7 Supplementary Materials

7.1 Aerial Image Dataset Roof Masks Downsampling Results

We also tested on another binary image dataset consisting of building roof masks in real-world aerial images of urban areas (see Table 4 and Figure 8). We built our dataset of 255 512x512 images by randomly subtracting sub-images from an satellites aerial image dataset (by Humans in the Loop group with the Mohammed Bin Rashid Space Center[4]), which originally contains 72 images of various sizes (from 509x544 to 2149x1479). As mentioned in Section 3, we temporarily added a ring of white pixels to the image boundaries to simplify boundary handling cases. We point out that this dataset is more challenging than the CNCB dataset we tested in Section 5, as it contains a lot of small and relatively thin components. This leads to even more topologically incorrect downsampling results (compared to CNCB dataset) by the traditional non topology-preserving methods. However, our IP-based method and the dilation-based method still produced completely topologically correct results. Our method's results still have similar levels of pixel-wise accuracy as the traditional methods, while the dilation method's accuracy is much worse. On the downside, this dataset is more challenging to have topologically correct results - there are 8, 3, 14, and 42 infeasible cases for downsampling factors 2, 4, 8, and 16, respectively.

Table 4: Quantitative and speed comparisons of different downsampling methods on the 255 segmentation masks in the aerial dataset to different sizes. Best and second-best results are marked in red and blue, respectively.

	512x512 to 256x256 (factor 2)					512x512 to 128x128 (factor 4)				
Method	↑ IoU	↑ Dice	↓ Betti num. error	↓ PH distance	↓ Avg. time (s)	↑ IoU	↑ Dice	↓ Betti num. error	↓ PH distance	↓ Avg. time (s)
Bicubic	94.56%	97.15%	2.47	0.077	0.001	86.75%	92.50%	4.62	0.135	0.001
Pooling	94.81%	97.29%	0.98	0.056	0.675	87.85%	93.32%	2.43	0.124	0.174
ACN [6]	93.59%	96.63%	5.15	0.215	0.190	81.98%	89.74%	5.67	0.240	0.285
Dilation	55.42%	67.81%	0.00	0.174	37.702	54.41%	67.68%	0.00	0.176	8.958
Ours	94.71%	97.24%	0.00	0.031	3.137	87.33%	92.97%	0.00	0.055	3.180

	512x512 to 64x64 (factor 8)					512x512 to 32x32 (factor 16)				
Method	↑ IoU	↑ Dice	↓ Betti num. error	↓ PH distance	↓ Avg. time (s)	↑ IoU	↑ Dice	↓ Betti num. error	↓ PH distance	↓ Avg. time (s)
Bicubic	78.45%	87.15%	10.14	0.213	0.001	63.13%	75.97%	13.57	0.292	0.001
Pooling	78.94%	87.39%	7.28	0.195	0.051	63.60%	76.11%	11.65	0.279	0.018
ACN [6]	66.77%	79.78%	10.57	0.249	0.365	48.28%	63.83&	12.84	0.287	0.445
Dilation	48.23%	62.75%	0.00	0.174	2.296	34.18%	48.55%	0.00	0.194	0.496
Ours	78.62%	87.41%	0.00	0.083	3.494	61.37%	74.60%	0.00	0.115	5.348

[4] https://humansintheloop.org/resources/datasets/semantic-segmentation-dataset-2/

