CuMPerLay: Learning Cubical Multiparameter Persistence Vectorizations

Supplementary Material

Abstract

This document supplements our submission titled CuMPerLay: Learning Cubical Multiparameter Persistence Vectorizations. We present proofs on the stability of our novel vectorization, provide additional background on multiparameter persistence theory, explain common multifiltrations for images, and extend the limited data results we have shared. In addition, we share example images from the datasets we have used and additional learned multiparameter persistence filtration examples on each dataset. We also share pseudocode and implementation details for our CUDA GPU implementation of Cubical Multiparameter Persistence.

A. Stability Results

A.1. Stability of Single Persistence Vectorizations

In machine learning, ensuring the stability of a particular persistence diagram (PD) vectorization is very crucial, as stability examines whether small changes in the PD result in significant alterations in the vectorization. To address this question, we need clear definitions of what constitutes "small" and "large" changes, which in turn requires a way to measure distance within the space of persistence diagrams. The most widely used metric for this purpose is the Wasserstein distance, also known as the matching distance.

Consider two images, $\mathcal{X}^+$ and $\mathcal{X}^-$, with their associated persistence diagrams $\mathrm{PD}(\mathcal{X}^+)$ and $\mathrm{PD}(\mathcal{X}^-)$, respectively (omitting dimension labels). Each diagram contains points $\{q_j^+\} \cup \Delta^+$ and $\{q_l^-\} \cup \Delta^-$, where $\Delta^\pm$ represents the diagonal (which stands for trivial cycles) with infinite multiplicity. Here, $q_j^+ = (b_j^+, d_j^+)$ in $\mathrm{PD}(\mathcal{X}^+)$ denotes the birth and death times of a feature σ_j in $\mathcal{X}^+$. Let $\phi : \mathrm{PD}(\mathcal{X}^+) \to \mathrm{PD}(\mathcal{X}^-)$ be a bijection, allowing mappings even when the cardinalities $|\{q_j^+\}|$ and $|\{q_l^-\}|$ differ. The p-th Wasserstein distance, denoted $\mathcal{W}_p$, is formally defined as follows:

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

Next, a vectorization, denoted as $\varphi(\mathrm{PD}(\mathcal{X}))$, is called to be *stable* if the distance between its outputs for two images, satisfies the inequality

$$\mathrm{d}(\varphi(\mathrm{PD}(\mathcal{X}^+)), \varphi(\mathrm{PD}(\mathcal{X}^-))) \leq C \cdot \mathcal{W}_p(\mathrm{PD}(\mathcal{X}^+), \mathrm{PD}(\mathcal{X}^-))$$

where $\mathrm{d}(.\ ,\ .)$ represents a suitable metric on the space of vectorizations. The constant $C > 0$ is independent of the images $\mathcal{X}^\pm$. This stability inequality interprets that changes in vectorizations are bounded by changes in persistence diagrams. Essentially, two nearby persistence diagrams correspond to nearby vectorizations. If a particular vectorization φ satisfy such a stability condition, we call them *stable vectorization* [3]. Notable examples of stable vectorizations include Persistence Landscapes [6], Silhouettes [11], Persistence Images [1], Stabilized Betti Curves [18], and various Persistence curves [12].

A.2. Stability of MultiPersistence Vectorizations

In this paper, because of technical problems in defining a multipersistence output like persistence diagrams (Suppl. B), we directly moved the vectorization of the multifiltrations by a well-known method called *slicing*. In particular, assuming we have 2-parameter filtration $\{\mathcal{X}_{i,j}\}$, in the multipersistence grid $\mathcal{G}$ of size $m \times n$, for each fixed i_0, we get single filtration $\widehat{\mathcal{X}}_{i_0}$, $\mathcal{X}_{i_0 1} \subset \mathcal{X}_{i_0 2} \subset \cdots \subset \mathcal{X}_{i_0 n} = \mathcal{X}_{i_0}$, which can be considered as *horizontal slicing* of the multipersistence grid. Then, applying vectorizations above to the single filtrations $\{\widehat{\mathcal{X}}_i\}_1^m$, we get m separate vectors $\vec{\varphi}_i = \varphi(\mathrm{PD}(\widehat{\mathcal{X}}_i))$. Then, collecting these vectors into a 2-tensor as rows, we obtain a 2-tensor $\mathcal{M}_\varphi$ which we call the induced MP vectorization of φ. For example, when φ is Betti vectorization, $\mathcal{M}_\varphi$ is simply correspond to multigraded Betti numbers [26], which are simply $m \times n$ matrices for each Betti dimension. We now show that when the source single parameter (SP) vectorization φ is stable, then so is its induced MP vectorization $\mathcal{M}_\varphi$.

Let $\mathcal{X}^+$ and $\mathcal{X}^-$ be two images of size $r \times s$. With the notation in Suppl. A.1, let φ be a stable SP vectorization with the stability equation

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

for some $1 \leq p_\varphi \leq \infty$. Note that for many common stable vectorization φ, $d(\cdot, \cdot)$ is taken as l^{p_φ} metric [30]. However, to keep the generality, we are not specifying it here.

Now, we consider the define these multiple single persistence diagrams $\{\mathrm{PD}(\widehat{\mathcal{X}}_i^\pm)\}$ as output for the multipersistence grid, and define a natural matching distance between as the sum of the corresponding distances for each row.

$$\mathbf{D}_p(\{\mathrm{PD}(\widehat{\mathcal{X}}_i^+)\}, \{\mathrm{PD}(\widehat{\mathcal{X}}_i^-)\}) = \textstyle\sum_{i=1}^m \mathcal{W}_p(\mathrm{PD}(\widehat{\mathcal{X}}_i^+), \mathrm{PD}(\widehat{\mathcal{X}}_i^-)). \tag{3}$$

Now, we define *the distance between induced MP vectorizations* as

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

where $p \geq 1$.

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

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

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

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

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

Corollary 1 (CumPerLay is stable)**.** *Let Ψ_{MP} represent CumPerLay vectorization as defined in Section 4.1. Let $\mathcal{K}$ be a cubical complex and let $F, G : \mathcal{K} \to \mathbb{R}^2$ be bifiltration functions. Let $\widehat{\mathcal{K}}_F, \widehat{\mathcal{K}}_G$ be the induced bipersistence modules. Then, we have*

$$\mathfrak{D}(\Psi_{\mathrm{MP}}(\widehat{\mathcal{K}}_F), \Psi_{\mathrm{MP}}(\widehat{\mathcal{K}}_G)) \leq \widehat{C}_{\Psi_{\mathrm{MP}}} \|F - G\|_\infty$$

