1. Theory - clDice in Digital Topology

In addition to our Theorem 1 in the main paper, we are providing intuitive interpretations of clDice from the digital topology perspective. Betti numbers describe and quantify topological differences in algebraic topology. The first three Betti numbers ($\beta_0$, $\beta_1$, and $\beta_2$) comprehensively capture the manifolds appearing in 2D and 3D topological space. Specifically,

- $\beta_0$ represents the number of connected-components,
- $\beta_1$ represents the number of circular holes, and
- $\beta_2$ represents the number of cavities (Only in 3D).

Using the concepts of Betti numbers and digital topology by Kong et al. [3, 6], we formulate the effect of topological changes between a true binary mask ($V$) and a predicted binary mask ($\hat{V}_p$) in Fig. 2. We will use the following definition of ghosts and misses, see Fig. 2.

1. Ghosts in skeleton: We define ghosts in the predicted skeleton ($S_p$) when $S_p \not\subseteq V_L$. This means the predicted skeleton is not completely included in the true mask. In other words, there exist false-positives in the prediction, which survive after skeletonization.

2. Misses in skeleton: We define misses in the predicted skeleton ($S_p$) when $S_p \not\subseteq V_L$. This means the true skeleton is not completely included in the predicted mask. In other words, there are false-negatives in the prediction, which survive after skeletonization.

The false positives and false negatives are denoted by $V_P \setminus V_L$ and $V_L \setminus V_P$, respectively, where $\setminus$ denotes a set difference operation. The loss function aims to minimize both errors. We call an error correction to happen when the value of a previously false-negative or false-positive voxel flips to a correct value. Commonly used voxel-wise loss functions, such as Dice-loss, treat every false-positive and false-negative equally, irrespective of the improvement in regards to topological differences upon their individual error correction. Thus, they cannot guarantee homotopy equivalence until and unless every single voxel is correctly classified. In stark contrast, we show in the following proposition that clDice guarantees homotopy equivalence under a minimum error correction.

**Proposition 1.** For any topological differences between $V_P$ and $V_L$, achieving optimal clDice to guarantee homotopy equivalence requires a minimum error correction of $V_P$.

**Proof.** From Fig 2, any topological differences between $V_P$ and $V_L$ will result in ghosts or misses in the foreground or background skeleton. Therefore, removing ghosts and misses are sufficient conditions to remove topological differences. Without the loss of generalizability, we consider the case of ghosts and misses separately:

For a ghost $g \subset S_P$, $\exists$ a set of predicted voxels $E1 \subset \{V_P \setminus V_L\}$ such that $V_P \setminus E1$ does not create any misses and removes $g$. Without the loss of generalizability, let’s assume that there is only one ghost $g$. Now, to remove $g$, under a minimum error correction of $V_P$, we have to minimize $|E1|$. Let’s say an optimum solution $E1_{\text{min}}$ exists. By construction, this implies that $V_P \setminus E1_{\text{min}}$ removes $g$.

For a miss $m \subset V_L$, $\exists$ a set of predicted voxels $E2 \subset \{V_L \setminus V_P\}$ such that $V_P \cup E2$ does not create any ghosts and removes $m$. Without the loss of generalizability, let’s assume that there is only one miss $m$. Now, to remove $m$, under a minimum error correction of $V_P$, we have to minimize $|E2|$. Let’s say an optimum solution $E2_{\text{min}}$ exists. By construction, this implies that $V_P \cup E2_{\text{min}}$ removes $m$.

Thus, in the absence of any ghosts and misses, from Lemma 1.1, clDice=1 for both foreground and background. Finally, Therefore, Theorem 1 (from the main paper) guarantees homotopy equivalence.

**Lemma 1.1.** In the absence of any ghosts and misses clDice=1.
Proof. The absence of any ghosts $S_P \in V_L$ implies $T_{prec} = 1$; and the absence of any misses $S_L \in V_P$ implies $T_{sens} = 1$. Hence, $clDice = 1$.

1.1. Interpretation of the Adaptation to Highly Unbalanced Data According to Digital Topology:

Considering the adaptions we described in the main text, the following provides analysis on how these assumptions and adaptions are funded in the concept of ghosts and misses, described in the previous proofs. Importantly, the described adaptions are not detrimental to the performance of $clDice$ for our datasets. We attribute this to the non-applicability of the necessary conditions specific to the background (i.e. II, IV, VI, VII, and IX in Figure 1), as explained below:

- II. → In tubular structures, all foreground objects are eccentric (or anisotropic). Therefore isotropic skeletonization will highly likely produce a ghost in the foreground.
- IV. → Creating a hole outside the labeled mask means adding a ghost in the foreground. Creating a hole inside the labeled mask is extremely unlikely because no such holes exist in our training data.
- VI. → The deletion of a hole without creating a miss is extremely unlikely because of the sparsity of the data.
- VII and IX. (only for 3D) → Creating or removing a cavity is very unlikely because no cavities exist in our training data.

