Large Scale Image Segmentation with Structured Loss Based Deep Learning for Connectome Reconstruction

Jan Funke ⓘ, Fabian Tschopp ⓘ, William Grisaitis ⓘ, Arlo Sheridan ⓘ, Chandan Singh, Stephan Saalfeld ⓘ, and Srinivas C. Turaga ⓘ

Abstract—We present a method combining affinity prediction with region agglomeration, which improves significantly upon the state of the art of neuron segmentation from electron microscopy (EM) in accuracy and scalability. Our method consists of a 3D U-NET, trained to predict affinities between voxels, followed by iterative region agglomeration. We train using a structured loss based on MALIS, encouraging topologically correct segmentations obtained from affinity thresholding. Our extension consists of two parts: First, we present a quasi-linear method to compute the loss gradient, improving over the original quadratic algorithm. Second, we compute the gradient in two separate passes to avoid spurious gradient contributions in early training stages. Our predictions are accurate enough that simple learning-free percentile-based agglomeration outperforms more involved methods used earlier on inferior predictions. We present results on three diverse EM datasets, achieving relative improvements over previous results of 27, 15, and 250 percent. Our findings suggest that a single method can be applied to both nearly isotropic block-face EM data and anisotropic serial sectioned EM data. The runtime of our method scales linearly with the size of the volume and achieves a throughput of ~ 2.6 seconds per megavoxel, qualifying our method for the processing of very large datasets.

Index Terms—Connectomics, electron microscopy, deep learning, structured loss, segmentation, affinity prediction, agglomeration

✦

1 INTRODUCTION

PRECISE reconstruction of neural connectivity is of great importance to understand the function of biological nervous systems. 3D electron microscopy (EM) is the only available imaging method with the resolution necessary to visualize and reconstruct dense neural morphology without ambiguity. At this resolution, however, even moderately small neural circuits yield image volumes that are too large for manual reconstruction. Therefore, automated methods for neuron tracing are needed to aid human analysis.

We present a method combining a structured loss for deep learning based instance separation with subsequent region agglomeration for neuron segmentation in 3D electron microscopy, which improves significantly upon state of the art in terms of accuracy and scalability. For an overview, see Fig. 1, top row. The main components of our method are: (1) Prediction of 3D affinity graphs using a 3D U-NET architecture [1], (2) a structured loss based on MALIS

[2] to train the U-NET to minimize topological errors, and (3) an efficient $O(n)$ agglomeration scheme based on quantiles of predicted affinities.

The choice of using a 3D U-NET architecture to predict voxel affinities is motivated by two considerations: First, U-NETs have already shown superior performance on the segmentation of 2D [3] and 3D [1] biomedical image data. One of their favourable properties is the multi-scale architecture which enables computational and statistical efficiency. Second, U-NETs efficiently predict large regions. This is of particular interest in combination with training on the MALIS structured loss, for which we need affinity predictions in a region.

We train our 3D U-NET to predict affinities using an extension of the MALIS loss function [2]. Like the original MALIS loss, we minimize a topological error on hypothetical thresholding and connected component analysis on the predicted affinities. We extended the original formulation to derive the gradient with respect to all predicted affinities (as opposed to sparsely sampling them), leading to denser and faster gradient computation. Furthermore, we compute the MALIS loss in two passes: In the *positive pass*, we constrain all predicted affinities between and outside of ground-truth regions to be 0, and in the *negative pass*, we constrain affinities inside regions to be 1 which avoids spurious gradients in early training stages.

Although the training is performed assuming subsequent thresholding, we found iterative agglomeration of *fragments* (or "supervoxels") to be more robust to small errors in the affinity predictions. To this end, we extract fragments running a watershed algorithm on the predicted affinities. The fragments are then represented in a region

---

- J. Funke is with the Institut de Robotica i Informatica Industrial, Universitat Politecnica de Catalunya, Barcelona 08034, Spain. E-mail: jfunke@iri.upc.edu.
- F. Tschopp is with the Institut fur Neuroinformatik UZH/ETH, Zurich 8057, Switzerland. E-mail: tschopfa@student.ethz.ch.
- W. Grisaitis, A. Sheridan, C. Singh, and S. C. Turaga are with the Turaga Lab, Howard Hughes Medical Institute, Janelia Research Campus, Ashburn, VA 20147. E-mail: {grisaitisw, sheridana, singhc, turagas}@janelia.hhmi.org.
- S. Saalfeld is with the Saalfeld Lab, Howard Hughes Medical Institute, Janelia Research Campus, Ashburn, VA 20147. E-mail: saalfelds@janelia.hhmi.org.

(a) 3D U-Net.

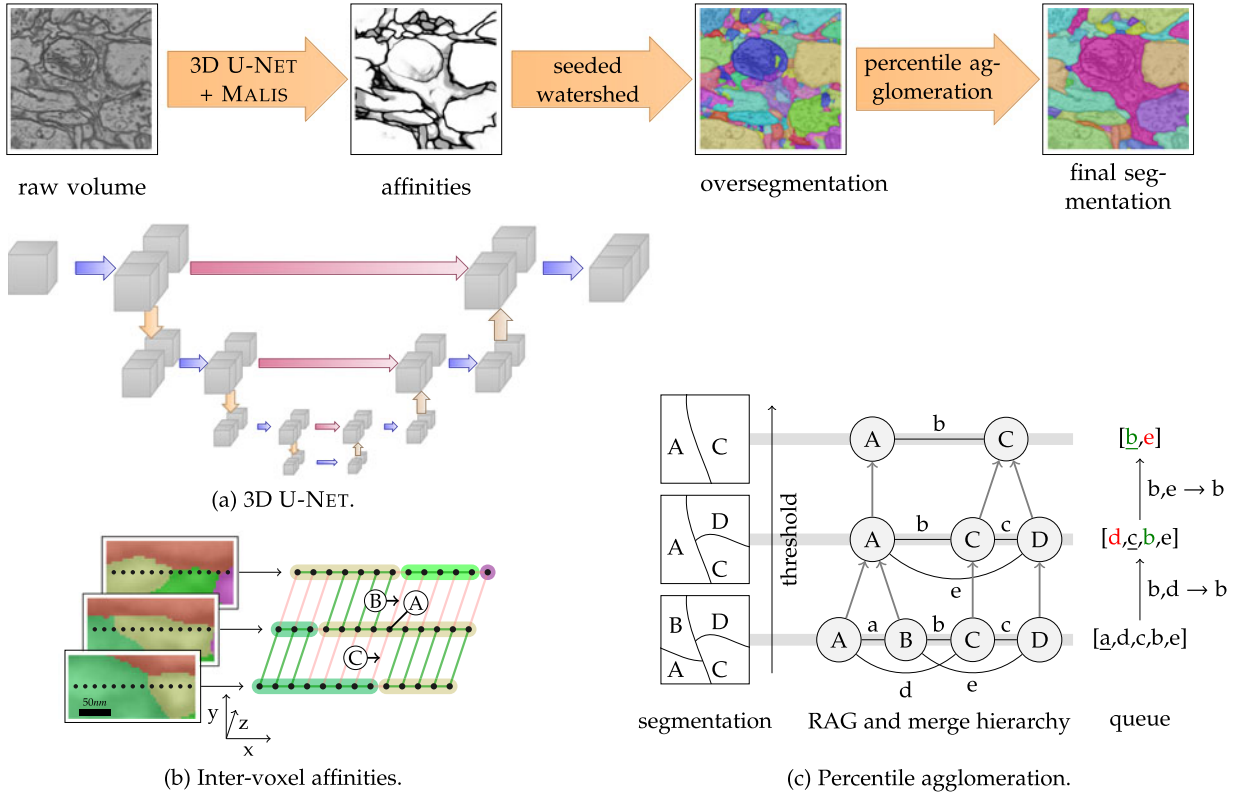(b) Inter-voxel affinities.

(c) Percentile agglomeration.

Fig. 1. Overview of our method (top row). Using a 3D U-Net (a), trained with the proposed constrained Malis loss, we directly predict inter-voxel affinities from volumes of raw data. Affinities provide advantages especially in the case of low-resolution data (b). In the example shown here, the *voxels* cannot be labeled correctly as foreground/background: If A were labeled as foreground, it would necessarily merge with the regions in the previous and next section. If it were labeled as background, it would introduce a split. The labeling of affinities on *edges* allows B and C to separate A from adjacent sections, while maintaining connectivity inside the region. From the predicted affinities, we obtain an over-segmentation that is then merged into the final segmentation using a percentile-based agglomeration algorithm (c).

adjacency graph (RAG), where edges are scored to reflect the predicted affinities between adjacent fragments: edges with small scores will be merged before edges with high scores. We discretize edge scores into k evenly distributed bins, which allows us to use a bucket priority queue for sorting. This way, the agglomeration can be carried out with a worst-case linear runtime.

