Self-Supervised Edge Detection Reconstruction for Topology-Informed 3D Axon Segmentation and Centerline Detection

Alec S. Xu Nina I. Shamsi Lars A. Gjesteby Laura J. Brattain

MIT Lincoln Laboratory, Lexington, MA 02421, USA
alecx@umich.edu
{nina.shamsi, lars.gjesteby, brattainl}@ll.mit.edu

Abstract

Many machine learning-based axon tracing methods rely on image datasets with segmentation labels. This requires manual annotation from domain experts, which is labor-intensive and not practical for large-scale brain mapping on hemisphere or whole brain tissue at cellular or sub-cellular resolution. Additionally, preserving axon structure topology is crucial to understanding neural connections and brain function. Self-supervised learning (SSL) is a machine learning framework that allows models to learn an auxiliary task on unannotated data to aid performance on a supervised target task. In this work, we propose a novel SSL auxiliary task of reconstructing an edge detector for the target task of topology-oriented axon segmentation and centerline detection. We pretrained 3D U-Nets on three different SSL tasks using a mouse brain dataset: our proposed task, predicting the order of permuted slices, and playing a Rubik's cube. We then evaluated these U-Nets and a baseline model on a different mouse brain dataset. Across all experiments, the U-Net pretrained on our proposed task improved the baseline's segmentation, topology-preservation, and centerline detection by up to 5.03%, 4.65%, and 5.41%, respectively. In contrast, there was no consistent improvement over the baseline observed with the slice-permutation and Rubik's cube pretrained U-Nets.

1. Introduction

Reconstructing neuron connections can help researchers better understand the relationship between brain structure and its function. Domain experts can accurately trace the connections manually, but this approach cannot practically scale to the billions of connections needed to map a single human brain. Therefore, it is crucial to develop automatic large-scale axon tracing methods that are as accurate as human abilities, and preserve the topology of the axon structures for brain connectivity analysis.

Deep learning methods have achieved state-of-the-art results on several biomedical image segmentation tasks. The U-Net architecture [2] in particular surpassed the human performance threshold in the 2017 SNEMI3D Connectomics Challenge [6]. Previous works [8, 12, 16, 18] have explored supervised 3D U-Net approaches for axon tracing through image segmentation. However, these supervised approaches rely on massive image datasets that are manually traced. The manual tracing process is a time and labor-intensive task and often leads to incomplete annotations.

Self-supervised learning (SSL) is a machine learning paradigm that can leverage large amounts of unannotated data. In this framework, a model is first trained on an *SSL auxiliary task* using data that are not human annotated. Any “annotations" needed for an SSL auxiliary task can be generated during SSL training time using only the data itself. Afterwards, the SSL-pretrained model is fine-tuned on a supervised *target task* (or *downstream task*) on a small set of annotated data. Common auxiliary tasks for 3D medical imaging problems include contrastive predictive coding [15], rotation prediction [15], patch location prediction [15], slice ordering [20], and playing a Rubik's cube [17, 21].

For the target task of 3D axon segmentation and centerline detection, previous works explored the SSL task of classifying the permutation used to shuffle the z-slices of a 3D image [5, 13]. However, the benefits of this SSL task are maximized only when the SSL training input size is larger than the segmentation input size, and after the target task loss function's hyperparameters are manually tuned. To our knowledge, no other auxiliary tasks have been investigated for this target task.

For the particular downstream task of topology-informed 3D axon segmentation and centerline detection, we propose a simple yet effective SSL auxiliary task: reconstructing the output of the Canny edge detector [1]. We empirically demonstrate that a 3D U-Net pretrained with this SSL task consistently achieves better segmentation, topology preservation, and centerline detection metrics than a baseline model (*i.e*. a 3D U-Net not pretrained on any SSL task) when **(1)** the SSL training dataset is different from the target task dataset, and **(2)** the SSL training and target task input sizes are equal. Meanwhile, 3D U-Nets pretrained on the slice permutation and Rubik's cube tasks often perform worse than the baseline under the same conditions. We also illustrate a 3D U-Net pretrained on our proposed task is generally more robust to the target task loss function hyperparameters than 3D U-Nets pretrained on the slice permutation and Rubik's cube tasks, which can reduce the hyperparameter tuning search space.

2. Related Work

2.1. Topologically-Aware 3D U-Nets

Pollack *et al*. [12] investigated two 3D U-Net approaches for axon segmentation, where the baseline method was a U-Net trained with voxel-wise cross-entropy loss. The first approach, called CasNet, cascaded two U-Nets to simultaneously obtain segmentation and centerline predictions. The CasNet was trained on a modified Dice loss [10]. The second approach was a 3D U-Net trained on clDice loss [14], which optimizes connections of tubular and curvilinear structures, such as axons, as well as emphasizes topology preservation:

$$\mathcal{L}_{\text{clDice}} = (1-\alpha)(1-\text{soft-Dice}) + \alpha(1-\text{soft-clDice}). \quad (1)$$

Computing the soft-clDice metric requires an iterative soft-skeletonization algorithm, where the number of iterations is a hyperparameter k. Generally, k should be chosen such that it is greater than or equal to the maximum pixel radius of the tubular structures in the given dataset. Otherwise, the skeletonization may be incomplete [14]. Note that the choice of k does not affect the $(1 - \text{soft-Dice})$ term. Likewise, α is another hyperparameter that weighs the soft-Dice and soft-clDice components, where $0 < \alpha < 1$.

Pollack *et al*. showed that the topologically-aware 3D U-Net with clDice loss typically achieved improved segmentation, topology preservation, and centerline detection metrics compared to other methods. Thus, in our work, we also use clDice loss for downstream task training.

