arXiv:1709.02974v1 [cs.CV] 9 Sep 2017

A Deep Structured Learning Approach Towards Automating Connectome Reconstruction from 3D Electron Micrographs

Jan Funke⋆
Institut de Robòtica i Informàtica Industrial
UPC/CSIC Barcelona
jfunke@iri.upc.edu

Fabian David Tschopp⋆
Institute of Neuroinformatics
University of Zurich and ETH Zurich
tschopfa@student.ethz.ch

William Grisaitis
Janelia Research Campus
Howard Hughes Medical Instutute
grisaitisw@janelia.hhmi.org

Chandan Singh
Janelia Research Campus
Howard Hughes Medical Instutute
chandan_singh@berkeley.edu

Stephan Saalfeld
Janelia Research Campus
Howard Hughes Medical Instutute
saalfelds@janelia.hhmi.org

Srinivas C. Turaga
Janelia Research Campus
Howard Hughes Medical Instutute
turagas@janelia.hhmi.org

Abstract

We present a deep learning method for neuron segmentation from 3D electron microscopy (EM), which improves significantly upon state of the art in terms of accuracy and scalability. Our method consists of a fully 3D extension of the U-NET architecture, trained to predict affinity graphs on voxels, followed by a simple and efficient iterative region agglomeration. We train the U-NET using a structured loss function based on MALIS that encourages topological correctness. The resulting affinity predictions are accurate enough that we obtain state-of-the-art results by a simple new learning-free percentile-based iterative agglomeration algorithm. We demonstrate the accuracy of our method on three different and diverse EM datasets where we significantly improve over the current state of the art. We also show for the first time that a common 3D segmentation strategy can be applied to both well-aligned nearly isotropic block-face EM data, and poorly aligned anisotropic serial sectioned EM data. The runtime of our method scales with $O(n)$ in the size of the volume and is thus ready to be applied to very large datasets.

1 Introduction

Precise reconstruction of neural connectivity is of great importance for the deep understanding of the function of biological nervous systems. 3D electron microscopy (EM) are the only available imaging methods 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 structured deep learning based image segmentation method for reconstructing neurons from 3D electron microscopy, which improves significantly upon state of the art in terms of accuracy

