filtration $\{\Delta_{i_0 j_0 k}\}$. This gives $m \times n$ persistence diagrams $\{PD(\mathcal{V}_{ij})\}$. After vectorization, we obtain a $3D$-vector of size $m \times n \times r$ as before.

B.4 Examples of MP Fingerprints

Here, we give explicit constructions of MP Fingerprints for most common SP vectorizations. As noted above, the framework is generalizable and can be applied to most SP vectorization methods. In all the examples below, we use the following setup: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, and let $f, g : \mathcal{V} \to \mathbb{R}$ be two filtering functions with threshold sets $\{\alpha_i\}_{i=1}^m$ and $\{\beta_j\}_{j=1}^n$ respectively. As explained above, we first apply sublevel filtering with f to get a sequence of nested subgraphs, $\mathcal{G}_1 \subseteq \ldots \subseteq \mathcal{G}_m = \mathcal{G}$. Then, for each $\mathcal{G}_i$, we apply sublevel filtration with g to get persistence diagram $PD(\mathcal{G}_i, g)$. Therefore, we will have m PDs. In the examples below, for a given SP vectorization φ, we explain how to obtain a vector $\vec{\varphi}(PD(\mathcal{G}_i, g))$, and define the corresponding MP Fingerprint $\mathbf{M}_\varphi$. Note that we skip the homology dimension (subscript k for $PD_k(\mathcal{G})$) in the discussion. In particular, for each dimension $k = 0, 1, \ldots$, we will have one MP Fingerprint $\mathbf{M}_\varphi(\mathcal{G})$ (a matrix) corresponding to $\{\vec{\varphi}(PD_k(\mathcal{G}_i, g))\}$. The most common dimensions are $k = 0$ and $k = 1$ in applications.

MP Landscapes Persistence Landscapes λ are one of the most common SP vectorizations introduced by [13]. For a given persistence diagram $PD(\mathcal{G}) = \{(b_i, d_i)\}$, λ produces a function $\lambda(\mathcal{G})$ by using generating functions Λ_i for each $(b_i, d_i) \in PD(\mathcal{G})$, i.e., $\Lambda_i : [b_i, d_i] \to \mathbb{R}$ is a piecewise linear function obtained by two line segments starting from $(b_i, 0)$ and $(d_i, 0)$ connecting to the same point $(\frac{b_i + d_i}{2}, \frac{b_i - d_i}{2})$. Then, the *Persistence Landscape* function $\lambda(\mathcal{G}) : [\epsilon_1, \epsilon_q] \to \mathbb{R}$ for $t \in [\epsilon_1, \epsilon_q]$ is defined as

$$\lambda(\mathcal{G})(t) = \max_i \Lambda_i(t),$$

where $\{\epsilon_k\}_1^q$ are thresholds for the filtration used.

Considering the piecewise linear structure of the function, $\lambda(\mathcal{G})$ is completely determined by its values at $2q - 1$ points, i.e., $\frac{b_i \pm d_i}{2} \in \{\epsilon_1, \epsilon_{1.5}, \epsilon_2, \epsilon_{2.5}, \ldots, \epsilon_q\}$ where $\epsilon_{k.5} = (\epsilon_k + \epsilon_{k+1})/2$. Hence, a vector of size $1 \times (2q - 1)$ whose entries the values of this function would suffice to capture all the information needed, i.e. $\vec{\lambda} = [\lambda(\epsilon_1)\ \lambda(\epsilon_{1.5})\ \lambda(\epsilon_2)\ \lambda(\epsilon_{2.5})\ \lambda(\epsilon_3)\ \ldots\ \lambda(\epsilon_q)]$

Considering we have threshold set $\{\beta_j\}_{j=1}^n$ for the second filtering function g, $\vec{\lambda}_i = \vec{\lambda}(PD(\mathcal{G}_i, g))$ will be a vector of size $1 \times 2n - 1$. Then, as $\mathbf{M}_\lambda^i = \vec{\lambda}_i$ for each $1 \leq i \leq m$, MP Landscape $\mathbf{M}_\lambda(\mathcal{G})$ would be a $2D$-vector (matrix) of size $m \times (2n - 1)$.