2.2. Slice-Permutation Classification

Self-Supervised Feature Extraction for 3D Axon Segmentation. Klinghoffer *et al*. [5] investigated slice-permutation classification as an SSL task for axon segmentation. During SSL training, they split the input image into 8 even slices along the z-axis, and shuffled their order based on one of 8 randomly chosen permutations. They trained a 3D U-Net encoder and auxiliary classifier to classify the permutation applied to the volume using an information-weighted cross-entropy loss, $\mathcal{L}_{ICE}$. At a high-level, the encoder learns where along the z-axis the axon paths are broken due to the shuffled slices. Voxel-wise binary cross-entropy loss was used for target task training.

Self-Supervised Learning to Improve Topology-Optimized Axon Segmentation and Centerline Detection. Shamsi *et al*. [13] applied the slice-permutation auxiliary task to topologically-aware 3D U-Nets and used the following SSL objective:

$$\mathcal{L}_{SSL} = \beta\mathcal{L}_{ICE} + (1-\beta)\mathcal{L}_R, \quad (2)$$

where $\mathcal{L}_R$ is the ℓ_2 loss between the decoder output and the original, non-shuffled image, with $0 < \beta < 1$. They trained on clDice loss (1) [14] for the target task. Their baseline model was a 3D U-Net trained on clDice loss from scratch, *i.e*. without any SSL training.

In their work, they showed the benefits of pretraining on the slice-permutation task are highly dependent on the choices of α, k, and the input size during SSL training. For instance, when $\alpha = 0.5$ and $k = 8$, their SSL-pretrained model achieved better performance metrics than the baseline model only when the SSL input size was larger than the target task input size. When the two sizes were equal, the baseline outperformed the SSL-pretrained model. However, when $\alpha = 0.8$ and $k = 5$, the SSL input size had very little effect on the SSL-pretrained model's target task performance. This suggests it is necessary to manually tune α and k, as well as use a larger input size during SSL training, to realize the benefits of this auxiliary task.

In our work, we empirically show that when a 3D U-Net is pretrained with our proposed SSL task, its downstream performance is less sensitive to α and k, and can improve the baseline even when using smaller SSL input sizes.

2.3. Playing a Rubik's Cube

Playing a Rubik's cube [21] requires splitting each training sample into N evenly sized sub-volumes, resulting in $P = N!$ total possible permutations. To guarantee no two permutations are highly similar, the $K \ll P$ permutations with the largest Hamming distances from each other are chosen to be kept. During training, one of K permutations is chosen randomly and applied to the input image. An encoder and an auxiliary classifier are trained to classify which permutation was applied to the image.

No previous work on 3D axon segmentation has explored pretraining on this auxiliary task, but it is closely related to the slice-permutation task. In our work, we pretrained a 3D U-Net to play a Rubik's cube for axon segmentation as one of the competing approaches against our proposed method. The encoder again learns where the axon paths are broken, but now across all three dimensions instead.

3. Proposed Method

For the specific target task of topology-oriented 3D axon segmentation and centerline detection, we propose the SSL auxiliary task of reconstructing the output of the Canny edge detection method [1]. During each SSL training epoch, we apply the Canny method to each slice along the z-axis of each raw input image. For each raw image, the Canny method returns a binary 3D image indicating which voxels are edges. The binary image serves as the "annotations" for SSL training. We train a Residual 3D U-Net to reconstruct this binary image at the output of the decoder.

Compared to the original raw input image, the Canny method output provides a sharper contrast between the axons and the background. By learning to reconstruct this output, we hypothesize the U-Net can better localize the axon paths.

Once trained on the auxiliary task, the entire U-Net is fine-tuned on axon segmentation and centerline detection in a supervised manner. The encoder is initialized using the resulting weights from auxiliary task training, and the decoder is initialized randomly. Figure 1 illustrates the end-to-end training pipeline. See the supplementary material for notes about our proposed method's computational complexity.

4. Experiments

We trained and evaluated four types of 3D Residual U-Nets for axon segmentation in a supervised manner. The baseline model (**BL U-Net**) was not pretrained on any SSL task. The other three models were pretrained on slice permutation classification (**SP U-Net**), Rubik's cube permutation classification (**RC U-Net**), and Canny edge detector reconstruction (**EDGE U-Net**, our proposed method). We aimed to evaluate how the choice of SSL auxiliary task *alone* affects downstream performance. Thus, the only difference between all of the methods are the SSL pretraining conditions.

4.1. Training and Implementation

We developed our implementation using PyTorch. Our 3D Residual U-Net consisted of four resolution blocks, each with a $3 \times 3 \times 3$ convolution layer followed by ELU activation and group normalization [19]. We used strided $2 \times 2 \times 2$ max-pooling to downsample between encoder blocks, and strided transpose convolutions with max pooling to upsample between decoder blocks.

To create the model input images, we randomly cropped the original data volume, and then applied random flips and $90°$ $X - Y$ plane rotations as augmentations.

For both auxiliary and target task training, we used the ADAM optimizer [4] with a cosine-annealing learning rate multiplier [9]. The initial learning rate was 10^{-4} and weight decay was 10^{-3}. We trained all of our models on an Intel Xeon G6 node with 2 NVIDIA Volta V100 GPUs.

Edge Detection Reconstruction. To create the edge detection output, we set the Canny method's Gaussian blur standard deviation σ to 1.5, the low threshold to 10% of the entire SSL training volume's maximum voxel intensity, and the high threshold to 20% of the maximum voxel intensity. We trained on ℓ_2 loss.

Slice Permutation Classification. We split each training sample into 8 evenly-sized slices along the z-axis and applied one of 8 permutations, chosen uniformly at random. Our auxiliary classifier consisted of a $3 \times 3 \times 3$ convolutional layer, ELU activation [3], group normalization, and a linear layer. We trained on the loss function (2) with $\beta = 0.8$.