Proof. The proof follows from the stability of Perslay Vectorizations [9], and the previous stability theorem. In particular, while the authors proved their stability theorem for clique complexes in [9], their proofs as single persistence vectorization extends to cubical complexes as all the relevant results extends to this context [30]. By [8, Theorem A.2], as single persistence module, for each row $1 \leq m_0 \leq M$ of the bipersistence module $\widehat{\mathcal{K}}$, we obtain $d_B(\mathrm{PD}(\widehat{\mathcal{K}}_F^{m_0}), \mathrm{PD}(\widehat{\mathcal{K}}_F^{m_0}) \leq \|F - G\|_\infty$ where $\widehat{\mathcal{K}}^{m_0}$ represents m_0^{th} row in bipersistence module $\widehat{\mathcal{K}}$. As Perslay vectorizations are stable, we obtain

$$\mathrm{d}(\Psi_{\mathrm{MP}}^{m_0}(\widehat{\mathcal{K}}_F), \Psi_{\mathrm{MP}}^{m_0}(\widehat{\mathcal{K}}_F) \leq C_{m_0} d_B(\mathrm{PD}(\widehat{\mathcal{K}}_F^{m_0}), \mathrm{PD}(\widehat{\mathcal{K}}_F^{m_0})$$

where $\Psi_{\mathrm{MP}}^{m_0}$ represents restriction of Ψ_{MP} to m_0^{th} row. After obtaining similar inequality for each row, the proof follows by following the same procedure in the proof of Thm. 1 where $\widehat{C}_{\Psi_{\mathrm{MP}}} = \sum_{m=1}^M C_m$. □

While we define the metrics and MP vectorizations for a 2-parameter case, it can naturally be adapted to any multiparameter case. Similarly, the proof can easily be adapted to higher dimensional images. In this paper, we utilize Silhouette and Betti curves as the vectorization method φ. while Betti curves are not stable with respect to the bottleneck ($\mathcal{W}_\infty$) distance [2], they are stable with respect to the $\mathcal{W}_1$-metric [14]. On the other hand, Silhouette vectorizations are stable as they are derived from persistence landscapes [7, 11]. Therefore, by applying Thm. 1 to these vectorizations, we have the stability for both MP Betti and MP Silhouette vectorizations, we employed in this paper. Furthermore, adapting the stability result given in [26] to our setting, one can obtain another proof for the stability of MP Betti vectorization with respect to a signed Wasserstein-1 distance, a bottleneck-type metric they introduce.

B. Multiparameter Persistence Theory

Multipersistence theory has garnered significant research interest due to its potential to enhance the performance and robustness of single persistence theory. While single persistence extracts topological features from a one-parameter filtration, a multidimensional filtration with multiple parameters should, in principle, provide a richer and more informative summary for machine learning applications. However, technical challenges in the theory have hindered its full realization, leaving multipersistence largely unexplored in the ML community. Here, we summarize these key challenges. For a more detailed discussion, [5] provides an overview of the current state of the theory and its major obstacles.

In single persistence, the threshold space $\{\alpha_i\}$ is a totally ordered subset of $\mathbb{R}$, meaning that any topological feature appearing in the filtration sequence $\{\Delta_i\}$ has a well-defined birth and death time, with birth occurring before death. This ordering property allows for the decomposition of the persistence module $M = \{H_k(\Delta_i)\}_{i=1}^N$ into barcodes, a concept formalized in the 1950s through the Krull-Schmidt-Azumaya Theorem [5] (Theorem 4.2). This decomposition forms the basis of what is known as a *Persistence Diagram*.

However, in higher dimensions ($d = 2$ or more), the threshold set $\{(\alpha_i, \beta_j)\}$ is only partially ordered (a poset), meaning that while some indices have a clear ordering (e.g., $(1, 2) < (4, 7)$), others do not (e.g., $(2, 3)$ vs. $(1, 5)$). Consequently, in a multipersistence grid $\{\Delta_{ij}\}$, birth and death times are no longer well-defined. Furthermore, the Krull-Schmidt-Azumaya Theorem does not extend to higher dimensions [5] (Section 4.2), making barcode decomposition impossible for general multipersistence modules. This fundamental limitation prevents a straightforward generalization of single persistence to multipersistence. Even in cases where a meaningful barcode decomposition exists, the challenge remains of faithfully representing these barcodes due to the inherent partial ordering. Multipersistence modules are an active area of research in commutative algebra, with further details available in [15].

Despite these challenges, several approaches have been proposed to leverage the multipersistence framework [21]. One of the earliest methods, introduced by [22], involves analyzing one-dimensional slices of the multipersistence

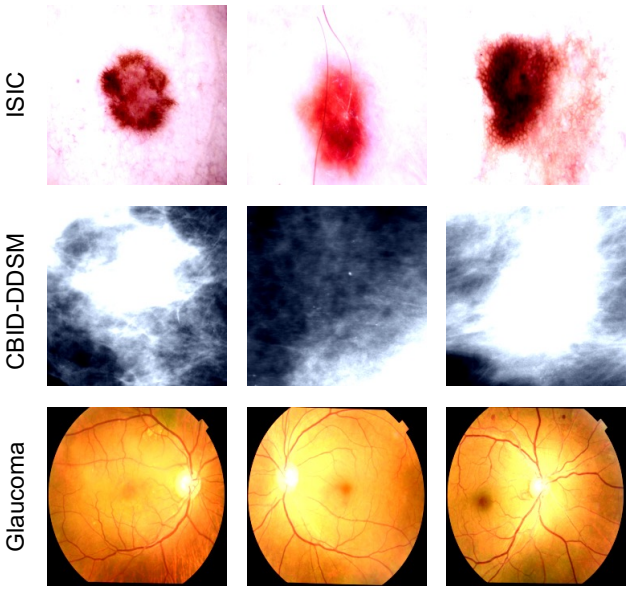

Figure 1. **Datasets.** Example images from ISIC dataset (top row) [13], CBIS-DDSM dataset (middle row) [20], Glaucoma dataset (bottom row) [27, 29].

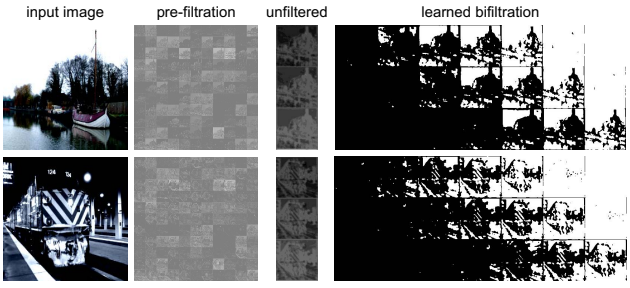

Figure 2. **Filtration learning.** Some of the compact multifiltration representations and the corresponding bifiltrations from the first layer of our U-Net backbone Topo-MP network on PASCAL VOC 2012 dataset.

grid to extract the most dominant features. Later, [8] expanded this idea by considering multiple slicing directions (vineyards) and summarizing multiple persistence diagrams (PDs) into a vectorized representation. These slicing-based techniques extract persistence diagrams from predetermined one-dimensional slices and aggregate them into a lower-dimensional summary [5]. However, this approach presents two major issues: (1) the topological summary heavily depends on the chosen slicing directions, making direction selection a nontrivial problem, and (2) compression of information from multiple persistence diagrams may lead to significant information loss.

A different approach to vectorizing multipersistence modules was introduced by Vipond [32], who extended persistence landscapes into higher dimensions. Unlike slicing-based methods, this approach does not rely on a predetermined global slice direction. Instead, for each point $\mathbf{x} \in \mathbb{R}^n$, where n is the dimension of the multipersistence module, the k^{th}-landscape explores the widest direction in which the rank invariant has a nontrivial image. From this perspective, the Multipersistence Landscape can be seen as a more faithful representation of the multipersistence module. While this method is particularly effective in settings where key insights come from a few dominant topological features (e.g., point clouds or sparse data), its computational complexity limits its practicality for large datasets with many topological features. Recently, [10] introduced the meta-rank and meta-diagram as novel invariants for 2-parameter persistence modules, demonstrating their equivalence to the rank invariant and signed barcode while providing computational improvements and an intuitive visualization as a persistence diagram of diagrams. Again, very recently, [24] extends stable vectorization techniques from one-parameter to multiparameter persistent homology by leveraging signed barcodes as signed Radon measures, enabling the computation of informative and provably stable feature vectors that enhance performance in topology-based data analysis.

