JMI Journal of Medical Imaging

RESEARCH PAPER

Medical volume segmentation by overfitting sparsely annotated data

Tristan Payer[a,*] Faraz Nizamani,[a] Meinrad Beer,[b] Michael Götz[b] and Timo Ropinski[a]

[a]Ulm University, Institute of Media Informatics, Visual Computing Group, Ulm, Germany
[b]University Hospital Ulm, Radiology Department, Ulm, Germany

ABSTRACT. **Purpose:** Semantic segmentation is one of the most significant tasks in medical image computing, whereby deep neural networks have shown great success. Unfortunately, supervised approaches are very data-intensive, and obtaining reliable annotations is time-consuming and expensive. Sparsely labeled approaches, such as bounding boxes, have shown some success in reducing the annotation time. However, in 3D volume data, each slice must still be manually labeled.

Approach: We evaluate approaches that reduce the annotation effort by reducing the number of slices that need to be labeled in a 3D volume. In a two-step process, a similarity metric is used to select slices that should be annotated by a trained radiologist. In the second step, a predictor is used to predict the segmentation mask for the rest of the slices. We evaluate different combinations of selectors and predictors on medical CT and MRI volumes. Thus we can determine that combination works best, and how far slice annotations can be reduced.

Results: Our results show that for instance for the Medical Segmentation Decathlon —heart dataset, some selector, and predictor combinations allow for a Dice score 0.969 when only annotating 20% of slices per volume. Experiments on other datasets show a similarly positive trend.

Conclusions: We evaluate a method that supports experts during the labeling of 3D medical volumes. Our approach makes it possible to drastically reduce the number of slices that need to be manually labeled. We present a recommendation in which selector predictor combination to use for different tasks and goals.

 [DOI: 10.1117/1.JMI.10.4.044007]

Keywords: image segmentation; data labeling; active learning; neural networks; artificial intelligence

Paper 22307GRR received Nov. 4, 2022; revised Jul. 14, 2023; accepted Jul. 31, 2023; published Aug. 17, 2023.

1 Introduction

Many state-of-the-art techniques for medical segmentation are supervised approaches, which require large amounts of annotated images to train neural networks. Especially when focusing on semantic segmentation, annotation is very time-consuming, as target structures in images have to be labeled with pixel-level precision. The situation is even more challenging when dealing with 3D volume data, which consists of many image slices to be annotated.[1] Naturally, such supervised approaches pose severe challenges in disciplines, such as medicine, as data need to be labeled by experts,[2] sometimes even redundantly.

*Address all correspondence to Tristan Payer, tristan.payer@uni-ulm.de

2329-4302/2023/$28.00

In medical 3D volumes, the annotations are often generated by annotating individual 2D slices. In this paper, we aim to answer the question as to how far this number can be reduced. A reduced number of slices that needs to be annotated would speed up the annotation process. A faster data labeling process has two benefits. First, it would free up expert annotators' time, so they can spend it on clinically relevant tasks. Second, a faster labeling process would result in the creation of larger annotated datasets. This would most likely improve the results of deep learning methods that require large amounts of labeled training data, as in the medical domain. Here, it is not the number of images, but rather the amount of labeled training data, that acts as the limiting factor.[1] The approaches we compare work as a two-step process. First, based on different slice similarity metrics, a selector selects slices that need to be manually annotated by an expert. Second, the annotated slices are then passed to a predictor which creates segmentations for all the unlabeled slices in the 3D volume. A schematic overview of the process can be seen in Ref. 1. What makes our approach different from other approaches is that we actively facilitate overfitting in order to generate annotations for sparsely annotated 3D volumes. For each volume, a new predictor is trained using only the previously selected slices. Each predictor thus overfits on a single specific volume. To determine the ideal combination of selector and predictor, we analyze different combinations and evaluate them on CT and MRI volumes. The analyzed selectors range from similarity metrics computed directly on the slices, over histogram-based approaches to perceptual similarity metrics.[3] For the predictors, we use nearest interpolation as a baseline and compare it against the state-of-the-art slice interpolation method implemented in the MITK (Medical Imaging Interaction Toolkit) workbench,[4] as well as three state-of-the-art deep segmentation models (U-Net,[5] DeepLabV3+,[6] and MAnet[7]).

The evaluated approaches facilitate ideas from active learning,[8] and the overfitting approaches facilitate techniques from the lower end of the bias-variance spectrum.[9] Thus we trade expert time with compute time, as the latter is cheaper and easier to come by. We believe that this is a sustainable trade-off, as compute time for standard processes will presumably be further reduced in the future,[10] which cannot be expected for the expert time. In very critical use cases, the predicted annotations could even be reviewed and/or edited by experts. This would still reduce the required expert time, as it is much faster to assess a high-quality annotation than to create one.

2 Related Work

While there is quite a lot of literature that focuses on semantic segmentation in medical image computation, we focus on a specific topic: how to reduce the number of slices to be annotated when creating training data for a medical volume segmentation task. Surprisingly, there is little work that focuses on the task of labeling images, while this task is very tedious and builds the underlying foundation for many machine learning algorithms.