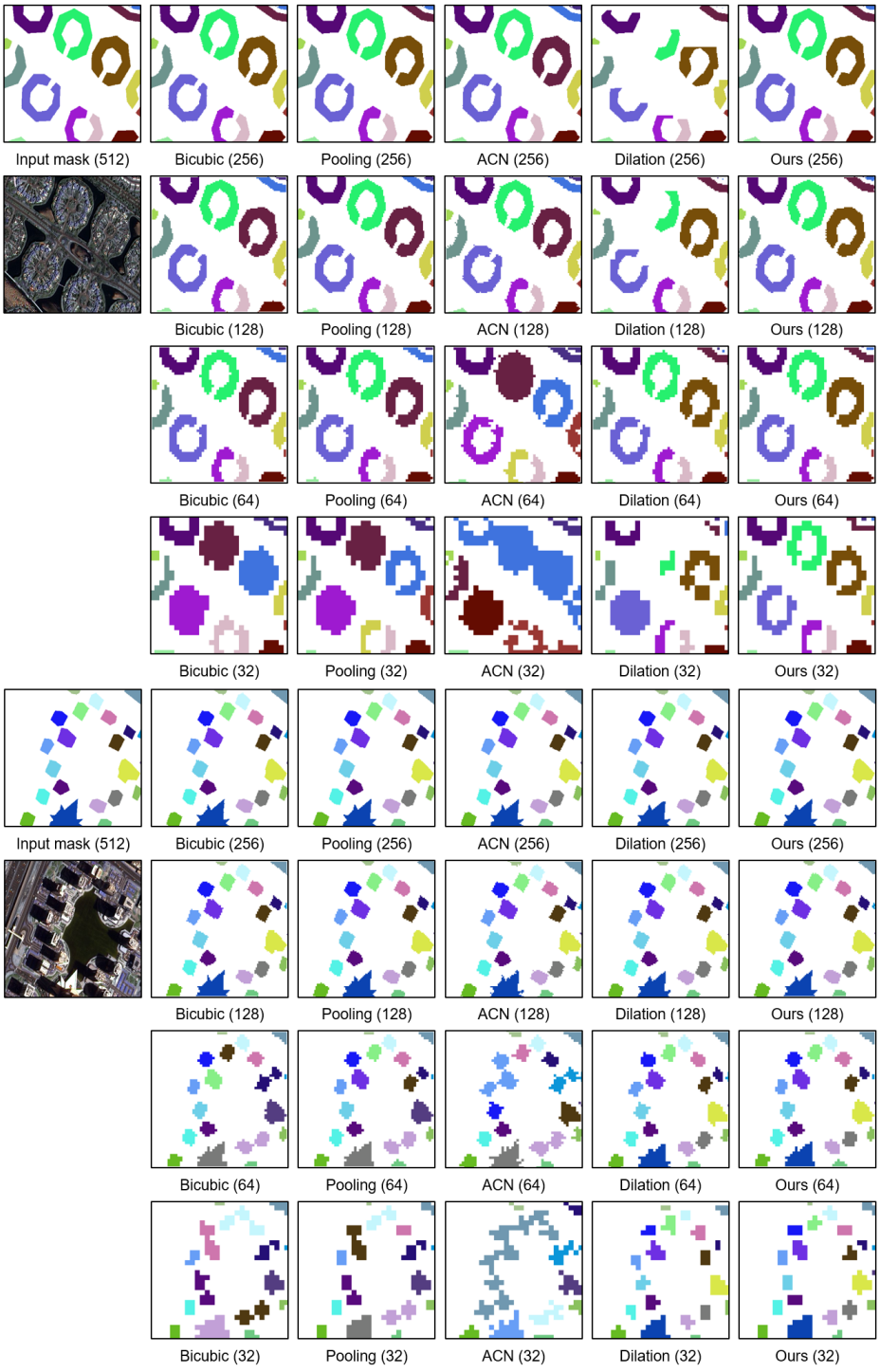

Fig. 8: Aerial roof mask dataset downsampling results. These are challenging problems that non topology-preserving methods often generate results with altered topology.

7.2 Comparisons to works by Passat et al. and Ngo et al. [16], [18], and [15]

Table 5: Comparing [16] and ours. We tested on both our datasets (CNCB and roof mask dataset) of 512x512 input images. A non-success for [1] means it cannot reach a feasible solution. For ours it means the solver declares the problem to be infeasible (with default coverage value) or could not finish in time. We find that [1] often cannot reach a feasible solution for downsampling tasks w/ factor > 2. Moreover, [1]'s results' PH distances are much worse.

	IOU, Dice	PH dist	success%	time
[1], factor2	94.07, 96.89	0.247	82.00%	1.08
Our, factor2	93.91, 96.79	0.018	98.00%	2.56
[1], factor4	85.64, 91.94	0.241	41.00%	2.35
Our, factor4	86.23, 92.26	0.034	99.00%	2.00
[1], factor8	71.18, 82.28	0.181	10.35%	4.81
Our, factor8	75.97, 85.48	0.057	96.95%	2.05