The resulting method (prediction of affinities, watershed, and agglomeration) scales favourably with $O(n)$ in the size n of the volume, a crucial property for neuron segmentation from EM volumes, where volumes easily reach several hundreds of terabytes. This is a major advantage over current state-of-the-art methods that all follow a similar pattern. First, voxel-wise predictions are made using a deep neural network. Subsequently, fragments are obtained from these predictions which are then merged using either greedy (Celis [4], Gala [5]) or globally optimal objectives (MultiCut [6] and lifted MultiCut [7], [8]). All these methods depend heavily on the quality of the initial fragments, which in turn depend on the quality of the boundary prediction. Despite this strong coupling, the boundary classifier is mostly trained unaware of the algorithm used to subsequently extract fragments. A noteworthy exception is a recent work [9] where a boundary classifier is trained using a structured loss to fill objects with seeded watershed regions. This work demonstrates the usefulness of structured boundary prediction, similar in spirit to the method described here. Nevertheless, the majority of current efforts focuses on the merging of fragments: Both Celis and Gala train a classifier to predict scores for hierarchical agglomeration which increases the computational complexity of agglomeration during inference. Similarly, the MultiCut variants train a classifier to predict the connectivity of fragments that are then clustered by solving a computationally expensive combinatorial optimization problem. Our proposed fragment agglomeration method drastically reduces the computation complexity compared to previous merge methods and does not require a separate training step.

We demonstrate the efficacy of our method on three diverse datasets of EM volumes, imaged by three different 3D electron microscopy techniques: Cremi (ssTEM, *Drosophila*), Fib-25 (FIBSEM, *Drosophila*), and SegEm (SBEM, mouse cortex). Our method significantly improves over the current state of the art in each of these datasets, outperforming in particular computationally more expensive methods without favorable worst-case runtime guarantees.

We made the source code for training[1] and agglomeration[2] publicly available, together with usage example scripts to reproduce our Cremi results.[3]

2 METHOD

2.1 Deep Multi-Scale Convolutional Network for Predicting 3D Voxel Affinities

We use a 3D U-Net architecture [1] to predict voxel affinities on 3D volumes. We use the same architecture for all

1. https://github.com/naibaf7/caffe
2. https://github.com/funkey/waterz
3. http://cremi.org/static/data/20170312_mala_v2.tar.gz

investigated datasets which we illustrate in Fig. 1a. In particular, our 3D U-NET consists of four levels of different resolutions. In each level, we perform at least one convolution pass (shown as blue arrows in Fig. 1a) consisting of two convolutions (kernel size $3 \times 3 \times 3$) followed by rectified linear units. Between the levels, we perform max pooling on variable kernel sizes depending on the dataset resolution for the downsampling pass (yellow arrows), as well as transposed convolution of the same size for upsampling (brown arrows). The results of the upsampling pass are further concatenated with copies of the feature maps of the same level in the downsampling pass (red arrows), cropped to account for context loss in the lower levels. Details of the individual passes are shown in Fig. 6. A more detailed description of the U-NET architectures for each of the investigated datasets can be found in Fig. 5.

We chose to predict voxel affinities on edges between voxels instead of labeling voxels as foreground/background to allow our method to handle low spatial resolutions. As we illustrate in Fig. 1b, a low z resolution (common for serial section EM) renders a foreground/background labeling of voxels impossible. Affinities, on the other hand, effectively increase the expressiveness of our model and allow to obtain a correct segmentation. Furthermore, affinities easily generalize to arbitrary neighborhoods and might thus allow the prediction of longer range connectivity.

2.2 Training Using Constrained Malis

We train our network using an extension of the MALIS loss [2]. This loss, that we term *constrained* MALIS, is designed to minimize topological errors in a segmentation obtained by thresholding and connected component analysis. Although thresholding alone will unlikely produce accurate results, it serves as a valuable proxy for training: If the loss can be minimized for thresholding, it will in particular be minimized for agglomeration. To this end, in each training iteration, a complete affinity prediction of a 3D region is considered. Between every pair of voxels, we determine the maximin affinity edge, i.e., the highest minimal edge over all paths connecting the pair. This edge is crucial as it determines the threshold under which the two voxels in question will be merged. Naturally, for voxels that are supposed to belong to the same region, we want the maximin edge affinity to be as high as possible, and for voxels of different regions as low as possible.

Our extension consists of two parts: First, we improve the computational complexity of the MALIS loss by presenting an $O(n \log(n) + kn)$ method for the computation of the gradient, where n is the size of the volume and k the number of ground-truth objects. We thus improve over the previous method that had a complexity of $O(n^2)$. Second, we compute the gradient in two separate passes, once for affinities inside ground-truth objects (positive pass), and once for affinities between and outside of ground-truth objects.

2.2.1 The MALIS Loss

Let $G = (V, E, a)$ be an affinity graph on voxels V with edges $E \subseteq V^2$ and affinities $a : E \mapsto [0, 1]$. A maximin edge between two voxels u and v is an edge $\mathrm{mm}(u, v) \in E$ with lowest affinity on the overall highest affinity path $P^*_{u,v}$ connecting u and v, i.e.,

$$P^*_{u,v} = \arg\max_{P \in \mathcal{P}_{u,v}} \min_{e \in P} a(e) \qquad \mathrm{mm}(u,v) = \arg\min_{e \in P^*_{u,v}} a(e), \quad (1)$$

where $\mathcal{P}_{u,v}$ denotes the set of all paths between u and v. If we imagine a simple thresholding on the affinity graph, such that edges with affinities below a threshold θ are removed from G, then the affinity of the maximin edge $\mathrm{mm}(u, v)$ is equal to the highest threshold under which nodes u and v would still be part of the same connected component. Acknowledging the importance of maximin edges, the MALIS loss favors high maximin affinities between voxels that belong to the same ground-truth segment, and low maximin affinities between voxels that belong to different ground-truth segments. We assume that a ground-truth segmentation is given as a labelling $s : V \mapsto \{0, \dots, k\}$ such that each segment has a unique label in $\{1, \dots, k\}$ and background is marked with 0. Let $F \subseteq V$ denote all foreground voxels $F = \{v \in V \mid s(v) \neq 0\}$ and $\delta(u, v)$ indicate whether u and v belong to the same ground-truth segment:

$$\delta(u,v) = \begin{cases} 1 & \text{if } u, v \in F \text{ and } s(u) = s(v), \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The MALIS loss $L(s, a)$ is the sum of affinity losses over the maximin edges of every pair of voxels that do not belong to the background:

$$L(s,a) = \sum_{u,v \in F} l(\delta(u,v), a(\mathrm{mm}(u,v))). \quad (3)$$

The affinity loss can be any continuous and differentiable function, we chose $l(x, y) = (x - y)^2$ for all experiments in this paper.

2.2.2 Quasilinear Loss Computation

Considering that we have $O(n^2)$, $n = |V|$, pairs of voxels, but—in the case of grid graphs considered here—only $O(n)$ edges, it follows that maximin edges are shared between voxel pairs. This observation generalizes to arbitrary graphs. In particular, the union of all maximin edges forms a maximal spanning tree (MST),

$$\{\mathrm{mm}(u,v) \mid (u,v) \in V^2\} = \mathrm{MST}(G), \quad (4)$$

i.e., there are always only $n - 1$ maximin edges in a graph.