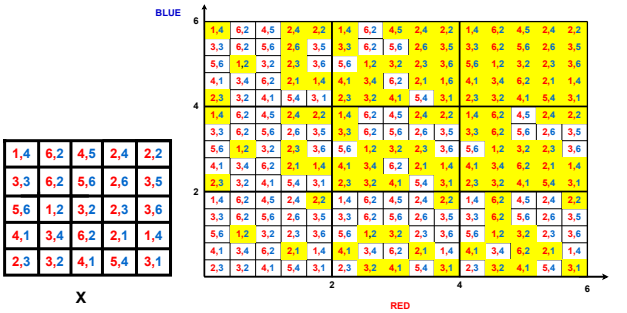

Figure 3. **Toy illustration of bifiltration.** For a given image X with two color channels, our color multifiltration produces 9 binary images in 3×3 grid. In horizontal directions, we activate (coloring yellow) the pixels whose red color value is below the given threshold. In vertical direction, we only consider blue color values to activate the pixels.

While these approaches effectively capture dominant topological patterns in sparse data, their applications are largely limited to point cloud and graph settings. In contrast, our approach offers a more computationally efficient and versatile vectorization for cubical multipersistence, where the key topological signatures mostly arise from tracking the density of small features, making it suitable for image analysis.

C. Common Multifiltrations for Images

There are several natural multifiltration methods for cubical persistence the images offer. Here, we list some of them.

Bifiltrations on binary images. Let $\mathcal{X}$ be an image of size $r \times s$. A bifiltration induced from $\mathcal{X}$ is a collection of binary images $\{\mathcal{X}_{m,n}\}$ of the same size where $\mathcal{X}_{m,n} \subset \mathcal{X}_{m+1,n}$ and $\mathcal{X}_{m,n} \subset \mathcal{X}_{m,n+1}$ for any $1 \leq m \leq M$ and $1 \leq n \leq N$.

Table 1. **Limited Data performances (ISIC and CBIS-DDSM).** Performance of the baseline Swin model and hybrid models incorporating Single Persistence (Swin-SP) and Multi-Persistence (Swin-MP) outputs under limited data conditions. The model was trained on X% of the dataset (specified in the first column) while using the original test set for evaluation, with early stopping based on validation accuracy. Both datasets are evaluated and averaged over 4 different seeds, with standard deviations reported after ±. Best AUC results are given in **bold**, while best accuracy results are given in **blue** for each setting.

	ISIC						CBIS-DDSM					
	Swin [23]		TopoSwin-SP*		TopoSwin-MP*		Swin [23]		TopoSwin-SP*		TopoSwin-MP*	
Data %	Acc	AUC	Acc	AUC	Acc	AUC	Acc	AUC	Acc	AUC	Acc	AUC
0.05	66.62 ± 4.23	**87.15 ± 3.85**	65.92 ± 4.44	84.20 ± 6.75	**68.74 ± 3.32**	85.94 ± 3.17	58.52 ± 4.44	**63.34 ± 1.68**	58.72 ± 2.03	62.81 ± 3.27	**59.35 ± 3.26**	61.10 ± 1.62
0.10	73.90 ± 2.15	91.48 ± 0.58	74.01 ± 1.88	**91.64 ± 1.01**	**74.45 ± 1.59**	90.71 ± 1.59	60.80 ± 2.00	60.60 ± 3.15	59.40 ± 2.37	61.70 ± 2.47	**61.79 ± 0.88**	**64.31 ± 3.13**
0.20	77.38 ± 1.08	93.65 ± 0.38	**77.62 ± 1.94**	**93.80 ± 1.17**	77.58 ± 0.68	93.07 ± 0.28	62.50 ± 1.82	69.04 ± 1.16	63.21 ± 3.69	**69.08 ± 0.96**	**63.38 ± 1.81**	67.25 ± 0.83
0.50	80.82 ± 1.73	**95.46 ± 0.83**	79.65 ± 0.50	94.80 ± 0.65	**81.64 ± 0.62**	95.33 ± 0.89	67.95 ± 1.46	**74.92 ± 1.24**	67.53 ± 0.94	73.22 ± 1.85	**68.61 ± 1.13**	74.46 ± 1.36
1.00	82.67 ± 1.26	95.62 ± 0.15	82.50 ± 1.17	95.63 ± 0.71	**83.02 ± 0.35**	**95.93 ± 0.55**	68.47 ± 1.76	75.53 ± 0.77	69.38 ± 0.97	75.89 ± 0.97	**69.60 ± 0.72**	**76.84 ± 0.94**

Table 2. **Limited Data performances (Glaucoma).** Performance of the baseline Swin model and hybrid models incorporating Single Persistence (Swin-SP) and Multi-Persistence (Swin-MP) outputs under limited data conditions. The model was trained on X% of the dataset (specified in the first column) while using the original test set for evaluation. Best AUC results are given in **bold**, while best accuracy results are given in **blue** for each setting.

	Glaucoma					
	Swin [23]		TopoSwin-SP*		TopoSwin-MP*	
Data %	Acc	AUC	Acc	AUC	Acc	AUC
0.05	**79.47**	85.30	67.57	72.76	78.10	**88.04**
0.10	78.52	**89.42**	73.26	89.33	**78.74**	89.05
0.20	**82.95**	**92.00**	82.31	91.34	82.52	90.63
0.50	81.68	90.94	**83.57**	91.63	81.89	**92.64**
1.00	82.73	92.24	82.95	**92.90**	**84.63**	92.87

In other words, a bifiltration defines a grid $\mathcal{G}$ of size $M \times N$ where each individual row and column represents a regular filtration as shown in Fig. 3, *i.e.*, for any fixed $1 \leq m_0 \leq M$ (row), the sequence $\{\mathcal{X}_{m_0,n}\}_1^N$ is a regular filtration of length N (column), and for any fixed $1 \leq n_0 \leq N$, the sequence $\{\mathcal{X}_{m,n_0}\}_1^M$ is a regular filtration of length M.

Color multifiltrations.. The primary objective of multi-parameter persistence lies in effectively leveraging multiple parameters, especially when data offers more than one function to utilize. In color images, each image inherently comprises three natural functions, denoted as R, G, and B. Thus, for every pixel Δ_{ij}, there exist corresponding color values: $R_{ij}, G_{ij}, B_{ij} \in [0, 255]$. To proceed, we establish a three-parameter multifiltration with parameters $\{s_m\}_1^{N_1}$, $\{t_n\}_1^{N_2}$, $\{v_r\}_1^{N_3}$, where $s_m, t_n, v_r \in [0, 255]$ are threshold values for color channels R, G, and B respectively. By simply defining binary images $\mathcal{X}_{m,n,r} = \{\Delta_{ij} \subset \mathcal{X} \mid R_{ij} \leq s_m, G_{ij} \leq t_n, B_{ij} \leq v_r\}$, we induce a three-parameter multipersistence module $\{H_k(\mathcal{X}_{m,n,r})\}$, yielding Betti tensors $\mathbf{B}_k(\mathcal{X}) = [\beta^k_{m,n,r}]$. These tensors constitute $3D$ arrays with dimensions $N_1 \times N_2 \times N_2$ (See Fig. 3).