[18] and [15] both analyzed the necessary and sufficient conditions for a 2D binary image to retain topology after arbitrary rigid transformations (translations and rotations, but no scaling). In [15], two algorithms to transform a 2D binary image to be a topology-invariant-under-affine-transformation one are proposed. The first is by fixing "at-fault" pixels (ones that can break topology after rigid transforms) one-by-one and the second is by doing a super-resolution (but no subsequent downsampling is discussed). Note that down-scaling is not a rigid transformation so their results are not directly applicable to our problems. In [16], the authors extend the discussions to affine transformations (including scaling) and proposed an algorithm to do arbitrary topology-invariant affine transformation. In short, the algorithm (Alg.3 in their paper) starts by subdividing the initial binary image to fit in the target (affine transformed) grid, with possibly lots of arbitrarily-shaped polygons. Next, the subdivided polygon mesh is converted to one that is fully compatible to the target grid by applying morphological operations (erosion and dilation) one polygon-by-one. The operations are picked in a greedy, gradient-descent manner, sorted by an elegant scoring function (their Eq.85). However, the algorithm doesn't have the guarantee that a feasible solution (if one exists) is always reached (note that it starts with an infeasible solution). In this regard, we consider it less reliable than our baseline greedy dilation-based approach (Sec. 4 in our paper) because ours instead start with a feasible solution and improves upon it by applying topology-preserving dilation operations iteratively. We got the source codes from the authors and fixed some issues to make it work for large downsampling tasks. See Table 5 for testing results.

7.3 Proof of Lemma 1

We recap the Lemma here:

Faces of the opposite of all the half-edges in a boundary all belong to the same component.

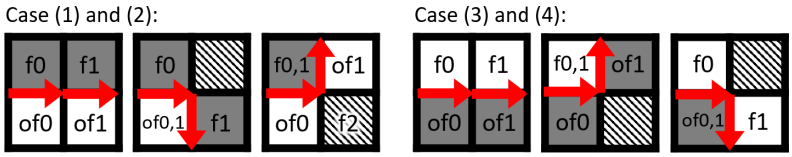

Fig. 9: Showing the cases for the proof of Lemma 1.

Proof. We prove it by showing that the faces of the opposite of every two consecutive half-edges must belong to the same component. We denote two consecutive boundary half-edges as $e0$ and $e1$,their faces as $f0$ and $f1$, and their opposite's faces as $of0$ and $of1$, respectively. We discuss four possible cases of a boundary: (1) a outer boundary of a black component, (2) an inner boundary of a black boundary, (3) a outer boundary of a white component, and (4) an inner boundary of a white component. Illustrations are shown in Figure 9.

For case (1) and (2), if $f0$ and $f1$ share an edge, then $of0$ and $of1$ must belong to the same white component because they share one edge. If $f0$ and $f1$ shares only a vertex, then $of0$ and $of1$ are actually the same face. Now for the case that $f0$ and $f1$ are the same face, assume for a moment that $of0$ and $of1$ belong to different white components. This means that the face, call it $f2$, that is adjacent to both $of0$ and $of1$, but not $f0/f1$, must be a black face (so to separate $of0$ and $of1$). However, this is a contradiction as $f2$ is adjacent to $f0/f1$ via a vertex and should have been included into the black component, which would violate the configuration of the two consecutive half-edges.

For case (3) and (4), if $f0$ and $f1$ share an edge, then $of0$ and $of1$ must belong to the same black component. If $f0$ and $f1$ are the same, then $of0$ and $of1$ are of the same black component because they share a vertex. If $f0$ and $f1$ are different, then $of0$ and $of1$ are the same. □

7.4 Proving that constraints (2), (3), and (4) of the IP problem have ensured that the original and the solved downsampled binary images have the same DAG

We show a sketch of proof as follows.

Proof. Constraint (2) ensures that every nodes of the original DAG (i.e., every black and white components) exist in the downsampled DAG. Constraint (3) ensures that a black big-pixel is adjacent to either another black big-pixel of the

same component, or to a white big-pixel. The same applies for white big-pixels. In other words, any two components of the same color won't touch each other.