That the previous equality holds can easily be proven by contradiction: Assume that for a pair (u, v), $\mathrm{mm}(u, v) \notin \mathrm{MST}(G)$. Let $P^+_{u,v} \subseteq \mathrm{MST}(G)$ denote the path connecting u and v on the MST, and let $\mathrm{mtp}(u, v)$ denote the edge with minimal affinity on $P^+_{u,v}$:

$$\mathrm{mtp}(u,v) = \arg\min_{e \in P^+_{u,v}} a(e). \quad (5)$$

Following our assumption, $P^+_{u,v}$ does not contain $\mathrm{mm}(u, v)$. By definition Eq. (1), the following inequalities hold:

$$a(\mathrm{mtp}(u,v)) \leq a(\mathrm{mm}(u,v)) \leq a(e) \ \ \forall e \in P^*_{u,v}. \quad (6)$$

We can now remove $\mathrm{mtp}(u, v)$ from the MST to obtain two disconnected sub-trees separating u from v. Since $P^*_{u,v}$ connects u and v, there exists an edge $e^* \in P^*_{u,v}$ that will reconnect the two sub-trees. However, $a(\mathrm{mtp}(u,v)) \leq a(e^*)$. If strict inequality holds, this will create a tree with a

(a) Predicted affinities.   (b) Positive pass.   (c) Negative pass.   (d) Gradient of loss.
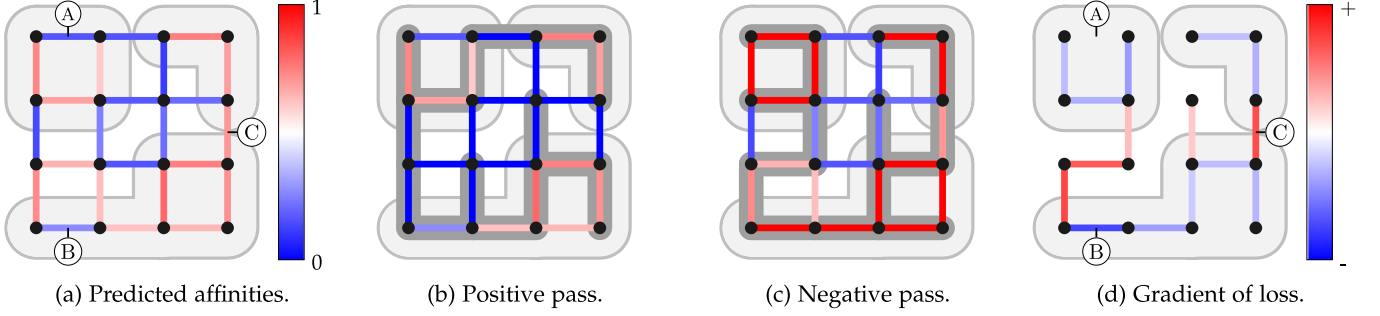
Fig. 2. Illustration of the constrained MALIS loss. Given predicted affinities (blue low, red high) and a ground-truth segmentation (a), losses on maximin edges are computed in two passes: In the positive pass, (b), affinities of edges between ground-truth regions are set to zero (blue), in the negative pass (c), affinities within ground-truth regions are set to one (red). In either case, a maximal spanning tree (shown as shadow) is constructed to identify maximin edges. Note that, in this example, edge A is not a maximin edge in the positive pass since the incident voxels are already connected by a high affinity path. In contrast, edge B is the maximin edge of the bottom left voxel to any other voxel in the same region and thus contributes to the loss. Similarly, C is the maximin edge connecting voxels of different ground-truth regions and contributes during the negative pass to the loss. The resulting gradients of the loss with respect to each edge affinity is shown in (d) (positive values in red, negative in blue).

larger sum of affinities than the MST, thus contradicting our assumptions. If equality holds and $a(\mathrm{mtp}(u,v)) = a(\mathrm{mm}(u,v)) = a(e^*)$, then there are more than one possible MSTs and hence $\mathrm{mm}(u,v)$ is contained in one of them.

Consequently, we are able to identify the maximin edge and compute its loss for each voxel pair by growing an MST on G. We use Kruskal's algorithm [10] to grow an MST, which consists of two steps: First, we sort all edges by affinity in descending order. Second, we iterate over all edges and grow the MST using a union-find data structure. Whenever a new edge e merges two trees $T_1, T_2 \subset MST(G)$ during construction of the MST, we compute the *positive* and *negative* weight of this edge on the fly. The positive weight $w_P(e)$ corresponds to the number of voxel pairs of the same ground-truth segment merged by e:

$$w_P(e) = |\{(u,v) \in F^2 \mid \delta(u,v) = 1,\ e = \mathrm{mm}(u,v)\}|. \quad (7)$$

By construction, e is the maximin edge to all pairs of voxels between the two trees it merges. Therefore, $w_P(e)$ equals the product of the number of voxels having label i in either tree, summed over all $i \in \{1, \dots, k\}$. Let V_T denote the set of voxels in T and $V_T^i \subseteq V_T$ the subset with ground-truth label i. The positive weight can then be rewritten as:

$$w_P(e) = \sum_{i \in \{1,\dots,k\}} \left| V_{T_1}^i \right| \left| V_{T_2}^i \right|. \quad (8)$$

Equivalently, the negative weight $w_N(e)$ is the number of voxel pairs of different ground-truth segments merged by e:

$$w_N(e) = |\{(u,v) \in F^2 \mid \delta(u,v) = 0,\ e = \mathrm{mm}(u,v)\}| \quad (9)$$

$$= \sum_{i \neq j \in \{1,\dots,k\}} \left| V_{T_1}^i \right| \left| V_{T_2}^j \right| \quad (10)$$

$$= \left| V_{T_1} \right| \left| V_{T_2} \right| - \sum_{i \in \{1,\dots,k\}} \left| V_{T_1}^i \right| \left| V_{T_2}^i \right|. \quad (11)$$

We can now rewrite the MALIS loss Eq. (3) as

$$L(s,a) = \sum_{e \in \mathrm{MST}(G)} w_P(e) l(1, a(e)) + w_N(e) l(0, a(e)), \quad (12)$$

and avoid the costly sum over all pairs of voxels. We keep track of the sizes of sets V_T and V_T^i used in each tree during

the construction of the MST. Consequently, the complexity of our algorithm is dominated by first sorting all edges by their affinity in $O(n \log(n))$ and subsequently evaluating Eqs. (8) and (11) while constructing the MST in $O(kn)$, resulting in a final complexity of $O(n \log(n) + kn)$. We thus improve over a previous method [2] that required $O(n^2)$ and therefore had to fall back to sparse sampling of voxel pairs. Note that this only affects the training of the network, the affinity prediction during test time scales linearly with the volume size.

2.2.3   Constrained MALIS

We further extend previous work by computing the maximin edge losses in two passes: In the first pass we compute only the weights w_P for edges within the same region (positive pass). In the second pass, we compute the weights w_N for edges between different regions (negative pass). As shown in Fig. 2, in the positive pass, we assume that all edges between regions have been predicted correctly and set their affinities to zero. Consequently, only maximin edges inside a region are found and contribute to the loss. This obviates an inefficiency in a previous formulation [2], where a spurious high-affinity (i.e., false positive) path leaving and entering a region might connect two voxels inside the same region. In this case, the maximin edge could lie outside of the considered region, resulting in an unwanted gradient contribution that would reinforce the false positive. Analogously, in the negative pass, all affinities inside the same region are set to one to avoid reinforcement of false negatives inside regions. Finally, the gradient contributions of both passes are added together.

Note that, similar to the original MALIS formulation [2], the constrained version presented here does not require precise location of the boundaries. In applications where the exact location of the boundary is less relevant, a broader background region around boundaries can be given. During the negative pass, any correctly predicted cut through this background region will result in a loss of zero.

2.3   Hierarchical Agglomeration

Our method for hierarchical agglomeration of segments from the predicted affinities consists of two steps. First, we use a heuristic to extract small fragments directly from the predicted affinities. Second, we iteratively score and merge

averaged affinities   threshold at 0.5
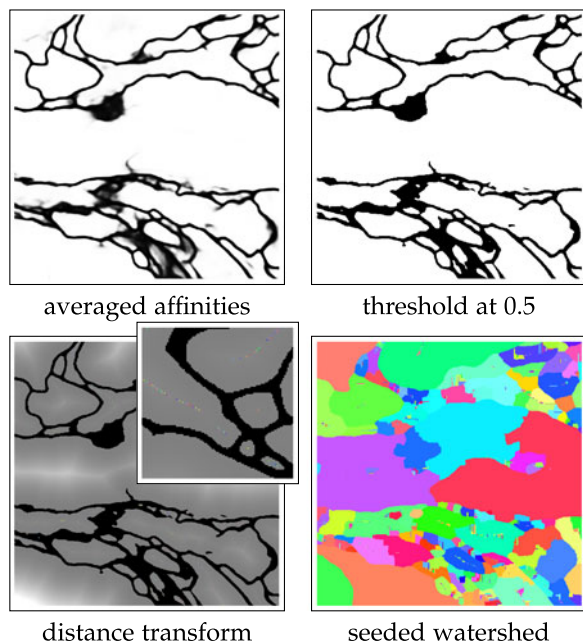
distance transform   seeded watershed

Fig. 3. Illustration of the seeded watershed heuristic.

adjacent fragments into larger objects until a predefined threshold is reached.