MP Persistence Images Next SP vectorization in our list is Persistence Images [3]. Different than the most SP vectorizations, Persistence Images produces $2D$-vectors. The idea is to capture the

location of the points in the persistence diagrams with a multivariable function by using the $2D$ Gaussian functions centered at these points. For $PD(\mathcal{G}) = \{(b_i, d_i)\}$, let ϕ_i represent a $2D$-Gaussian centered at the point $(b_i, d_i) \in \mathbb{R}^2$. Then, one defines a multivariable function, *Persistence Surface*, $\widetilde{\mu} = \sum_i w_i \phi_i$ where w_i is the weight, mostly a function of the life span $d_i - b_i$. To represent this multivariable function as a $2D$-vector, one defines a $k \times l$ grid (resolution size) on the domain of $\widetilde{\mu}$, i.e., threshold domain of $PD(\mathcal{G})$. Then, one obtains the *Persistence Image*, a $2D$-vector (matrix) $\vec{\mu} = [\mu_{rs}]$ of size $k \times l$ where $\mu_{rs} = \int_{\Delta_{rs}} \widetilde{\mu}(x, y)\, dxdy$ and Δ_{rs} is the corresponding pixel (rectangle) in the $k \times l$ grid.

This time, the resolution size $k \times l$ is independent of the number of thresholds used in the filtering, the choice of k and l is completely up to the user. Recall that by applying the first function f, we have the nested subgraphs $\{\mathcal{G}_i\}_{i=1}^m$. For each $\mathcal{G}_i$, the persistence diagram $PD(\mathcal{G}_i, g)$ obtained by sublevel filtration with g induces a $2D$ vector $\vec{\mu}_i = \vec{\mu}(PD(\mathcal{G}_i, g))$ of size $k \times l$. Then, define MP Persistence Image as $\mathbf{M}_\mu^i = \vec{\mu}_i$, where $\mathbf{M}_\mu^i$ is the i^{th}-floor of the matrix $\mathbf{M}_\mu$. Hence, $\mathbf{M}_\mu(\mathcal{G})$ would be a $3D$-vector of size $m \times k \times l$ where m is the number of thresholds for the first function f and $k \times l$ is the chosen resolution size for the Persistence Image $\vec{\mu}$.

MP Betti Summaries Next, we give an important family of SP vectorizations, Persistence Curves [25]. This is an umbrella term for several different SP vectorizations, i.e., Betti Curves, Life Entropy, Landscapes, et al. Our MP Fingerprint framework naturally adapts to all Persistence Curves to produce multidimensional vectorizations. As Persistence Curves produce a single variable function in general, they all can be represented as $1D$-vectors by choosing a suitable mesh size depending on the number of thresholds used. Here, we describe one of the most common Persistence Curves in detail, i.e., Betti Curves. It is straightforward to generalize the construction to other Persistence Curves.

Betti curves are one of the simplest SP vectorization as it gives the count of topological feature at a given threshold interval. In particular, $\beta_k(\Delta)$ is the total count of k-dimensional topological feature in the simplicial complex Δ, i.e., $\beta_k(\Delta) = rank(H_k(\Delta))$. Then, $\beta_k(\mathcal{G}) : [\epsilon_1, \epsilon_{q+1}] \to \mathbb{R}$ is a step function defined as

$$\beta_k(\mathcal{G})(t) = rank(H_k(\widehat{\mathcal{G}}_i))$$

for $t \in [\epsilon_i, \epsilon_{i+1})$, where $\{\epsilon_i\}_1^q$ represents the thresholds for the filtration used. Considering this is a step function where the function is constant for each interval $[\epsilon_i, \epsilon_{i+1})$, it can be perfectly represented by a vector of size $1 \times q$ as $\vec{\beta}(\mathcal{G}) = [\beta(1)\ \beta(2)\ \beta(3)\ \ldots\ \beta(q)]$.