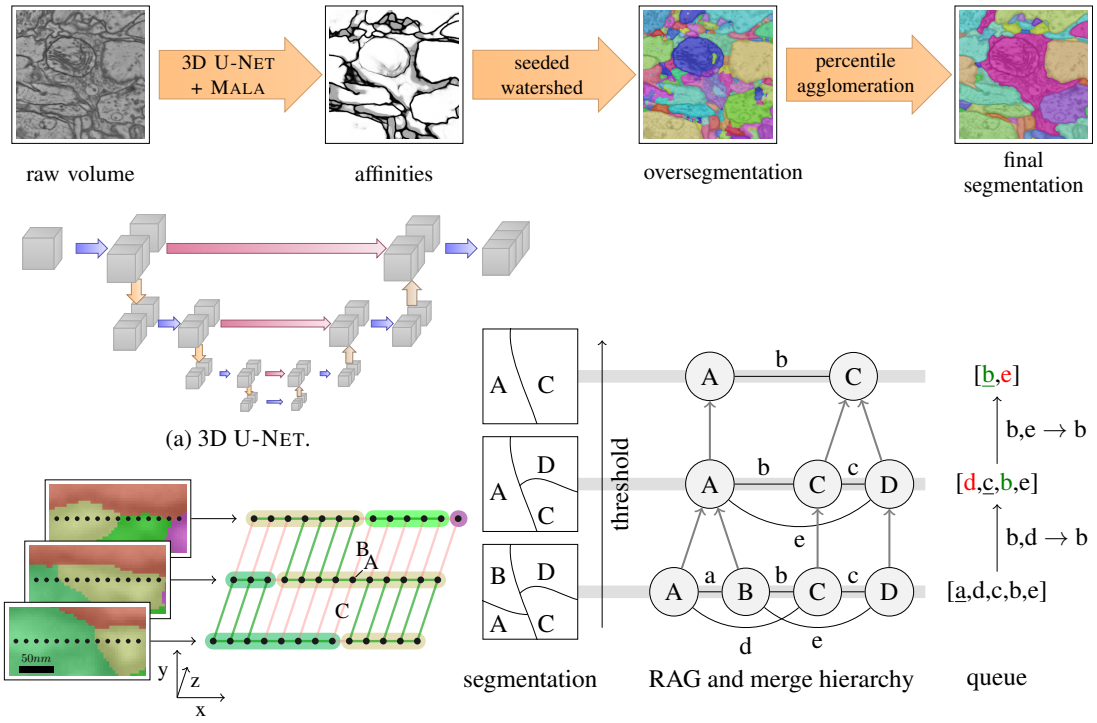


Figure 1: Overview of our method (top row). Using a fully 3D U-NET (a), trained with the proposed constrained MALIS loss, we directly predict inter-voxel affinities from volumes of raw data. Affinities provide advantages especially in the case of low-resolution data (b). In the example shown here, there would be no way to correctly label the *voxels* as foreground/background: If A was labelled as foreground, it would necessarily merge with with the regions in the previous and next section. If it was labelled as background, it would introduce a split. The labelling of affinities on *edges*, on the other hand, allows B and C to separate A from the adjacent sections, while maintaining connectivity inside the region. We obtain an oversegmentation from the predicted affinities, which is then merged into the final segmentation using a percentile-based agglomeration algorithm (c).

and scalability. The main contributions in our work are: (1) A fully 3D extension of the U-NET architecture [1] for the prediction of affinity graphs, (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 biomedical image data [1]. 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 constraint 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 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. 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. This is a crucial property for neuron segmentation from EM volumes, where volumes easily reach several hundreds of terabytes. The efficiency of our method is in contrast to current state-of-the-art methods, which all follow a similar pattern. First, voxel-wise predictions are made using a deep neural network. Subsequently, fragments are obtained from these predictions, which is then merged using either greedy (CELIS [3], GALA [4]) or globally optimal objectives (MULTICUT [5] and lifted MULTICUT [6, 7]). Current efforts focus mostly on the merging of fragments: Both CELIS and GALA train a classifier to predict scores for hierarchical agglomeration, which increase the computational complexity of agglomeration during inference. The MULTICUT variants also train a separate classifier to predict connectivity of fragments, which are then clustered by solving a computationally expensive combinatorial optimization problem. The fragment agglomeration method we propose here does drastically reduce 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 (FIB-SEM, *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.

2 Method

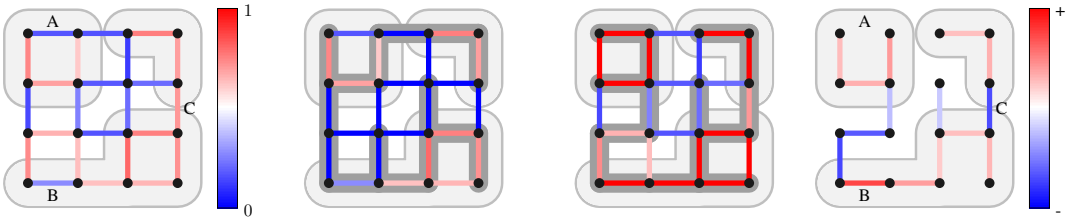
2.1 Deep multi-scale convolutional network for predicting 3D voxel affinities

We generalized the U-NET architecture [1] for predict voxel affinities on 3D volumes. We use the same architecture for all 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 $3\times3\times3$) 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 a transposed convolution of the same size for the upsampling (brown arrows). The results of the upsampling pass are further concatenated with copies of the original feature maps of the same level (red arrows), cropped to account for context loss in the lower level. A more detailed description, as well as the design choices for each of the investigated datasets, can be found in the supplement.

We chose to predict voxel affinities on edges between voxels instead of labelling voxels into 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 labelling 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



(a) Predicted affinities. (b) Positive pass. (c) Negative pass. (d) Gradient of loss.

Figure 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 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. In this example, note that 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 for each edge affinity is shown in (d) (positive values in red, negative in blue).

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 $O(n \log(n))$ method for the computation of the gradient (thus improving over previous $O(n^2)$. 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.