One way to reduce labeling efforts is to use weakly supervised methods. Weakly supervised models are trained on some form of simpler annotations. Those can include but are not limited to image-level labels, bounding boxes, or point annotations.[11–14] Creating annotations for those methods is much faster than creating full pixel-level labeled segmentation masks. A drawback of this approach is that methods trained with weakly supervised labels often perform worse than fully supervised methods.

Valindria et al.[15] used overfitting a segmentation model on a single image to estimate the performance of the said model in the absence of ground truth data. In their reverse classification accuracy (RCA), a segmentation model generates a segmentation for an image where no GT exists. To evaluate this segmentation, an RCA model is trained on this image. This is done by extracting pixel-based image features and then training a classifier to map them to a given class. To evaluate the RCA model, it is applied to all pixels of images where GT segmentations exist, thus creating a segmentation. This approach will not work on arbitrary images but shows good results on images that are similar to the one where the RCA model is trained on. The assumption that the model will perform good on similar images even though it only has been trained on one single image is similar to the assumptions we made for our approach.

Top et al.[16] used an active learning framework in which a random walker algorithm is used to segment slices in a 3D volume. During training, the active learning framework queries an

annotator for regions in which the segmentation algorithm has high uncertainty. Those newly annotated regions are then used to improve the segmentation performance. This overall setup is very similar to our approach. Although the semisupervised methods try to reduce the annotation time by making the annotation process itself simpler, we aim to reduce the overall amount of annotations needed.

Tajbakhsh et al. compared a number of active learning approaches that are used for cost-effective labeling of medical images. However, only two of the eight methods presented in the paper fall into the "one-shot" category that is comparable to our approach. The other methods require an iterative workflow where multiple training and labeling iterations are needed.[17]

Mahapatra et al.[18] used a conditional generative adversarial network (cGAN) in an active learning framework to leverage small datasets. The cGAN takes an input image and its manually annotated mask to generate similar looking images. A Bayesian network[19] calculates the informativeness of the generated image samples. The samples that are rated with a high informativeness are used to fine tune a classifier. In an active learning method, this cycle is repeated. The Bayesian network makes use of noise that is inherent to the data sample (aleatoric) and combines it with uncertainty in the models parameters.

The method presented by Zhang et al. relies on a feature extractor network that needs to be pretrained in an unsupervised manner.[20] Both approaches show that the one-shot methods achieve competitive results compared to the iterative ones.[17]

Similar to our approach, Zheng et al. had experts annotate representative 2D slices from 3D volumes. The slices are then used to train a network and predict labels for the whole volumes. However, in contrast to our approach, the prediction is not shown to the experts but used as pseudolabels for weakly supervised learning.[21]

Sugino et al. also proposed an approach in which they trained a fully convolutional network on 3D volumes with missing slices. They experimented with 2D and 3D networks but found that both approaches can achieve good results.[22] For this reason, we only focus on 2D models in our work. A key difference is that they used an entire dataset to train their model whereas we use one predictor per volume. Furthermore, we experiment with different slice selectors.

3 Method

Annotating medical volume data for the purposes of training deep neural networks for segmentation tasks can be a time-consuming process. To tackle this challenge, we evaluate different combinations of selectors and predictors, which allow for reducing the number of slices to be annotated. While a selector facilitates a similarity metric, in order to select the slices to be annotated, a predictor predicts annotations for the remaining slices based on the provided annotations (see Fig. 1). This procedure is done individually for each volume. This means that for each individual volume a new predictor is trained which then overfits on the given volume. The reasoning behind this is that many slices inside a single 3D volume should be similar to each other. The task of the selector is to find a set of representative slices. When the predictor network learns to accurately segment a slice, it should also be able to accurately segment other slices that

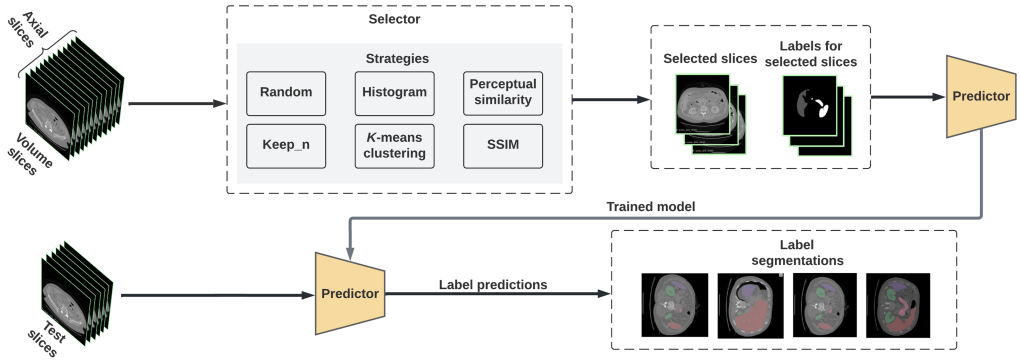

Fig. 1 Functionality of the presented approaches. A selector chooses representative slices from a given input volume. These slices are used to train a predictor, which is then used to create segmentations for the remaining slices.

look very similar. Our goals are to find the ideal combination of selector and predictor and to determine the minimal amount of slices to be annotated in order to obtain sufficient segmentation accuracy. In the following two sections, we briefly describe the selectors and predictors we have incorporated into our evaluation.