However, the above two constraints still don't guarantee each original component remains one single component in the downsampled image. Now, constraint (4) dictates that every boundary between a black and a white component (which means all kinds of component-component boundaries as it is impossible to have two components of the same color to be adjacent to each other) remain a single close loop. This rules out the possibility of one original component become multiple connected components. □

7.5 IP Problem Solving Outcomes Analysis

In short, any solution that our method outputs will have the same topology as the input image. To be precise, there are 4 possible outcomes:

- 1) and 2): The problem is solved optimally or sub-optimally (due to time limit) and a topologically correct solution with an optimal or sub-optimal objective value (in terms of pixel-wise similarity to the input image) is found.
- 3): The Gurobi solver finds the problem to be infeasible (w.r.t. the predefined "coverage" dx / dy value - using a larger coverage value may, but not always, leads to more feasible solutions). Note that this is a certain conclusion (no feasible solution w.r.t. the coverage value would exist if the problem is found to be infeasible).
- 4): Gurobi could not find a solution nor being able to declare the problem to be infeasible within the time limit (we used 60 seconds). In our experiments, we found such cases to be rare (such cases are included in average time calculations throughout our tests).

7.6 Ablation Study of the IP Formulation Constraints Design

The necessities of constraint (1) (i.e., the image shall be fully covered by big-pixels without overlaps), (2) (i.e., every component shall appear), and (4) (the boundary between every pair of adjacent components shall be a single closed loop) in Section 3 are apparent. However, it may not be clear if constraint (3) is needed when constraint (4) is already in place. We show a counter example and a discussion in Figure 10.

7.7 More CNCB Dataset Downsampling Qualitative Results

We show more qualitative results on the CNCB dataset with more downsampling factors (from 2 to 16) in Figure 11, Figure 12, and Figure 13. We point out that at big downsampling factors, results done by non topology-preserving methods (e.g., bicubic, pooling, and ACN) often have altered connected components, such as small components (or holes) got erased, multiple components/holes got connected, or one component/hole got separated into many.

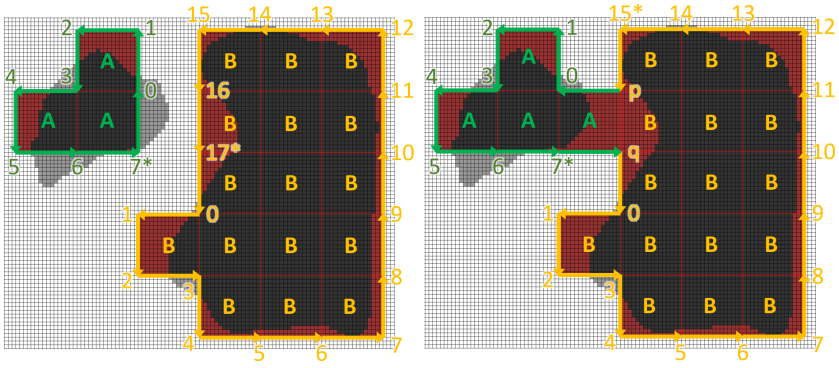

Fig. 10: We show the necessity of constraint (3) in Section 3. In the right, we show that the two black components become a single one when constraint (3) is not enforced even when constraint (4) is being applied. The two components are represented by faces marked in A and B, respectively. Observe that in the right, constraint (4) actually still applies. This is because at point p and q, none of the corner configurations are applicable, so their Boolean variables are not enumerated. This allowed the outer boundaries of component A and B to become non-closed intervals.

7.8 Persistent Diagram Comparisons

In Section 5, we have shown quantitative comparisons (in terms of PH distances) of persistent diagrams of original binary images and downsampled images by our method and other methods. To have an qualitative comparison, in Figure 14, we show persistent diagrams of an original 512x512 binary image and its downsampled versions by our method, and some results done by other methods. We see that persistent diagrams of our downsampled images are visually similar to the original. In comparison, persistent diagrams of results done by other methods, having significantly higher PH distances, look more different to the original.

7.9 Shortest Path Comparisons

In Figure 15, we show approximate shortest paths in our downsampled images. We also show a false negative (FN) case and a false positive (FP) case, both led to highly incorrect shortest path estimations, that may happen in topologically incorrect downsampled images.

7.10 Bigger Resolution Inputs Testing Results

To see how our method performs on bigger ($\geq$ 512x512) input binary images, we tested on two big images found in the aerial image dataset. Statistics are shown in Table 6 and downsampling results are shown in Figure 16. By comparing to Table 4, we see that the computational cost of downsampling a 1024x1024 image to 64x64 is only slightly higher then converting a 512x512 image to 64x64 (from 3.494 sec to 3.647 sec). The same can be said about comparing converting 1024x1024 and 512x125 images to 32x32 images (from 5.348 sec to 5.550 sec).