A maximin edge between two voxels u and v is an edge $\mathrm{mm}(u, v)$ with lowest affinity on the overall highest affinity path $P^*_{u,v}$ connecting u and v, *i.e.*,

$$P^*_{u,v} = \operatorname*{arg\,max}_{P \in \mathcal{P}_{u,v}} \min_{(i,j) \in P} A_{i,j} \qquad \mathrm{mm}(u, v) = \operatorname*{arg\,min}_{(i,j) \in P^*_{u,v}} A_{i,j}, \tag{1}$$

where $\mathcal{P}_{u,v}$ denotes the set of all paths between u and v and A_e the predicted affinity of edge e. If we imagine a simple thresholding on the affinity graph, such that edges with affinities below a threshold θ are removed, 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. Taking advantage of the importance of maximin edges, the MALIS loss favours high maximin affinities between nodes with the same label, and low otherwise:

$$l(A) = \sum_{u<v} \left(\delta_{\mathrm{id}(u)\,\mathrm{id}(v)} - A_{\mathrm{mm}(u,v)}\right)^2, \tag{2}$$

where δ is the Kronecker delta and $\mathrm{id}(u)$ is the ground-truth label of voxel u. It can easily be seen that maximin edges are shared between voxel pairs. In fact, the union of all maximin edges forms a maximal spanning tree (MST),

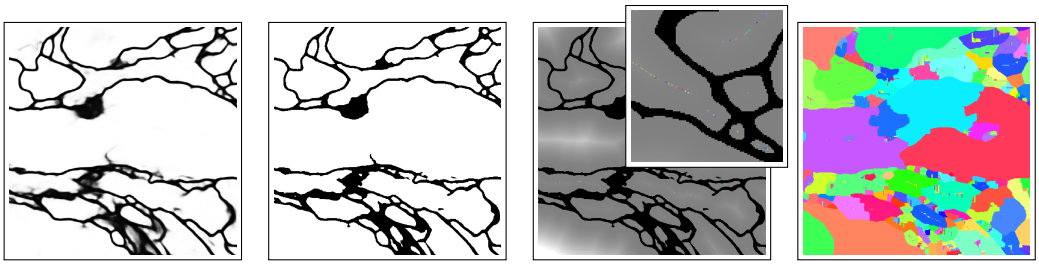
$$\bigcup_{u,v} \mathrm{mm}(u, v) = \mathrm{MST}(A). \tag{3}$$

Consequently, we are able to identify the maximin edge and compute its loss for each voxel pair in $O(n \log(n))$ time. We thus improve over a previous method [2] that required $O(n^2)$ and thus 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.

We further extend previous work by computing the maximin edge losses in two passes: First for edges within the same region (positive pass), second for edges between different regions (negative pass). As shown in Fig. 2, in the positive pass we assume that all edges between regions have correctly been predicted 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.

2.3 Hierarchical agglomeration



averaged affinities | threshold at 0.5 | distance transform | seeded watershed

Figure 3: Illustration of the seeded watershed heuristic.

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

2.3.1 Fragment extraction

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

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 maxima is taken as a seed, from which we grow basins using a standard watershed algorithm [8] on the boundary predictions. For an example, 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 prediction 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 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 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

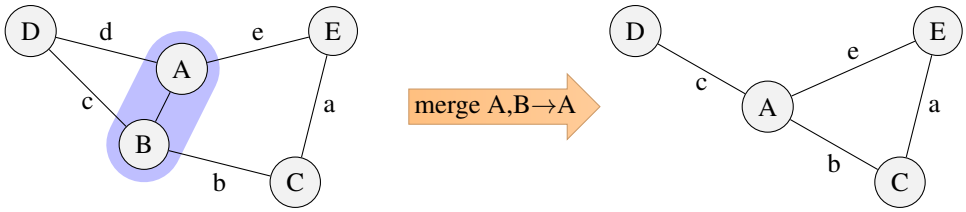


Figure 4: Illustration of the three different edge update cases during a merge: (1) a is not affected by the merge operation, (2) b and e link one of the merge nodes to a neighbor, which is not adjacent to the other merge node, and (c) c and d link one of the merge nodes to a node D, which is adjacent to both A and B. c and d thus get merged into one edge and its score needs to be updated.