Rubik's Cube Permutation Classification. We split each training sample into 8 evenly-sized sub-volumes and applied one of 8 permutations, chosen uniformly at random. Our auxiliary classifier consisted of four blocks, each with a $3 \times 3 \times 3$ convolutional layer, Leaky-ReLU, and batch normalization. The output of the last block was passed into a linear layer. We again trained on (2) with $\beta = 0.8$.

Target Task. We used clDice loss (1) for downstream training. The values of α and k we used are described in Section 4.3. The U-Net outputs the predicted segmentation. To obtain the centerline predictions, we applied a 3D skeletonization method [7] to the segmentation predictions.

4.2. Datasets

We used Dataset 1 for all SSL training, and Dataset 2 for all target task training and testing.

Dataset 1. As previously described in [5] and [12], Dataset 1 consists of mouse brain tissue. Under $3\times$ expansion, the tissue was stained using calretinin antibody immunostaining, prepared with the SHIELD technique [11], and imaged using a Lifecanvas SMARTSPIM light-sheet imager. The image contains areas of globus pallidus externa, globus pallidus interna substantia nigra reticulata, and subthalamic nucleus. The full volume is $2048 \times 2048 \times 1024$ voxels with a voxel resolution of $0.65 \times 0.65 \times 2\mu$m. We pre-processed the data by clipping the highest and lowest 0.01% of the values, applying a median filter, and then scaling the voxels to lie in the range $[0, 1]$.

Dataset 2. As previously described in [13], the data were acquired using a Leica confocal microscope with $20\times$ magnification from mouse thalamus sections labeled via coritcal injection with recombinant adeno-associated virus expressing tdTomato (red) and synaptophysin (green). We used the tdTomato channel after converting it to grayscale. The full volume is $4096 \times 4096 \times 52$ voxels with a voxel resolution of $0.14 \times 0.14 \times 2\mu$m, We partitioned the volume into training, validation, and testing sets using a 50:25:25 split, respectively. We applied the same pre-processing steps on this dataset as we did on Dataset 1 [12].

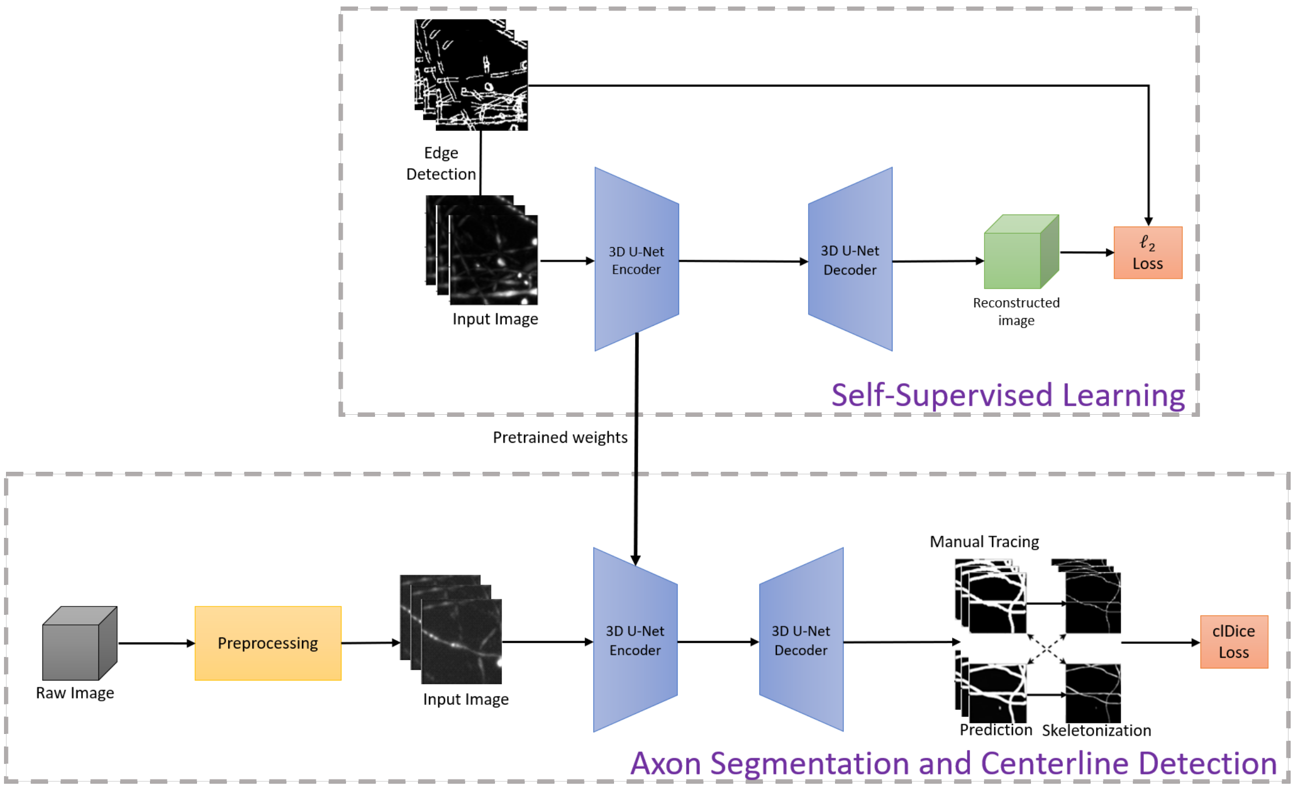

Figure 1. **Training pipeline using edge detection reconstruction SSL auxiliary task.** Top: edge detection-based SSL. The pretrained encoder is used as the initial weights for the downstream task. Bottom: topology-oriented axon segmentation and centerline detection.

