A skeletonization algorithm for gradient-based optimization

Martin J. Menten[1,2]   Johannes C. Paetzold[1,2]   Veronika A. Zimmer[1]   Suprosanna Shit[1]
Ivan Ezhov[1]   Robbie Holland[2]   Monika Probst[1]   Julia A. Schnabel[1]   Daniel Rueckert[1,2]
[1]Technical University of Munich   [2]Imperial College London

Abstract

The skeleton of a digital image is a compact representation of its topology, geometry, and scale. It has utility in many computer vision applications, such as image description, segmentation, and registration. However, skeletonization has only seen limited use in contemporary deep learning solutions. Most existing skeletonization algorithms are not differentiable, making it impossible to integrate them with gradient-based optimization. Compatible algorithms based on morphological operations and neural networks have been proposed, but their results often deviate from the geometry and topology of the true medial axis. This work introduces the first three-dimensional skeletonization algorithm that is both compatible with gradient-based optimization and preserves an object's topology. Our method is exclusively based on matrix additions and multiplications, convolutional operations, basic non-linear functions, and sampling from a uniform probability distribution, allowing it to be easily implemented in any major deep learning library. In benchmarking experiments, we prove the advantages of our skeletonization algorithm compared to non-differentiable, morphological, and neural-network-based baselines. Finally, we demonstrate the utility of our algorithm by integrating it with two medical image processing applications that use gradient-based optimization: deep-learning-based blood vessel segmentation, and multimodal registration of the mandible in computed tomography and magnetic resonance images.

1. Introduction

Skeletonization algorithms aim at extracting the medial axis of an object, which is defined as the set of points that have more than one closest point on the object's boundary [7]. This lower-dimensional representation compactly encodes various geometric, topological, and scale features. As such, it is useful for many tasks in computer vision, including object description, compression, recognition, tracking, registration, and segmentation [19, 23, 26, 41, 46]. While there are several efficient approaches to calculate the medial axis in continuous space, extracting the skeleton of a discrete digital

image is not trivial. The search for accurate skeletonization algorithms, which can process two-dimensional and three-dimensional digital images, has spawned a plethora of research works [5, 6, 8, 16, 19, 32, 43, 47]. For a comprehensive overview and taxonomy of skeletonization algorithms and their applications, we refer to the excellent survey by Saha *et al.* [33].

Today, computer vision tasks are commonly solved using deep learning. Skeletonization may be used as building block or inductive bias in these image processing applications [13, 17, 39]. However, most established skeletonization methods are not compatible with backpropagation and gradient-based optimization [33]. The few works that have integrated skeletonization with deep learning pipelines rely on morphological skeletonization algorithms [39]. This class of algorithms is based on simple morphological operations, such as erosion and dilation [21, 45]. However, they will often result in breaks in the skeleton, causing it to diverge from the geometry and topology of the true medial axis. Recently, learning-based methods have also been harnessed for skeletonization [9, 13, 17, 24, 25, 27, 37, 38]. Most of these works train an encoder-decoder neural network on pairs of input images and ground truth skeletons, which have previously been obtained using a classical skeletonization algorithm. While learned approaches are intrinsically compatible with backpropagation, they are not guaranteed to preserve the topology of the input. Additionally, they are susceptible to domain shifts between the training and inference data [10, 44].

Our contribution This work bridges the gap between traditional skeletonization principles with strict topology guarantees and their integration with gradient-based optimization. We introduce a skeletonization algorithm that is topology-preserving, domain-agnostic, and compatible with backpropagation (see Figure 1). Our algorithm is exclusively based on matrix additions and multiplications, convolutional operations, basic non-linear functions, and sampling from a uniform probability distribution, allowing it to be easily implemented in any major deep learning library, such as PyTorch or Tensorflow [2, 28]. In benchmarking experiments, we establish that our algorithm outperforms non-differentiable,
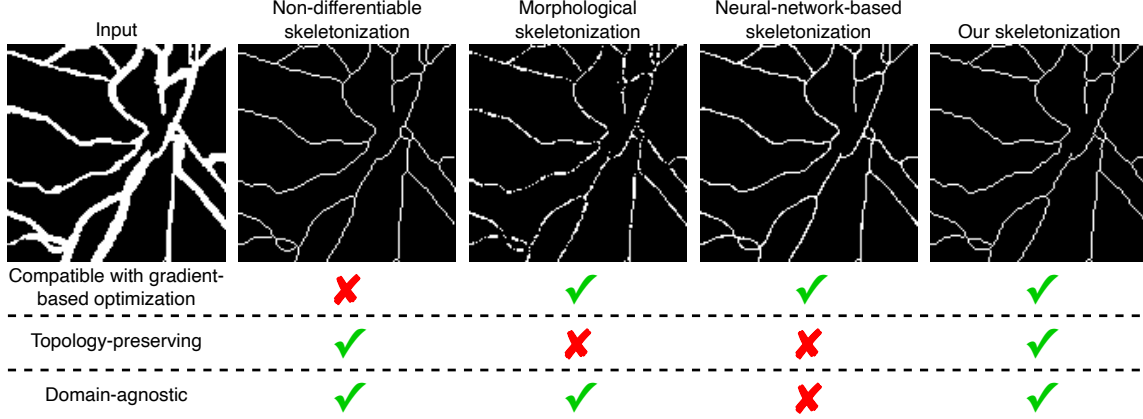
Figure 1. Most existing skeletonization algorithms are not differentiable, making it impossible to integrate them with gradient-based optimization. Morphological and neural-network-based solutions can be used with backpropagation, but alter the topology of the object by introducing breaks along the skeleton. Our proposed skeletonization algorithm preserves the topology while simultaneously being compatible with gradient-based optimization.

morphological, and neural-network-based baselines. Finally, we directly integrate our skeletonization algorithm with two medical image processing applications that rely on gradient-based optimization. We show that it enhances both deep-learning-based blood vessel segmentation, and multimodal registration of the mandible in computed tomography (CT) and magnetic resonance (MR) images.

2. Prerequisites

Digital images A discrete three-dimensional image is an array of points $P = \{p(x, y, z)\}$, which are each assigned an intensity value, on a lattice defined by Cartesian coordinates $x, y, z \in \mathbb{Z}$ [15]. Owing to the grid nature of the image, we can define the 6-neighborhood $N_6(p)$, 18-neighborhood $N_{18}(p)$, and 26-neighborhood $N_{26}(p)$ of a point [31]:

$$N_6(p) = \{p'; (|x - x'| + |y - y'| + |z - z'|) \leq 1\},$$
$$N_{26}(p) = \{p'; \max(|x - x'|, |y - y'|, |z - z'|) \leq 1\}, \quad (1)$$
$$N_{18}(p) = \{p'; (|x - x'| + |y - y'| + |z - z'|) \leq 2\} \cap N_{26}(p).$$

Points that are inside each other's n-neighborhood are called n-adjacent. Two points p and p' are said to be n-connected, if there is a sequence of points $p = p_0, ..., p_k = p'$ so that each p_i is n-adjacent to p_{i-1} for $1 \leq i \leq k$.