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 by updated accordingly. Acknowledging the merge hierarchy of edges (as opposed to nodes), we will refer to the leafs 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, 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 $O(n)$. As we show in Section 3, with $k = 256$ bins we notice no sacrifice 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. 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 edges $f \in E_{i-1}$ into $e \in E_{i-1}$ is always greater equal its previous score. We thus just mark e as *stale* and proceed merging. 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 deleted as a consequence of a nearby merge.

3 Results

We present results on three different and diverse datasets: CREMI[1], FIB-25 [9], and SEGEM [10] (see Table 1 for an overview). These datasets sum up to almost 15 gigavoxels of testing data, with

[1] `https://cremi.org`

Name	Imaging	Tissue	Resolution	Training Data	Testing Data
CREMI	ssTEM	*Drosophila*	4×4×40 nm	3 volumes of 1250×1250×125	3 volumes of 1250×1250×125
FIB-25	FIBSEM	*Drosophila*	8×8×8 nm	520×520×520	13.8 gigavoxels
SEGEM	SBEM	mouse cortex	11×11×26 nm	279 volumes of 100×100×100	400×400×350 (skeletons)

Table 1: Overview of used datasets.

method	VOI split	VOI merge	VOI sum
U-NET MALA	1.060	0.229	**1.288**
U-NET	1.421	0.358	1.778
FlyEM [9]	1.619	0.495	2.114
CELIS [3]	1.533	0.344	1.877
CELIS+MC [3]	1.169	0.347	1.516

(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 [9]	3.160	0.251	3.411
CELIS [3]	3.401	0.166	3.568
CELIS+MC [3]	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 [7]	0.597	0.272	0.868	0.398
CRunet	1.081	0.389	1.470	0.566
LFC	1.085	0.140	1.225	0.616

(c) Results on CREMI.

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 [10]	2.121	3.951	1.380

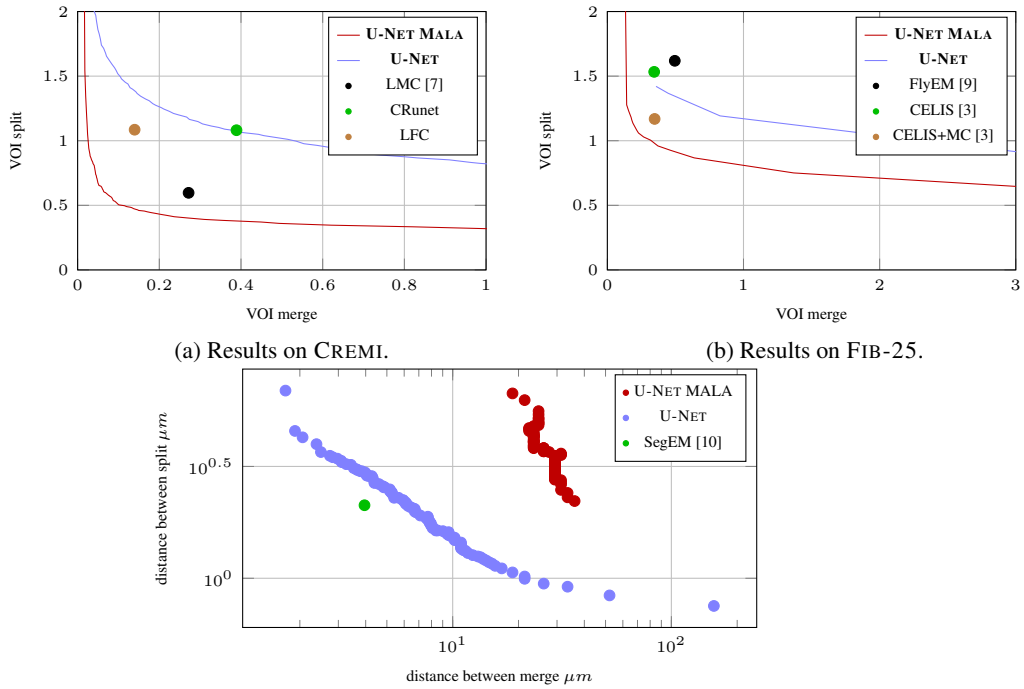
(d) Results on SEGEM.