3.1 Selectors

To analyze the more advanced similarity-based selectors, we include two simple baselines in our evaluation. The first baseline, to which we refer as random, randomly samples a given number of slices from the volume. The second baseline, to which we refer as Keep_n, selects every n'th slice to be annotated. This selector was also used by Sugino et al.[22]

All other selectors facilitate similarity-based clustering. For the simplest version, each slice is flattened into a single vector which is used as input for the k-means algorithm. The standard sklearn k-means implementation was used. The k-means algorithm converged for all cases in <10 iterations, without reaching the max number of iterations.

The second approach is based on histogram analysis, an aggregated technique often used for image comparison. The histogram with 255 bins is computed for each image and used as input for a k-means clustering algorithm. The third approach relies on perceptual similarity that is often used as loss function in neural networks if images are compared.[3] The perceptual similarity is computed with the lpips Python library[23] using a ImageNet-pretrained AlexNet.[24]

Finaly, structural similarity index measure (SSIM) is an image similarity measure that has shown good results on various image comparison tasks, which is why we also include it here.[25] In the case of SSIM and perceptual similarity, the similarities between all slices are computed and the resulting matrix is used for the clustering.

For all results displayed in Table 2, k-means with $k = 15$ was used as we found this to give the best results compared to $k = 5$ and $k = 10$. The times different selector strategies need to return slices can be seen in Table 1.

To obtain the desired number of slices from these cluster-based selectors, we randomly sample a fixed number of slices from each cluster. If a cluster is too small to sample the required number of slices, slices are sampled from the most similar cluster to obtain the desired number of slices.

To keep the number of experiments and thus compute time at bay, we only consider axial slices. Only for the FLARE21 dataset, we ran a subset of our experiments on the coronal and sagittal axis as well.

3.2 Predictors

After the slices obtained from the selector have been manually annotated, a predictor can be used to predict segmentation masks for the remaining slices. To analyze more advanced predictors, we use nearest-neighbor (NN) interpolation as a baseline. In contrast to linear interpolation, NN ensures that the interpolation of two segment IDs remains a valid segment ID.

One more advanced predictor is provided by the MITK[26] framework and graphical tool for medical image analysis. Its graphical interface is designed to handle the complete clinical workflow, including image analysis, data retrieval, and diagnosis. For segmentation, it includes not only basic tools like draw or fill but also advanced interpolation techniques. Being based on other

Table 1 Times a selector needs to select slices of a single volume from the BraTS19 dataset. Times have been averaged over 7 runs.

	Times
Keep_n	764 ms ± 26.5 ms
k-means	7.27 s ± 203 ms
Hist.	1.77 s ± 27.2 ms
Percept.	1 min 4 s ± 706 ms
SSIM	2 min 23 s ± 795 ms

well-established frameworks like ITK or VTK, a wide range of typical file formats as well as multiview images, such as CT, or color images, such as pathological images, can be handled with MITK. The framework and toolkit is completed open source and can be extended with custom extensions.

We use its 2D workbench interpolation[4] to predict labels for the remaining slices. It uses a shape-based interpolation algorithm introduced by Herman et al.[27] Within a slice, the distance to a segmentation border is determined. The distance has a positive value if the pixel is within the segmentation and a negative value if it is outside. The distances to the segmentation border are computed using the 3×3 matrix, as described in the original paper. To then interpolate between slices, cubic spline interpolation is used on those distance maps.

Because the MITK could only be used for binary datasets, we also used the 3D Slicer software package[28] as a predictor. 3D Slicer provides functionality that is very similar to the MITK. To interpolate between missing slices, it uses a morphology-based approach that only uses the segmentation maps. The algorithm was proposed by Albu et al.[29] It works in an iterative way, computing transitions between pairs of slices. Interpolated slices can be used in the following iterations. This allows for a smooth but not over-smooth transition between slices. Like the MITK, the 3D Slicer GUI does not allow to change any parameters of the algorithms so the defaults have been used.

Furthermore, we analyze the value of deep-learning-based predictors, whereby we overfit state-of-the-art segmentation models to single 3D volumes, to predict labels for the nonannotated slices. This means that the models need to be trained/overfitted for/to each individual volume. This is fairly straightforward and does not differ from the usual way a neural network is trained in a semantic segmentation task. The only big difference is that for each new 3D volume we train a new network. The models are initialized with weights obtained by pretraining on ImageNet[30] (BraTS19) or Instagram images (FLARE21 and MSD-Heart). All weights are provided by the segmentation models pytorch library.[31] Note that we do only train 2D models. Previous work has shown that 2D and 3D models are often on par.[22,32]

To analyze the impact of such deep overfittings, we decided to include a comparison of three state-of-the-art semantic segmentation models (U-Net,[5] DeepLabV3+,[6] and MAnet[7]) in our study. We chose those models based on the results of public leader boards for different segmentation datasets.[33] Using different architectures allows us to show that the approach presented is independent of a given backbone network.