Euler characteristic In the special case of a binary image the value of each point is either 1 or 0. The foreground of a binary image is represented by the set S of points with value 1, while the set $\overline{S}$ denotes the remaining points with value 0. Based on above's definition of connectedness, we can define an object O as a set of n-connected points in S. An object in $\overline{S}$ that is completely surrounded by points in S is called a cavity C in S. Finally, a hole H can be intuitively described as a tunnel through S. Objects, cavities and holes can be analogously defined for $\overline{S}$. By combining the number of

objects, cavities, and holes the Euler characteristic, or genus, of a 6-connected set of points G_6 can be determined [15]:

$$G_6 = \#O - \#H + \#C. \quad (2)$$

It is also possible to span a graph between all 6-neighbors. On this graph, simplicial complexes consisting of one, two, four, or eight points are called vertex v, edge e, face f, or octant oct, respectively [15]. G_6 can also be calculated based on the number of these complexes via

$$G_6 = \#v - \#e + \#f - \#oct. \quad (3)$$

In the following, we only consider the case in which objects in S are 26-connected and objects in $\overline{S}$ are 6-connected. To calculate the genus of a 26-connected S $G_{26}(S)$ we can derive $G_6(\overline{S})$ using Equation 2 or 3 and use the following relation [15]:

$$G_{26}(S) = G_6(\overline{S}) - 1 \quad (4)$$

Simple points Crucial to the skeletonization of digital images is the definition of a simple point. A point belonging to S is simple if it can be deleted, that is changed from 1 to 0, without altering the image's topology. Morgenthaler [22] shows that this is the case if the deletion of a point does not change the number of objects and holes in S and $\overline{S}$. Using δ to denote the difference in topology between S and $S \setminus \{p\}$, we can write this relation as:

$$p \text{ is simple} \iff$$
$$\delta O(S) = 0, \delta O(\overline{S}) = 0, \delta H(S) = 0, \delta H(\overline{S}) = 0. \quad (5)$$

Lee *et al.* [16] prove that these conditions are equivalent to calculating the change in the number of objects and Euler characteristic of S in a point's local 26-neighborhood:

$$p \text{ is simple} \iff \delta O(S) = 0, \delta G_{26}(S) = 0. \quad (6)$$

3. Method

Arguably, the most common class of skeletonization algorithms for digital images are iterative boundary-peeling methods [33]. These algorithms are based on the repeated removal of simple points until only the skeleton remains. [4, 16, 18, 22]. At its core, our skeletonization algorithm follows the same paradigm (see Figure 2). To ensure that our method is compatible with gradient-based optimization while remaining topology-preserving, we make the following contributions:

- We introduce two methods to differentiably identify simple points (see Section 3.1). One solution relies on the calculation of the Euler characteristic, and the other one is based on a set of Boolean rules that evaluate a point's 26-neighborhood.

- We adopt a scheme to safely delete multiple simple points at once, enabling the parallelization of our algorithm (see Section 3.2).

- We introduce a strategy to apply our algorithm to non-binary inputs and integrate it with gradient-based optimization by employing the reparametrization trick and a straight-through estimator (see Section 3.3).

- All of above's contributions are formulated using matrix additions and multiplications, convolutional operations, basic non-linear functions, and sampling from a uniform probability distribution. We combine them into a single PyTorch module, which we make publicly available (see Section 3.4).
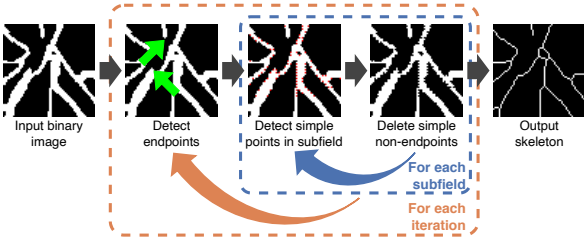

Figure 2. Data flow through an iterative boundary-peeling skeletonization algorithm. Our method follows the same paradigm, while ensuring that the identification of simple points, endpoints, and the subfield-based parallelization are all compatible with gradient-based optimization.

3.1. Identification of simple points

3.1.1 Euler characteristic to identify simple points

Lobregt *et al.* [18] base their detection of simple points on the observation that the removal of a simple point does not alter the genus of its 26-neighborhood:

$$p \text{ is simple} \implies \delta G_{26}(S) = 0, \tag{7}$$

which is a relaxation of Equation 6. To efficiently determine $\delta G_{26}(S)$, their algorithm assesses the change of the genus in each of the 26-neighborhood's eight octants and sums their contributions. Thereby, they rely on a look-up table in which each entry corresponds to one of the 2^8 possible configurations of an octant.

In order to reduce the number of comparisons and provide a smoother learning signal for backpropagation, we use eight convolutions with pre-defined kernels to determine the number of vertices, edges, faces, and octants (see Figure 3). By inserting these into Equation 3, we calculate $G_6(\overline{S})$ of each 26-neighborhood. Afterwards, we repeat this process with the central point of each neighborhood set to zero and assess whether the Euler characteristic has changed. This process is parallelized while ensuring that only one point in each 26-neighborhood is deleted at a given time. To this end, we use multiple sets of points given by

$$S_{i,j,k} \in \{p'(x+i, y+j, z+k)\};$$
$$x, y, z \in \{0, 2, 4, \ldots\}, \ i, j, k \in \{0, 1\} \tag{8}$$

Cycling through all combinations of i, j, and k yields eight subfields of points that can be processed simultaneously. The same subfields are also used during the later removal of simple points (see Section 3.2).

Lee *et al.* [16] show that the invariance of the Euler characteristic under deletion of a point is a necessary but not sufficient condition for it being simple (cf. Equation 6). Consequently, above's strategy slightly overestimates the number of simple points [16]. On the set of all possible 2^{26} configurations of a 26-neighborhood, above's algorithm characterizes the central point as simple in 40.07% of cases when in fact only 38.72% are truly simple.

3.1.2 Boolean characterization of simple points

For this reason, we propose a second method that identifies the exact set of simple points. It is based on work by Bertrand *et al.* [4] who introduce the following Boolean characterization of a simple point:

$$p \text{ is simple} \iff (\#\overline{X_6} = 1) \text{ or } (\#X_{26} = 1)$$
$$\text{or } (\#B_{26} = 0, \#X_{18} = 1)$$
$$\text{or } (\#\overline{A_6} = 0, \#B_{26} = 0, \#B_{18} = 0,$$
$$\#\overline{X_6} - \#\overline{A_{18}} + \#\overline{A_{26}} = 1) \tag{9}$$

where $\#X_n$ and $\#\overline{X_n}$ are the number of n-neighbors of a point belonging to S and $\overline{S}$, respectively. $\#B_{26}$, $\#\overline{A_6}$, $\#B_{18}$, $\#\overline{A_{18}}$, and $\#\overline{A_{26}}$ correspond to the number of specific cell configurations depicted in Figure 4.