2.3.1 Fragment Extraction

The extraction of fragments is a crucial step for the subsequent agglomeration. Too many fragments slow down the agglomeration unnecessarily and increase its memory footprint. Too few fragments, on the other hand, are subject to undersegmentation that cannot be corrected.

Empirically, we found a seeded watershed to deliver the best trade-off between fragment size and segmentation accuracy across all investigated datasets. For the seeded watershed, we first average the predicted affinities for each voxel to obtain a volume of boundary predictions. We subsequently threshold the boundary predictions at 0.5 and perform a distance transform on the resulting mask. Every local maximum is taken as a seed, from which we grow basins using a standard watershed algorithm [11] on the boundary predictions. For an example,
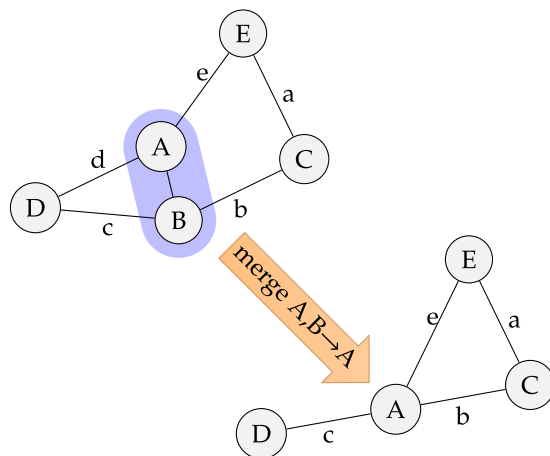

Fig. 4. Illustration of the three different edge update cases during a merge. Case 1: The edge is not involved in the merge at all (a). Case 2: One of the edge's nodes is involved in the merge, but the boundary represented by the edge does not change (b and e). Case 3: The boundaries represented by two edges get merged (c and d). Only in this case the score needs to be updated.

see Fig. 3. As argued above, voxel-wise predictions are not fit for anisotropic volumes with low z-resolution (see Fig. 1b). To not re-introduce a flaw that we aimed to avoid by predicting affinities instead of voxel-wise labels in the first place, we perform the extraction of fragments xy-section-wise for anisotropic volumes.

2.3.2 Fragment Agglomeration

For the agglomeration, we consider the region adjacency graph (RAG) of the extracted fragments. The RAG is an annotated graph $G = (V, E, f)$, with V the set of fragments, $E \subseteq V \times V$ edges between adjacent fragments, and $f : E \mapsto \mathbb{R}$ an edge scoring function. The edge scoring function is designed to prioritize merge operations in the RAG, i.e., the contraction of two adjacent nodes into one, such that edges with lower scores are merged earlier. Given an annotated RAG, a segmentation can be obtained by finding the edge with the lowest score, merge it, recompute the scores of edges affected by the merge, and iterate until the score of the lowest edge hits a predefined threshold θ. In the following, we will denote by G_i the RAG after i iterations (and
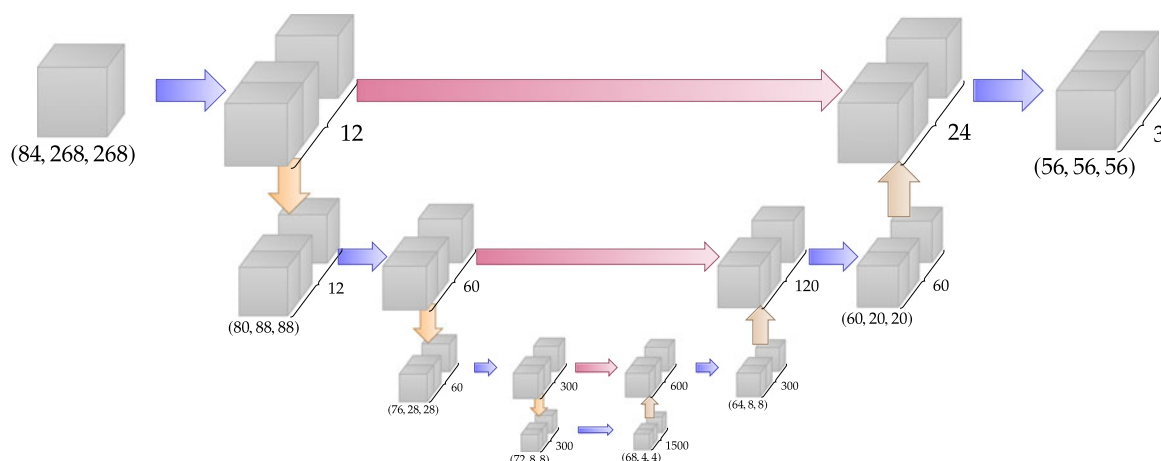

Fig. 5. Overview of the U-net architecture used for the CREMI dataset. The architectures for FIB-25 and SEGEM are similar, with changes in the input and output sizes (in: $(132, 132, 132)$, out: $(44, 44, 44)$ for FIB-25 and in: $(188, 188, 144)$, out: $(100, 100, 96)$ for SEGEM) and number of feature maps for FIB-25 (24 in the first layer, increased by a factor of 3 for lower layers).
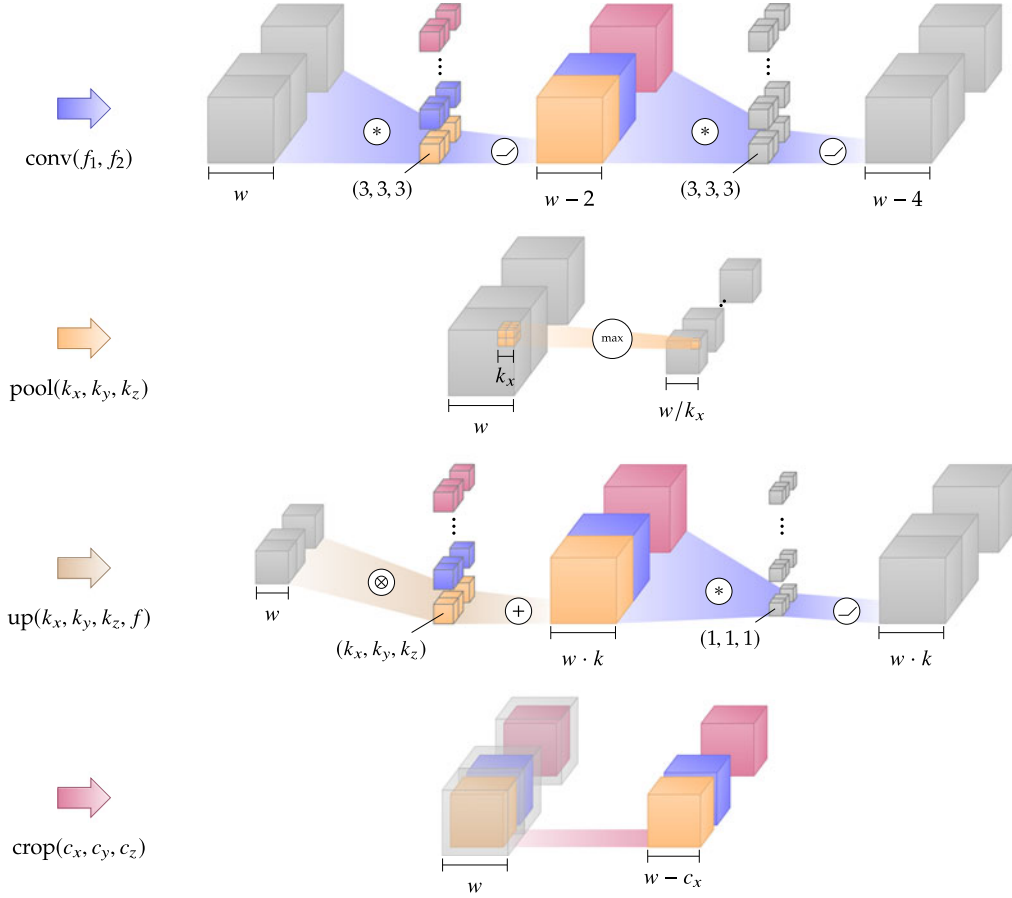
Fig. 6. Details of the convolution (blue), max-pooling (yellow), upsampling (brown), and copy-crop operations (red). "$*$" denotes a convolution, "⟋" a rectified linear unit, and "$\otimes$" the Kronecker matrix product.

analogously by V_i, E_i, and f_i its nodes, edges, and scores), with $G_0 = G$ as introduced above. We will "reuse" nodes and edges, meaning $V_{i+1} \subset V_i$ and $E_{i+1} \subset E_i$.