Erosion bifiltrations.. One significant limitation of sublevel filtration is its inability to provide information about the sizes of topological features. Instead, it only focuses on the difference in function values between when a topological feature is born and when it dies. To illustrate, consider a grayscale image where all pixels have a grayscale value of 0 except for one pixel in the center with a value of 255. The resulting persistence diagram would feature a single large bar $[0, 255)$, indicating a very small topological feature—a hole with a diameter of 1. Conversely, a binary image in $\mathcal{X}_{100}$ might contain a large hole with a diameter of 20, where the pixels of the hole have color values between 101 and 105, then it will be completely filled in by $\mathcal{X}_{105}$. Despite the significant change in the hole's size, the grayscale sublevel filtration would only yield a small bar $(100, 105)$ reflecting the difference in grayscale values (color contrast of the hole), without conveying any information about the hole's size.

While persistence homology aims to identify the topological features present in a filtration, sublevel filtrations cannot inherently capture size information for these features. However, alternative filtration methods such as erosion, dilation, and signed distance filtrations have been proposed to address this issue [17]. These methods offer avenues to incorporate size information into the analysis of topological features, complementing the capabilities of sublevel filtration.

To capture topological features created by the images as well as their sizes, a natural approach is to combine grayscale sublevel filtration with erosion filtration. Erosion filtration is defined for binary images, and basically, it is a sublevel filtration for a special function, called *erosion*. Let $\mathcal{X}$ be a grayscale image of size $r \times s$, and let $\{\mathcal{X}_m\}_1^M$ be the sublevel filtration induced by grayscale values described in

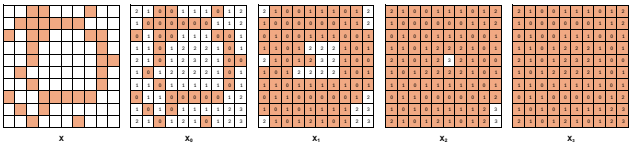

Figure 4. **Erosion Filtration.** For a given binary image $\mathcal{X}$, we first define the erosion function (given in $\mathcal{X}_0$). Then, we obtain the filtration of binary images $\mathcal{X}_0 \subset \mathcal{X}_1 \subset \ldots \mathcal{X}_3$, by activating the pixels reaching the threshold value.

Section 3. Each $\mathcal{X}_m$ is a binary image with some topological features. Let Ω_m represent all black pixels in $\mathcal{X}_m$ and $\mathcal{G}$ is $r \times s$ grid. Then, for each $\mathcal{X}_m$, we define an erosion function $\xi_m : \mathcal{G} \to \mathcal{N}$ such that $\xi_m(i,j) = \mathcal{D}(\Omega_m, \Delta_{ij})$ where $\mathcal{D}$ is the Manhattan (L_1) metric between the pixel Δ_{ij} and the black region Ω_m. In other words, $\xi_m(i_0, j_0) = \inf\{|i_0 - i'| + |j_0 - j'| \mid \Delta_{i'j'} \subset \Omega_m\}$. In particular, for each $\Delta_{ij} \in \Omega_m$, $\xi_m(i,j) = 0$. Intuitively, erosion function ξ_m gives 0 values to all black pixels in the binary image $\mathcal{X}_m$, and measures the L_1 distance of each white pixel to black region Ω_m in $\mathcal{X}_m$ (See Fig. 4). Then, for each m, the bifiltration $\{\mathcal{X}_{m,n}\}$ is defined as the collection of binary images $\mathcal{X}_{m,n} = \{\Delta_{ij} \in \mathcal{X} \mid \xi_m(i,j) < n\}$. Hence, if there is a white hole in $\mathcal{X}_m$, the value d of the farthest pixel in the hole to Ω_m measures the radius of the whole, and produces a bar of size $[0, d)$ in the persistence diagram. In a way, erosion filtration is enabling to measure the size of the wholes in binary images. Recall that the persistence bars for sublevel filtration for color values only give the color difference (contrast) of the hole independent of the size of the hole. Therefore, color filtration and erosion filtration are completely complementary to each other, and combining them produces a very powerful filtration model for the image. Signed distance and dilation are other types of filtrations to capture the size of the topological features in binary images, similar to erosion filtration [17].

D. Implementation Details

In Algorithm 1, we present the main algorithm for our CUDA GPU implementation of Cubical Persistence and Cubical Multiparameter Persistence. The two main C++/CUDA functions are further explained in Algorithm 2 and Algorithm 3. Our implementation is influenced by the CPU-based Cubical-Ripser software [19] for SP. We utilize the duality between top-dimensional cell and vertex constructions in persistent homology [4] to efficiently compute 0^{th} and 1^{st} dimensional homology of 2D images with only minor modifications to the algorithm. The compact multifiltration inputs (corresponding to N MP filtration inputs with C row-wise slices) are used to construct an input grid whose structure depends on the target dimension. For the 0^{th} dimension, the compact filtration input is padded with a threshold value (a constant input representing infinity), corresponding to a primal grid. For the 1^{st} dimension, the input values are padded with threshold, inverted, and then padded again, corresponding to an Alexander Duality-based dual grid. Following this, Algorithm 2 iterates over all edges (horizontal and vertical for both dimensions, plus diagonal edges for the 1^{st} dimension), computes filtration values based on the compact filtration inputs in parallel over the batch, width, and height dimensions, and finally sorts the values in a descending order.

In Algorithm 3, we use a union-find-based algorithm with a CUDA-adapted implementation of Union-Find [31] with path compression and union by rank. The initialization of the union-find data structure is parallelized over each batch element and each pixel. The main loop, which calculates persistence pairs by iterating over edges sorted by decreasing death times, is parallelized only over the batch dimension. Since the number of persistence pairs can vary per input and is not known beforehand, we use a chunking mechanism. Based on a provided chunk size hyperparameter, the loop is executed until a maximum of chunk size pairs are generated for each batch element, while the number of persistence pairs is tracked separately. The kernel for this loop is executed iteratively until the computation for each batch element is complete. The resulting chunks are then concatenated and processed in parallel. This approach allows variable-length persistence pair outputs and can be further optimized by fine-tuning the chunk size hyperparameter for a given task.

Finally, we extract the final persistence pairs using the coordinate locations that correspond to the birth and death values in the original MP filtration. The algorithm's output for variable-length persistence pairs consists of two tensors: a zero-padded tensor containing the persistence pairs for each batch element and row-wise slice, and a separate tensor storing the lengths for each batch element and row-wise slice. This output is then processed by our CuMPerLay vectorization layer, which takes masked persistence pair tensors as inputs. While we typically use a chunk size of 1024, this hyperparameter can be dynamically adjusted during training based on the maximum number of persistence pairs observed in recent batches.