For our experiments, we specifically used pytorch model implementations provided by "segmentation models pytorch."[31]

On the BraTS19 dataset, a Resnet152[34] was used as encoder, and on the FLARE21 and MSD-Heart dataset, we went even larger with a ResNext101[35] with cardinality 32. The architecture was chosen because it performed well on a test of a single volume from the dataset.

To set the hyperparameters for our networks, we ran a hyperparameter sweep on 12 representative 3D volumes. We use those hyperparameters for all networks and volumes in the same dataset. So even though each model is trained on a different individual 3D volume each network has the same hyperparameters.

4 Datasets

All possible combinations of selectors and predictors, which we evaluate in this work, are initially evaluated on the BraTS19 dataset,[36] a high-resolution MRI dataset, containing high-grade glioma images. We choose to work with T1 contrast-enhanced images from the four available contrasts in the dataset, because of its common use in medical image segmentation problems. To make it feasible to test a large number of selector and predictor combinations, we further choose 12 volumes out of 259 available. We do this by applying PCA within each volume and doing k-means clustering between the resultant PCs to select volumes based on maximum variability.

The medical segmentation decathlon MSD-Heart and the FLARE21 dataset. The MSD-Heart dataset[37] contains 30 mono-modal MRI volumes, from which we selected 7 volumes having an equal number of 120 total slices, whereby the target ROI is left ventricle. We choose this dataset because of its large target segmentation area. The FLARE21[38] dataset contains 360 CT volumes from which we select 6 volumes, which have an equal total number of slices for each

volume. We selected FLARE21 dataset because of its diversity in target sites, such as liver, kidney, spleen, and pancreas.

The FLARE21 and MSD-Heart datasets contain volumes with different numbers of slices in the axial direction. For comparability, all volumes that have been chosen from one dataset had to have the same number of slices. Volumes have been grouped by the number of slices and a group has been chosen at random while ensuring that it contained enough and not too many slices.

5 Results

5.1 BraTS19

Table 2 shows the Dice scores for the BraTS19 dataset using different selectors and predictors. The Dice scores are the mean Dice scores over the different volumes in the dataset. The standard deviation is reported after the $\pm$ sign. This setup is kept for all the following tables in this paper. The first column displays the train size, meaning that how many slices have been sampled from each volume. The best result for each train size is highlighted in bold. As can be seen, usually the deep-learning-based predictors outperform the NN interpolation baseline. Naturally, with increasing train size, the performance increases, independent of the model. Furthermore, one can see that usually the Keep_*n*-based selector gives the best results followed by the SSIM selector. The overfitting baseline results for U-Net, DeepLabV3+, and MAnet are 0.973, 0.917, and 0.958, respectively. This shows that the models did not completely overfit the data. The overfitting baselines are networks that have been trained on all slices. They represent the upper limit of what results one could achieve with the given training time. The MAnet further is the best predictor in most of the cases.

5.2 FLARE21 and MSD-Heart Dataset

For the FLARE21 and MSD-Heart datasets, we decided to only use the Keep_*n* and the SSIM selectors, as they showed the most promising results on the BraTS19 dataset. The obtained results are shown in Tables 3 and 4. In addition, those tables also show the average Hausdorff distances[39] between the ground truth and the predictions. All distances are reported in millimeters. Best results are highlighted for Dice scores in bold.

Because the results from the BraTS19 dataset show a high Dice score already, we decided that no experiments with train sizes bigger than 50% were needed. Furthermore, this observation motivated us to experiment with even fewer slices than the 20% train size. For the FLARE21 dataset, the deep-learning-based method outperforms the NN interpolation method on all train sizes. The Keep_*n* strategy outperforms the SSIM strategy on all selectors and on all train sizes. The U-Net predictor generally outperforms the NN interpolation as well as the 3D Slicer. The interpolation method used in the 3D Slicer software achieves better results than the NN interpolation for slices selected with the Keep_*n* selector but is worse for slices using the SSIM selector. This indicates that the 3D Slicer needs slices sampled with a regular step size.

For the MSD-Heart dataset, the interactive interpolation method from the MITK workbench achieved the highest Dice scores and small Hausdorff distances on most train sizes. On small train sizes (12.5% and 20%), the performance of the deep learning models is very similar to the MITK one, while it outperforms it in some cases. The performance of the 3D Slicer lies between the U-Net and the NN interpolation, which performs worst for this dataset. Similar to the FLARE21 dataset, the Keep_*n* strategy outperforms the SSIM selector on all train sizes and selectors. The overfitting baseline for the FLARE21 and MSD-Heart dataset are a Dice score of 0.996 and 0.998, respectively. Showing that those models overfitted well on both datasets.

Figure 2 shows some visual results for the segmentations on all three datasets. All results have been obtained using the 50% Keep_*n* selector and a U-Net or NN interpolation as predictor.