2. Additional Qualitative Results
Figure 3. Qualitative results: for the Massachusetts Road dataset and for the DRIVE retina dataset (last row). From left to right, the real image, the label, the prediction using soft-dice and the predictions using the proposed $\mathcal{L}_c(\alpha = 0.5)$, respectively. The first three rows are U-Net results and the fourth row is an FCN result. This indicates that soft-$\alpha$Dice segments road connections which the soft-dice loss misses. Some, but not all, missed connections are indicated with solid red arrows, false positives are indicated with red-yellow arrows.
Figure 4. Qualitative results: 2D slices of the 3D vessel dataset for different sized field of views. From left to right, the real image, the label, the prediction using soft-dice and the U-Net predictions using $L_c(\alpha = 0.4)$, respectively. These images show that soft-clDice helps to better segment the vessel connections. Importantly the networks trained using soft-dice over-segment the vessel radius and segments incorrect connections. Both of these errors are not present when we train including soft-clDice in the loss. Some, but not all, false positive connections are indicated with red-yellow arrows.

3. Comparison to Other Literature:

A recent pre-print proposed a region-separation approach, which aims to tackle the issue by analysing disconnected foreground elements [5]. Starting with the predicted distance map, a network learns to close ambiguous gaps by referring to a ground truth map which is dilated by a five-pixel kernel, which is used to cover the ambiguity. However, this does not generalize to scenarios with a close or highly varying proximity of the foreground elements (as is the case for e.g. capillary vessels, synaptic gaps or irregular road intersections). Any two foreground objects which are placed at a twice-of-kernel-size distance or closer to each other will potentially be connected by the trained network. This is facilitated by the loss function considering the gap as a foreground due to performing dilation in the training stage. Generalizing their approach to smaller kernels has been described as infeasible in their paper [5].
4. Datasets and Training Routine

For the DRIVE vessel segmentation dataset, we perform three-fold cross-validation with 30 images and deploy the best performing model on the test set with 10 images. For the Massachusetts Roads dataset, we choose a subset of 120 images (ignoring imaged without a network of roads) for three-fold cross-validation and test the models on the 13 official test images. For CREMI, we perform three-fold cross-validation on 324 images and test on 51 images. For the 3D synthetic dataset, we perform experiments using 15 volumes for training, 2 for validation, and 5 for testing. For the Vessap dataset, we use 11 volumes for training, 2 for validation, and 4 for testing. In each of these cases, we report the performance of the model with the highest cIDice score on the validation set.

5. Network Architectures

We use the following notation: $In(input\ channels)$, $Out(output\ channels)$, $B(output\ channels)$ present input, output, and bottleneck information (for U-Net); $C(filter\ size, output\ channels)$ denote a convolutional layer followed by ReLU and batch-normalization; $U(filter\ size, output\ channels)$ denote a transposed convolutional layer followed by ReLU and batch-normalization; $\downarrow 2$ denotes maxpooling; $\oplus$ indicates concatenation of information from an encoder block. We had to choose a different FCN architecture for the Massachusetts road dataset because we realize that a larger model is needed to learn useful features for this complex task.

5.1. Drive Dataset

5.1.1 FCN :

$IN(3\ ch) \rightarrow C(3, 5) \rightarrow C(5, 10) \rightarrow C(5, 20) \rightarrow C(3, 50) \rightarrow C(1, 1) \rightarrow Out(1)$

5.1.2 Unet :

ConvBlock : $C_B(3, output\ size) \equiv C(3, output\ size) \rightarrow C(3, output\ size) \rightarrow \downarrow 2$

UpConvBlock : $U_B(3, output\ size) \equiv U(3, output\ size) \rightarrow \oplus \rightarrow C(3, output\ size)$

Encoder : $IN(3\ ch) \rightarrow C_B(3, 64) \rightarrow C_B(3, 128) \rightarrow C_B(3, 256) \rightarrow C_B(3, 512) \rightarrow C_B(3, 1024) \rightarrow B(1024)$

Decoder : $B(1024) \rightarrow U_B(3, 1024) \rightarrow U_B(3, 512) \rightarrow U_B(3, 256) \rightarrow U_B(3, 128) \rightarrow U_B(3, 64) \rightarrow Out(1)$

5.2. Road Dataset

5.2.1 FCN :

$IN(3\ ch) \rightarrow C(3, 10) \rightarrow C(5, 20) \rightarrow C(7, 30) \rightarrow C(11, 30) \rightarrow C(7, 40) \rightarrow C(5, 50) \rightarrow C(3, 60) \rightarrow C(1, 1) \rightarrow Out(1)$

5.2.2 Unet :

Same as Drive Dataset, except we used $2 \times 2$ up-convolutions instead of bilinear up-sampling followed by a 2D-convolution with kernel size 1.