Table 6: Quantitative and speed comparisons of different downsampling methods on two bigger (1024x1024) input binary images in the aerial image dataset to different sizes. Best and second-best results are marked in red and blue, respectively.

	1024x1024 to 64x64 (factor 16)					1024x1024 to 32x32 (factor 32)				
Method	↑ IoU	↑ Dice	↓ Betti num. error	↓ PH distance	↓ Avg. time (s)	↑ IoU	↑ Dice	↓ Betti num. error	↓ PH distance	↓ Avg. time (s)
Bicubic	76.62%	86.56%	5.00	0.158	0.004	59.87%	74.33%	10.58	0.179	0.004
Pooling	77.00%	86.81%	4.42	0.175	0.066	59.53%	73.83%	10.67	0.202	0.038
ACN [6]	62.39%	76.42%	9.17	0.212	0.801	43.83%	60.49&	12.17	0.229	0.879
Dilation	56.56%	71.50%	0.00	0.107	2.232	37.12%	53.31%	0.00	0.159	0.674
Ours	76.52%	86.51%	0.00	0.075	3.647	54.43&	69.69%	0.00	0.095	5.550

7.11 Anisotropic Downsampling Results

Our method supports anisotropic downsampling by using different downsampling factors in width and height. In Figure 17, we show one such result of downsampling a 512x512 image into a 32x64 one with downsampling factor 16 in width and 8 in height.

Fig. 11: CNCB dataset downsampling results at all factors (2, 4, 8, and 16). Observe that results done by non topology-preserving methods often have altered connected components, especially at large factors (16).