5.3 Random Crops

In earlier results, we have seen that the Keep_*n* selection strategy performs better than the other selectors. We came to the conclusion that this is because there is a strong concordance along one image axis in our data. To investigate this, we decided to add a padding around each slice of the volume and crop a patch at a random location from the padded image. This way the sampled slices should have less concordance compared to the original approach. We experimented with

Table 2 Experiment results for the BraTS19 dataset for different selector/predictor combinations at various train sizes. The bold highlighted results show that the Keep_*n* and SSIM selectors show the most promising results. The MAnet is the best predictor.

%	Predictor	Rand.	Keep_*n*	*k*-means	Hist.	Percept.	SSIM
20	U-Net	0.842 (±0.036)	0.842 (±0.086)	0.834 (±0.104)	0.822 (±0.057)	0.808 (±0.094)	0.852 (±0.090)
	DeepLabV3+	0.789 (±0.046)	0.852 (±0.088)	0.804 (±0.044)	0.780 (±0.045)	0.790 (±0.041)	0.808 (±0.053)
	MAnet	0.839 (±0.045)	**0.874** (±**0.026**)	0.854 (±0.042)	0.844 (±0.019)	0.813 (±0.044)	0.859 (±0.038)
	NN	0.765 (±0.054)	0.831 (±0.027)	0.790 (±0.037)	0.764 (±0.054)	0.732 (±0.037)	0.813 (±0.034)
33	U-Net	0.874 (±0.026)	0.847 (±0.029)	0.872 (±0.030)	0.860 (±0.033)	0.862 (±0.034)	0.883 (±0.028)
	DeepLabV3+	0.823 (±0.035)	**0.898** (±**0.030**)	0.817 (±0.037)	0.803 (±0.034)	0.814 (±0.038)	0.828 (±0.038)
	MAnet	0.872 (±0.023)	0.885 (±0.088)	0.882 (±0.039)	0.865 (±0.024)	0.874 (±0.024)	0.894 (±0.032)
	NN	0.834 (±0.033)	0.876 (±0.024)	0.815 (±0.033)	0.796 (±0.045)	0.804 (±0.041)	0.835 (±0.029)
50	U-Net	0.894 (±0.027)	0.894 (±0.087)	0.898 (±0.027)	0.887 (±0.028)	0.889 (±0.029)	0.900 (±0.025)
	DeepLabV3+	0.846 (±0.036)	0.870 (±0.034)	0.845 (±0.032)	0.837 (±0.034)	0.841 (±0.037)	0.848 (±0.029)
	MAnet	0.866 (±0.089)	**0.915** (±**0.021**)	0.881 (±0.024)	0.888 (±0.021)	0.886 (±0.023)	0.883 (±0.027)
	NN	0.855 (±0.027)	0.875 (±0.024)	0.854 (±0.025)	0.842 (±0.030)	0.848 (±0.038)	0.858 (±0.027)
67	U-Net	0.906 (±0.026)	0.906 (±0.024)	0.910 (±0.027)	0.907 (±0.022)	0.912 (±0.029)	0.901 (±0.025)
	DeepLabV3+	0.860 (±0.034)	0.875 (±0.097)	0.866 (±0.030)	0.906 (±0.030)	0.860 (±0.035)	0.860 (±0.024)
	MAnet	0.907 (±0.024)	**0.916** (±**0.021**)	0.914 (±0.022)	0.905 (±0.029)	0.902 (±0.032)	0.904 (±0.027)
	NN	0.863 (±0.029)	0.875 (±0.026)	0.873 (±0.025)	0.860 (±0.024)	0.870 (±0.031)	0.867 (±0.034)
80	U-Net	0.913 (±0.030)	0.903 (±0.020)	0.900 (±0.020)	0.916 (±0.020)	0.901 (±0.027)	**0.922** (±**0.035**)
	DeepLabV3+	0.866 (±0.033)	0.885 (±0.154)	0.872 (±0.030)	0.904 (±0.032)	0.871 (±0.039)	0.868 (±0.041)
	MAnet	0.908 (±0.042)	0.903 (±0.032)	0.918 (±0.028)	0.902 (±0.025)	0.914 (±0.089)	0.889 (±0.043)
	NN	0.870 (±0.032)	0.876 (±0.024)	0.842 (±0.025)	0.874 (±0.023)	0.856 (±0.034)	0.872 (±0.025)

Table 3 Dice scores and Hausdorff distances for the Keep_*n* and SSIM strategies on the FLARE21 dataset. For comparison with a state-of-the-art method, we choose the interpolation algorithm from the 3D Slicer software package. For each training percentage, the best results (Dice score) have been highlighted in bold. For each training size, the first row depicts the Dice score and the second row the Hausdorff distance.