Table 1. **Mean and standard deviation evaluated on the test set of Dataset 2 across 10 trials when trained at various k with $\alpha = 0.5$.** SSL input size = $128 \times 128 \times 32$ and target task input size = $128 \times 128 \times 32$.

Metric	Model	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$
Dice	BL	0.756 ± 0.025	0.763 ± 0.024	0.769 ± 0.021	0.764 ± 0.025	0.769 ± 0.029
	SP	0.736 ± 0.026	0.713 ± 0.023	0.729 ± 0.028	0.725 ± 0.015	0.743 ± 0.019
	RC	0.728 ± 0.008	0.752 ± 0.021	0.731 ± 0.010	0.747 ± 0.021	0.730 ± 0.008
	EDGE	$\mathbf{0.794 \pm 0.010}$	$\mathbf{0.774 \pm 0.025}$	$\mathbf{0.785 \pm 0.019}$	$\mathbf{0.786 \pm 0.020}$	$\mathbf{0.793 \pm 0.014}$
clDice	BL	0.690 ± 0.015	0.675 ± 0.014	0.679 ± 0.021	0.667 ± 0.025	0.668 ± 0.054
	SP	0.624 ± 0.026	0.596 ± 0.035	0.646 ± 0.042	0.651 ± 0.044	0.656 ± 0.024
	RC	0.649 ± 0.033	0.657 ± 0.023	0.657 ± 0.041	0.654 ± 0.022	0.644 ± 0.014
	EDGE	$\mathbf{0.699 \pm 0.021}$	$\mathbf{0.679 \pm 0.027}$	$\mathbf{0.691 \pm 0.030}$	$\mathbf{0.692 \pm 0.027}$	$\mathbf{0.700 \pm 0.028}$
ρ-Dice	BL	0.769 ± 0.016	0.756 ± 0.019	0.765 ± 0.018	0.752 ± 0.029	0.754 ± 0.054
	SP	0.695 ± 0.029	0.660 ± 0.039	0.714 ± 0.043	0.716 ± 0.049	0.731 ± 0.027
	RC	0.725 ± 0.036	0.745 ± 0.024	0.735 ± 0.045	0.740 ± 0.030	0.724 ± 0.022
	EDGE	$\mathbf{0.787 \pm 0.023}$	$\mathbf{0.764 \pm 0.033}$	$\mathbf{0.779 \pm 0.034}$	$\mathbf{0.781 \pm 0.031}$	$\mathbf{0.788 \pm 0.032}$

4.3. clDice Loss Hyperparameter Sweeps

As discussed in Section 2.2, Shamsi *et al.* [13] demonstrated the importance of choosing proper α-k values to observe the benefits of pretraining on the slice-permutation auxiliary task. To determine how sensitive each model type's downstream performance is to α and k, we used various α-k combinations during target task training.

Table 2. **Mean and standard deviation evaluated on the test set of Dataset 2 across 10 trials when trained at various α with $k = 5$.** SSL input size = $128 \times 128 \times 32$ and target task input size = $128 \times 128 \times 32$.

Metric	Model	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$
Dice	BL	0.783 ± 0.013	0.778 ± 0.018	0.782 ± 0.011	0.791 ± 0.013	0.769 ± 0.021
	SP	0.752 ± 0.016	0.768 ± 0.023	0.774 ± 0.007	0.749 ± 0.023	0.729 ± 0.028
	RC	0.767 ± 0.010	0.768 ± 0.019	0.784 ± 0.009	0.782 ± 0.012	0.731 ± 0.010
	EDGE	$\mathbf{0.791 \pm 0.009}$	$\mathbf{0.796 \pm 0.008}$	$\mathbf{0.795 \pm 0.004}$	$\mathbf{0.794 \pm 0.009}$	$\mathbf{0.785 \pm 0.019}$
clDice	BL	0.678 ± 0.024	0.665 ± 0.035	0.673 ± 0.023	0.696 ± 0.025	0.679 ± 0.021
	SP	0.604 ± 0.036	0.642 ± 0.051	0.653 ± 0.014	0.601 ± 0.047	0.646 ± 0.042
	RC	0.646 ± 0.019	0.650 ± 0.040	0.686 ± 0.018	0.680 ± 0.028	0.657 ± 0.041
	EDGE	$\mathbf{0.691 \pm 0.020}$	$\mathbf{0.701 \pm 0.016}$	$\mathbf{0.702 \pm 0.007}$	$\mathbf{0.700 \pm 0.019}$	$\mathbf{0.691 \pm 0.030}$
ρ-Dice	BL	0.768 ± 0.024	0.753 ± 0.033	0.763 ± 0.024	0.781 ± 0.027	0.765 ± 0.018
	SP	0.679 ± 0.038	0.720 ± 0.058	0.733 ± 0.016	0.678 ± 0.047	0.714 ± 0.043
	RC	0.731 ± 0.025	0.740 ± 0.042	0.771 ± 0.014	0.764 ± 0.030	0.735 ± 0.045
	EDGE	$\mathbf{0.780 \pm 0.023}$	$\mathbf{0.788 \pm 0.019}$	$\mathbf{0.789 \pm 0.007}$	$\mathbf{0.793 \pm 0.020}$	$\mathbf{0.779 \pm 0.034}$

Table 3. **Mean and standard deviation evaluated on the test set of Dataset 2 when trained at various k with $\alpha = 0.5$.** SSL input size = $192 \times 192 \times 48$ and target task input size = $128 \times 128 \times 32$.