Similar as before, the presence of these five configurations and their 6-, 8-, and 12-rotational equivalents can be efficiently checked by convolving the image with pre-defined
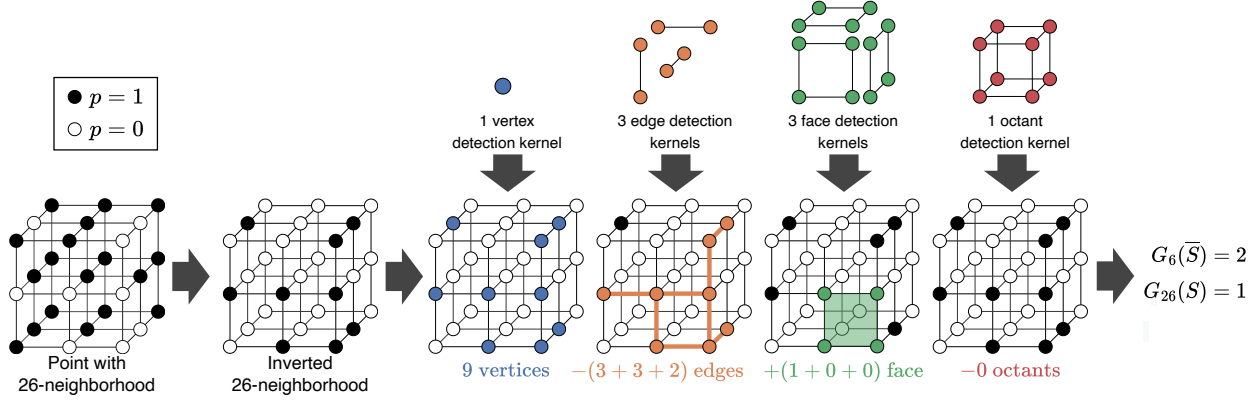
Figure 3. In order to calculate the Euler characteristic of a point, we initially invert its 26-neighborhood. Next, we determine the number of vertices, edges, faces, and octants of the background via eight simple convolutions with predefined kernels. Finally, Equations 3 and 4 are used to derive the Euler characteristic of the foreground.

kernels. Compared to our first strategy, this algorithm trades off computational efficiency for accuracy. It requires a total of 57 convolutions with three-dimensional kernels, but is guaranteed to correctly classify all points of a binary image as either simple or not.
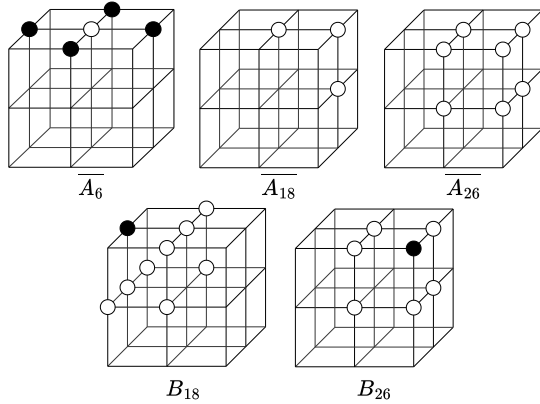

Figure 4. The five cell configuration introduced by Bertrand *et al.* [4] used for Boolean characterization of simple points.

3.2. Parallelization and endpoint conditions

Sequentially checking each point using above's conditions and deleting them if they are simple already constitutes a functioning skeletonization algorithm. However, this strategy is very inefficient when applied to large three-dimensional images. Naively deleting all simple points at once is not possible as simultaneous removal of neighboring points may affect the object's topology even if both points are simple. For this reason, previous works have researched strategies to safely remove multiple simple points in parallel [5, 16, 19, 22, 32, 43]. We adopt a subiterative scheme based on the same eight subfields that are already used during the calculation of the Euler characteristic (see Equation 8) [5].

In conventional skeletonization algorithms, the program terminates once a full iteration does not result in any points being deleted. To keep the number of operations comparable during repeated application, we explicitly provide the number of outer-loop iterations. This simple scalar hyperparameter can be easily tuned on a few representative samples of the considered dataset.

Merely preserving non-simple points would lead to a topological skeleton. For example, a solid object without any holes or cavities would be reduced to a single point. For many applications in image processing, it is desirable to also preserve some information about the image's geometry, such as the existence and position of extremities. This can be achieved by also preserving so-called endpoints. Our algorithm uses the following definition of an endpoint:

$$p \text{ is endpoint} \iff \#X_{26} \leq 1 \qquad (10)$$

Other definitions for endpoints could potentially be integrated with our algorithm and would result in different properties of the obtained skeleton. For example, endpoint conditions could be chosen to extract a medial surface instead of a medial axis, or the number of short extremities, sometimes called spurs, may be reduced [35, 47].

3.3. Processing of continuous inputs

The previously introduced definitions for simple points and endpoints are only valid for binary images. However, in many applications the input is often a matrix of continuous values, such as probability maps output by a learning algorithm. Simply rounding these inputs inhibits learning via backpropagation. We circumvent this issue by treating each point as a discrete, stochastic node modeled by a Bernoulli distribution, and use the reparametrization trick and a straight-through estimator to facilitate sampling and gradient estimation from it [3, 12, 20].
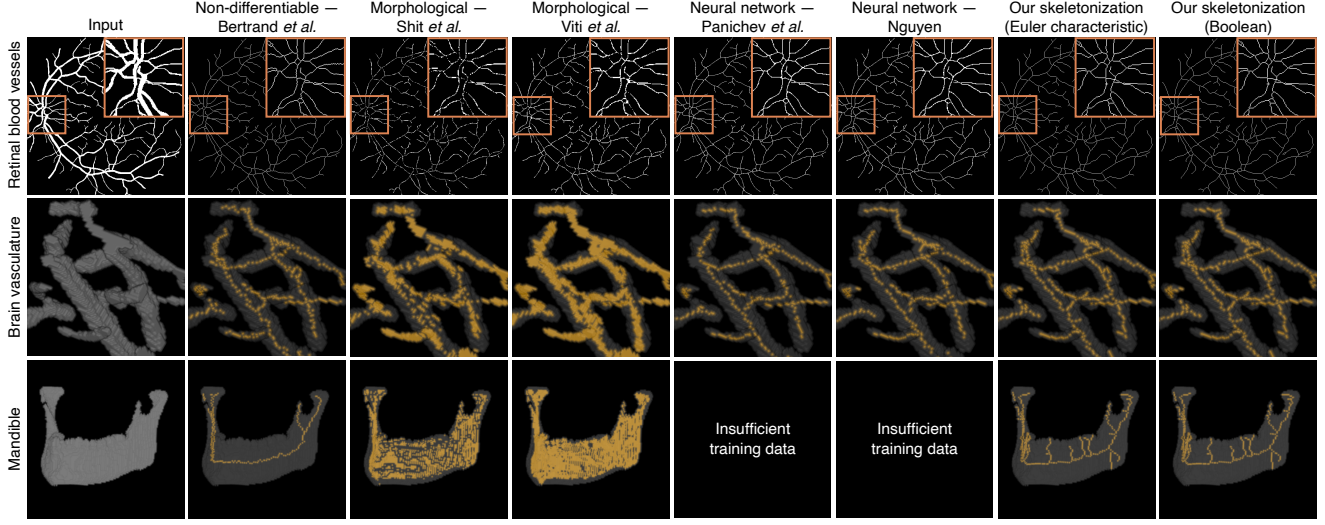
Figure 5. The results of applying the seven tested skeletonization algorithm to representative samples of three diverse datasets. Of the six algorithms that are compatible with gradient-based optimization, only our two methods are able to extract a thin, topology-preserving skeleton, similar to the one obtained using the non-differential baseline. Additional samples are shown in the Supplementary Material.

The reparametrization trick splits each node into a differentiable function, the raw grayscale input, and a fixed noise distribution [11, 12, 20]. We can sample from each node via

$$X = \sigma \left(\frac{(\log \alpha + \beta L)}{\tau} \right) , \ \alpha = \frac{\alpha_1}{(1 - \alpha_1)} , \quad (11)$$

where $\alpha_1 \in (0, 1)$ is the probability of the input being 1, σ denotes the sigmoid function, L is a sample from a Logistic distribution that is scaled by factor $\beta \in [0, \infty)$, and $\tau \in (0, \infty)$ is the Boltzmann temperature. Both β and τ control the entropy of the distribution. Others have proposed gradually annealing these parameters as learning progresses or even updating them via backpropagation [11, 12, 20]. In this work, we treat them as simple tunable hyperparameter.