Algorithm 1: Cubical Persistence for 2D Images

Input : Image filtration batches $I_{batch} \in \mathbb{R}^{N \times C \times H \times W}$ (C rows), threshold T

Output : Persistence pairs P, Lengths L

$I \leftarrow \text{Reshape}(I_{batch}, (N \cdot C, H, W))$;
$P_{list}, L_{list} \leftarrow [], []$;
for $d \in \{0, 1\}$ **do**
  `// Prepare grid for dimension d`
  $G \leftarrow \text{PadGrid}(I, \text{value} = T, d)$
  `// Compute persistence`
  $V, Idx \leftarrow \text{EnumerateAndSortEdges}(G, d)$;
  $C, L_d \leftarrow \text{JointPairs}(G, V, Idx, d)$
  $P_d \leftarrow \text{ExtractPersistenceValues}(C, I, d)$;
  $P_{list}.\text{append}(P_d)$;
  $L_{list}.\text{append}(L_d)$;
`// Combine and reshape results`
$P \leftarrow \text{PadAndStack}(P_{list})$;
$L \leftarrow \text{Stack}(L_{list})$;
$P, L \leftarrow \text{ReshapeToOriginal}(P, L, (N, C, \dots))$;
return P, L;

Algorithm 2: EnumerateAndSortEdges

Input : Grid $G \in \mathbb{R}^{B \times H \times W}$, dimension $d \in \{0, 1\}$
Output : Sorted filtration values V, Sorted indices Idx

`// Define edge connectivity`
if $d = 0$ **then**
  `// Horizontal and vertical edges`
  Offsets $\leftarrow \{(1, 0), (0, 1)\}$
else
  `// Include diagonal edges`
  Offsets $\leftarrow \{(1, 0), (0, 1), (1, 1), (1, -1)\}$
$E_{list} \leftarrow []$
`// Compute filtration values (CUDA)`
for *each image in batch* G ***in parallel*** **do**
  for *each pixel* $v = (x, y)$ *in image* ***in parallel*** **do**
    for *each offset* $o \in$ *Offsets* **do**
      $u \leftarrow v + o$;
      value $\leftarrow \max(G(v), G(u))$;
      E_{list}.append(value);
`// Sort edges by filtration values`
$V, Idx \leftarrow$ SortAndGetIndices(E_{list}, order = descending);
return V, Idx;

Algorithm 3: JointPairs

Input : Grid $G \in \mathbb{R}^{B \times H \times W}$, Sorted filtr. values V, Sorted indices Idx, dimension $d \in \{0, 1\}$
Output : Paired cell coordinates C, Lengths L

`// Initialize components for pairing`
$UF \leftarrow$ InitializeUnionFind();
for *each vertex* $v \in G$ ***in parallel*** **do**
  UF.Add(v, birth = GetVertexBirth(G, v));
$C_{list} \leftarrow []$;
`// Parallel over batch` B `(CUDA)`
for *each edge* e *in the sorted filtration* (V, Idx) **do**
  $u, v \leftarrow$ GetVerticesOfEdge(e);
  $root_u, root_v \leftarrow UF$.Find($u$), UF.Find(v);
  if $root_u \neq root_v$ **then**
    birth$_u$, birth$_v \leftarrow$ UF.GetBirth($root_u$), UF.GetBirth($root_v$);
    if *birth*$_u \geq$ *birth*$_v$ **then**
      C_{list}.append((coord($root_u$), coord(e)));
      UF.Union($root_u, root_v$, birth = birth$_v$);
    else
      C_{list}.append((coord($root_v$), coord(e)));
      UF.Union($root_u, root_v$, birth = birth$_u$);
if $d = 0$ **then**
  $root_{essential} \leftarrow$ GetFinalComponent(UF);
  C_{list}.append((coord($root_{essential}$), coord$_{inf}$));
$C, L \leftarrow$ FormatAndPack(C_{list});
return C, L;

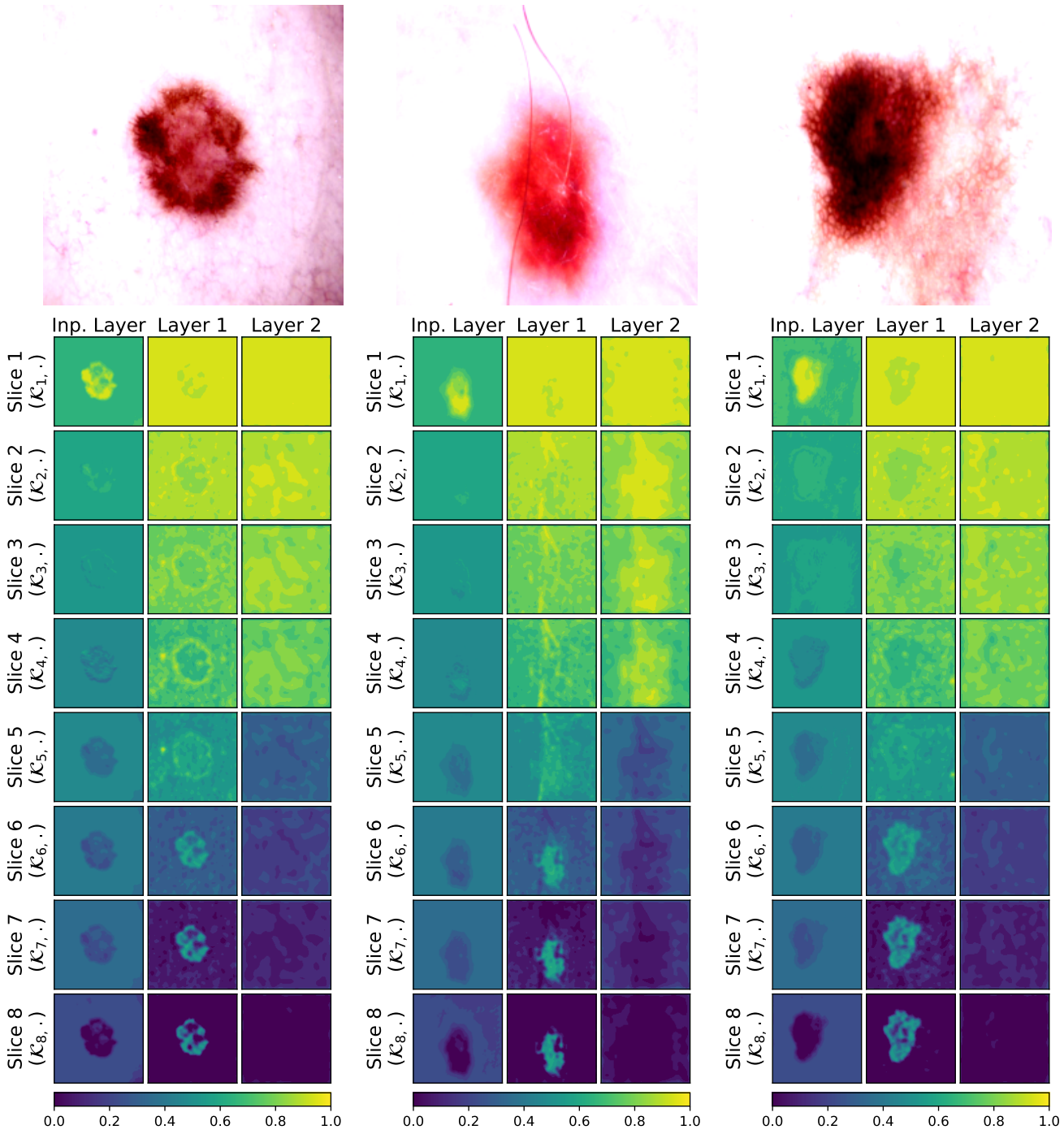

Figure 5. Learned MP Filtration examples for sample images from ISIC dataset [13] for our model TopoSwin-MP with corresponding input images on top. Each slice is a compact representation of one row-wise slice of the learned multifiltration from the input layer, layer 1 and layer 2 learnable cubical multipersistence modules.