	Keep_*n*			SSIM		
%	U-Net	NN	3D Slicer	U-Net	NN	3D Slicer
12.5	**0.885** (±0.018)	0.849 (±0.018)	0.870 (±0.019)	0.844 (±0.028)	0.789 (±0.031)	0.674 (±0.067)
	3.282 (±0.631)	6.105 (±0.568)	5.948 (±0.468)	3.363 (±0.764)	6.407 (±0.402)	6.212 (±0.881)
20	**0.926** (±0.009)	0.900 (±0.011)	0.913 (±0.014)	0.857 (±0.038)	0.812 (±0.035)	0.793 (±0.067)
	2.982 (±0.36)	4.962 (±0.83)	5.439 (±0.344)	3.242 (±0.752)	6.131 (±0.428)	5.71 (±0.489)
33	0.930 (±0.015)	0.930 (±0.009)	**0.939** (±0.011)	0.898 (±0.015)	0.871 (±0.013)	0.805 (±0.145)
	2.932 (±0.384)	4.966 (±0.338)	5.008 (±0.358)	3.125 (±0.801)	5.577 (±0.509)	4.824 (±0.517)
50	**0.962** (±0.005)	0.932 (±0.005)	0.960 (±0.006)	0.952 (±0.013)	0.919 (±0.012)	0.937 (±0.015)
	2.719 (±0.247)	4.955 (±0.418)	4.888 (±0.353)	2.752 (±0.77)	5.154 (±0.724)	3.296 (±0.585)

Table 4 Dice scores and Hausdorff distances for the Keep_*n* and SSIM selectors on the MSD-Heart dataset. For comparison with a state-of-the-art method, we chose the interpolation algorithm from the MITK and the 3D Slicer software packages. For each training percentage, the best results (Dice score) have been highlighted in bold. For each training size, the first row depicts the Dice score and the second row the Hausdorff distance.

	Keep_*n*				SSIM			
%	U-Net	NN	MITK	3D Slicer	U-Net	NN	MITK	3D Slicer
12.5	0.943 (±0.041)	0.921 (±0.010)	**0.948** (±0.009)	0.940 (±0.007)	0.900 (±0.086)	0.865 (±0.036)	0.899 (±0.051)	0.867 (±0.054)
	1.294 (±0.348)	1.644 (±0.181)	1.498 (±0.180)	1.278 (±0.201)	1.378 (±0.511)	1.928 (±0.198)	1.100 (±0.216)	1.074 (±0.109)
20	**0.969** (±0.013)	0.947 (±0.006)	0.964 (±0.010)	0.957 (±0.003)	0.944 (±0.050)	0.914 (±0.015)	0.940 (±0.023)	0.927 (±0.021)
	1.123 (±0.23)	1.469 (±0.143)	1.365 (±0.199)	1.444 (±0.164)	1.204 (±0.44)	1.586 (±0.199)	1.047 (±0.177)	1.209 (±0.171)
33	0.978 (±0.012)	0.965 (±0.029)	**0.985** (±0.004)	0.977 (±0.003)	0.961 (±0.016)	0.934 (±0.006)	0.965 (±0.014)	0.950 (±0.016)
	1.022 (±0.198)	1.353 (±0.145)	1.258 (±0.154)	1.378 (±0.165)	1.113 (±0.469)	1.469 (±0.253)	0.861 (±0.126)	0.979 (±0.18)
50	0.984 (±0.008)	0.965 (±0.003)	**0.993** (±0.003)	0.989 (±0.001)	0.973 (±0.210)	0.950 (±0.005)	0.988 (±0.003)	0.967 (±0.011)
	0.945 (±0.173)	1.356 (±0.129)	1.114 (±0.155)	1.268 (±0.114)	0.911 (±0.423)	1.238 (±0.344)	0.645 (±0.083)	0.675 (±0.13)