Afterwards, we discretize the obtained samples using a straight-through gradient estimator [3, 30]. It returns the rounded binary value during the forward pass. Instead of using the zero gradient of the rounding operation during the backward pass, the modified chain rule is applied and the identity function is used as proxy gradient.

3.4. Implementation in PyTorch

Our proposed algorithm consists exclusively of matrix additions, multiplications, convolutional operations, basic activation functions, and sampling from a uniform probability distribution. As such, it can easily be implemented in any major deep learning library and runs efficiently on graphics processing units. Our skeletonization module, which we make publicly available[1], is implemented in PyTorch

---
[1] https://github.com/martinmenten/skeletonization-for-gradient-based-optimization

[28] and is fully integrated with its automatic differentiation engine.

4. Experiments and results

Initially, we benchmark the performance of our skeletonization algorithm with regard to spatial and topological correctness, run time, and the ability to combine it with back-propagation (see Section 4.1). Afterwards, we showcase the utility of our method by integrating it with two medical image processing pipelines: deep-learning-based blood vessel segmentation and multimodal registration of the mandible (see Section 4.2).

For the experiments, we use three different datasets:

- the DRIVE dataset consisting of 40 two-dimensional retinal color fundus photographs and matching annotations of the visible blood vessels [40],

- the VesSAP dataset comprising 24 three-dimensional light-sheet microscopy images of murine brains after tissue clearing, staining, and labeling of the vascular network, which we split into 2,400 patches, [42].

- an in-house dataset of 34 matched three-dimensional CT and MR images and manually extracted segmentation masks of the mandible.

Additional information about each dataset and our experimental setup can be found in the Supplementary Material.

Table 1. Quantitative comparison of the topological accuracy and run time of seven skeletonization algorithms on three datasets.

Dataset	Skeletonization algorithm	# points	β_0 error	β_1 error	β_2 error	Run time [ms]
DRIVE	Non-differentiable – Bertrand *et al.* [6]	8316±618	0±0	0±0	-	-
	Morphological – Shit *et al.* [39]	9926±667	1156±197	50±23	-	19±1
	Morphological – Viti *et al.* [45]	11834±976	266±62	45±19	-	23±3
	Neural network – Panichev *et al.* [27]	10420±915	6±3	13±11	-	14±1
	Neural network – Nguyen [25]	10619±806	10±5	18±8	-	117±1
	Ours – Euler characteristic	8393±611	0±0	0±0	-	101±3
	Ours – Boolean	8393±611	0±0	0±0	-	540±2
VesSAP	Non-differentiable – Bertrand *et al.* [6]	471±212	0±0	0±0	0±0	-
	Morphological – Shit *et al.* [39]	1914±809	173±79	3±5	0±1	20±1
	Morphological – Viti *et al.* [45]	3783±1797	2±2	23±18	0±1	21±2
	Neural network – Panichev *et al.* [27]	423±182	64±38	3±5	0±1	16±1
	Neural network – Nguyen [25]	422±189	33±19	4±6	0±1	115±1
	Ours (Euler characteristic)	540±245	0±1	0±1	0±1	100±2
	Ours (Boolean)	540±243	0±0	0±0	0±0	520±26
Mandible	Non-differentiable – Bertrand *et al.* [6]	236±37	0±0	0±0	0±0	-
	Morphological – Shit *et al.* [39]	2698±387	131±24	13±10	0±0	37±1
	Morphological – Viti *et al.* [45]	4866±758	1±1	81±26	0±0	42±1
	Neural network – Panichev *et al.* [27]	Insufficient training data				
	Neural network – Nguyen [25]	Insufficient training data				
	Ours (Euler characteristic)	409±67	1±1	1±1	0±0	160±2
	Ours (Boolean)	405±68	0±0	0±0	0±0	1081±10

4.1. Benchmarking experiments

4.1.1 Spatial and topological accuracy

We compare our two skeletonization algorithms with five baselines:

- a well-established non-differentiable skeletonization algorithm by Bertrand *et al.* [6], which has been implemented in the open-source DGtal library [1],

- two morphological skeletonization algorithm based on repeated opening and erosion proposed by Shit *et al.* [39] and Viti *et al.* [45], respectively,

- two neural-network-based methods by Panichev *et al.* [27] and Nguyen [25], respectively, that each train a encoder-decoder network to output the skeleton of a binary input image.

On all three datasets, both of our proposed algorithms produce continuous, thin skeletons that agree well with the non-differentiable baseline (see Figure 5). When applying the morphological skeletonization algorithms, we observe that continuous blood vessels are split into many small objects along the medial axis. Similarly, the mandible is broken into small components that are positioned at the medial surface of the input structure. The neural-network-based algorithms also cannot preserve the topology of the vascular network when applied to data from the DRIVE and VesSAP datasets, and completely fail to converge during training on the small mandible dataset.

These observations are corroborated by quantitative measurements (see Table 1). We assess the topological correctness of all skeletons by evaluating the error of the first three Betti numbers, β_0, β_1, and β_2. These measure the absolute difference of the number of objects, holes, and cavities, respectively, between the input structure and obtained skeleton. Our skeletonization algorithm based on the Boolean characterization of simple points preserves the exact topology of the base object in all cases as does the non-differentiable baseline. The skeletons obtained by the morphological and neural-network-based algorithms both contain topological errors in all three Betti numbers. Furthermore, their produced skeletons are often thicker than one voxel. This is reflected by the substantially larger number of points included in the results.

4.1.2 Run time analysis

Table 1 also lists the average run time of each skeletonization algorithm when processing images of varying sizes and dimensions. We report the duration of a single forward and backward pass through each skeletonization module as required during gradient-based optimization. All measurements were conducted using a workstation equipped with a Nvidia RTX A6000 GPU (Nvidia Corporation, Santa Clara, California, United States), 128 GB of random access memory, and a AMD Ryzen Threadripper 3970X 32-core central processing unit (Advanced Micro Devices, Inc., Santa Clara, California, United States). We find that our algorithms are slower than both morphological and neural-network-based methods. Still, all algorithms run in a second or less and are fast enough to be effectively employed for the applications described in the following (see Section 4.2).

4.1.3 Compatibility with gradient-based optimization

In order to make our skeletonization algorithms compatible with gradient-based optimization, logistic noise is added to the input during the reparametrization trick (see Section 3.3). We demonstrate the efficiency of this approach and the importance of well-tuned noise parameters, β and τ, in a simple experiment (see Figure 6). Hereby, an input tensor is initialized with random values and passed through our skeletonization module. Using backpropagation, the tensor's values are learned so that its ultimate output resembles that of the ground truth skeleton. While several degenerate solutions that all yield the same skeleton, exist, we expect the learned tensor to ultimately resemble the skeleton itself.
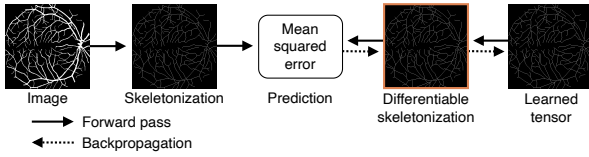

Figure 6. Experiment to test the compatibility of our skeletonization algorithm (orange box) with backpropagation.