Table 2: 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). 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), adapted RAND error (ARAND, lower is better), CREMI score (geometric mean of VOI and ARAND, 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 [11] with a distance threshold of $52nm$ (corresponding to the thickness of two z-sections). CREMI results are reported as average over all testing samples, individual results can be found in the supplement.

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 [9, 3]. Another challenge is posed by the CREMI dataset: Coming from serial section EM, this dataset is highly anisotropic, contains registration errors, and 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.*, 213×213×29 for CREMI, 89×89×89 for FIB-25, and 89×89×49 for SEGEM. For the CREMI dataset, we also pre-aligned training and testing data with an elastic alignment method [12], using the padded volumes provided by the challenge.

On each of the investigated datasets, we see a significant 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 (labelled U-NET MALA) against different other methods. We also include a baseline (labelled U-NET) in our analysis, which is our method, but trained without the constrained MALIS loss. In Table 2, we report the segmentation obtained on the best threshold found in the respective training datasets. In Fig. 5, we show the split/merge curve for varying thresholds on CREMI and FIB-25.

Note that for SEGEM, we do not use the metric proposed by Berning et al. [10], 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 completely ignoring many 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, rendering the



(a) Results on CREMI. (b) Results on FIB-25.

(c) Results on SEGEM.

Figure 5: 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).

	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 [13]	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 [13]	1.054	1.017	2.071

(b) FIB-25.

Table 3: Results for different merge functions of our method compared with the agglomeration strategy proposed in [13]. 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 2.

results incomparable. Therefore, we performed a new IED evaluation using TED [11], a metric that allows slight displacement of skeleton nodes (we chose $52nm$ in this case) in an attempt to minimize splits and merges. This metric reveals (see Fig. 5c) that our segmentations (U-NET MALA) improve over both split and merge errors, over all thresholds of agglomeration (this includes our initial supervoxels).

U-NET MALA generalizes to block-face and serial sectioned datasets Our method improves the accuracy on both near-isotropic block-face datasets (FIB-25, SEGEM) as well as on highly anisotropic serial-section datasets (CREMI). With that, we eliminate the need for specialized constructions like dedicated features for anisotropic volumes or separate classifiers trained for merging of fragments within or across sections. Furthermore, the presence of artifacts in the CREMI dataset (missing sections, support film folds, and staining precipitations).

Merge function 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,

dataset	U-NET [s/Mv]	watershed [s/Mv]	agglomeration [s/Mv]	total [s/Mv]
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

Table 4: Throughput of our method for each of the investigated datasets in seconds per megavoxel.

and an agglomeration baseline proposed in [13] on datasets CREMI and FIB-25. Even across these very different datasets, we see best results for affinity quantiles between 50% and 75%.

Throughput Table 4 shows the throughput of our method for each dataset, broken down into affinity prediction (U-NET), fragment extraction (watershed), and fragment agglomeration (agglomeration). For CREMI and SEGEM, most time is spent in the prediction of affinities. The faster predictions in FIB-25 are due to less feature maps used in the network for this dataset.

4 Discussion

A remarkable property of our method is that it almost requires no tuning to operate on datasets of different characteristics, except for trivial changes in the size of the receptive field of the U-NET. With this contribution, we see no further need for the development of dedicated algorithms for different EM modalities. Our results indicate that affinity predictions on voxels can be sufficiently accurate across different datasets to not make a difference for fragment extraction and agglomeration in subsequent processing. It remains an open question whether fundamentally different approaches, like the recently reported flood-filling network [14], also generalize in a similar way. At the time of writing, neither code nor data are 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 in contrast to current state-of-the-art methods, which require careful splitting of precious training data into non-overlapping sets used to train voxel-wise predictions and to train a fragment merge classifier (or accepting the disadvantages of having the sets overlap).