Given that the initial fragments are indeed an oversegmentation, it is up to the design of the scoring function and the threshold θ to ensure a correct segmentation. The design of the scoring function can be broken down into the initialization of $f_0(e)$ for $e \in E_0$ (i.e., the initial scores) and the update of $f_i(e)$ for $e \in E_i$; $i > 0$ after a merge of two regions $a, b \in V_{i-1}$. For the update, three cases can be distinguished (for an illustration see Fig. 4): (1) e was not affected by the merge, (2) e is incident to a or b but represents the same contact area between two regions as before, and (3) e results from merging two edges of E_{i-1} into one (the other edge will be deleted). In the first two cases, the score does not change, i.e., $f_i(e) = f_{i-1}(e)$, since the contact area between the nodes linked by e remains the same. In the latter case, the contact area is the union of the contact area of the merged edges, and the score needs to be updated accordingly. Acknowledging the merge hierarchy of edges (as opposed to nodes), we will refer to the leaves under a merged edge e as *initial edges*, denoted by $E^*(e) \subseteq E_0$.

In our experiments, we initialize the edge scores $f(e)$ for $e \in E_0$ with one minus the maximum affinity between the fragments linked by e and update them using a quantile value of scores of the initial edges under e. This strategy has been found empirically over a range of possible implementations of f (see Section 3).

Implemented naively, hierarchical agglomeration has a worst-case runtime complexity of at least $O(n \log(n))$, where $n = |E_0|$ is the number of edges in the initial RAG. This is due to the requirement of finding, in each iteration, the cheapest edge to merge, which implies sorting of edges based on their scores. Furthermore, the edge scoring function has to be evaluated $O(n)$ times, once for each affected edge of a node merge (assuming nodes have a degree bounded by a constant). For the merge function suggested above, a quantile of $O(n)$ initial edge scores has to be found in the worst case would a balanced merge tree improve this figure to $O(\log(n))$?, resulting in a total worst-case runtime complexity of $O(n \log(n) + n^2)$.

To avoid this prohibitively high runtime complexity, we propose to discretize the initial scores f_0 into k bins, evenly spaced in the interval $[0, 1]$. This simple modification has two important consequences: First, a bucket priority queue for sorting edge scores can be used, providing constant time insert and pop operations. Second, the computation of quantiles can be implemented in constant time and space by using histograms of the k possible values. This way, we obtain constant-time merge iterations (pop an edge, merge nodes, update scores of affected edges), applied at most n times, thus resulting in an overall worst-case complexity of $O(n)$. With $k = 256$ bins, we noticed no sacrifice of accuracy in comparison to the non-discretized variant.

The analysis above holds only if we can ensure that the update of the score of an edge e, and thus the update of the priority queue, can be performed in constant time.

TABLE 1
Qualitative Results of our Method (U-Net Mala) Compared
to the Respective State of the Art on the Testing Volumes
of Each Dataset and a Baseline (U-Net)

method	VOI split	VOI merge	VOI sum
U-Net Mala	0.891	0.180	**1.071**
U-Net	1.205	0.316	1.520
FlyEM [12]	1.490	0.462	1.952
CELIS [4]	1.426	0.208	1.634
CELIS+MC [4]	1.037	0.229	1.266

(a) Results on Fib-25, evaluated on whole test volume.

method	VOI split	VOI merge	VOI sum
U-Net Mala	1.953	0.198	**2.151**
U-Net	2.442	0.471	2.914
FlyEM [12]	3.160	0.251	3.411
CELIS [4]	3.401	0.166	3.568
CELIS+MC [4]	2.354	0.216	2.570

(b) Results on Fib-25, evaluated on synaptic sites.

method	VOI split	VOI merge	VOI sum	CREMI score
U-Net Mala	0.425	0.181	**0.606**	**0.289**
U-Net	0.979	0.546	1.524	0.793
LMC [8]	0.597	0.272	0.868	0.398
CRunet [13]	1.081	0.389	1.470	0.566
LFC [14]	1.085	0.140	1.225	0.616

(c) Results on Cremi (from leaderboard in [15]).

method	IED split	IED merge	IED total
U-Net Mala	6.259	21.337	**4.839**
U-Net	6.903	1.719	1.377
SegEM [16]	2.121	3.951	1.380

(d) Results on SegEm.

Highlighted in bold are the names of our method and the best value in each column. Measures shown are variation of information (VOI, lower is better), CREMI score (geometric mean of VOI and adapted RAND error, lower is better), and inter-error distance in μm (IED, higher is better) evaluated on traced skeletons of the test volume. The IED has been computed using the TED metric on skeletons [12] with a distance threshold of 52 nm (corresponding to the thickness of two z-sections). Cremi results are reported as average over all testing samples, individual results can be found in Fig. 8.

In particular, it is to be avoided to search for e in its respective bucket. We note that for the quantile scoring function (and many more), the new edge score $f_i(e)$ after merging an edge $f \in E_{i-1}$ into $e \in E_{i-1}$ is always greater than or equal to its previous score. We can therefore mark e as *stale* and f as *deleted* and proceed merging without resorting the queue or altering the graph. Whenever a stale edge is popped from the priority queue, we compute its actual score and insert it again into the queue. Not only does this ensure constant time updates of edge scores and the priority queue, it also avoids computing scores for edges that are never used for a merge. This can happen if the threshold is hit before considering the edge, or if the edge got marked as deleted as a consequence of a nearby merge.

3 RESULTS

Datasets We present results on three different and diverse datasets: Cremi [13], Fib-25 [14], and SegEm [15] (see Table 2 for an overview). These datasets sum up to almost 15 gigavoxels of testing data, with Fib-25 alone contributing 13.8 gigavoxels, thus challenging automatic segmentation methods for their efficiency. In fact, only two methods have so far been evaluated on Fib-25 [4], [14]. Another challenge is

posed by the Cremi dataset: Coming from serial section EM, this dataset is highly anisotropic and contains artifacts like support film folds, missing sections, and staining precipitations. Regardless of the differences in isotropy and presence of artifacts, we use the same method (3D U-Net training, prediction, and agglomeration) for all datasets. The size of the receptive field of the U-Net was set for each dataset to be approximately one μm^3, i.e., $213 \times 213 \times 29$ for Cremi, $89 \times 89 \times 89$ for Fib-25, and $89 \times 89 \times 49$ for SegEm. For the Cremi dataset, we also pre-aligned training and testing data with an elastic alignment method [16], using the padded volumes provided by the challenge.

Training. We implemented and trained our network using the caffe library on modestly augmented training data for which we performed random rotations, transpositions and flips, as well as elastic deformations. On the anisotropic Cremi dataset, we further simulated missing sections by setting intensity values to 0 ($p = 0.05$) and low contrast sections by multiplying the intensity variance by 0.5 ($p = 0.05$). We used the Adam optimizer [17] with an initial learning rate of $\alpha = 10^{-4}$, $\beta_1 = 0.95$, $\beta_2 = 0.99$, and $\epsilon = 10^{-8}$.

Quantitative Results. On each of the investigated datasets, we see a clear improvement in accuracy using our method, compared to the current state of the art. We provide quantitative results for each of the datasets individually, where we compare our method (labeled U-Net Mala) against different other methods.[4] We also include a baseline (labeled U-Net) in our analysis, which is our method, but trained without the constrained Malis loss. In Table 1, we report the segmentation obtained on the best threshold found in the respective training datasets. In Fig. 7, we show the split/merge curve for varying thresholds of our agglomeration scheme.

For SegEm, we do not use the metric proposed by Berning et al. [15], as we found it to be problematic: The authors suggest an overlap threshold of 2 to compensate for inaccuracies in the ground-truth, however this has the unintended consequence of ignoring some neurons in the ground-truth for poor segmentations. For the SegEm segmentation (kindly provided by the authors), 195 out of 225 ground-truth skeletons are ignored because of insufficient overlap with any segmentation label. On our segmentation, only 70 skeletons would be ignored, thus the results are not directly comparable. Therefore, we performed a new IED evaluation using TED [12], a metric that allows slight displacement of skeleton nodes (we chose 52 nm in this case) in an attempt to minimize splits and merges. This metric reveals that our segmentations (U-Net Mala) improve over both split and merge errors, over all thresholds of agglomeration, including the initial fragments (see Fig. 7c).