Then, with the threshold set $\{\beta_j\}_{j=1}^n$ for the second filtering function g, $\vec{\beta}_i = \vec{\beta}(PD(\mathcal{G}_i, g))$ will be a vector of size $1 \times n$. Then, as $\mathbf{M}_\beta^i = \vec{\beta}_i$ for each $1 \leq i \leq m$, MP Betti Summary $\mathbf{M}_\beta(\mathcal{G})$ would be a $2D$-vector (matrix) of size $m \times n$. In particular, each entry $\mathbf{M}_\beta = [m_{ij}]$ is just the Betti number of the corresponding clique complex in the bifiltration $\{\widehat{\mathcal{G}}_{ij}\}$, i.e., $m_{ij} = \beta(\widehat{\mathcal{G}}_{ij})$. This matrix $\mathbf{M}_\beta$ is also called *bigraded Betti numbers* in the literature, and computationally much faster than other vectorizations [58, 51].

B.5 MP Vectorization with Other Filtrations

In our paper, other than the simple bifiltration explained in Section 4, we also used the following two filtrations. In the Vietoris-Rips filtration, we use graph geodesic (VR-filtration) as our natural slicing direction. The motivation for this choice is that *VR-filtration captures fine intrinsic structure of the graph by using the pairwise distances between the nodes (atoms).* With the weight filtration, we can utilize the bond strength of compounds effectively in our construction.

Vietoris-Rips filtration: Here, we describe our VR construction for $2D$ multipersistence. The construction can easily be extended to $3D$ or higher dimensions (See Appendix B.3). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, and let $f : \mathcal{V} \to \mathbb{R}$ be a filtering function (e.g., atomic mass, partial charge, bond type, electron affinity, ionization energy) with threshold sets $\{\alpha_i\}_{i=1}^m$. Let $\mathcal{V}_i = \{v_r \in \mathcal{V} \mid f(v_r) \leq \alpha_i\}$. This defines a hierarchy $\mathcal{V}_1 \subset \mathcal{V}_2 \subset \cdots \subset \mathcal{V}_m = \mathcal{V}$ among the nodes with respect to the function f.

Before constructing simplicial complexes, compute the distances between each node in graph $\mathcal{G}$, i.e., $d(v_r, v_s) = d_{rs}$ is the length of the shortest path from v_r to v_s where each edge has length 1. Let $K = \max d_{rs}$. Then, for each $1 \leq i_0 \leq m$, define VR-filtration for the vertex set $\mathcal{V}_{i_0}$ with the distances $d(v_r, v_s) = d_{rs}$ as described in Section 3.1, i.e., $\Delta_{i_0 0} \subseteq \Delta_{i_0 1} \subseteq \ldots \subseteq \Delta_{i_0 K}$ (See Figure 5). This gives $m \times (K + 1)$ simplicial complexes $\{\Delta_{ij}\}$ where $1 \leq i \leq m$ and $0 \leq j \leq K$.