Although linear in runtime and memory, parallelization of our method is not trivial due to the hierarchical agglomeration. However, we can take advantage of the fact that scores of merged edges only increase (as discussed in Section 2.3.2): This allows to split a given volume into regions, where within a region agglomeration can be performed independently, as long as the score does not exceed any boundary score to a neighboring region.

References

[1] Ronneberger, O., Fischer, P. & Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 234–241 (Springer, 2015).

[2] Briggman, K., Denk, W., Seung, S., Helmstaedter, M. N. & Turaga, S. C. Maximin affinity learning of image segmentation. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I. & Culotta, A. (eds.) *Advances in Neural Information Processing Systems 22*, 1865–1873 (Curran Associates, Inc., 2009).

[3] Maitin-Shepard, J. B., Jain, V., Januszewski, M., Li, P. & Abbeel, P. Combinatorial energy learning for image segmentation. In *Advances in Neural Information Processing Systems 29*, 1966–1974 (Curran Associates, Inc., 2016).

[4] Nunez-Iglesias, J., Kennedy, R., Plaza, S. M., Chakraborty, A. & Katz, W. T. Graph-based active learning of agglomeration (gala): a python library to segment 2d and 3d neuroimages. *Frontiers in neuroinformatics* **8**, 34 (2014).

[5] Andres, B. *et al.* Globally optimal closed-surface segmentation for connectomics. In *European Conference on Computer Vision*, 778–791 (Springer, 2012).

[6] Keuper, M. *et al.* Efficient decomposition of image and mesh graphs by lifted multicuts. In *The IEEE International Conference on Computer Vision (ICCV)* (2015).

[7] Beier, T. *et al.* Multicut brings automated neurite segmentation closer to human performance. *Nature Methods* **14**, 101–102 (2017).

[8] Coelho, L. P. Mahotas: Open source software for scriptable computer vision. *Journal of Open Research Software* **1(1):e3** (2013).

[9] Takemura, S.-y. *et al.* Synaptic circuits and their variations within different columns in the visual system of drosophila. *Proceedings of the National Academy of Sciences* **112**, 13711–13716 (2015).

[10] Berning, M., Boergens, K. M. & Helmstaedter, M. Segem: efficient image analysis for high-resolution connectomics. *Neuron* **87**, 1193–1206 (2015).

[11] Funke, J., Klein, J., Moreno-Noguer, F., Cardona, A. & Cook, M. Ted: A tolerant edit distance for segmentation evaluation. *Methods* **115**, 119–127 (2017).

[12] Saalfeld, S., Fetter, R., Cardona, A. & Tomancak, P. Elastic volume reconstruction from series of ultra-thin microscopy sections. *Nature methods* **9**, 717–720 (2012).

[13] Zlateski, A. & Seung, H. S. Image segmentation by size-dependent single linkage clustering of a watershed basin graph. *CoRR* **abs/1505.00249** (2015). URL `http://arxiv.org/abs/1505.00249`.

[14] Januszewski, M. *et al.* Flood-filling networks. *CoRR* **abs/1611.00421** (2016). URL `http://arxiv.org/abs/1611.00421`.

A U-NET architectures

Details of the used U-NET architectures can be found in Fig. 6, where we provide detailed schematics of the architectures used for the different datasets. The passes (convolution, max pooling, upsampling, and copy-crop) used in the U-NET are detailed in Fig. 7.

B Additional results

Testing results on the individual CREMI samples A+, B+, and C+ can be found in Fig. 8c.