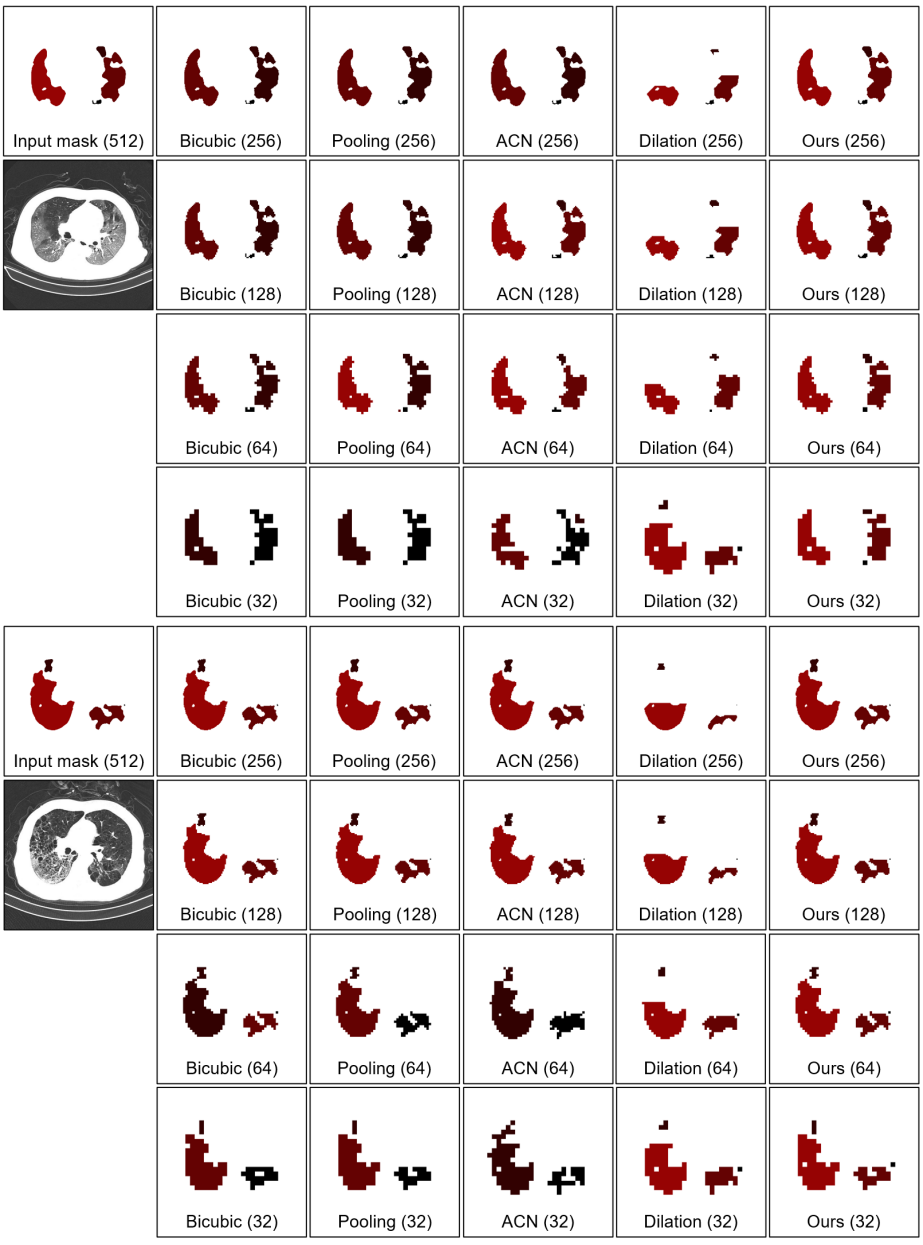

Fig. 12: CNCB dataset downsampling results at all factors (2, 4, 8, and 16).

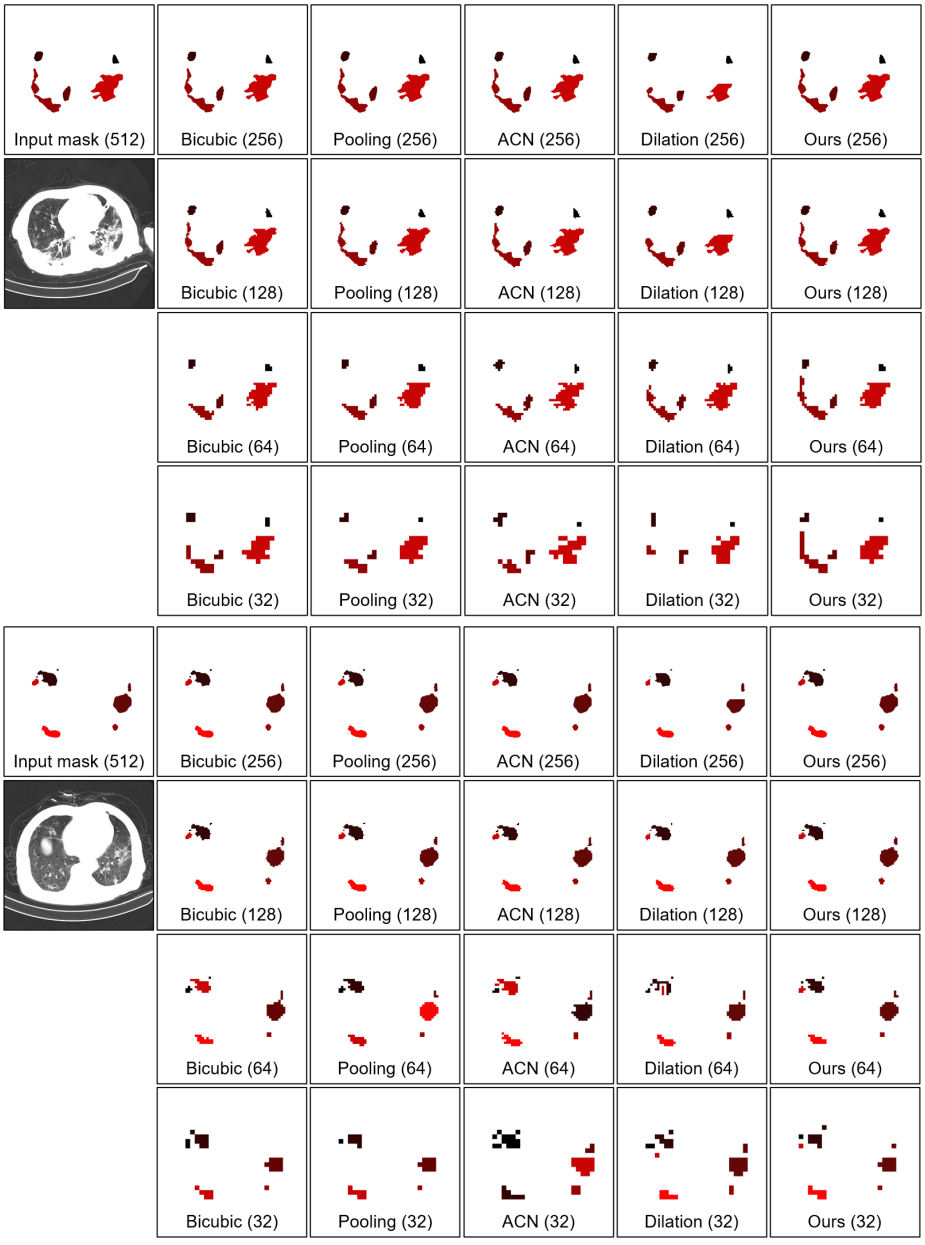

Fig. 13: CNCB dataset downsampling results at all factors (2, 4, 8, and 16).