Qualitative Results. Renderings of 11 and 23 randomly selected neurons, reconstructed using the proposed method, are shown for the test regions of Cremi and Fib-25 in Figs. 9 and 10, respectively.

Dataset (an)isotropy. Save for minor changes in the network architectures and the generation of initial fragments, our method works unchanged on both near-isotropic blockface datasets (Fib-25, SegEm) as well as on highly anisotropic serial-section datasets (Cremi). These findings suggest that

---

4. The presented results reflect the state of the Cremi challenge at the time of writing, see [13].

TABLE 2
Overview of Used Datasets

| Name	Imaging	Tissue	Resolution	Training Data	Testing Data
CREMI	ssTEM	*Drosophila*	$4 \times 4 \times 40$ nm	3 volumes of $1250 \times 1250 \times 125$ voxels	3 volumes of $1250 \times 1250 \times 125$ voxels
FIB-25	FIBSEM	*Drosophila*	$8 \times 8 \times 8$ nm	$520 \times 520 \times 520$ voxels	13.8 gigavoxels
SEGEM	SBEM	mouse cortex	$11 \times 11 \times 26$ nm	279 volumes of $100 \times 100 \times 100$ voxels	$400 \times 400 \times 350$ voxels (skeletons)
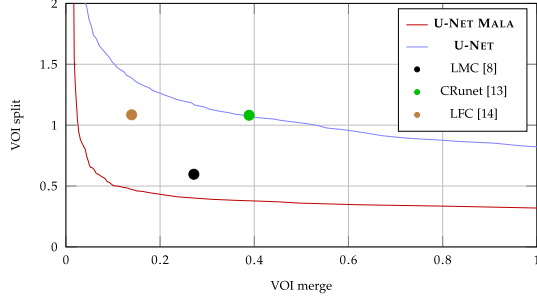
(a) VOI split/merge on CREMI (lower is better).
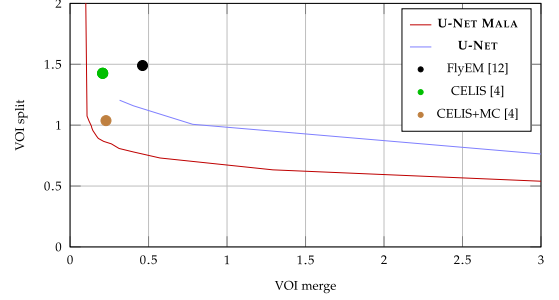
(b) VOI split/merge on FIB-25 (lower is better).
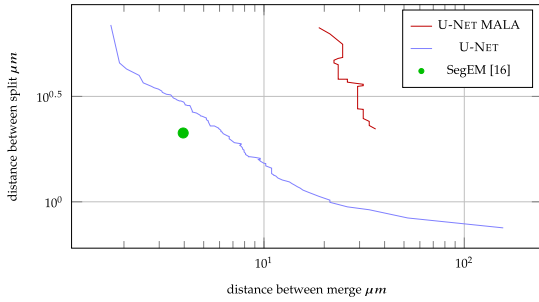
(c) IED split/merge on SEGEM (higher is better).
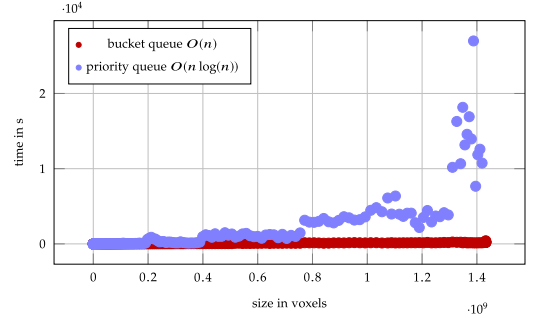
(d) Runtime analysis.

Fig. 7. (a-c) Split merge curves of our method (lines) for different thresholds on the CREMI, FIB-25, and SEGEM datasets, compared against the best-ranking competing methods (dots). (d) Performance comparison of a naive agglomeration scheme (priority queue, $O(n \log (n))$) versus our linear-time agglomeration (bucket queue, $O(n)$).

TABLE 3
Results for Different Merge Functions of Our Method Compared with the Agglomeration Strategy Proposed in [18]

	VOI split	VOI merge	VOI sum	CREMI score
15%	0.583	0.063	0.646	0.212
25%	0.441	0.092	0.533	0.188
50%	0.397	0.056	0.453	0.156
75%	0.347	0.074	**0.421**	**0.146**
85%	0.347	0.084	0.431	0.156
mean	0.380	0.058	0.438	0.149
Zlateski [20]	1.015	1.010	2.025	0.364

(a) CREMI (training data).

	VOI split	VOI merge	VOI sum
15%	1.480	0.364	1.844
25%	1.393	0.163	1.555
50%	1.115	0.234	**1.350**
75%	1.085	0.318	1.402
85%	1.176	0.394	1.570
mean	1.221	0.198	1.418
Zlateski [20]	1.054	1.017	2.071

(b) FIB-25.

We show the results at the threshold achieving the best score in the respective dataset (CREMI score for CREMI, VOI for FIB-25). Note that, for this analysis, we used the available training datasets which explains deviations from the numbers shown in Table 1.

there is no need for specialized constructions like dedicated features for anisotropic volumes or separate classifiers trained for merging of fragments within or across sections.

Merge Functions. Our method for efficient agglomeration allows using a range of different merge functions. In Table 3, we show results for different choices of quantile merge functions, mean affinity, and an agglomeration baseline proposed in [18] on datasets CREMI and FIB-25. Even across these very different datasets, we see best results for affinity quantiles between 50 and 75 percent. All initial edge scores have been set to one minus the maximum predicted affinity between the regions. Theoretically, this is the ideal choice since the MALIS training optimizes the maximin affinity between regions. Also empirically we found this initialization to perform consistently better than others (like the mean or a quantile affinity).

Throughput. Table 4 shows the throughput of our method for each dataset, broken down into affinity prediction

TABLE 4
Throughput of our Method for Each of the Investigated Datasets in Seconds Per Megavoxel

dataset	U-NET	watershed	agglomeration	total
CREMI	3.04	0.23	0.83	4.10
FIB-25	0.66	0.92	1.28	2.86
SEGEM	2.19	0.25	0.14	2.58