With very low levels of entropy, we observe that learning with our skeletonization module is very slow (see Figure 7). Increasing the entropy results in single passes through the skeletonization to be less faithful to the geometry and topology of the true medial axis (see Figure 8). However, averaging over repeated samples mostly recovers the true skeleton and enables learning of the correct structure. At too high entropy, convergence slows down as the obtained skeleton is not sufficiently accurate anymore.
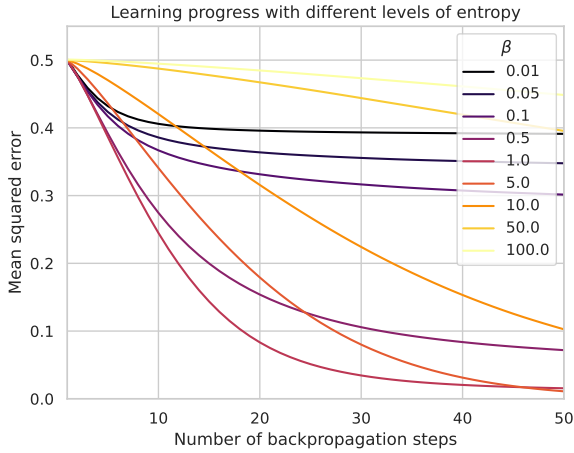

Figure 7. Effect of the scale β of the added logistic noise on the ability to propagate a gradient through our skeletonization module. Both very low entropy and very high entropy inhibit learning. Similar results can be found when varying τ (see Supplementary Material).
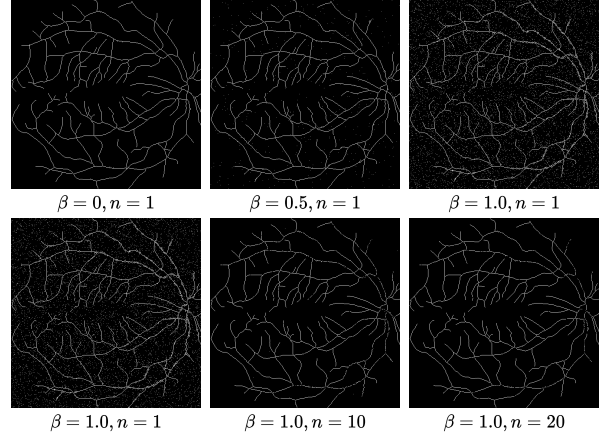

Figure 8. Effect of scaling the added logistic noise (β) in our skeletonization algorithm. Repeated sampling (n) mostly recovers the true skeleton.

4.2. Application experiments

4.2.1 Topology-aware blood vessel segmentation

We explore the utility of our skeletonization methods by integrating them with a deep-learning-based segmentation algorithm for the VesSAP dataset (see Figure 9). The training of the basic U-Net incorporates the centerline Dice (clDice) loss function that encourages topology preservation across different vessel scales by comparing skeletons of the prediction and ground truth [29, 39]. The loss formulation requires a skeletonization function that is compatible with backpropagation. In all cases, we tune the weighting factor λ, which balances the clDice loss with the standard Dice loss.
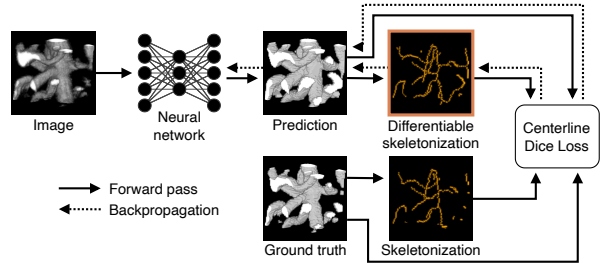

Figure 9. Deep learning pipeline for training a vessel segmentation network using the centerline Dice loss [39]. The loss formulation requires the use of a skeletonization algorithm that is compatible with backpropagation (orange box).

Using the clDice loss instead of a vanilla Dice loss slightly improves the topological agreement between prediction and ground truth as indicated by a lower error of the first three Betti numbers (see Table 2). Moreover, we find that using our skeletonization methods yield slightly better results than using a morphological skeletonization algorithm. Spatial accuracy, quantified by the Dice similarity coefficient (DSC),

was nearly identical in all cases. We obtain similar findings when conducting the same experiment using the DRIVE dataset (see Supplementary Material).

Table 2. Performance of the vessel segmentation network using either a standard Dice loss ('Without') or clDice loss with different skeletonization algorithms.

Skeletonization	DSC	β_0 error	β_1 error	β_2 error
Without	0.85±0.01	5.1±0.8	3.1±0.1	0.8±0.3
Morphological	0.85±0.01	4.3±0.5	2.8±0.2	0.7±0.1
Ours (Euler)	0.86±0.01	3.5±0.2	2.7±0.1	0.4±0.1
Ours (Boolean)	0.86±0.01	3.7±0.3	2.8±0.1	0.5±0.2

4.2.2 Multimodal registration of the mandible in CT and MR images

Finally, we explore whether incorporating skeletonization can improve multimodal registration of the mandible (see Figure 10). This application is motivated by the fact that bones often appear larger in MR images than in CT images. When registering segmentation masks from both modalities, the smaller mask can be orientated flexibly inside the larger one. We propose extracting the skeleton of both structures and calculating their overlap as image distance function instead. We employ a conventional registration algorithm that optimizes the image distance with respect to the rigid transformation between both images using a gradient-based optimization method, thus requiring a compatible skeletonization method. We implement this application in AirLab [34], which uses Pytorch's autograd functionality to compute the gradient based on the objective function. This allows a seamless integration of our skeletonization module in a conventional registration algorithm.
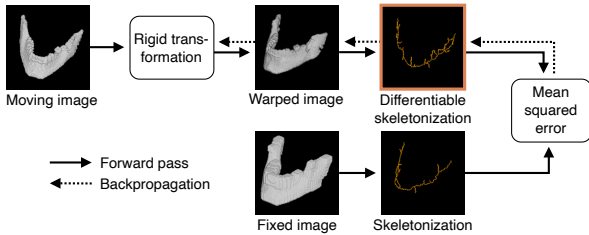

Figure 10. Workflow for multimodal registration of the mandible. To compensate for the different size of the mandible in CT and MR images, the skeleton of both images are calculated (orange box) and registered instead.

We report the DSC, the Hausdorff distance (HD) and the average surface distance (ASD) between the fixed and warped segmentation as proxy measure for registration accuracy (see Table 3). Our findings show registering the images based on the skeleton of their segmentation map slightly improves the alignment of both structures.

Table 3. Registration accuracy using two different loss functions: either the image distance is calculated using the full binary mask of the mandible ('Without'), or based on their skeletons obtained via one of three skeletonization algorithms.

Skeletonization	DSC	HD [mm]	ASD [mm]
Without	0.38±0.01	29.9±0.9	6.6±0.2
Morphological	0.32±0.01	29.2±1.1	6.7±0.2
Ours (Euler)	0.37±0.02	28.6±1.2	6.6±0.2
Ours (Boolean)	0.37±0.01	28.0±1.1	6.5±0.2

5. Discussion and conclusion

This work bridges the gap between classical skeletonization algorithms and the ability to integrate these with gradient-based optimization. We have introduced a three-dimensional skeletonization algorithm that is compatible with backpropagation, domain-agnostic and preserves an object's topology. Our method combines a characterizations of simple points, a parallelization scheme for their efficient removal, and a strategy for discretization of non-binary inputs that are all compatible with gradient-based optimization. In benchmarking experiments, we have proved the superior spatial and topological accuracy of our method compared to morphological, and neural-network-based baselines.