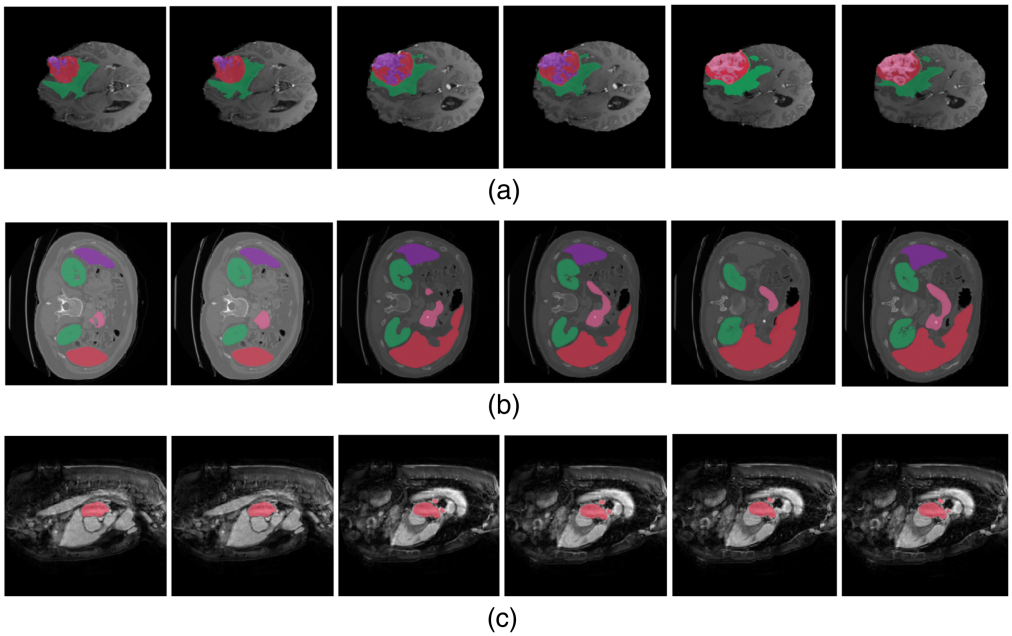

Fig. 2 Visual segmentation results for the three different datasets. (a) For the BraTS19 dataset, the labels are: necrotic and nonenhancing tumor (red), peritumoral edema (green), and enhancing tumor (purple). (b) For the FLARE21 dataset, the labels are: liver (red), kidney (green), spleen (purple), and pancreas (pink). (c) For the MSD-Heart dataset, the segmentation target is the left atrium (red). First two image pairs show the results for U-Net and the last pair for NN interpolation. The first image in a pair is the prediction the second the ground truth. All methods show good performance with slight differences mainly around the edges of the segmentations. The biggest differences are in the FLARE21 dataset (b). Especially, the spleen seems to be hard to segment.

Table 5 The slices in the FLARE21 dataset were padded and cropped at random locations have been taken. The IoU was 0.01587 for padding with 50 pixels at each border and 0.01087 for padding with 100 pixels. For small training dataset sizes and fewer overlap, the SSIM selector outperforms the Keep_*n* selector (bold). For each training percentage, the first row depicts the Dice score and the second row the average Hausdorff distance.

	+50 px padding		+100 px padding	
%	Keep_*n*	SSIM	Keep_*n*	SSIM
12.5	0.640 (± 0.309)	0.622 (± 0.263)	0.489 (± 0.210)	**0.628 (± 0.281)**
	3.594 (± 1.721)	3.078 (± 1.678)	3.017 (± 1.478)	1.912 (± 0.948)
20	0.789 (± 0.211)	0.736 (± 0.266)	0.642 (± 0.167)	**0.676 (± 0.201)**
	2.965 (± 1.652)	2.852 (± 1.828)	2.504 (± 1.504)	1.756 (± 1.058)
33	0.842 (± 0.175)	0.824 (± 0.217)	0.760 (± 0.142)	0.724 (± 0.170)
	3.061 (± 1.536)	2.752 (± 2.045)	2.369 (± 1.535)	1.69 (± 1.262)
50	0.884 (± 0.148)	0.852 (± 0.146)	0.767 (± 0.149)	0.762 (± 0.135)
	2.438 (± 1.525)	2.542 (± 2.001)	2.174 (± 1.72)	1.496 (± 1.216)

Table 6 Dice scores and Hausdorff distances for segmentation when slices from different image axes are used. (FLARE21 dataset). Again, the Keep_*n* selector outperforms the SSIM selector. A U-Net was used as a predictor. Bold results (Dice) show the best selector for each axis and percentage. For each training size, the first row depicts the Dice score and the second row the Hausdorff distance.

	Coronal		Sagittal	
%	Keep_*n*	SSIM	Keep_*n*	SSIM
12.5	**0.965** (±0.017)	0.944 (±0.043)	**0.954** (±0.026)	0.939 (±0.054)
	4.857 (±0.884)	11.386 (±2.303)	7.386 (±1.236)	11.186 (±1.823)
20	**0.974** (±0.008)	0.960 (±0.027)	**0.970** (±0.014)	0.937 (±0.057)
	4.754 (±0.638)	9.183 (±3.002)	7.035 (±1.137)	10.170 (±2.713)
33	**0.979** (±0.007)	0.967 (±0.017)	**0.975** (±0.009)	0.928 (±0.063)
	4.631 (±0.642)	8.938 (±2.215)	6.987 (±0.998)	8.650 (±2.493)
50	**0.981** (±0.004)	0.974 (±0.012)	**0.978** (±0.006)	0.947 (±0.047)
	4.857 (±0.884)	5.027 (±1.324)	6.712 (±0.837)	7.775 (±2.085)

two different amounts of padding, either padding 50 pixels at each side or padding 100 pixels at each side. The mean intersection over union (IoU) for the resulting patches was 0.1587 and 0.1087, respectively. The results for the Keep_*n* and SSIM selector strategies can be found in Table 5. One can see that in the setting where there is only very few training data available and the overlap between the patches is minimal the SSIM strategy outperforms the Keep_*n* strategy. For the other settings, Keep_*n* performs better.

5.4 Other Image Axes

For the FLARE21 dataset, we also ran some experiments using other image axes. Similar to the experiments using axial image slices, the Keep_*n* selector outperforms the SSIM selector. This leads us to believe that there is also a strong concordance between image slices on the other axes for the FLARE21 dataset. Comparing these results with the results from Tables 3 and 4, one can see that using the other image axes gives higher Dice scores. The voxel size for the FLARE21 dataset is 2.5 mm in axial direction and 0.7 mm in sagital and coronal direction (Table 6).