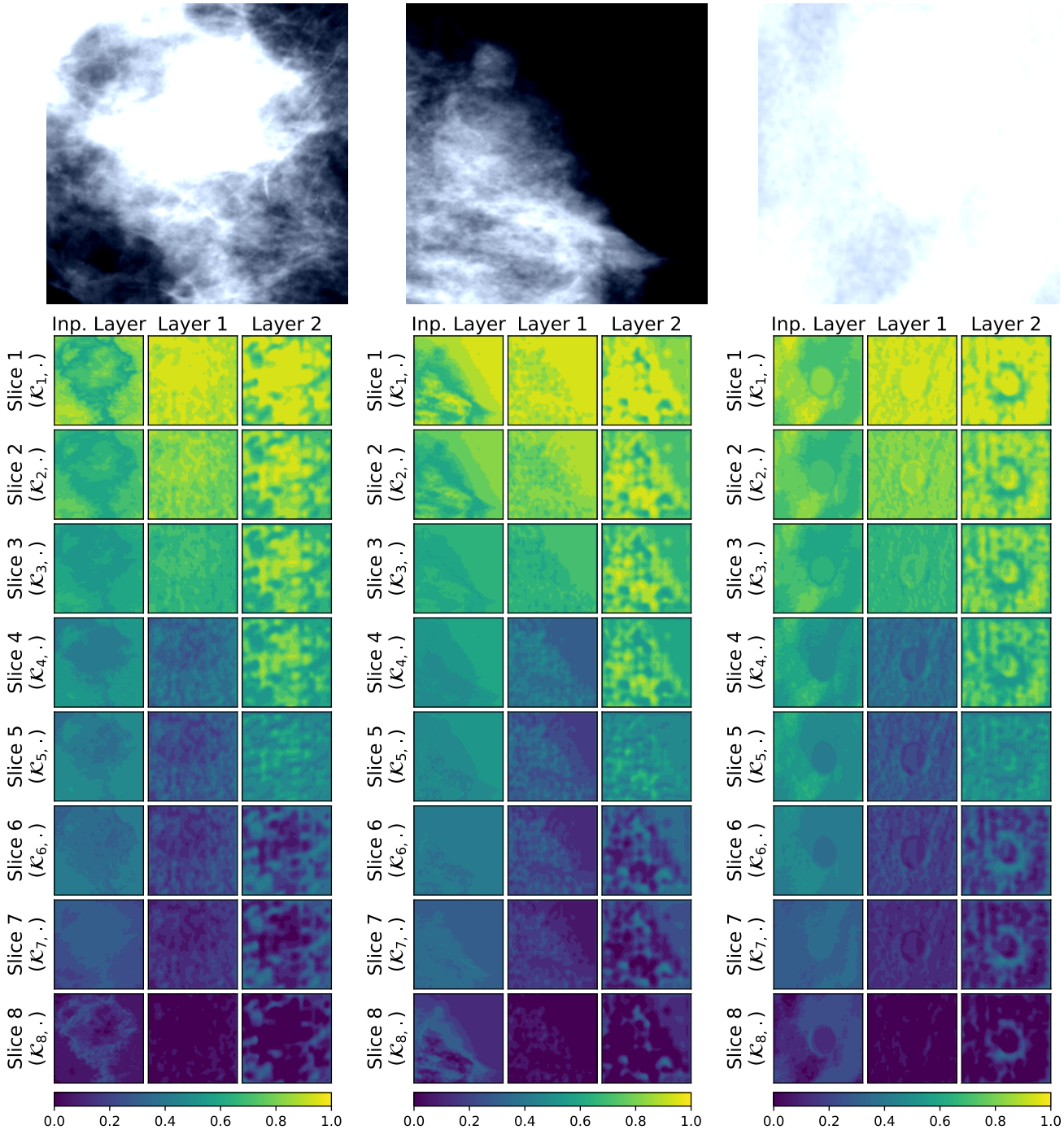

Figure 6. Learned MP Filtration examples for sample images from CBIS-DDSM dataset [20] for our model TopoSwin-MP with corresponding input images on top. Each slice is a compact representation of one row-wise slice of the learned multifiltration from the input layer, layer 1 and layer 2 learnable cubical multipersistence modules.