Our algorithm consists exclusively of matrix additions, multiplications, convolutional operations, activation functions and sampling from basic probability distributions. Consequently, it can be implemented in any deep learning library and can be seamlessly integrated with diverse image processing pipelines that use gradient-based optimization. We showcase this utility by applying it in two realistic medical image processing applications: semantic segmentation of blood vessels with deep learning, and automated multimodal image registration. In both cases, we find that our skeletonization algorithms allows the incorporation of topological and geometric information within the respective optimization objective, leading to modest performance gains.

To our knowledge, this work introduces the first topology-preserving skeletonization algorithm for gradient-based optimization. Still, we discern that there may be other, potentially more effective, approaches to create such algorithms. We hope that this work can serve as a blueprint for others to further explore skeletonization. Building upon a rich body of literature on classical skeletonization algorithms, future work could further explore alternative strategies to identify simple points [33, 36]. Similarly, past works have extensively studied schemes to efficiently remove simple points in parallel of which some may be better suited for processing on graphics processing units. Finally, the endpoint condition used during skeletonization influences the properties of the created skeleton [36, 47]. In other applications skeletal surfaces may be preferable over a medial axis or a different trade-off between a geometric and topological skeleton may be chosen. Ultimately, we envision that our method may

also be beneficial in many computer vision applications that have historically utilized skeletonization, but have since been increasingly solved using deep learning.

Acknowledgments

References

[1] Dgtal: Digital geometry tools and algorithms library. http://dgtal.org. 6

[2] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. 1

[3] Yoshua Bengio et al. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 4, 5

[4] Gilles Bertrand. A boolean characterization of three-dimensional simple points. *Pattern recognition letters*, 17(2):115–124, 1996. 3, 4

[5] Gilles Bertrand et al. Three-dimensional thinning algorithm using subfields. In *Vision Geometry III*, volume 2356, pages 113–124. SPIE, 1995. 1, 4

[6] Gilles Bertrand et al. Powerful parallel and symmetric 3d thinning schemes based on critical kernels. *Journal of Mathematical Imaging and Vision*, 48:134–148, 2014. 1, 6, 13

[7] Harry Blum. A transformation for extracting new descriptions of shape. *Models for the perception of speech and visual form*, pages 362–380, 1967. 1

[8] Gunilla Borgefors et al. Computing skeletons in three dimensions. *Pattern recognition*, 32(7):1225–1236, 1999. 1

[9] Ilke Demir et al. Skelneton 2019: Dataset and challenge on deep learning for geometric shape understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019. 1, 11

[10] Yaroslav Ganin et al. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016. 1

[11] Iris AM Huijben et al. A review of the gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 5

[12] Eric Jang et al. Categorical reparameterization with gumbel-softmax. *ICLR*, 2017. 4, 5

[13] Koteswar Rao Jerripothula et al. Object co-skeletonization with co-segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3881–3889. IEEE, 2017. 1

[14] Diederik P Kingma et al. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 11

[15] T Yung Kong et al. Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing*, 48(3):357–393, 1989. 2

[16] Ta-Chih Lee et al. Building skeleton models via 3-D medial surface axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994. 1, 2, 3, 4

[17] Jiang-Jiang Liu et al. Dynamic feature integration for simultaneous detection of salient object, edge, and skeleton. *IEEE Transactions on Image Processing*, 29:8652–8667, 2020. 1

[18] Steven Lobregt et al. Three-dimensional skeletonization: principle and algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 2(1):75–77, 1980. 3

[19] C Min Ma et al. A fully parallel 3D thinning algorithm and its applications. *Computer vision and image understanding*, 64(3):420–433, 1996. 1, 4

[20] Chris J. Maddison et al. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on learning Representations*. International Conference on Learning Representations, 2017. 4, 5

[21] Petros Maragos et al. Morphological skeleton representation and coding of binary images. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(5):1228–1244, 1986. 1, 13

[22] David G Morgenthaler. Three-dimensional simple points: serial erosion, parallel thinning and skeletonization. *TR-1005*, 1981. 2, 3, 4

[23] Bryan S Morse et al. Multiscale medial analysis of medical images. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 112–131. Springer, 1993. 1

[24] Sabari Nathan et al. Skeletonet: Shape pixel to skeleton pixel. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 1

[25] Nam Hoang Nguyen. U-net based skeletonization and bag of tricks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2105–2109, 2021. 1, 6, 11, 13

[26] Kálmán Palágyi et al. A sequential 3D thinning algorithm and its medical applications. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 409–415. Springer, 2001. 1

[27] Oleg Panichev et al. U-net based convolutional neural network for skeleton extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019. 1, 6, 11, 13

[28] Adam Paszke et al. Automatic differentiation in pytorch. *NIPS 2017 Workshop Autodiff*, 2017. 1, 5

[29] Olaf Ronneberger et al. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 7, 11

[30] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton*. Cornell Aeronautical Laboratory, 1957. 5

[31] Azriel Rosenfeld et al. *Digital picture processing*. Academic press, 1976. 2

[32] Punam K Saha et al. A new shape preserving parallel thinning algorithm for 3D digital images. *Pattern recognition*, 30(12):1939–1955, 1997. 1, 4

[33] Punam K Saha et al. A survey on skeletonization algorithms and their applications. *Pattern recognition letters*, 76:3–12, 2016. 1, 3, 8

[34] Robin Sandkühler et al. Airlab: autograd image registration laboratory. *arXiv preprint arXiv:1806.09907*, 2018. 8

[35] Doron Shaked et al. Pruning medial axes. *Computer vision and image understanding*, 69(2):156–169, 1998. 4

[36] Wei Shen et al. Skeleton pruning as trade-off between skeleton simplicity and reconstruction error. *Science China Information Sciences*, 56:1–14, 2013. 8

[37] Wei Shen et al. Object skeleton extraction in natural images by fusing scale-associated deep side outputs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 222–230, 2016. 1

[38] Wei Shen et al. Deepskeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images. *IEEE Transactions on Image Processing*, 26(11):5298–5311, 2017. 1

[39] Suprosanna Shit et al. cldice-a novel topology-preserving loss function for tubular structure segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16560–16569, 2021. 1, 6, 7, 11, 13

[40] Joes Staal et al. Ridge-based vessel segmentation in color images of the retina. *IEEE transactions on medical imaging*, 23(4):501–509, 2004. 5, 11

[41] David Thibault et al. Terrain reconstruction from contours by skeleton construction. *GeoInformatica*, 4:349–373, 2000. 1

[42] Mihail Ivilinov Todorov et al. Machine learning analysis of whole mouse brain vasculature. *Nature methods*, 17(4):442–449, 2020. 5, 11

[43] YF Tsao et al. A parallel thinning algorithm for 3-D pictures. *Computer graphics and image processing*, 17(4):315–331, 1981. 1, 4

[44] David Vazquez et al. Virtual and real world adaptation for pedestrian detection. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):797–809, 2013. 1

[45] Mario Viti et al. Coronary artery centerline tracking with the morphological skeleton loss. In *Proc. ICIP*, pages 2741–2745, 2022. 1, 6, 13

[46] Feng Zhao et al. Preprocessing and postprocessing for skeleton-based fingerprint minutiae extraction. *Pattern Recognition*, 40(4):1270–1281, 2007. 1

[47] Yong Zhou et al. Efficient skeletonization of volumetric objects. *IEEE Transactions on visualization and computer graphics*, 5(3):196–209, 1999. 1, 4, 8

Supplementary Material

Used datasets

DRIVE The widely used DRIVE dataset consists of 40 two-dimensional retinal color fundus photographs and matching annotations of the visible blood vessels [40]. We normalize the images to an intensity range between 0 and 1 and crop them to a size of 512×512 pixel. Then, we divide the dataset into training, validation and testing splits with a ratio of 60% to 20% to 20%.