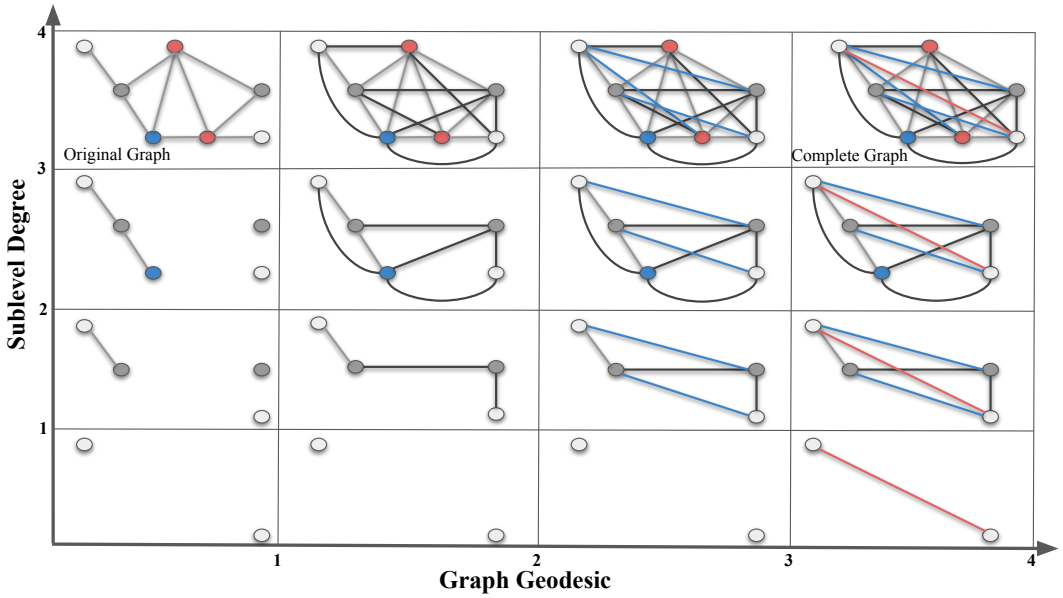

Figure 5: **Vietoris-Rips Filtrations.** In this toy example, we give a bifiltration composed of a sublevel (vertical) and a VR filtration (horizontal) of a simple graph (top box in the first column). In the vertical direction, we apply sublevel filtration by degree function with thresholds $1, 2, 3$ and 4. In the horizontal direction, we apply VR-filtration with respect to graph distance (geodesic length). In the first column, we have an (gray) edge between two nodes if their graph distance is 1. In the second column, we have an (black) edge between two nodes if their graph distance is ≤ 2. Blue edges in the third column represent the edges for graph distance 3. Red edges in the last column represent the edges for graph distance 4.

This is called the *bipersistence module*. One can imagine increasing sequence of $\{\mathcal{V}_i\}$ as vertical direction, and induced VR-complexes $\{\Delta_{ij}\}$ as the horizontal direction. In our construction, we fix the slicing direction as the horizontal direction (VR-direction) in the bipersistence module, and obtain the persistence diagrams in these slices.

In the toy example in Figure 5, we use a small graph $\mathcal{G}$ instead of a real compound to keep the exposition simple. Our sublevel filtration (vertical direction) comes from the degree function. Degree of a node is the number of edges incident to it. In the first column, we simply see the single sublevel filtration of $\mathcal{G}$ by the degree function. In each row, we develop VR-filtration of the subgraph by using the graph distances between the nodes. Here, graph distance between nodes means the length of the shortest path (geodesic) in the graph where each edge is taken as length 1. Then, in the second column, we add the edges for the nodes whose graph distance is equal to 2. In the third column, we add the (blue) edges for the nodes whose graph distance is equal to 3. Finally, in the last column, we add the (red) edges for the nodes whose graph distance is equal to 4. By construction, all the graphs in the last column must be a complete graph as there is no more edge to add.

After getting the bifiltration, the following steps are the same as in Section 4. In particular, for each $1 \leq i_0 \leq m$, one obtains a single filtration $\mathcal{V}_{i_0} = \Delta_{i_0 0} \subseteq \Delta_{i_0 1} \subseteq \ldots \subseteq \Delta_{i_0 K}$ in horizontal direction. This single filtration gives a persistence diagram $PD(\mathcal{V}_{i_0})$ as before. Hence, one obtains m persistence diagrams $\{PD(\mathcal{V}_i)\}$. Again, by applying a vectorization φ, one obtains m row vectors of fixed size r, i.e. $\vec{\varphi}_i = \varphi((\mathcal{V}_i))$. This induces a $2D$-vector $\mathbf{M}_\varphi$ (a matrix) of size $m \times (K+1)$ as before.

While computationally more expensive than others, VR-filtration can be very effective for some VS tasks, as it detects the graph distances between atoms, and size of the topological features [59, 2]. Note that VR-filtration when used for unweighted graphs with graph distance is known as "power filtration" in the literature. For further details on VR-filtration, see [5, 27].

Weight filtration For a given weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, it is common to use edge weights $\mathcal{W} = \{\omega_{rs} \in \mathbb{R}^+ \mid \epsilon_{rs} \in \mathcal{E}\}$ to describe filtration. For example, in our case, one can take bond strength in the compounds as edge weights. By choosing the threshold set similarly $\mathcal{I} = \{\alpha_i\}_1^m$ with $\alpha_1 = \min\{\omega_{rs} \in \mathcal{W}\} < \alpha_2 < \ldots < \alpha_m = \max\{\omega_{rs} \in \mathcal{W}\}$. For $\alpha_i \in \mathcal{I}$, let $\mathcal{E}_i = \{e_{rs} \in \mathcal{V} \mid \omega_{rs} \leq \alpha_i\}$. Let $\mathcal{G}^i$ be a subgraph of $\mathcal{G}$ induced by $\mathcal{V}_i$. This induces a nested sequence of subgraphs $\mathcal{G}_1 \subset \mathcal{G}_2 \subset \cdots \subset \mathcal{G}_m = \mathcal{G}$.