Metric	Model	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$
Dice	BL	0.756 ± 0.025	0.763 ± 0.024	0.769 ± 0.021	0.764 ± 0.025	0.769 ± 0.029
	SP	0.734 ± 0.024	0.723 ± 0.003	$\mathbf{0.770 \pm 0.025}$	0.732 ± 0.019	0.751 ± 0.028
	RC	$\mathbf{0.779 \pm 0.019}$	0.762 ± 0.023	$\mathbf{0.770 \pm 0.022}$	0.763 ± 0.023	$\mathbf{0.784 \pm 0.018}$
	EDGE	0.775 ± 0.018	$\mathbf{0.786 \pm 0.017}$	0.759 ± 0.035	$\mathbf{0.775 \pm 0.023}$	0.781 ± 0.021
clDice	BL	$\mathbf{0.690 \pm 0.015}$	0.675 ± 0.014	0.679 ± 0.021	0.667 ± 0.025	0.668 ± 0.054
	SP	0.660 ± 0.015	0.650 ± 0.011	0.668 ± 0.015	0.664 ± 0.009	0.662 ± 0.017
	RC	0.677 ± 0.023	0.681 ± 0.010	0.678 ± 0.009	0.681 ± 0.010	$\mathbf{0.694 \pm 0.009}$
	EDGE	0.685 ± 0.035	$\mathbf{0.684 \pm 0.035}$	$\mathbf{0.681 \pm 0.033}$	$\mathbf{0.698 \pm 0.023}$	0.682 ± 0.041
ρ-Dice	BL	$\mathbf{0.769 \pm 0.016}$	0.756 ± 0.019	$\mathbf{0.765 \pm 0.018}$	0.752 ± 0.029	0.754 ± 0.054
	SP	0.738 ± 0.014	0.725 ± 0.015	0.748 ± 0.017	0.737 ± 0.012	0.739 ± 0.015
	RC	0.765 ± 0.027	$\mathbf{0.772 \pm 0.011}$	0.764 ± 0.010	0.769 ± 0.011	$\mathbf{0.778 \pm 0.013}$
	EDGE	0.764 ± 0.037	0.765 ± 0.040	0.759 ± 0.035	$\mathbf{0.779 \pm 0.023}$	0.762 ± 0.045

For each of the four model types, each trial consisted of training on the target task using a particular α-k combination. We first kept $\alpha = 0.5$ constant, and swept through $k = 3, 4, 5, 6,$ and 7. We then swept through $\alpha = 0.1, 0.2, 0.3$ and 0.4 while keeping $k = 5$ constant. We conducted 10 trials for each model type and α-k combination, and then we evaluated the models on the test set.

4.4. SSL Training Input Size

To realize the benefits of the slice permutation task, the input size during SSL training should be larger than the input size during segmentation training, otherwise the performance gains are minimal [5] [13]. However, larger SSL input sizes increase the time and memory needed for SSL training. To determine if this size requirement is necessary for the Rubik's cube task and our proposed method, we kept the target task input size at $128 \times 128 \times 32$, while using two SSL input sizes: $128 \times 128 \times 32$ and $192 \times 192 \times 48$.

4.5. Evaluation Metrics

We recorded the mean and standard deviation of each model's Dice, clDice [14], and ρ-Dice metrics [12] across the 10 trials for each α-k pair. The Dice coefficient measures segmentation performance via voxel-wise similarity, clDice measures topology preservation, and ρ-Dice measures accuracy of centerline detections.

Table 4. **Mean and standard deviation evaluated on the test set of Dataset 2 when trained at various α with $k = 5$.** SSL input size = $192 \times 192 \times 48$ and target task input size = $128 \times 128 \times 32$.

Metric	**Model**	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$
Dice	BL	0.783 ± 0.013	0.778 ± 0.018	0.782 ± 0.011	0.791 ± 0.013	0.769 ± 0.021
	SP	0.792 ± 0.007	0.764 ± 0.012	0.757 ± 0.011	0.767 ± 0.008	$\mathbf{0.770 \pm 0.025}$
	RC	0.758 ± 0.008	0.773 ± 0.019	0.794 ± 0.003	0.788 ± 0.002	$\mathbf{0.770 \pm 0.022}$
	EDGE	$\mathbf{0.795 \pm 0.011}$	$\mathbf{0.788 \pm 0.014}$	$\mathbf{0.796 \pm 0.006}$	$\mathbf{0.794 \pm 0.007}$	0.759 ± 0.035
clDice	BL	0.678 ± 0.024	0.665 ± 0.035	0.673 ± 0.023	0.696 ± 0.025	0.679 ± 0.021
	SP	0.688 ± 0.017	0.630 ± 0.024	0.613 ± 0.023	0.636 ± 0.018	0.668 ± 0.015
	RC	0.617 ± 0.016	0.651 ± 0.043	0.699 ± 0.007	0.689 ± 0.004	0.678 ± 0.009
	EDGE	$\mathbf{0.700 \pm 0.026}$	$\mathbf{0.682 \pm 0.031}$	$\mathbf{0.705 \pm 0.015}$	$\mathbf{0.701 \pm 0.016}$	$\mathbf{0.681 \pm 0.033}$
ρ-Dice	BL	0.768 ± 0.024	0.753 ± 0.033	0.763 ± 0.024	0.781 ± 0.027	$\mathbf{0.765 \pm 0.018}$
	SP	0.774 ± 0.017	0.708 ± 0.029	0.706 ± 0.025	0.725 ± 0.015	0.748 ± 0.017
	RC	0.697 ± 0.018	0.729 ± 0.046	0.783 ± 0.009	0.774 ± 0.005	0.764 ± 0.010
	EDGE	$\mathbf{0.784 \pm 0.026}$	$\mathbf{0.768 \pm 0.030}$	$\mathbf{0.789 \pm 0.017}$	$\mathbf{0.784 \pm 0.016}$	0.759 ± 0.035

For each model type, we also computed the standard deviation of their mean Dice, clDice, and ρ-Dice scores across k, as well as across α. We did this to measure each model type's sensitivity to the clDice loss hyperparameters.