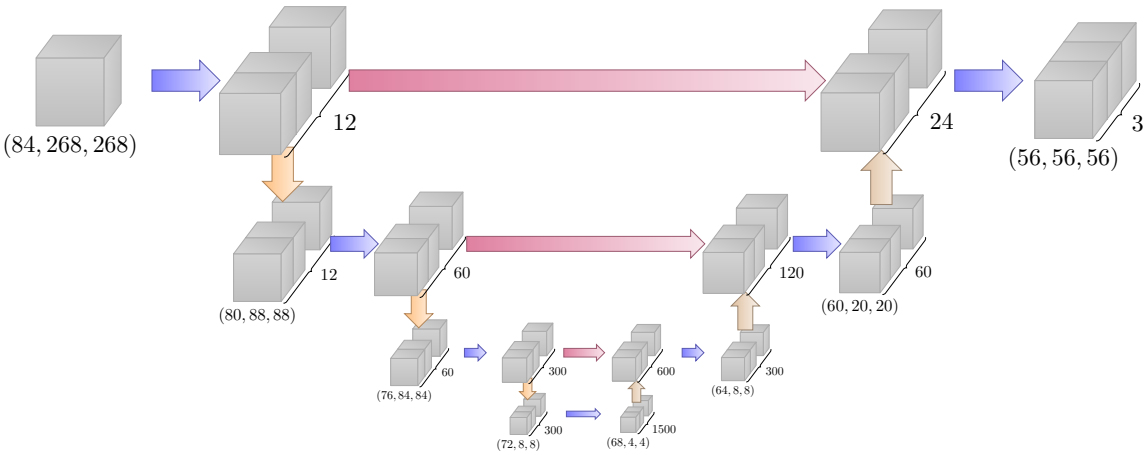


Figure 6: Overview of the U-net architecture used for the CREMI dataset. The architectures for FIB-25 and SEGEM are the same, with changes in the input and output sizes which are: in= $(132, 132, 132)$, out= $(44, 44, 44)$ for FIB-25 and in= $(188, 188, 144)$, out= $(100, 100, 96)$ for SEGEM.

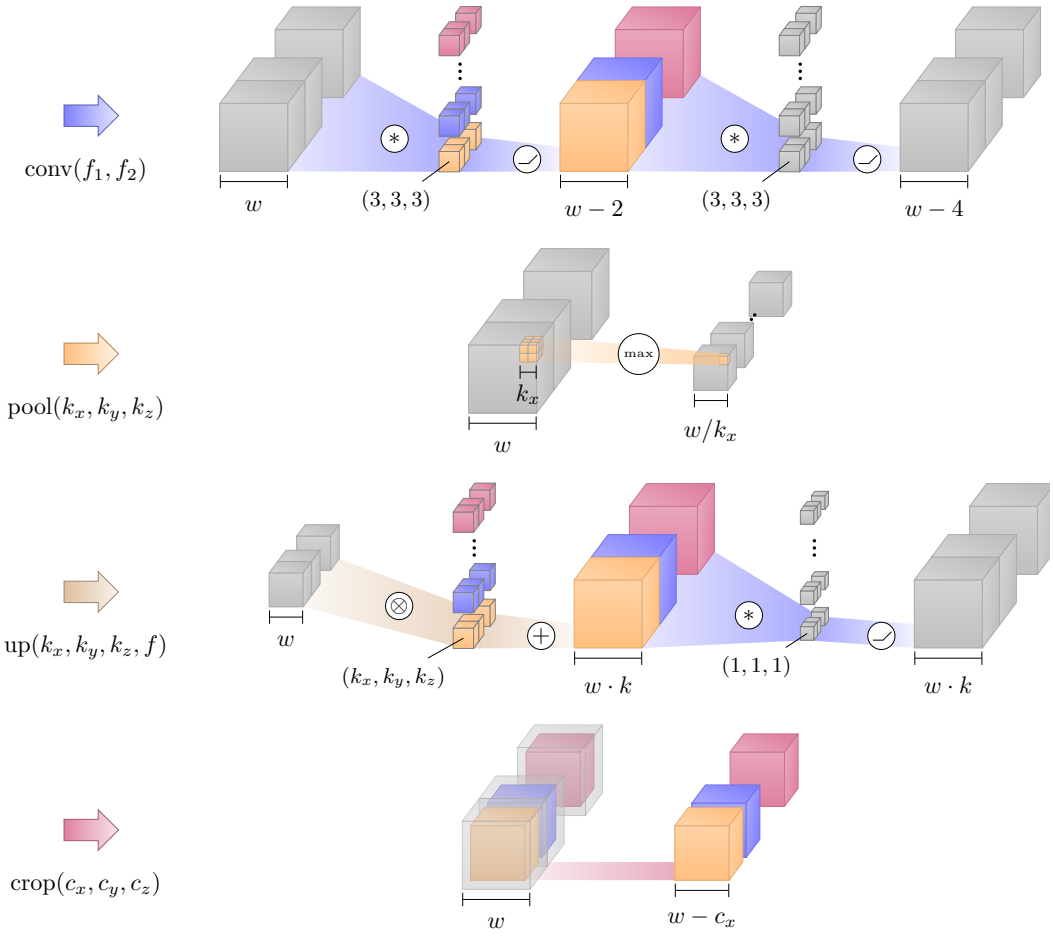
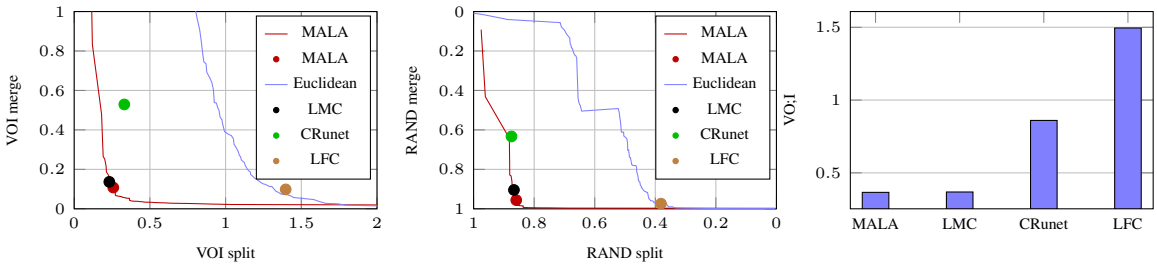
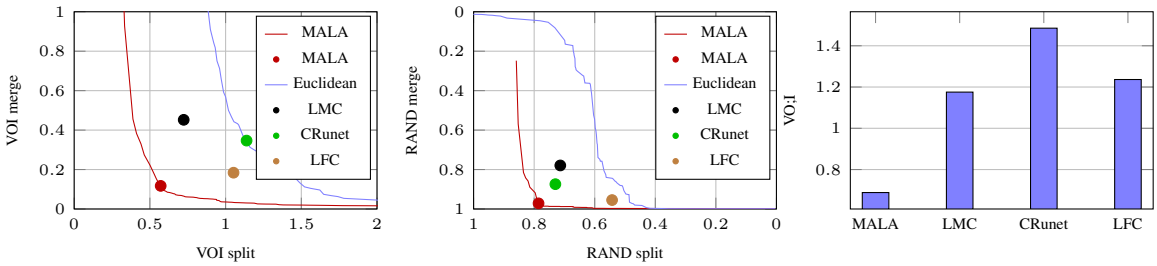