In the case of weighted graphs, one can apply the MP vectorization framework just by replacing the first filtering (via f) with weight filtering. In particular, let $g : \mathcal{V} \to \mathbb{R}$ be a filtering function with threshold set $\{\beta_j\}_{j=1}^n$. Then, one can first apply weight filtering to get $\mathcal{G}_1 \subset \cdots \subset \mathcal{G}_m = \mathcal{G}$ as above, and then apply f to each $\mathcal{G}_i$ to get a bilfiltration $\{\mathcal{G}_{ij}\}$ ($m \times n$ resolution). One gets m PDs as $PD(\mathcal{G}_i, g)$ and induce the corresponding $\mathbf{M}_\varphi$. Alternatively, one can change the order by applying g first, and get a different filtering $\mathcal{G}_1 \subset \mathcal{G}_2 \subset \cdots \subset \mathcal{G}_n = \mathcal{G}$ induced by g. Then, apply to edge weight filtration to any $\mathcal{G}_j$, one gets a bifiltration $\{\widehat{\mathcal{G}}_{ji}\}$ ($n \times m$ resolution) this time. As a result, one gets n PDs as $PD(\mathcal{G}_i, \omega)$ and induce the corresponding $\mathbf{M}_\varphi$. The difference is that in the first case (first apply weights, then g), the filtering function plays more important role as $\mathbf{M}_\varphi$ uses $PD(\mathcal{G}_i, g)$ while in the second case (first apply g, then apply weights) weights have more important role as $\mathbf{M}_\varphi$ is induced by $PD(\mathcal{G}_j, \omega)$. Note also that there is a different filtration method for weighted graphs by applying the following the following VR-complexes method.

In our applications, we used weight filtration to express bond strength in the compounds. Single bond has weight 1, double bond has weight 2, triple bond has weight 3, and finally aromatic bond has weight 4 on the edges.

C Further Experimental Results

C.1 Dataset Statistics

Table 3: Summary statistics of the Cleves-Jain dataset.

Target	# Training Samples	# Test Samples
a	3	6
b	3	22
c	2	13
d	3	6
e	2	5
f	2	4
g	2	5
h	2	5
i	2	5
j	3	14
k	3	14
l	3	10
m	3	9
n	2	10
o	3	30
p	3	23
q	3	11
r	2	14
s	3	15
t	2	5
u	3	9
v	3	7
Decoy	0	850

Table 4: Summary statistics of the DUD-E Diverse dataset.

Target	Description	# Active	# Decoy
AMPC	beta-lactamase	62	2902
CXCR4	C-X-C chemokine receptor type 4	122	3414
KIF11	kinesin-like protein 1	197	6912
CP3A4	cytochrome P450 3A4	363	11940
GCR	glucocorticoid receptor	563	15185
AKT1	serine/threonine-protein kinase Akt-1	423	16576
HIVRT	HIV type 1 reverse transcriptase	639	19134
HIVPR	HIV type 1 protease	1395	36278

C.2 Baselines

We compare our methods against the 23 state-of-the-art baselines including USR [8], ROCS [38], PS [42], GZD [92], PH_VS [50], USR + GZD [84], USR + PS [84], USR + ROCS [84], GZD + PS [84], GZD + ROCS [84], PS + ROCS [84], Findsite [101], Fragsite [102], Gnina [87], GOLD-EATL [96], Glide-EATL [96], CompM [96], CompScore [75], CNN [77], DenseFS [44], SIEVE-Score [98], DeepScore [94], and RF-Score-VSv3 [98].

In particular,, we compare our methods against the well-known $3D$-methods Ultrafast Shape Recognition (USR) [8], shape-based, ligand-centric method (ROCS) [38], PatchSurfer (PS) [42], Zernike (GZD) [92] and PH_VS [50] in Cleves-Jain dataset. In Table 1, we report the performances of all these $3D$ methods with 50 conformations [84] except PH_VS with 1 conformation [50].