VesSAP The VesSAP dataset contains 24 three-dimensional light-sheet microscopy images of murine brains after tissue clearing, staining, and labeling of the vascular network. It has been made publicly available and has been extensively described by Todorov *et al.* [42]. We split the $500 \times 500 \times 50$ voxel large images into non-overlapping patches of size $50 \times 50 \times 50$. We remove the patches that only contain background. Finally, we split the remaining ones into a training, validation and testing partition with a ratio of 80% to 10% to 10%, while ensuring a subject-wise split.

Mandible The mandible dataset consists of 34 matched CT and MR images of the lower head and neck. In all images the mandible bone was outlined by a clinical expert. We resample all images to a resolution of $0.25 \times 0.25 \times 0.25$ cm^3 and subsequently remove all smaller cavities of the segmentation mask by alternatingly applying dilation and erosion operations. For the benchmarking experiments of the skeletonization algorithm we exclusively use the CT images (cf. Section 4.1 of the main paper). For the multimodal registration workflow that incorporates our skeletonization module we use the matched image pairs (cf. Section 4.3 of the main paper).

Neural network architecture and training

This work uses neural networks either for explicit skeletonization (cf. Section 4.1 of the main paper) or for vessel segmentation (cf. Section 4.2 of the main paper). In the first case networks are provided with a binary mask and asked to provide the ground truth skeleton while being evaluated using the Dice loss. In the second case the neural network is provided images from either the DRIVE or VesSAP dataset and trained to output a blood vessel segmentation map. Hereby, we use the topology-preserving clDice loss in combination with various skeletonization algorithms [39].

The two neural-network-based skeletonization methods are implemented according to the works and accompanying public software code by Panichev *et al.* [27] and Nguyen [25], respectively. In order to facilitate processing of volumetric images we replace all two-dimensional operations, such as convolutions, pooling and normalization network layers, with their three-dimensional equivalents.

The segmentation neural networks follow a basic U-Net architecture with four downsampling and four upsampling blocks with skip connections [29]. Each block consists of two sequences of convolutional layer (either two- or three-dimensional convolutions, kernel size: 3, same padding), instance normalization layer and leaky-ReLU non-linearity (slope: 0.01). Downsampling is achieved by using a stride of 2 in the second convolutional layer of each block. At each downsampling step, the number of feature maps is also doubled and copied to the skip connection. Upsampling is achieved via a transposed convolution with a kernel size of 2 and stride of 2. After upsampling, the respective skip connection is concatenated with the main feature map. Network weights are optimized using the ADAM optimizer with a learning rate of 10^{-4} [14]. In experiments using the DRIVE dataset, networks are trained with a batch size of 2 for 1,000 epochs. We apply random shifts ($\pm 10\%$) and rotations ($\pm 45°$) as data augmentation. For the VesSAP dataset, networks are trained with a batch size of 16 for 200 epochs. In each case, we pick the best performing network based on the validation dataset before reporting the results on the test dataset. All experiments are repeated three times using different random seeds.

Additional qualitative skeletonization results

Figure 11 presents additional representative results of applying the five skeletonization algorithms to the three datasets (cf. Section 4.1 of the main paper).

Experiments using the SkelNetOn dataset

The SkelNetOn dataset was published in the scope of the Deep Learning for Geometric Shape Understanding workshop held in conjunction with the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition [9]. It consists of binary images depicting a range of stylized objects and their corresponding skeletons. Compared to the complex topologies of biological structures, the dataset exclusively features closed two-dimensional surfaces.

We repeat the same benchmarking experiments as described in Section 4.1 of the main paper using the SkelNetOn dataset. At the time of our study the challenge's public leaderboard had been taken offline, so that we could not use the official test split to benchmark our skeletonization algorithms. Instead, we split the training dataset into a training, validation and testing partition. We observe the same characteristic behavior of all skeletonization algorithms as in the experiments with the other three datasets (see Figure 12). Both morphological baseline algorithms introduce breaks along the skeleton and in some cases omit large parts of the medial axis. The neural-network-based solutions also alter the topology of the object, whereas our skeletonization algorithms result in a topologically correct, thin skeleton. This is also reflected in the quantitative measurements reported in Table 4.
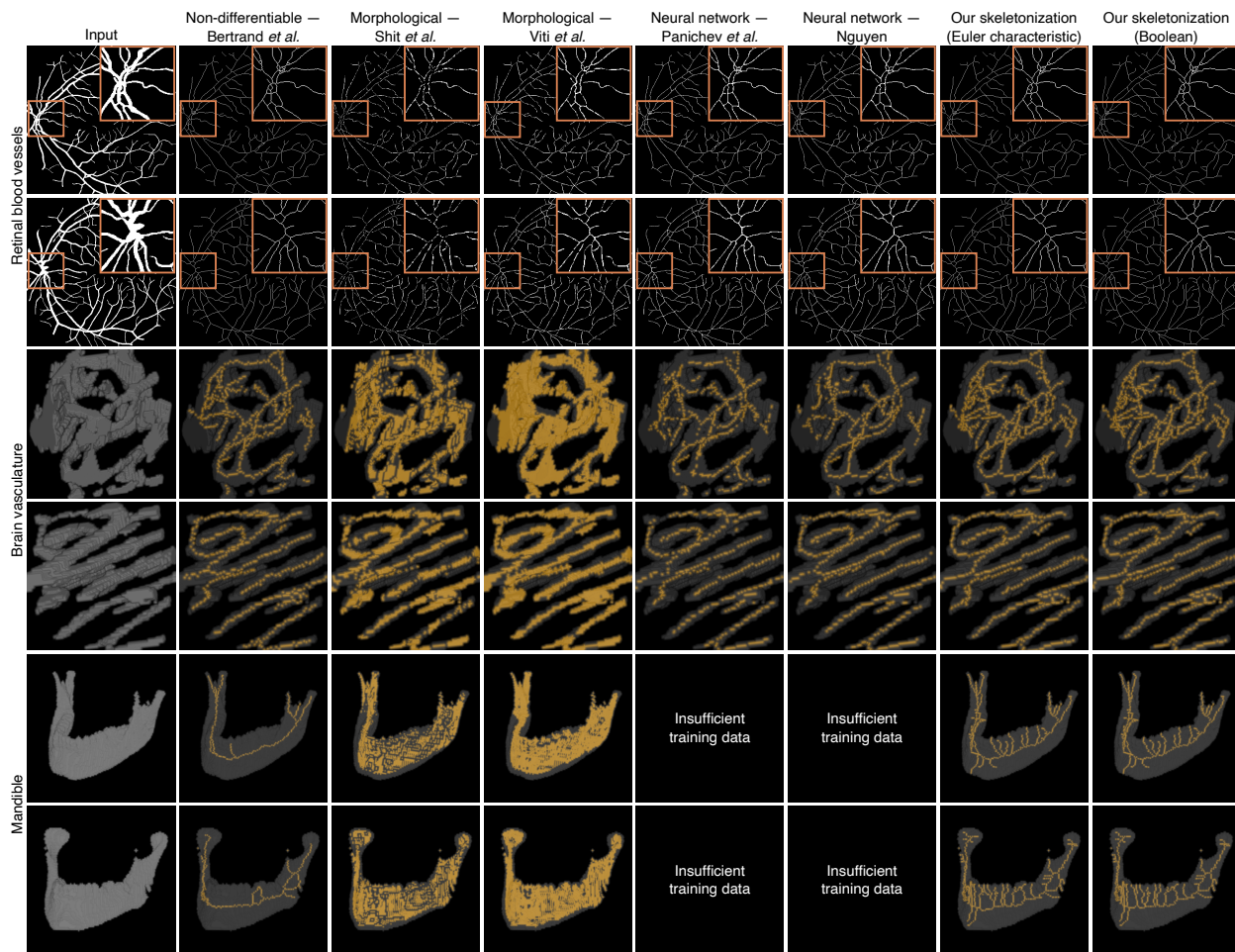
Figure 11. Additional results of applying the seven tested skeletonization algorithm to representative samples of three diverse datasets. Of the six algorithms that are compatible with gradient-based optimization, only our two methods are able to extract a thin, topology-preserving skeleton, similar to the one obtained using the non-differential baseline.
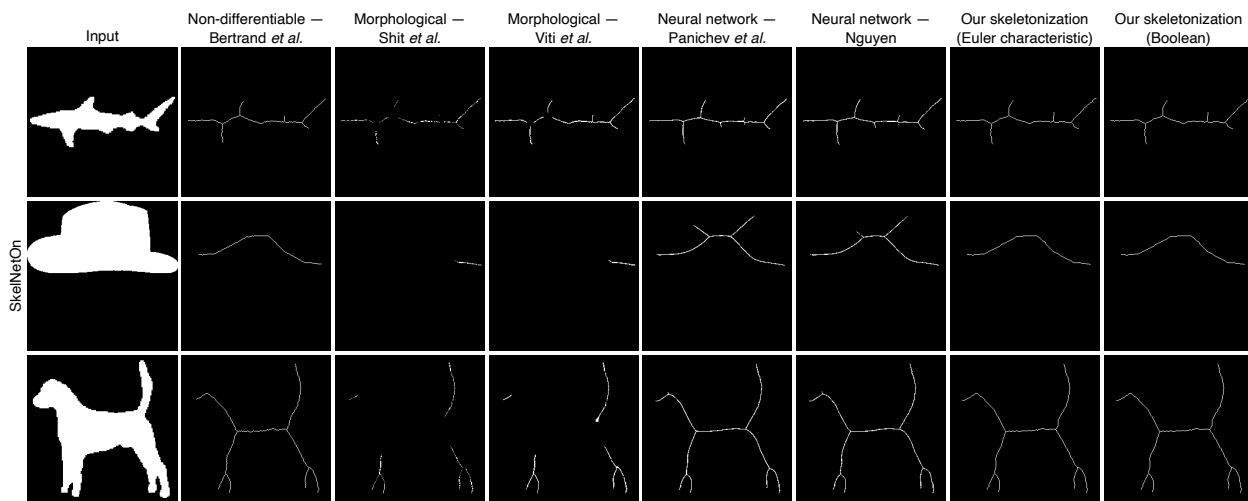