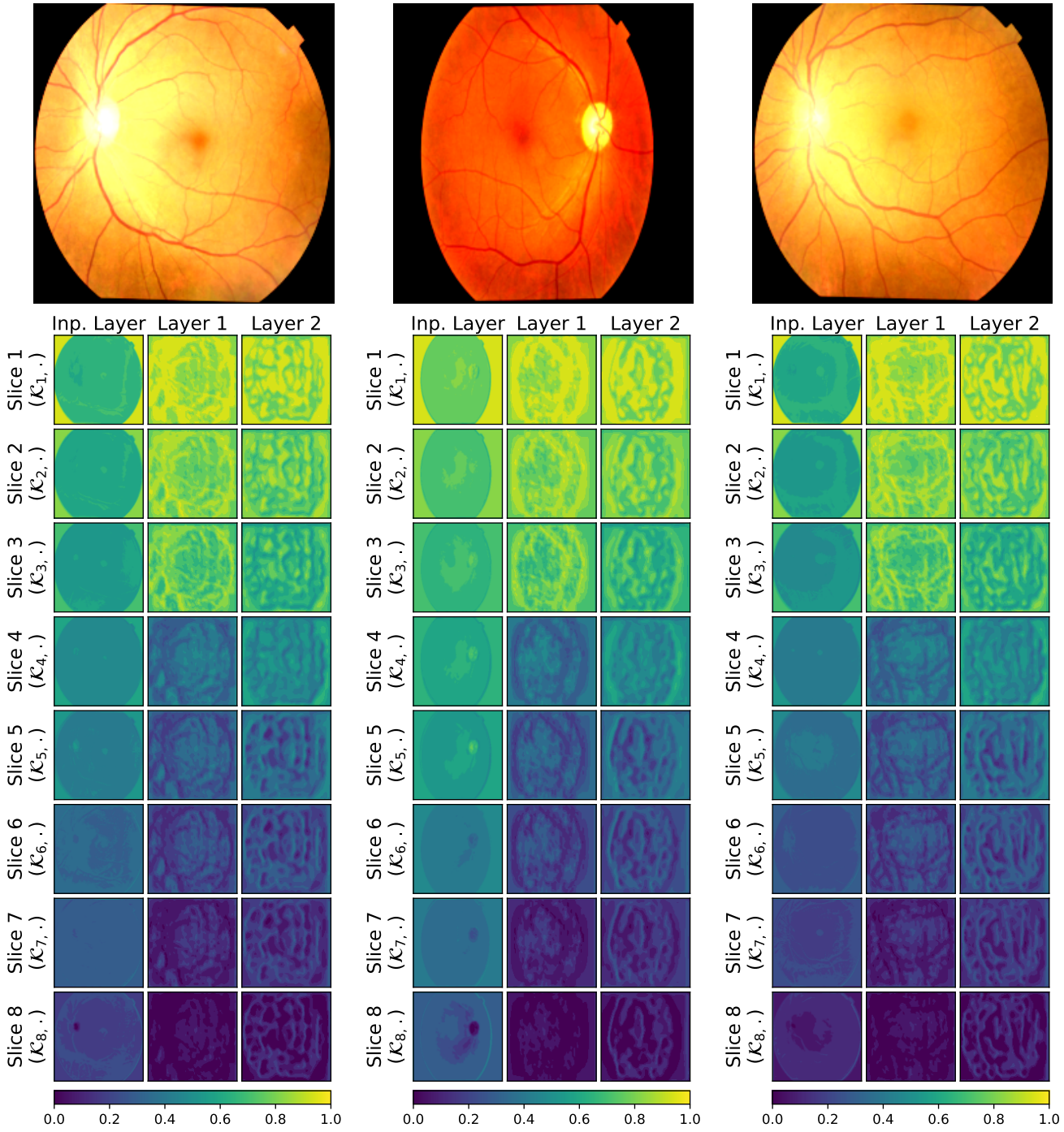

Figure 7. Learned MP Filtration examples for sample images from Glaucoma dataset [27, 29] for our model TopoSwin-MP with corresponding input images on top. Each slice is a compact representation of one row-wise slice of the learned multifiltration from the input layer, layer 1 and layer 2 learnable cubical multipersistence modules.

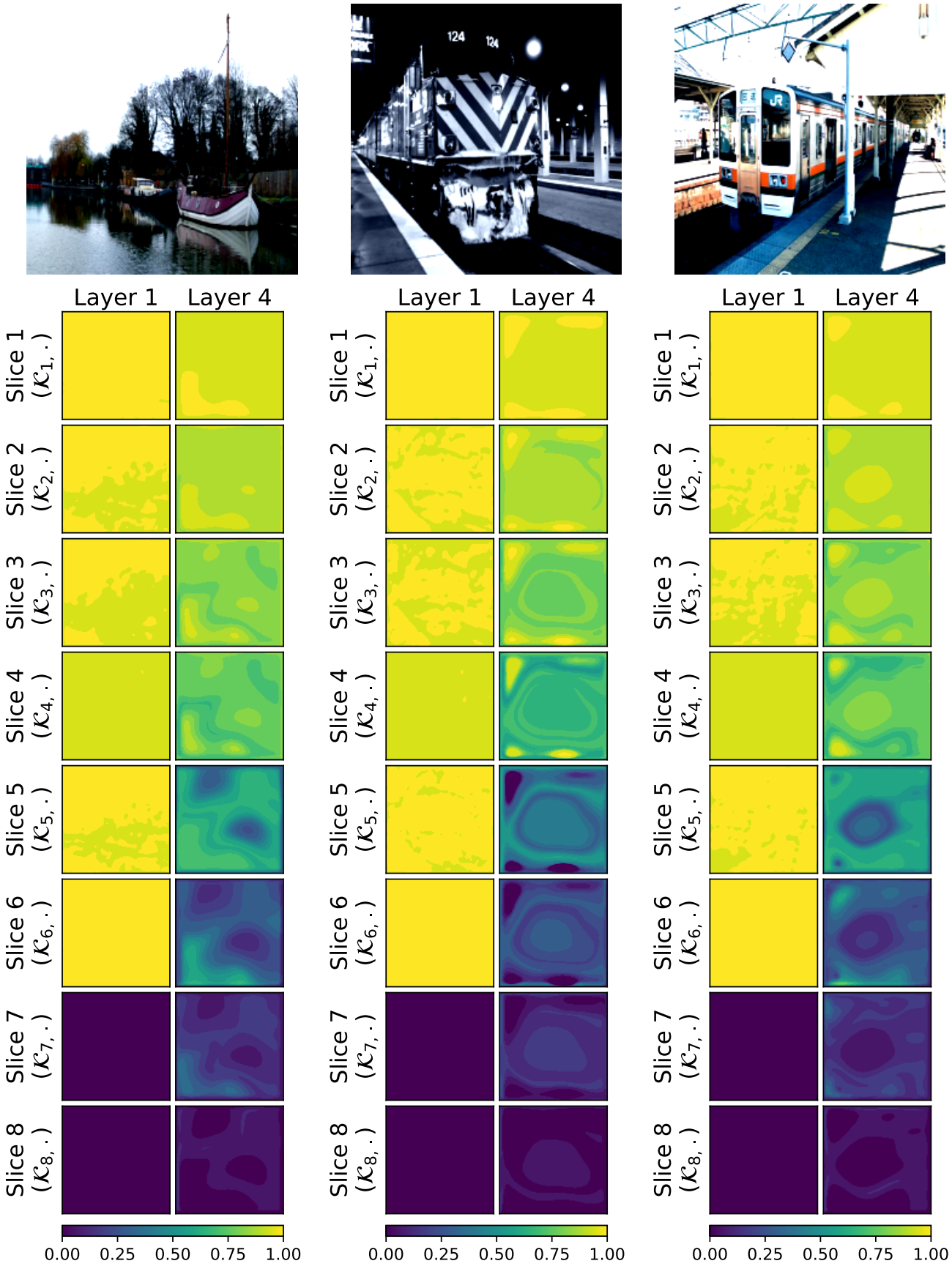

Figure 8. Learned MP Filtration examples for sample images from PASCAL VOC 2012 dataset [16] trained with FCN-Resnet 50 [25] backbone Topo-MP with corresponding input images on top. Each slice is a compact representation of one row-wise slice of the learned multifiltration from the layer 1 and layer 4 learnable cubical multipersistence modules.