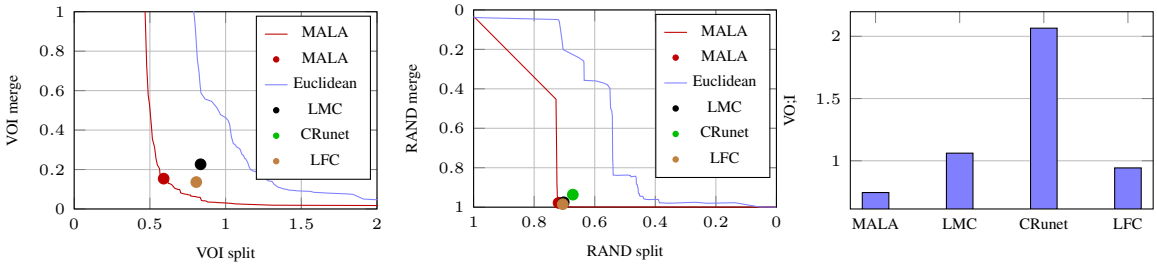
Figure 7: Details of the convolution (blue), max-pooling (orange), upsampling (brown), and copy-crop operations (purple). "$*$" denotes a convolution, "$_\!\!\diagup$" a rectified linear unit, and "$\otimes$" the Kronecker matrix product.



(a) CREMI sample A+



(b) CREMI sample B+



(c) CREMI sample C+

Figure 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 *Euclidean* is our method, but without MALIS training (*i.e.*, only minimizing the Euclidean distance to the ground-truth affinities during training).