Figure 12. Qualitative results of applying the seven tested skeletonization algorithm to representative samples of SkelNetOn dataset.

Table 4. Quantitative comparison of the topological accuracy of seven skeletonization algorithms on the SkelNetOn dataset.

Dataset	Skeletonization algorithm	# points	β_0 error	β_1 error	β_2 error	Run time [ms]
	Non-differentiable – Bertrand *et al.* [6]	355±206	0±0	0±0	-	-
	Morphological – Shit *et al.* [39]	158±150	40±30	0±1	-	23±2
	Morphological – Viti *et al.* [45]	355±295	5±5	0±1	-	26±2
SkelNetOn	Neural network – Panichev *et al.* [27]	524±247	2±2	0±1	-	30±1
	Neural network – Nguyen [25]	494±236	2±3	0±1	-	160±2
	Ours – Euler characteristic	406±241	0±0	0±0	-	189±3
	Ours – Boolean	406±241	0±0	0±0	-	948±5

Effect of Boltzmann temperature on learning with the differentiable skeletonization module

The entropy of the stochastic discretization can be controlled by varying either the scale of the noise β or the Boltzmann temperature τ (cf. Equation 11 of the main paper). We have also conducted the simple experiment presented in Figure 6 of the main paper while varying τ instead of β. Hereby, an input tensor is initialized with random values and passed through our skeletonization module. Using backpropagation, the tensor's values are learned so that its ultimate output resembles that of the ground truth skeleton. Analogously to our our results with varying noise scales (cf. Figure 7 of the main paper), we find that both a too low and too high Boltzmann temperature inhibit efficient learning with our skeletonization module (see Figure 13). Empirically, we find that it suffices to tune either the noise scale or Boltzmann temperature and proceed to tune β throughout all other presented experiments.

DRIVE dataset are shown in Table 5. Similar to the results for the VesSAP dataset (see Table 2 of the main paper), we find that using the clDice loss instead of a vanilla Dice loss slightly improves the topological agreement between prediction and ground truth as indicated by a lower error of the first two Betti numbers (β_2 indicating the difference in the number of cavities is always 0 in two dimensions). Moreover, we find that using our skeletonization methods yield slightly better results than using a morphological skeletonization algorithm. Spatial accuracy, quantified by the Dice similarity coefficient (DSC), is nearly identical in all cases.

Table 5. Performance of the vessel segmentation network using either a standard Dice loss ('Without') or clDice loss with one of three skeletonization algorithms.

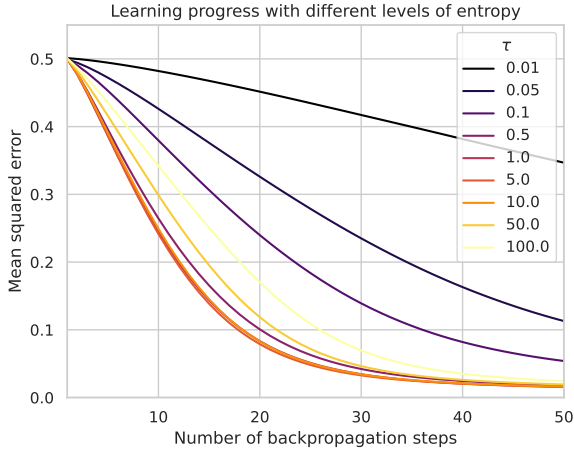| Skeletonization	DSC	β_0 error	β_1 error
Without	0.79±0.01	135.6±5.7	23.3±2.1
Morphological [21]	0.79±0.01	58.2±2.1	21.3±1.2
Ours (Euler)	0.79±0.01	58.3±18.6	18.6±0.5
Ours (Boolean)	0.79±0.01	49.6±8.5	20.6±2.1

Figure 13. Effect of the Boltzmann temperature τ on the ability to propagate a gradient through our skeletonization module. Both very low entropy and very high entropy inhibit learning.

Vessel segmentation in the DRIVE dataset

As described above and in Section 4.2 of the main paper, we integrated our skeletonization modules with a neural network that learns to segment blood vessels in either the VesSAP or DRIVE dataset. The results on the two-dimensional