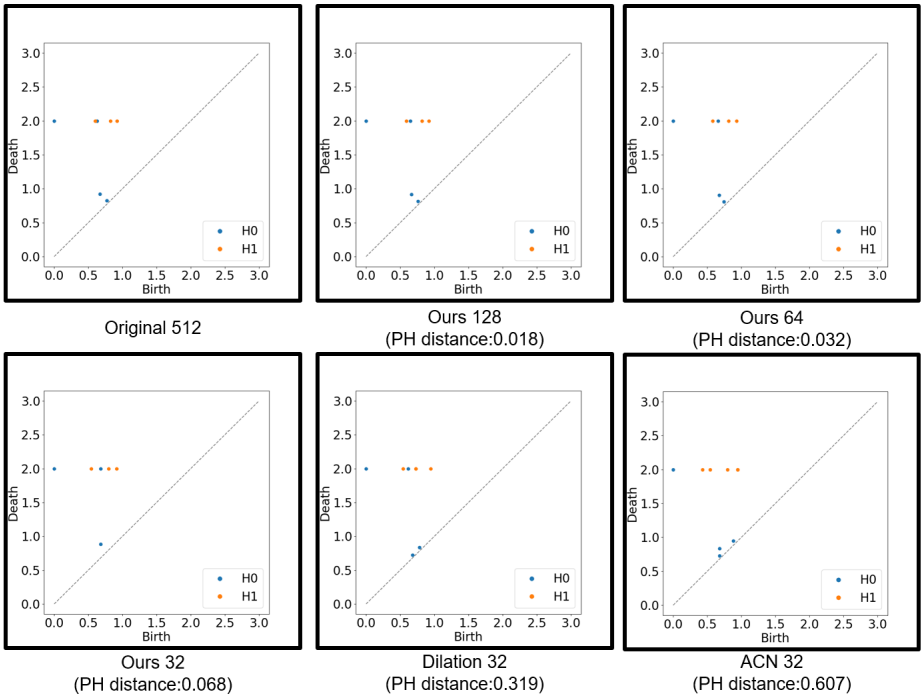

Fig. 14: Persistent diagrams of an original 512x512 binary image and its downsampled versions by our method (to 128x128, 64x64, and 32x32), and some results done by other methods (32x32).