Table 5. **Standard deviations across k of the average target task metrics (lower is better)**

SSL Input Size	**Model**	Dice	clDice	ρ-Dice
$128 \times 128 \times 32$	BL	$\mathbf{0.0048}$	0.0084	$\mathbf{0.0066}$
	SP	0.0102	0.0222	0.0244
	RC	0.0099	$\mathbf{0.0050}$	0.0082
	EDGE	0.0072	0.0075	0.0086
$192 \times 192 \times 48$	BL	$\mathbf{0.0048}$	0.0084	0.0066
	SP	0.0167	$\mathbf{0.0060}$	0.0073
	RC	0.0087	0.0061	$\mathbf{0.0051}$
	EDGE	0.0091	0.0062	0.0069

Table 6. **Standard deviations across α of the average target task metrics (lower is better)**

SSL Input Size	**Model**	Dice	clDice	ρ-Dice
$128 \times 128 \times 32$	BL	0.0072	0.0102	$\mathbf{0.0090}$
	SP	0.0158	0.0221	0.0224
	RC	0.0190	0.0162	0.0162
	EDGE	$\mathbf{0.0040}$	$\mathbf{0.0049}$	$\mathbf{0.0090}$
$192 \times 192 \times 48$	BL	$\mathbf{0.0072}$	$\mathbf{0.0102}$	$\mathbf{0.0090}$
	SP	0.0118	0.0272	0.0258
	RC	0.0129	0.0296	0.0320
	EDGE	0.0140	$\mathbf{0.0102}$	0.0114

5. Results

Tables 1 and 2 show each model's Dice, clDice, and ρ-Dice score mean and standard deviations on Dataset 2's test set when the SSL input size was $128 \times 128 \times 32$. Tables 3 and 4 provide the same results, but with an SSL input size of $192 \times 192 \times 48$. In Tables 2 and 4, we also include results for $\alpha = 0.5$, but note that these are duplicates of the $k = 5$ results from Tables 1 and 3, respectively. In Table 5, we provide the standard deviation of the mean Dice, clDice, and ρ-Dice scores across k, (*i.e.*, across the rows of Tables 1 and 3). Table 6 provides the same results, but across α instead. In all of the tables, we **bold** the best metrics. Figure 2 shows examples of segmentation and centerline predictions from the different model types after applying a maximum intensity projection along the z-axis.

6. Discussion

When SSL pretrained on Dataset 1, our proposed EDGE U-Net outperformed the baseline model and SP and RC U-Nets. We hypothesize that the background artifacts in Dataset 1 skewed the feature representations the SP and RC U-Nets learned during SSL training, which impacted their downstream performances on Dataset 2. However, the EDGE U-Net reconstructed images with binary voxel values, which presumably reduced the impact of the background artifacts in its learned feature representations. We believe this helped its downstream performance on Dataset 2.