(a) CREMI sample A+

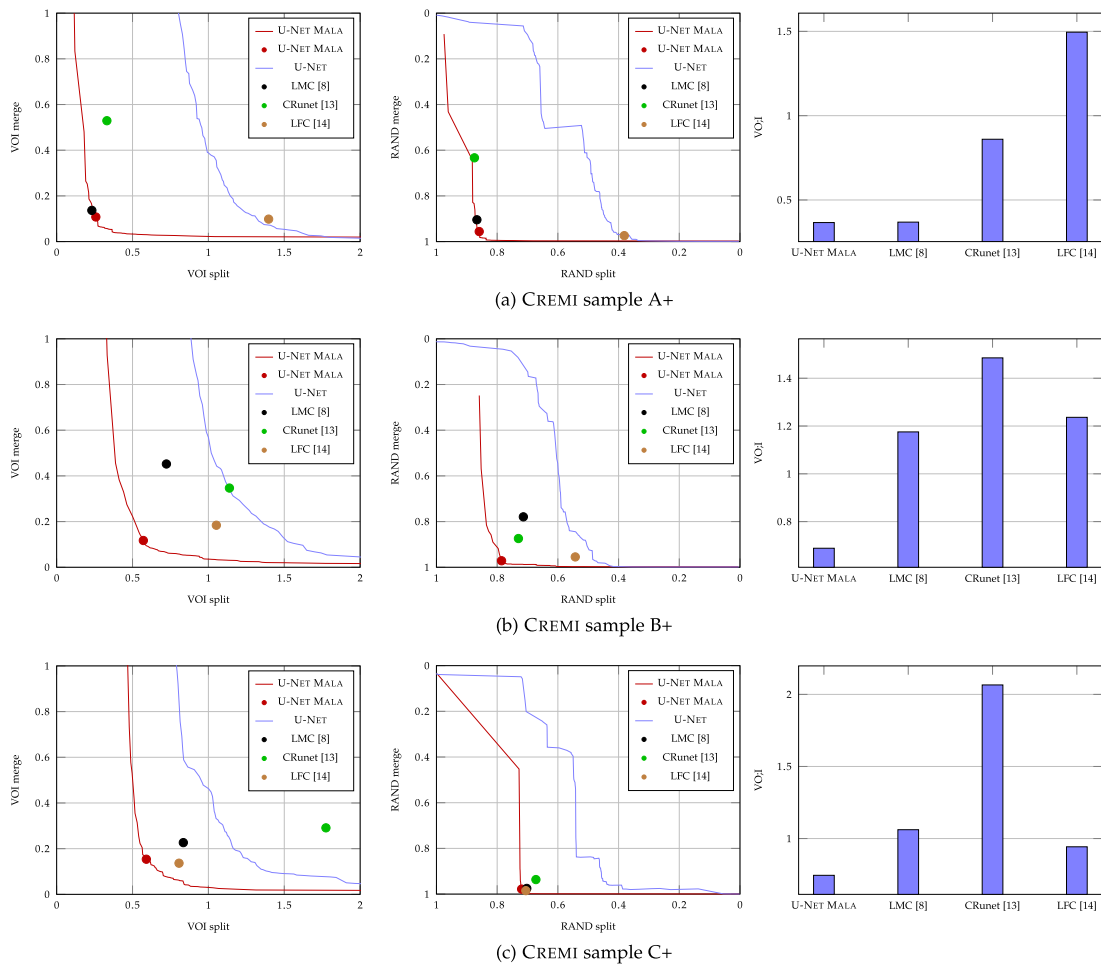(b) CREMI sample B+

(c) CREMI sample C+

Fig. 8. Comparison of the proposed method against competing methods on the CREMI testing datasets A+, B+, and C+. Shown are (from left to right) variation of information (VOI, split and merge contribution), Rand index (RAND, split and merge contribution), and the total VOI (split and merge combined). Baseline U-NET is our method, but without MALIS training (i.e., only minimizing the euclidean distance to the ground-truth affinities during training) For U-NET MALA, the red dot indicates the best threshold found on the training data.
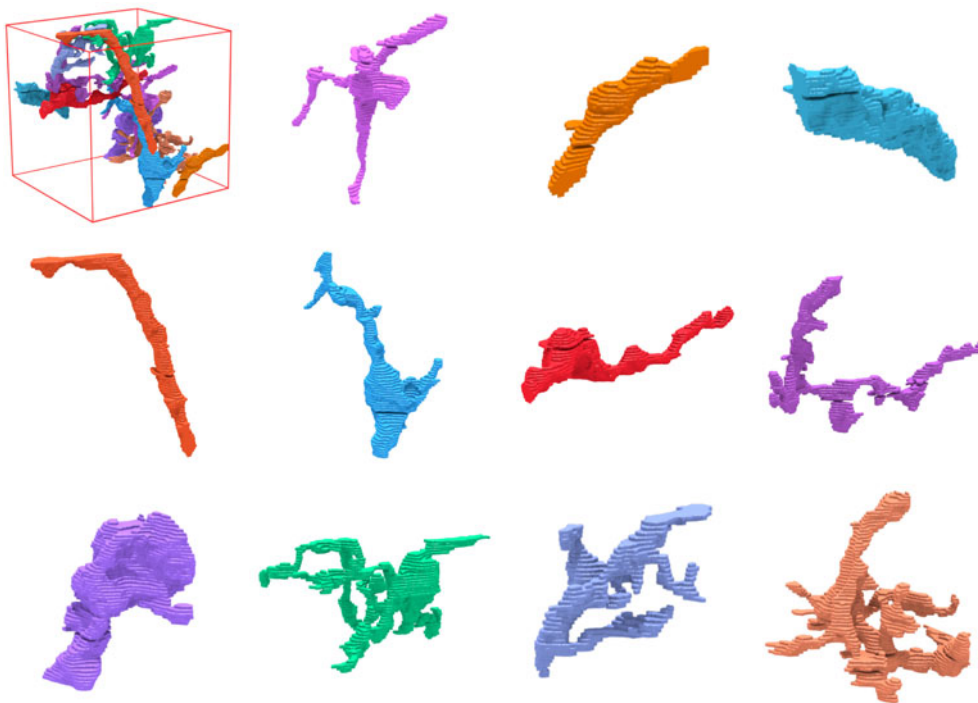

Fig. 9. Reconstructions of 11 randomly selected neurons of the 100 largest found in the CREMI test volume C+.
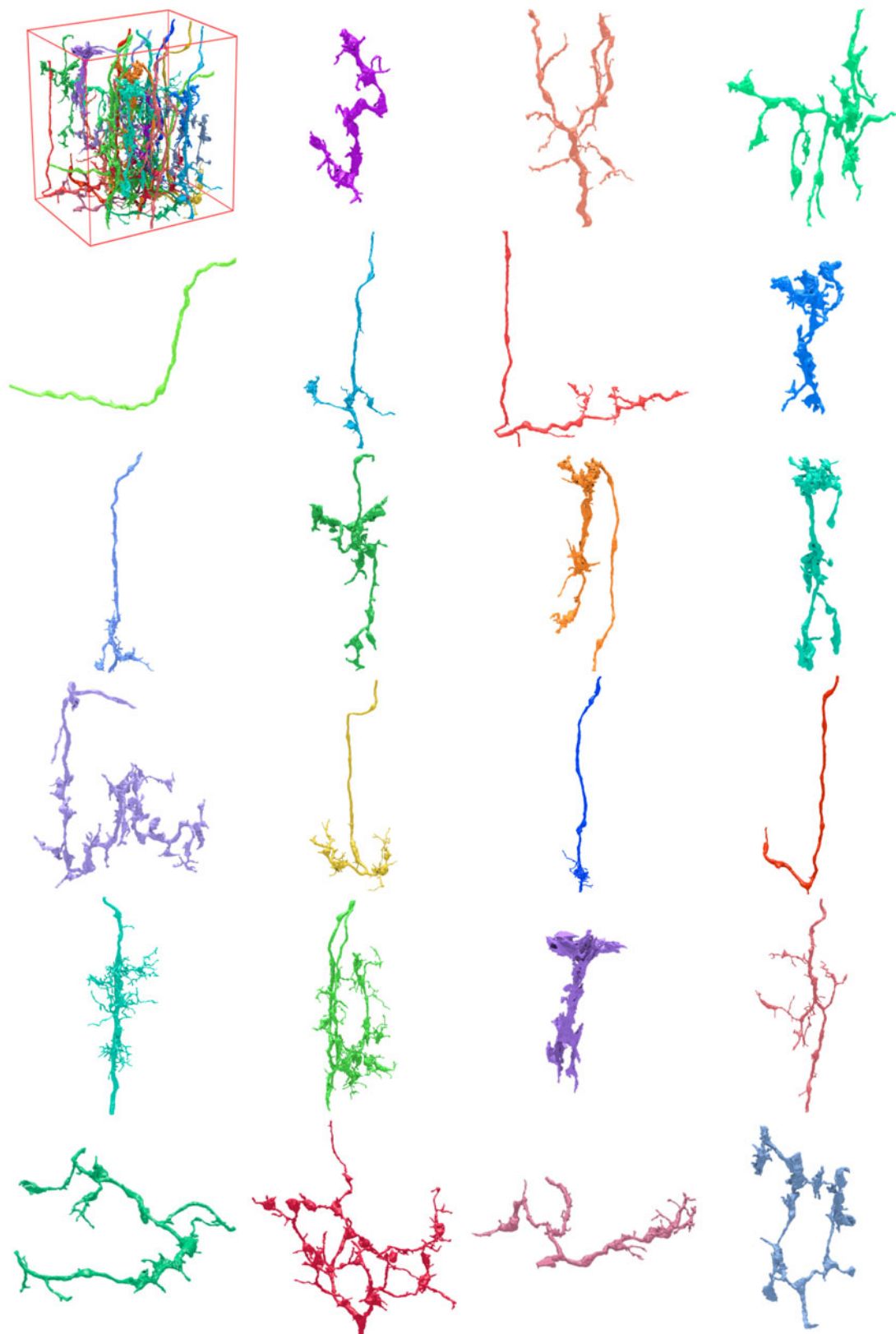
Fig. 10. Reconstructions of 23 randomly selected neurons of the 500 largest found in the Fib-25 test volume.

(U-Net), fragment extraction (watershed), and fragment agglomeration (agglomeration). For Cremi and SegEm, most time is spent on the prediction of affinities. The faster predictions in Fib-25 are due to less feature maps used in the network for this dataset.