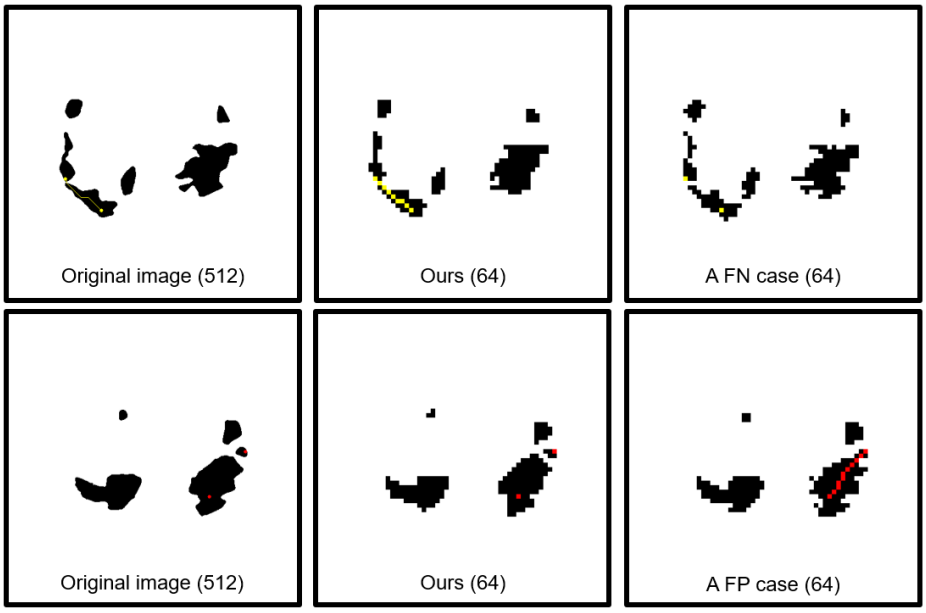

Fig. 15: Left: a shortest path (between two yellow pixels) and a pair of pixels without a path (red) in an original image. Middle: shortest path finding results in our down-sampled image. Right: we show a false negative (FN) case because the component containing the two yellow pixels now becomes two. We also show a false positive (FP) case because the two red pixels now belong to the same component and a path emerges.

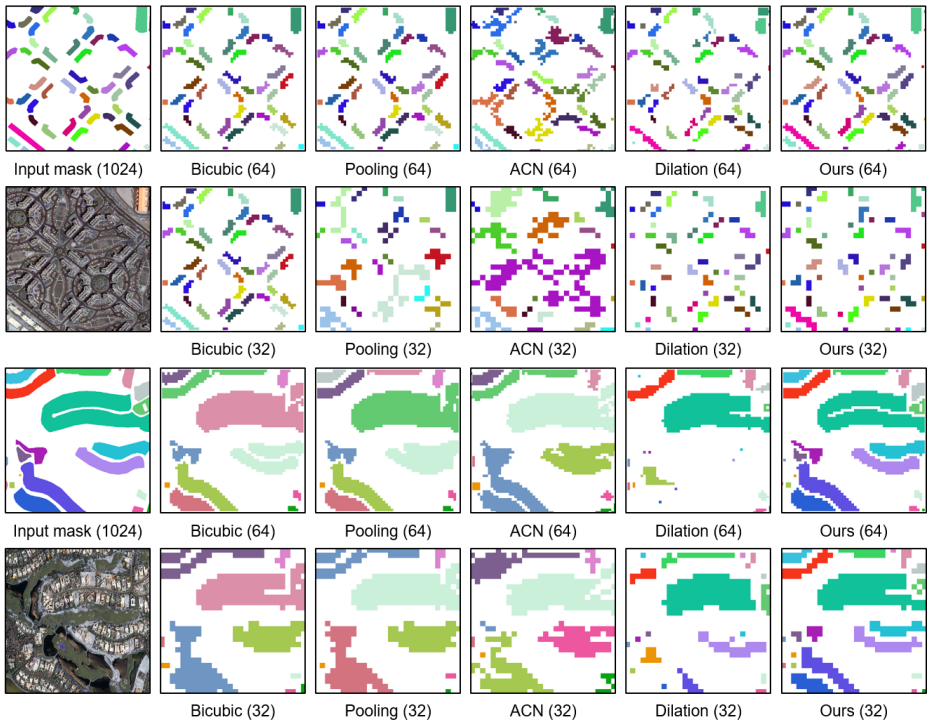

Fig. 16: Downsampling 1024x1024 images to 64x64 (factor 16) and 32x32 (factor 32).

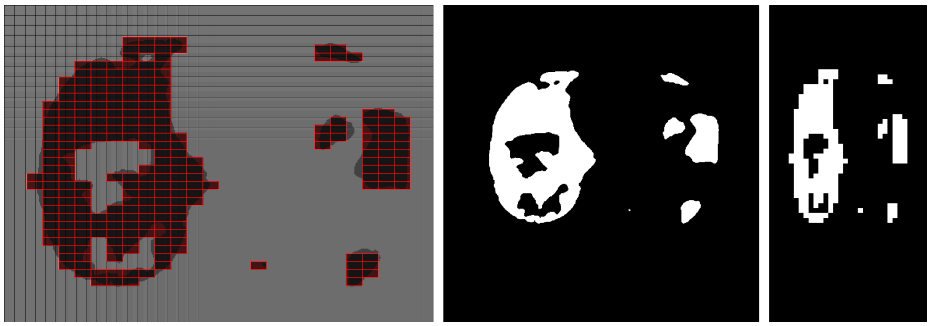

Fig. 17: Left: visualizing a downsampling result with factor 16 in width and 8 in height. Observe that each big-pixel is a 16-by-8 rectangle. Middle: original image, Right: downsampled image.