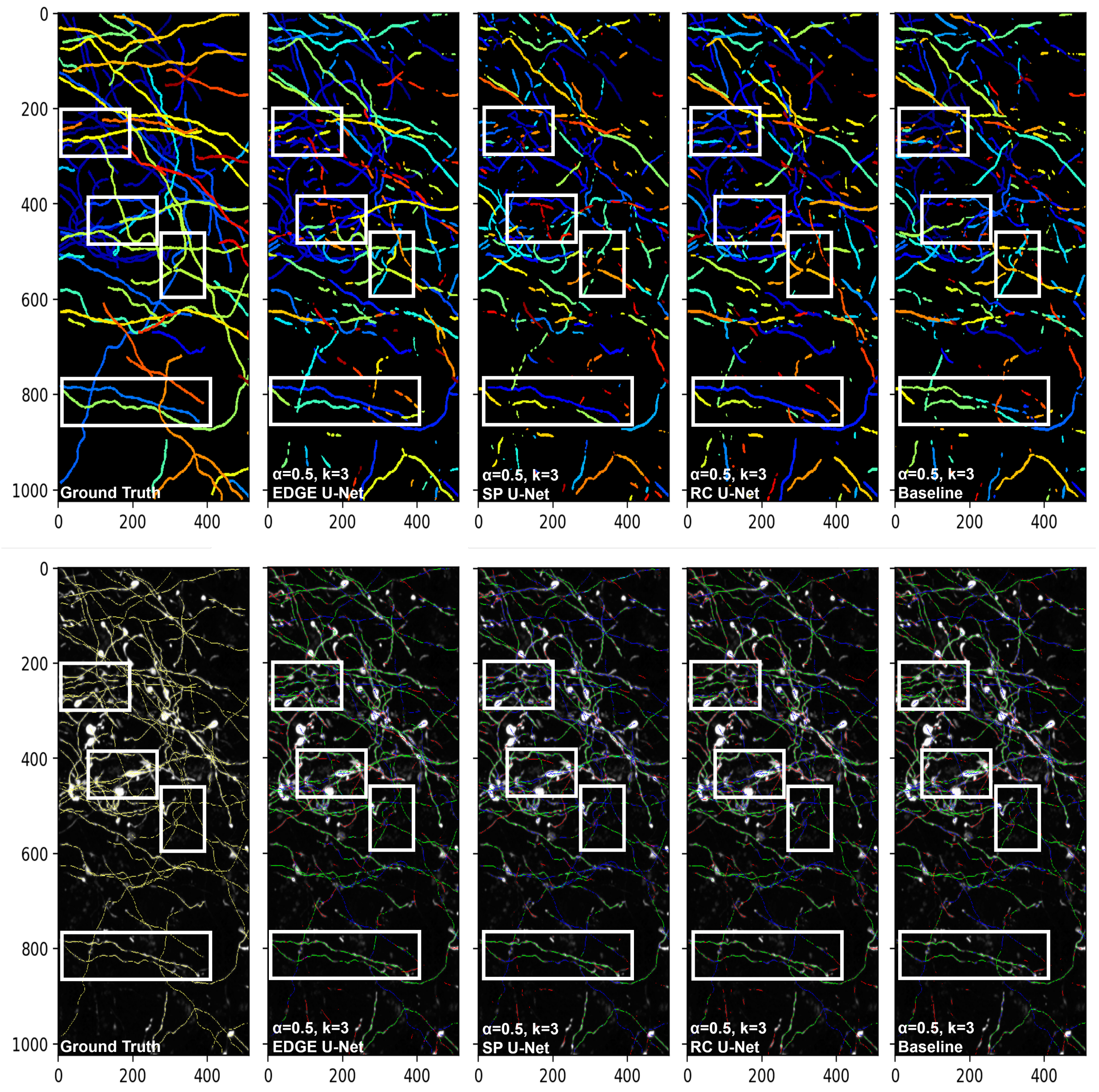

Figure 2. **Connected component (top) and centerline prediction (bottom) examples for $\alpha = 0.5$, $k = 3$, and SSL input size = $128 \times 128 \times 32$.** In the centerline predictions (bottom), green markings indicate true positive, red is false positive, and blue is false negative. The white boxes indicate regions where the four models provided different predictions. In these regions, our proposed model generally had more contiguous connected components and fewer false negative centerline predictions compared to the other models.

6.1. SSL Training Input Size

$\mathbf{128 \times 128 \times 32}$. Tables 1 and 2 show the EDGE U-Net always performed better than the baseline across all of the α-k pairs we used, while the SP and RC U-Nets almost always performed worse than the baseline. Our proposed method's mean Dice, clDice, and ρ-Dice scores were, on average, 2.21%, 2.71%, and 2.75% higher than the baseline's, respectively. Meanwhile, the SP U-Net's mean Dice, clDice, and ρ-Dice scores were 3.83%, 6.81%, and 7.77% lower than those of the baseline on average, and the RC U-Net's mean metrics were 2.40%, 2.75%, and 2.71% lower on average.

Even when the SSL the target task input sizes were equal, our approach improved on the baseline's target task metrics, while the other SSL pretrained models had trouble doing so. Thus, if a shorter SSL training time or smaller memory usage is desired, pretraining on our proposed approach can still result in downstream performance gains, while pretraining on the other two SSL tasks would likely not.

$\mathbf{192} \times \mathbf{192} \times \mathbf{48}$. The baseline model outperformed the EDGE U-Net in only four instances out of 27 total: in clDice and ρ-Dice when $\alpha = 0.5$, $k = 3$, as well as in Dice and ρ-Dice when $\alpha = 0.5$, $k = 5$. In contrast, the baseline outperformed the SP U-Net in 23 out of 27 instances, and the RC U-Net in 15 out of 27 instances. On average, across all of the α-k pairs we used, our proposed method's mean Dice, clDice, and ρ-Dice scores were 1.36%, 2.10%, and 1.36% higher than those of the baseline, respectively. Meanwhile, the SP U-Net's mean Dice, clDice, and ρ-Dice scores were 2.38%, 3.59%, and 3.80% lower than those of the baseline on average. The RC U-Net's mean Dice score was 0.24% higher than the baseline's, while its mean clDice and ρ-Dice scores were 0.38% and 0.42% lower.

At the larger SSL input size, the benefits of the slice permutation task should be maximized [5, 13]. Our results seem to indicate this is true for the Rubik's cube task as well, although no previous work has shown this. Regardless, out of all of the SSL pretrained models, our EDGE U-Net frequently achieved the highest target task metrics, and the greatest average improvements over the baseline.

It is worth noting that when the SSL input size increased, our EDGE U-Net's downstream performance worsened, while the SP and RC U-Nets' performances both improved. This may indicate with a large enough SSL input size, the SP and RC U-Nets would eventually consistently outperform the baseline and EDGE U-Net on the downstream task.

6.2. clDice Loss Hyperparameter Sensitivity

Sweeping through k with $\alpha = 0.5$. Table 5 shows the SP U-Net had the largest average performance spread across k among all of the models when the SSL input size was $128 \times 128 \times 32$. When the SSL input size was $192 \times 192 \times 48$, the SP U-Net's average Dice score standard deviation across k was the highest as well. The EDGE and RC U-Nets' average metric standard deviations across k were mostly very similar. Overall, our proposed method appears to be less sensitive to the choice of k compared to the SP U-Net, and similarly sensitive to the choice of k as the RC U-Net.

Sweeping through α with $k = 5$. Table 6 shows at both SSL input sizes, our EDGE U-Net's average metric standard deviations across α were almost always lower than those of the other two SSL pretrained models. The lone exception is in the mean Dice score when the SSL input size was $192 \times 192 \times 48$. However, at that SSL input size, our proposed method's average Dice score was still remarkably consistent from $\alpha = 0.1$ to $\alpha = 0.4$.

The SP and RC U-Nets appear more sensitive to the choice of α than our EDGE U-Net. Additionally, Table 4 shows the SP and RC U-Nets achieved similar or better metrics than the baseline at some values of α, but noticeably worse metrics at other values. Thus, to maximize the benefits of slice permutation and Rubik's cube tasks, it is likely necessary to tune α with a large search space during downstream training. Meanwhile, our proposed method performed more consistently *and* almost always achieved better metrics than the baseline across different α values. The benefits of our proposed SSL task are evident at almost all values of α we used. While some tuning of α may still be beneficial, we believe the search space can be greatly reduced.