5.3. CREMI Dataset

5.3.1 Unet :

Same as Road Dataset.

5.4. 3D Dataset

5.4.1 3D FCN :

$IN(1\ or\ 2\ ch) \rightarrow C(3, 5) \rightarrow C(5, 10) \rightarrow C(5, 20) \rightarrow C(3, 50) \rightarrow C(1, 1) \rightarrow Out(1)$

5.4.2 3D Unet :

ConvBlock : $C_B(3, output\ size) \equiv C(3, output\ size) \rightarrow C(3, output\ size) \rightarrow \downarrow 2$

UpConvBlock : $U_B(3, output\ size) \equiv U(3, output\ size) \rightarrow \oplus \rightarrow C(3, output\ size)$

Encoder : $IN(1\ or\ 2\ ch) \rightarrow C_B(3, 32) \rightarrow C_B(3, 64) \rightarrow C_B(3, 128) \rightarrow C_B(5, 256) \rightarrow C_B(5, 512) \rightarrow B(512)$

Decoder : $B(512) \rightarrow U_B(3, 512) \rightarrow U_B(3, 256) \rightarrow U_B(3, 128) \rightarrow U_B(3, 64) \rightarrow U_B(3, 32) \rightarrow Out(1)$

Table 1. Total number of parameters for each of the architectures used in our experiment.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Network</th>
<th>Number of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drive</td>
<td>FCN</td>
<td>15.52K</td>
</tr>
<tr>
<td></td>
<td>U-Net</td>
<td>28.94M</td>
</tr>
<tr>
<td>Road</td>
<td>FCN</td>
<td>279.67K</td>
</tr>
<tr>
<td>CREMI</td>
<td>U-Net</td>
<td>31.03M</td>
</tr>
<tr>
<td>3D</td>
<td>FCN 2ch</td>
<td>58.66K</td>
</tr>
<tr>
<td></td>
<td>U-Net 2ch</td>
<td>19.21M</td>
</tr>
</tbody>
</table>
6. Soft Skeletonization Algorithm

Figure 5. Scheme of our proposed differentiable skeletonization. On the top left the mask input is fed. Next, the input is reatedly eroded and dilated. The resulting erosions and dilations are compared to the image before dilation. The difference between these images is part of the skeleton and will be added iteratively to obtain a full skeletonization. The ReLu operation eliminates pixels that were generated by the dilation but are not part of the original or eroded image.

7. Code for the clDice similarity measure and the soft-clDice loss (PyTorch):

7.1. clDice measure

from skimage.morphology import skeletonize
import numpy as np

def cl_score(v, s):
    return np.sum(v*s)/np.sum(s)

def clDice(v, v_1):
    tprec = cl_score(v, skeletonize(v_1))
    tsens = cl_score(v_1, skeletonize(v))
    return 2*tprec*tsens/(tprec+tsens)

7.2. soft-skeletonization in 2D

import torch.nn.functional as F

def soft_erode(img):
    p1 = -F.max_pool2d(-img, (3, 1), (1, 1), (0, 0))
    p2 = -F.max_pool2d(-img, (1, 3), (1, 1), (0, 0))
    return torch.min(p1, p2)

def soft_dilate(img):
    return F.max_pool2d(img, (3, 3), (1, 1), (1, 1))

def soft_open(img):
    return soft_dilate(soft_erode(img))

def soft_skel(img, iter):
    img1 = soft_open(img)
    skel = F.relu(img-img1)
    for j in range(iter):
        img = soft_erode(img)
        img1 = soft_open(img)
        delta = F.relu(img-img1)
        skel = skel + F.relu(delta-skel*delta)
    return skel