In Table 2, we compare our models against other state-of-the-art VS methods on DUD-E Diverse dataset. All of these ML methods came in recent years. Among these, CNN [77] uses a convolutional neural network based framework with GPU-accelerated layer (i.e., MolGridDataLayer) to define a scoring function for protein-ligand interactions. DenseFS [44] improves the previous model CNN by a densely connected convolutional neural network architecture. Later, Sieve-Score proposes an effective scoring function: similarity of interaction energy vector score (SIEVE) [98] where they extract van der Waals, Coulomb and hydrogen bonding interactions to represent docking poses for ML. Random Forest Based Protein-ligand Scoring Function (RF-Score) [98] is another VS method proposed in the same work. Compscore [75] uses genetic algorithms to find the best combination of scoring components. Recently, energy auxiliary terms learning (EATL) [96] proposes an approach based on the scoring components extracted from the output of the scoring functions implemented in several popular docking programs like Genetic Optimisation for Ligand Docking (GOLD) [49], Molecular Operating Environment (MOE) [1] and Grid-based Ligand Docking with Energetics (GLIDE) [32, 35]. In the same work, they also combine these 3 methods and produced comprehensive EATL models, CompF and CompM. DeepScore [94] defines a deep learning based target specific scoring function. Findsite [101] proposes a threading/structure based method, where they improved it later with Fragsite [102] by using tree regression ML framework. Finally, Gnina [87] is a recently released molecular docking software, which uses deep convolutional networks to score protein-ligand structures. Note that all methods use 5-fold CV except Findsite-Fragsite (3-fold CV) and Gnina.

C.3 Model performance across different modalities

Table 5, 6, 7, 8 show detailed ablations of the modalities used in the graph filtration step of ToDD. The success of each periodic property varies per drug target and trained ML model. However merging MP fingerprints derived from each one of these domain functions used for graph filtration has always improved the performance.

Our results also show that MP fingerprints ensure making successful predictions while training few-labeled data such as Cleves-Jain (with 2-3 labeled compounds per drug target) using a transformer-based model.

RF shows worse performance than ViT on small-scale dataset such as Cleves-Jain, despite regularization by bootstrapping and using pruned, shallow trees. Additionally, RF is more sensitive to the small variations in the training set, and imbalanced data can hamper its accuracy. In order to effectively handle the large-scale datasets that have long-tailed distributions, we undersample from the majority class (decoys). Specifically, while training RF for the binary classification task on the drug targets of DUD-E Diverse (class distributions are summarized in Table 4), we use 80% of the active compounds and the same number of randomly chosen decoys for training. Undersampling decoys to avoid heavy class imbalance achieves better trade-offs between the accuracies of active compounds and decoys.

Table 5: EF 2% values and ROC-AUC scores across different modalities on Cleves-Jain dataset using **ToDD-RF**.

Target	Atomic Mass	Partial Charge	Bond Type	Atomic Mass & Partial Charge	All Modalities
a	33.3	33.3	33.3	33.3	41.7
b	25.0	29.5	31.8	27.3	25.0
c	19.2	7.7	15.4	26.9	34.6
d	33.3	33.3	41.7	50.0	50.0
e	30.0	30.0	30.0	40.0	40.0
f	25.0	50.0	37.5	50.0	37.5
g	30.0	30.0	30.0	30.0	40.0
h	40.0	50.0	30.0	50.0	50.0
i	40.0	40.0	30.0	40.0	40.0
j	17.9	39.3	35.7	28.6	28.6
k	21.4	21.4	17.9	35.7	32.1
l	15.0	15.0	15.0	30.0	25.0
m	44.4	50.0	33.3	50.0	38.9
n	15.0	25.0	10.0	25.0	10.0
o	21.7	20.0	25.0	23.3	23.3
p	10.9	8.7	13.0	17.4	26.1
q	45.5	27.3	22.7	40.9	40.9
r	42.9	42.9	42.9	39.3	32.1
s	26.7	16.7	20.0	20.0	30.0
t	30.0	50.0	50.0	50.0	50.0
u	33.3	38.9	27.8	38.9	50.0
v	21.4	28.6	28.6	28.6	28.6
Mean	28.3	31.3	28.3	35.2	35.2
ROC-AUC	0.92	0.90	0.88	0.94	0.93

Table 6: EF 2% values and ROC-AUC scores across different modalities on Cleves-Jain dataset using **ToDD-ViT**.