7. Conclusion

In this work, we showed reconstructing the output of the Canny edge detection method is an effective SSL auxiliary task for topology-oriented axon segmentation and centerline detection. A Residual 3D U-Net pretrained on our proposed task consistently achieved higher downstream metrics than the baseline model, even when different data was used for SSL and target task training, and when the SSL input size was equal to the target task input size. Meanwhile, under the same conditions, 3D U-Nets pretrained on the slice permutation and Rubik's cube tasks often performed worse than the baseline. Finally, a 3D U-Net pretrained on our proposed method appears to be more robust to the choice of clDice loss hyperparameter values compared to 3D U-Nets pretrained on the slice permutation and Rubik's cube tasks.

7.1. Future Work

One possible next step is to investigate these SSL approaches on larger datasets across other organisms. In addition, it is not obvious how the Canny edge detection parameters should be set during SSL training, and downstream performance likely depends on these parameter choices. Investigating the effect of these parameters on downstream performance, or having a 3D U-Net reconstruct the output of a parameter-free edge detection or skeletonization method as an SSL task, are other possibilities for future work.

Additionally, as discussed in Section 6, we hypothesize that the noise and background artifacts in Dataset 1 caused the SP and RC U-Nets to perform worse than the baseline on the target task on Dataset 2. Performing edge detection or skeletonization on the input images before permuting the slices or sub-volumes during SSL training might mitigate these effects, thus improving downstream performance.

8. Acknowledgements

The authors would like to acknowledge the laboratory of Kwanghun Chung at MIT for providing data, and MIT Lincoln Laboratory Supercomputing Center for their support of high performance computing tasks. This material is based upon work supported by the National Institutes of Health (NIH) U01MH117072, NIMH 1R43MH128076-01, and MBF Bioscience. Any opinions, conclusions, or recommendations in this material are those of the author(s) and do not necessarily reflect the views of NIH/NIMH.

References

[1] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. 2, 3

[2] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432. Springer, 2016. 1

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 3

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[5] Tzofi Klinghoffer, Peter Morales, Young-Gyun Park, Nicholas Evans, Kwanghun Chung, and Laura J Brattain. Self-supervised feature extraction for 3d axon segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 978–979, 2020. 1, 2, 3, 5, 8

[6] Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H Sebastian Seung. Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017. 1

[7] Ta-Chih Lee, Rangasami L Kashyap, and Chong-Nam Chu. Building skeleton models via 3-d medial surface axis thinning algorithms. *CVGIP: graphical models and image processing*, 56(6):462–478, 1994. 3

[8] Qiufu Li and Linlin Shen. 3d neuron reconstruction in tangled neuronal image with deep networks. *IEEE transactions on medical imaging*, 39(2):425–435, 2019. 1

[9] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 3

[10] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016. 2

[11] Young-Gyun Park, Chang Ho Sohn, Ritchie Chen, Margaret McCue, Dae Hee Yun, Gabrielle T Drummond, Taeyun Ku, Nicholas B Evans, Hayeon Caitlyn Oak, Wendy Trieu, et al. Protection of tissue physicochemical properties using polyfunctional crosslinkers. *Nature biotechnology*, 37(1):73–83, 2019. 3

[12] Dylan Pollack, Lars A Gjesteby, Michael Snyder, David Chavez, Lee Kamentsky, Kwanghun Chung, and Laura J Brattain. Axon tracing and centerline detection using topologically-aware 3d u-nets. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 238–242. IEEE, 2022. 1, 2, 3, 5

[13] Nina I Shamsi, Lars A Gjesteby, David Chavez, Michael Snyder, Brian S Eastwood, Matthew G Fay, Nathan J O'Connor, Jack R Glaser, Charles R Gerfen, and Laura J Brattain. Self-supervised learning to improve topology-optimized axon segmentation and centerline detection. In *2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI)*, pages 1–4. IEEE, 2023. 1, 2, 3, 4, 5, 8

[14] Suprosanna Shit, Johannes C Paetzold, Anjany Sekuboyina, Ivan Ezhov, Alexander Unger, Andrey Zhylka, Josien PW Pluim, Ulrich Bauer, and Bjoern H Menze. 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. 2, 5

[15] Aiham Taleb, Winfried Loetzsch, Noel Danz, Julius Severin, Thomas Gaertner, Benjamin Bergner, and Christoph Lippert. 3d self-supervised methods for medical imaging. *Advances in neural information processing systems*, 33:18158–18172, 2020. 1

[16] Wenbo Tang, Huiwen Huang, Yongyi Gong, Fang Li, Ji Li, and Xiaonan Luo. 3d neuron segmentation based on 3d dsac u-net. In *2020 8th International Conference on Digital Home (ICDH)*, pages 322–326. IEEE, 2020. 1

[17] Xing Tao, Yuexiang Li, Wenhui Zhou, Kai Ma, and Yefeng Zheng. Revisiting rubik's cube: self-supervised learning with volume-wise transformation for 3d medical image segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part IV 23*, pages 238–248. Springer, 2020. 1

[18] Donglai Wei, Kisuk Lee, Hanyu Li, Ran Lu, J Alexander Bae, Zequan Liu, Lifu Zhang, Márcia dos Santos, Zudi Lin, Thomas Uram, et al. Axonem dataset: 3d axon instance segmentation of brain cortical regions. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part I 24*, pages 175–185. Springer, 2021. 1

[19] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 3

[20] Pengyue Zhang, Fusheng Wang, and Yefeng Zheng. Self supervised deep representation learning for fine-grained body part recognition. In *2017 IEEE 14th international symposium on biomedical imaging (ISBI 2017)*, pages 578–582. IEEE, 2017. 1

[21] Xinrui Zhuang, Yuexiang Li, Yifan Hu, Kai Ma, Yujiu Yang, and Yefeng Zheng. Self-supervised feature learning for 3d medical images by playing a rubik's cube. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part IV 22*, pages 420–428. Springer, 2019. 1, 2