To empirically confirm the theoretical speedup of using a bucket queue for agglomeration, we show in Fig. 7d a speed comparison of the proposed linear-time agglomeration against a naive agglomeration scheme for volumes of different sizes.

4 DISCUSSION

A remarkable property of the MALA method is that it requires almost no tuning to operate on datasets of different characteristics, except for minor changes in the size of the receptive field of the U-NET, training data augmentation to model dataset specific artifacts, and initial fragment generation. This suggests that there is no need for the development of dedicated algorithms for different EM modalities. Across all datasets, our results indicate that affinity predictions on voxels are sufficiently accurate to render sophisticated post-precessing obsolete. It remains an open question whether fundamentally different approaches, like the recently reported flood-filling network [19], also generalize in a similar way. At the time of writing, neither code nor data were publicly available for a direct comparison.

Furthermore, the U-NET is the only part in our method that requires training, so that all training data can (and should) be used to correctly predict affinities. This is an advantage over current state-of-the-art methods that require careful splitting of precious training data into non-overlapping sets used to train voxel-wise predictions and an agglomeration classifier (or accepting the disadvantages of having the sets overlap).

Although linear in runtime and memory, correct parallelization of hierarchical agglomeration is not trivial and will require further research. However, as demonstrated on the FIB-25 dataset, naive block-based agglomeration followed by empirical stitching based on region overlap generates very satisfying practical results.

ACKNOWLEDGMENTS

REFERENCES

[1] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning dense volumetric segmentation from sparse annotation," in *Proc. Int. Conf. Medical Image Comput. Comput.-Assisted Intervention*, 2016, pp. 424–432.

[2] K. Briggman, W. Denk, S. Seung, M. N. Helmstaedter, and S. C. Turaga, "Maximin affinity learning of image segmentation," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 1865–1873.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Medical Image Comput. Comput.-Assisted Intervention*, 2015, pp. 234–241.

[4] J. B. Maitin-Shepard, V. Jain, M. Januszewski, P. Li, and P. Abbeel, "Combinatorial energy learning for image segmentation," in *Proc. Advances Neural Inf. Process. Syst.*, 2016, pp. 1966–1974.

[5] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza, A. Chakraborty, and W. T. Katz, "Graph-based active learning of agglomeration (gala): A python library to segment 2D and 3D neuroimages," *Frontiers Neuroinformatics*, vol. 8, 2014, Art. no. 34.

[6] B. Andres, T. Kroeger, K. L. Briggman, W. Denk, N. Korogod, G. Knott, U. Koethe, and F. A. Hamprecht, "Globally optimal closed-surface segmentation for connectomics," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 778–791.

[7] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoue, T. Brox, and B. Andres, "Efficient decomposition of image and mesh graphs by lifted multicuts," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1751–1759.

[8] T. Beier, C. Pape, N. Rahaman, T. Prange, S. Berg, D. D. Bock, A. Cardona, G. W. Knott, S. M. Plaza, L. K. Scheffer, U. Koethe, A. Kreshuk, and F. A. Hamprecht, "Multicut brings automated neurite segmentation closer to human performance," *Nature Methods*, vol. 14, pp. 101–102, 2017.

[9] S. Wolf, L. Schott, U. Kothe, and F. Hamprecht, "Learned watershed: End-to-end learning of seeded segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2030–2038.

[10] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. Amer. Math. Society*, vol. 7, no. 1, pp. 48–50, 1956.

[11] L. P. Coelho, "Mahotas: Open source software for scriptable computer vision," *J. Open Res. Softw.*, vol. 1, no. 1, 2013, Art. no. e3.

[12] J. Funke, J. Klein, F. Moreno-Noguer, A. Cardona, and M. Cook, "TED: A tolerant edit distance for segmentation evaluation," *Methods*, vol. 115, pp. 119–127, 2017.

[13] J. Funke, E. Perlman, S. Turaga, D. Bock, and S. Saalfeld, Cremi Challenge Leaderboard, as of 2017/22/09. (2016). [Online]. Available: https://cremi.org/leaderboard

[14] S.-Y. Takemura, C. S. Xu, Z. Lu, P. K. Rivlin, T. Parag, D. J. Olbris, S. Plaza, T. Zhao, W. T. Katz, L. Umayam, et al., "Synaptic circuits and their variations within different columns in the visual system of drosophila," *Proc. Nat. Acad. Sci.*, vol. 112, no. 44, pp. 13 711–13 716, 2015.

[15] M. Berning, K. M. Boergens, and M. Helmstaedter, "Segem: Efficient image analysis for high-resolution connectomics," *Neuron*, vol. 87, no. 6, pp. 1193–1206, 2015.

[16] S. Saalfeld, R. Fetter, A. Cardona, and P. Tomancak, "Elastic volume reconstruction from series of ultra-thin microscopy sections," *Nature Methods*, vol. 9, no. 7, pp. 717–720, 2012.

[17] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980, 2014.

[18] A. Zlateski and H. S. Seung, "Image segmentation by size-dependent single linkage clustering of a watershed basin graph," *CoRR*, vol. abs/1505.00249, 2015. [Online]. Available: http://arxiv.org/abs/1505.00249

[19] M. Januszewski, J. Maitin-Shepard, P. Li, J. Kornfeld, W. Denk, and V. Jain, "Flood-filling networks," *CoRR*, vol. abs/1611.00421, 2016. [Online]. Available: http://arxiv.org/abs/1611.00421

Jan Funke received the diplom (MS) degree in computer science from the Technical University Dresden, in 2010, and the PhD degree in computer science from ETH Zurich, in 2014. Until 2018, he was a post-doctoral fellow with the Institute of Neuroinformatics, a joint institute of ETH Zurich and University of Zurich, and later at the Institut Robotica i Informatica Industrial at UPC, Barcelona. In June 2018, Jan Funke joined the Janelia Research Campus of the Howard Hughes Medical Institute in Ashburn (VA) as a group leader. His research interests include computer vision and machine learning methods for the automatic analysis of microscopy images.

Fabian Tschopp received the BSc degree in computer science from ETH, Zrich, Switzerland, in 2016, and the MSc degree in neural systems and computation from the Institute of Neuroinformatics, University of Zrich. He is an external collaborator with the Howard Hughes Institute, Janelia, and at Advanced Micro Devices since 2015. His research interests include deep learning, 3D graphics, high-performance computing and connectomics.
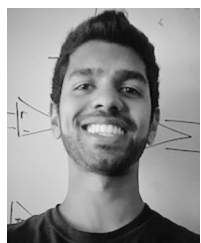
William Grisaitis received the BA in physics from Duke University, Durham (NC), in 2012, where he also worked as an undergraduate research fellow. After working at Capital One Bank in McLean (VA) as a Business and later Senior Data Analyst between 2013 and 2016, William Grisaitis joined Srini Turaga's group at the Janelia Research Campus of the Howard Hughes Medical Institute as a Senior Software Engineer.

Arlo Sheridan received the BA degree in neuroscience from Knox College, IL, in 2016. He then joined the Janelia Research Campus of the Howard Hughes Medical Institute to help create the first large scale, collaborative, connectome annotation team (CAT) from the ground up. He worked directly for Vivek Jayaraman on the underlying intrinsic circuits implicated in spatial navigation and sensorimotor learning/memory within the drosophila central complex. In 2017, he joined the Turaga and Saalfeld labs at Janelia to assist with algorithm analysis. His research interests include circuit neuroscience and 2d/3d visualization.

Chandan Singh received the engineering degree in computer science and mathematics from the University of Virginia, in 2017. He is working toward the PhD degree at the University of California, Berkeley, under the supervision of Prof. Bin Yu. His research interests include statistical neuroscience and interpreting machine learning models.

Stephan Saalfeld received the diplom (MS) degree in computer science and media, and the PhD degree in computer science from the Technische Universitt Dresden, Germany, in 2008 and 2013, respectively. For his PhD from 2008 to 2013, and as a postdoctoral fellow in 2013, he worked with Pavel Tomancak at the Max Planck Institute of Molecular Cell Biology and Genetics in Dresden, Germany. Since 2013, he is a group leader with the Janelia Research Campus of the Howard Hughes Medical Institute in Ashburn, VA. His research addresses scalable image analysis and software design.

Srinivas C. Turaga received the PhD degree from MIT, in 2009. He is a group leader with the Janelia Research Campus of the Howard Hughes Medical Institute. He was previously a postdoctoral fellow with the Gatsby Unit at University College London. His research interests include machine learning and computational neuroscience, with a special focus on connectomics.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.