Target	Atomic Mass	Partial Charge	Bond Type	Atomic Mass & Partial Charge	All Modalities
a	25.0	33.3	33.3	33.3	50.0
b	20.5	6.8	34.1	6.8	34.1
c	11.5	15.4	23.1	7.7	46.2
d	25.0	41.7	50.0	33.3	50.0
e	40.0	20.0	30.0	20.0	30.0
f	25.0	25.0	37.5	25.0	50.0
g	20.0	30.0	40.0	30.0	50.0
h	40.0	50.0	50.0	50.0	50.0
i	30.0	20.0	20.0	20.0	50.0
j	10.7	14.3	21.4	17.9	21.4
k	21.4	21.4	25.0	21.4	39.3
l	15.0	30.0	30.0	40.0	35.0
m	22.2	44.4	22.2	44.4	50.0
n	10.0	25.0	15.0	20.0	35.0
o	20.0	16.7	16.7	18.3	20.0
p	26.1	17.4	26.1	15.2	32.6
q	36.4	36.4	50.0	36.4	18.2
r	14.3	17.9	42.9	21.4	32.1
s	30.0	13.3	23.3	13.3	33.3
t	20.0	30.0	50.0	30.0	50.0
u	33.3	38.9	38.9	33.3	50.0
v	21.4	28.6	28.6	21.4	42.9
Mean	23.5	26.2	32.2	25.4	39.5
ROC-AUC	0.87	0.85	0.86	0.84	0.90

Table 7: EF 1% values and ROC-AUC scores across different modalities on DUD-E Diverse using **ToDD-RF**.

	Atomic Mass		Partial Charge		Bond Type		Atomic Mass & Partial Charge		All Modalities	
Model	**EF 1%**	**ROC-AUC**	**EF 1%**	**ROC-AUC**	**EF 1%**	**ROC-AUC**	**EF 1%**	**ROC-AUC**	**EF 1%**	**ROC-AUC**
AMPC	42.9	0.90	42.9	0.92	28.6	0.84	42.9	0.84	28.6	0.86
CXCR4	84.6	0.98	76.9	0.99	84.6	0.97	92.3	0.99	92.3	0.99
KIF11	55.0	0.96	55.0	0.98	70.0	0.97	70.0	0.98	75.0	0.98
CP3A4	54.1	0.96	48.6	0.87	40.5	0.93	62.2	0.95	67.6	0.96
GCR	54.4	0.96	43.9	0.95	57.9	0.97	63.2	0.97	78.9	0.97
AKT1	62.8	0.97	86.0	0.99	79.1	0.98	81.4	0.98	90.7	0.99
HIVRT	53.1	0.90	46.9	0.93	64.1	0.97	54.7	0.91	64.1	0.95
HIVPR	85.0	0.99	86.4	0.99	78.6	0.99	87.9	0.99	92.1	0.99
Mean	61.5	0.95	60.8	0.95	62.9	0.95	69.3	0.95	73.7	0.96

Table 8: EF 1% values and ROC-AUC scores across different modalities on DUD-E Diverse using **ToDD-ConvNeXt**.

	Atomic Mass		Partial Charge		Bond Type		Atomic Mass & Partial Charge		All Modalities	
Model	**EF 1%**	**ROC-AUC**	**EF 1%**	**ROC-AUC**	**EF 1%**	**ROC-AUC**	**EF 1%**	**ROC-AUC**	**EF 1%**	**ROC-AUC**
AMPC	30.8	0.90	15.4	0.83	30.8	0.73	38.5	0.92	46.2	0.81
CXCR4	52.0	0.98	44.0	0.92	32.0	0.94	60.0	0.97	84.0	0.99
KIF11	47.5	0.96	45.0	0.88	37.5	0.92	60.0	0.96	72.5	0.97
CP3A4	19.2	0.86	17.8	0.86	15.1	0.86	28.8	0.90	28.8	0.91
GCR	25.7	0.95	30.1	0.95	19.5	0.94	43.4	0.96	46.0	0.97
AKT1	60.0	0.91	51.8	0.91	41.2	0.96	77.6	0.99	81.2	0.98
HIVRT	26.6	0.93	21.9	0.89	17.2	0.94	35.9	0.94	37.5	0.95
HIVPR	65.6	0.98	50.9	0.97	45.5	0.94	70.3	0.99	74.6	0.99
Mean	40.9	0.93	34.6	0.90	29.9	0.90	51.8	0.95	58.8	0.95

C.4 Computation Time

See 6.3 in the main text for details.