7.3. soft-skeletonization in 3D

import torch.nn.functional as F

def soft_erode(img):
    p1 = -F.max_pool3d(-img, (3, 1, 1), (1, 1, 1), (1, 0, 0))
    p2 = -F.max_pool3d(-img, (1, 3, 1), (1, 1, 1), (0, 1, 0))
    p3 = -F.max_pool3d(-img, (1, 1, 3), (1, 1, 1), (0, 0, 1))
    return torch.min(torch.min(pl, p2), p3)

def soft_dilate(img):
    return F.max_pool3d(img, (3, 3, 3), (1, 1, 1), (1, 1, 1))

def soft_open(img):
    return soft_dilate(soft_erode(img))

def soft_skel(img, iter):
    img1 = soft_open(img)
    skel = F.relu(img-img1)
    for j in range(iter):
        img = soft_erode(img)
        img1 = soft_open(img)
        delta = F.relu(img-img1)
        skel = skel + F.relu(delta-skel*delta)
    return skel

8. Evaluation Metrics

As discussed in the text, we compare the performance of various experimental setups using three types of metrics: volumetric, graph-based and topology-based.

8.1. Overlap-based:

Dice coefficient, Accuracy and clDice, we calculate these scores on the whole 2D/3D volumes. clDice is calculated using a morphological skeleton (skeletonize3D from the skimage library).

8.2. Graph-based:

We extract graphs from random patches of 64×64 pixels in 2D and 48×48×48 in 3D images. For the StreetmoveDistance (SMD) [1] we uniformly sample a fixed number of points from the graph of the prediction and label, match them and calculate the Wasserstein-distance between these graphs. For the junction-based metric (Opt-J) we compute the F1 score of junction-based metrics, recently proposed by [2]. According to their paper this metric is advantageous over all previous junction-based metrics as it can account for nodes with an arbitrary number
of incident edges, making this metric more sensitive to endpoints and missed connections in predicted networks. For more information please refer to their paper.

8.3. Topology-based:

For topology-based scores we calculate the Betti Errors for the Betti Numbers $\beta_0$ and $\beta_1$. Also, we calculate the Euler characteristic, $\chi = V - E + F$, where $E$ is the number of edges, $F$ is the number of faces and $V$ is the number of vertices. We report the relative Euler characteristic error ($\chi$ ratio), as the ratio of the $\chi$ of the predicted mask and that of the ground truth. Note that a $\chi$ ratio closer to one is preferred. All three topology-based scores are calculated on random patches of $64 \times 64$ pixels in 2D and $48 \times 48 \times 48$ in 3D images.

9. Additional Quantitative Results

Table 2. Quantitative experimental results for the 3D synthetic vessel dataset. Bold numbers indicate the best performance. We trained baseline models of binary-cross-entropy (BCE), softDice and mean-squared-error loss (MSE) and combined them with our soft-clDice and varied the $\alpha > 0$. For all experiments we observe that using soft-clDice in $L_c$ results in improved scores compared to soft-Dice. This improvement holds for almost $\alpha > 0$. We observe that soft-clDice can be efficiently combined with all three frequently used loss functions.

<table>
<thead>
<tr>
<th>Loss</th>
<th>Dice</th>
<th>clDice</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCE</td>
<td>99.81</td>
<td>98.24</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.5$</td>
<td>99.74</td>
<td>97.07</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.4$</td>
<td>99.77</td>
<td>98.04</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.3$</td>
<td>99.76</td>
<td>98.20</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.2$</td>
<td>99.78</td>
<td>98.29</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.1$</td>
<td>99.82</td>
<td>98.39</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.01$</td>
<td>99.83</td>
<td>98.46</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.001$</td>
<td>99.85</td>
<td>98.42</td>
</tr>
<tr>
<td>soft-Dice</td>
<td>99.74</td>
<td>97.35</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.5$</td>
<td>99.74</td>
<td>97.07</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.4$</td>
<td>99.77</td>
<td>98.04</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.3$</td>
<td>99.80</td>
<td>98.13</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.2$</td>
<td>99.74</td>
<td>98.08</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.1$</td>
<td>99.74</td>
<td>98.08</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.01$</td>
<td>99.74</td>
<td>97.07</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.001$</td>
<td>99.74</td>
<td>97.12</td>
</tr>
<tr>
<td>MSE</td>
<td>99.71</td>
<td>97.03</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.5$</td>
<td>99.62</td>
<td>98.22</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.4$</td>
<td>99.65</td>
<td>97.04</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.3$</td>
<td>99.67</td>
<td>98.16</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.2$</td>
<td>99.70</td>
<td>97.10</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.1$</td>
<td>99.74</td>
<td>98.21</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.01$</td>
<td>99.82</td>
<td>98.32</td>
</tr>
<tr>
<td>$L_c, \alpha = 0.001$</td>
<td>99.84</td>
<td>98.37</td>
</tr>
</tbody>
</table>

References