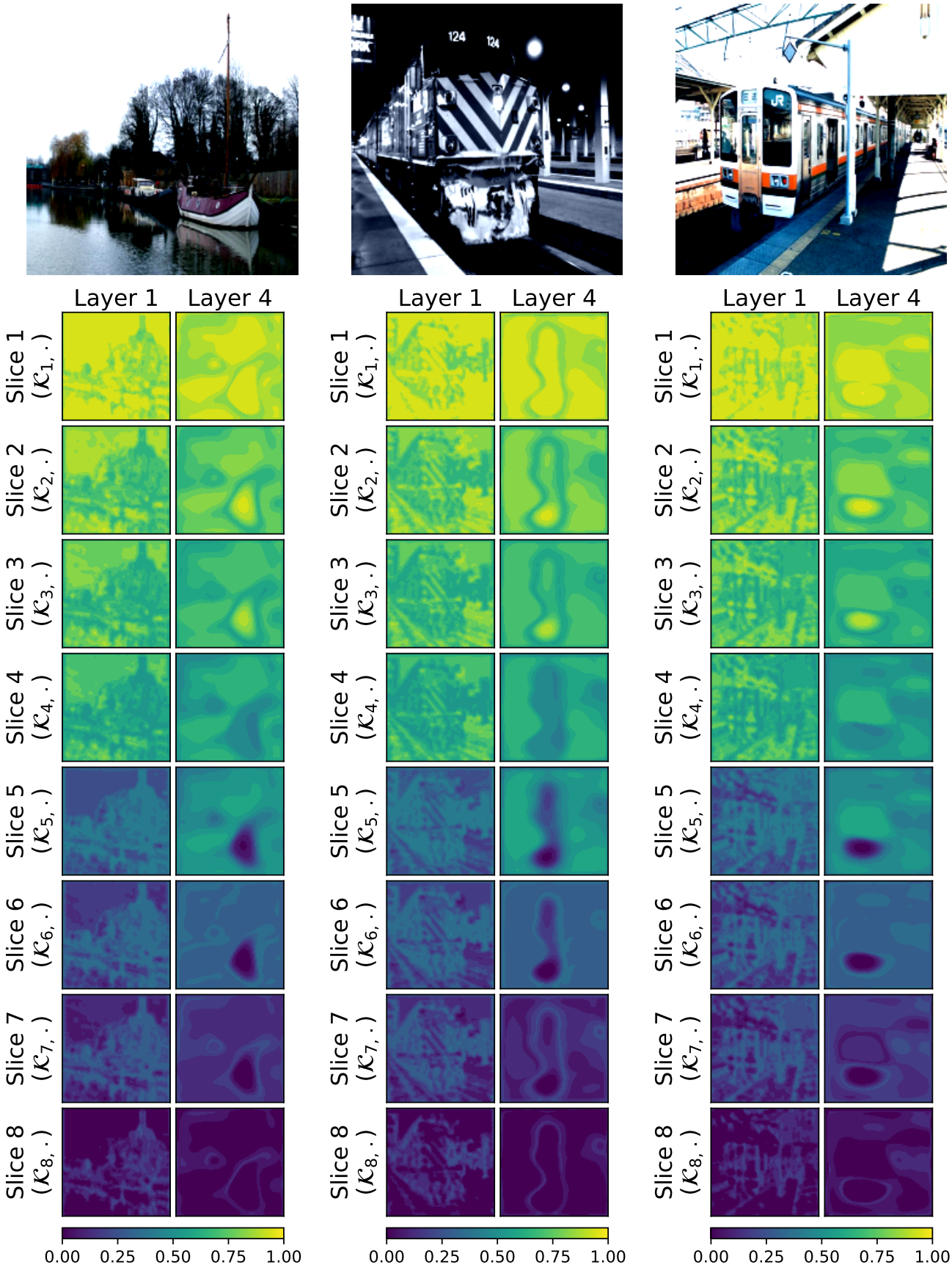

Figure 9. Learned MP Filtration examples for sample images from PASCAL VOC 2012 dataset [16] trained with U-Net [28] backbone Topo-MP with corresponding input images on top. Each slice is a compact representation of one row-wise slice of the learned multifiltration from the layer 1 and layer 4 learnable cubical multipersistence modules.

References

[1] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18 (8):1–35, 2017. 1

[2] Dashti Ali, Aras Asaad, Maria-Jose Jimenez, Vidit Nanda, Eduardo Paluzo-Hidalgo, and Manuel Soriano-Trigueros. A survey of vectorization methods in topological data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 2

[3] Nieves Atienza, Rocío González-Díaz, and Manuel Soriano-Trigueros. On the stability of persistent entropy and new summary functions for topological data analysis. *Pattern Recognition*, 107:107509, 2020. 1

[4] Bea Bleile, Adélie Garin, Teresa Heiss, Kelly Maggs, and Vanessa Robins. *The Persistent Homology of Dual Digital Image Constructions*, page 1–26. Springer International Publishing, 2022. 5

[5] Magnus Bakke Botnan and Michael Lesnick. An introduction to multiparameter persistence. *arXiv preprint arXiv:2203.14289*, 2022. 2, 3

[6] P. Bubenik. Statistical topological data analysis using persistence landscapes. *JMLR*, 16(1):77–102, 2015. 1

[7] Peter Bubenik. Statistical topology using persistence landscapes. *Journal of Machine Learning Research*, 16:77–102, 2015. 2

[8] Mathieu Carrière and Andrew Blumberg. Multiparameter persistence image for topological machine learning. *NeurIPS*, 33, 2020. 2, 3

[9] Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *International Conference on Artificial Intelligence and Statistics*, pages 2786–2796, 2020. 2

[10] Erin W Chambers and Joachim Gudmundsson. Metadiagrams for 2-parameter persistence. In *39th International Symposium on Computational Geometry (SoCG 2023)*, page 25, 2023. 3

[11] Frédéric Chazal, Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, and Larry Wasserman. Stochastic convergence of persistence landscapes and silhouettes. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 474–483, 2014. 1, 2

[12] Yu-Min Chung and Austin Lawson. Persistence curves: A canonical framework for summarizing persistence diagrams. *arXiv preprint arXiv:1904.07768*, 2019. 1

[13] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019. 3, 7

[14] Paweł Dłotko and Davide Gurnari. Euler characteristic curves and profiles: a stable shape invariant for big data problems. *GigaScience*, 12:giad094, 2023. 2

[15] David Eisenbud. *Commutative algebra: with a view toward algebraic geometry*. Springer Science & Business Media, 2013. 2

[16] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 10, 11

[17] Adélie Garin and Guillaume Tauzin. A topological" reading" lesson: Classification of mnist using tda. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1551–1556. IEEE, 2019. 4, 5

[18] Megan Johnson and Jae-Hun Jung. Instability of the betti sequence for persistent homology and a stabilized version of the betti sequence. *Journal of the Korean Society for Industrial and Applied Mathematics*, 25(4):296–311, 2021. 1

[19] Shizuo Kaji, Takeki Sudo, and Kazushi Ahara. Cubical ripser: Software for computing persistent homology of image and volume data, 2020. 5

[20] Rebecca Sawyer Lee, Francisco Gimenez, Assaf Hoogi, Kanae Kawai Miyake, Mia Gorovoy, and Daniel L Rubin. A curated mammography data set for use in computer-aided detection and diagnosis research. *Scientific data*, 4(1):1–9, 2017. 3, 8

[21] Michael Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015. 2

[22] Michael Lesnick and Matthew Wright. Interactive visualization of 2-D persistence modules. *arXiv preprint arXiv:1512.00180*, 2015. 2

[23] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12009–12019, 2022. 4

[24] David Loiseaux, Luis Scoccola, Mathieu Carrière, Magnus Bakke Botnan, and Steve Oudot. Stable vectorization of multiparameter persistent homology using signed barcodes as measures. *Advances in Neural Information Processing Systems*, 36:68316–68342, 2023. 3

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 10

[26] Steve Oudot and Luis Scoccola. On the stability of multigraded betti numbers and hilbert functions. *SIAM Journal on Applied Algebra and Geometry*, 8(1):54–88, 2024. 1, 2

[27] Riadur Rashid, Mohammad Sharmin, Shayla Khatun, Tania Hasan, Md Zahid, and Mohammad Shorif Uddin. Eye disease image dataset, 2024. 3, 9

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 11

[29] Shayla Sharmin, Mohammad Riadur Rashid, Tania Khatun, Md Zahid Hasan, Mohammad Shorif Uddin, et al. A dataset of color fundus images for the detection and classification of eye diseases. *Data in Brief*, 57:110979, 2024. 3, 9

[30] Primoz Skraba and Katharine Turner. Wasserstein stability for persistence diagrams. *arXiv preprint arXiv:2006.16824*, 2020. 1, 2

[31] Robert E. Tarjan and Jan van Leeuwen. Worst-case analysis of set union algorithms. *J. ACM*, 31(2):245–281, 1984. 5

[32] Oliver Vipond. Multiparameter persistence landscapes. *Journal of Machine Learning Research*, 21(61):1–38, 2020. 3