Table 9: Clock time performance to extract Vietoris Rips persistent homology features.

Dataset	**Atomic Mass**	**Partial Charge**	**Bond Type**
Cleves-Jain	00 h : 03 min : 21 sec	00 h : 03 min : 14 sec	00 h : 01 min : 13 sec
DUD-E Diverse	06 h : 10 min : 51 sec	05 h : 37 min : 12 sec	02 h : 14 min : 54 sec

C.5 Model performance using Morgan fingerprints

See 6.4 in the main text for details.

Table 10: EF 2% values on Cleves-Jain Dataset using ViT model trained with Morgan fingerprints vs. ToDD fingerprints.

Target	Morgan	ToDD
a	25.0	50.0
b	11.4	34.1
c	3.8	46.2
d	50.0	50.0
e	10.0	30.0
f	37.5	50.0
g	30.0	50.0
h	40.0	50.0
i	20.0	50.0
j	17.9	21.4
k	14.3	39.3
l	45.0	35.0
m	38.9	50.0
n	15.0	35.0
o	13.3	20.0
p	2.2	32.6
q	18.2	18.2
r	14.3	32.1
s	10.0	33.3
t	20.0	50.0
u	27.8	50.0
v	35.7	42.9
Mean	22.7	39.5
ROC-AUC	0.86	0.90

Table 11: EF 1% values and ROC-AUC scores on DUD-E Diverse dataset using ConvNeXt model trained with Morgan fingerprints vs. ToDD fingerprints.

	Morgan		ToDD	
Model	**EF 1%**	**ROC-AUC**	**EF 1%**	**ROC-AUC**
AMPC	38.5	0.87	46.2	0.81
CXCR4	48.0	0.97	84.0	0.99
KIF11	57.5	0.95	72.5	0.97
CP3A4	20.5	0.84	28.8	0.91
GCR	46.7	0.94	46.0	0.97
AKT1	60.0	0.98	81.2	0.98
HIVRT	50.0	0.96	37.5	0.95
HIVPR	61.3	0.98	74.6	0.99
Mean	47.8	0.94	58.8	0.95

D Societal Impact and Limitations

D.1 Societal Impact

We perform in silico experiments and use high-throughput screening to recognize active compounds that can bind to a drug target of interest, e.g., an enzyme or protein receptor without conducting research on any human or animal subjects. Our overarching goal is to augment the capabilities of AI to enhance the in silico virtual screening and drug discovery processes, develop new drugs that have less side effects but are more effective to cure diseases, and minimize the participation of human and animal subjects as much as possible to ensure their humane and proper care.

D.2 Limitations

We discuss in detail the computational complexity of our model in 6.3. Our model is versatile and can be scaled for large libraries by customizing the allocated computational resources. Please note that the analysis in C.4 shows the execution time of our computation pipeline when the feature extraction task is distributed across 8 cores of a single Intel Core i7 CPU. It is possible to parallelize

computationally costlier operations such as VR-filtration by allocating more CPU cores on the HPC platform and optimize array operations (e.g., numpy) via the joblib library. Furthermore, all ToDD models require substantially fewer computational resources during training compared to current graph-based models that encode a compound through mining common molecular fragments, a.k.a., motifs [47]. For instance, training a motif based GNN on GuacaMol dataset which has approximately 1.5M drug-like molecules takes 130 hours of GPU time [64]. In contrast, once we generate the topological fingerprints via Vietoris-Rips filtration, training time of ToDD-ViT and ToDD-ConvNeXt for each individual drug target takes less than 1 hour on a single GPU (NVIDIA RTX 2080 Ti).

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix D.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix D.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Given in Appendix B.1
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Dataset links are provided. See Section 5.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 6.1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We reported the standard deviation of our experiments evaluated by 5-fold cross-validation. See Table 1 and 2 in Section 6.2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 6.3
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] They use public-domain-equivalent license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]