6 Discussion

Our results show that there are no clear favoring strategies for all situations. On a binary segmentation task (MSD-Heart), the MITK interpolation method provides the best results in terms of Dice score. Especially in cases where the amount of labeled training slices gets reduced only by half, very high Dice scores (>0.99) can be achieved. For cases where the number of labeled training slices gets reduced more drastically, the U-Net can achieve results similar to the MITK method. In practice, we would still recommend the use of the MITK method because of the reduced compute requirements. On a desktop GPU, depending on the volume size and underlying model used, the overfitting on the other hand can take up to 30 min per volume. A clear downside of the MITK interpolation method is that it is only implemented for binary segmentation tasks. A workaround would be to split the labels into separate binary labels per target structure and run the interpolation on each of them. After that, the binary labels could be combined again to have a multi-class segmentation. For a multi-class segmentation task (BraTS19 and FLARE21), we would recommend the deep-learning-based method over the NN interpolation, as we believe that the longer training time is out-weight by the performance. Especially, when the annotations are used in a downstream task to train or evaluate other machine learning models appropriate annotations are essential.

Table 7 Training times in minutes for the U-Net on the BraTS19 and FLARE21 datasets. A larger encoder led to longer times for the FLARE21 dataset. For all examples, 50% of the slices have been used for training.

	Keep_*n*	*k*-means	Hist.	Percept.	SSIM
U-Net BraTS19	25.39	22.43	25.67	23.81	28.12
U-Net FLARE21	88.47	NA	NA	NA	82.5

As a selector, we would recommend the Keep_*n* strategy. It achieved the best results while at the same time it is very fast and also easy to understand.

Our results show that the deep-learning-based models that have achieved higher results in the overfitting baseline also achieve higher results when trained with lower train sizes. This shows that there are indeed a lot of similar slices in a medical 3D volume. It also shows that a relatively small number of representative slices is sufficient for a neural network to segment the other slices.

In our earlier experiments, the Keep_*n* selector outperformed the other more elaborate selectors. We believe that this hints at strong interslice correlations along the given image axis. To investigate this, we performed the experiment in which we cropped patches at random slice locations. This reduced the concordance along the axis drastically. In cases with little training data and reduced concordance, the SSIM selector outperformed the Keep_*n* selector. This leads us the the recommendation that for datasets where very little concordance along a given image axis is expected the SSIM selector should lead to better results.

While the neural network-based predictors take longer time to be trained, it can be used in a practical setting. One possible solution is an updated workflow. An annotator could annotate the selected slices of a volume, whereas a predictor is trained on a second volume or the predictions are created over night. In both settings, the annotator has to look twice into a volume, but this can be accounted for in the software and time saving during the annotation process more than compensates for that. Another possible solution would be further software or hardware optimizations. Relying on nonoptimized research code, we believe that an improved setting could lead to significant time reductions.

Our method lends a lot of ideas from active learning, yet does not exactly fall in the category of active learning. In active learning, an annotator would label some image slices, feed them to the model, wait for some output, label new slices, and repeat the whole process. We see it as a benefit that in our method, the process is not iterative and an annotator does not have to wait for some results and then work on the same volume again as in other methods.[16]

7 Conclusions and Future Work

In Sec. 1, we mention that we trade human annotation time with compute time, as the second is cheaper, easier to come by and will most likely reduce in the future. An overview of compute times can be seen in Table 7. This table shows that when using big encoders, as done in the FLARE21 setting, the time the selector needs is negligible.

Our paper provides an overview of how far the number of slices in a 3D volume that need annotations can be reduced, to obtain sufficient segmentation quality. We tested different selectors and predictors on CT and MRI data to answer this question. Based on our results, we see no reason to assume that the imaging modality influences the performance and therefore believe that the methods are domain agnostic. In general, the MITK interpolation method seems to lead to sufficient results, without strong performance penalties. For datasets where only a very small number of slices can be labeled manually in a given time frame, the deep-learning-based predictors also seem to perform sufficiently well.

As our evaluation naturally required a vast amount of compute cycles, we see several endeavors, which still could be investigated in the future. For example, we are interested in the influence of different target sizes and variations in how slices are sampled from clusters. In addition, it would be interesting to investigate the effects of initializing the networks with weights that

have been pretrained on medical data, possibly even from annotated slices of the same dataset. This might even speed up the deep-learning-based predictors.

Disclosures

The authors have no relevant financial interests in the manuscript and no other potential conflicts of interest to disclose.

Code, Data, and Materials Availability

The data presented in this article are publicly available at BraTS19: https://www.med.upenn.edu/cbica/brats2019/data.html; FLARE21: https://flare.grand-challenge.org/; and MSD-Heart: http://medicaldecathlon.com/. Code will be available at https://github.com/sirtris/slice_segmentation.

References

1. G. Litjens et al., "A survey on deep learning in medical image analysis," *Med. Image Anal.* **42**, 60–88 (2017).
2. S. Wang et al., "Annotation-efficient deep learning for automatic medical image segmentation," *Nat. Commun.* **12**(1), 5915 (2021).
3. A. Dosovitskiy and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," in *Adv. Neural Inf. Process. Syst. 29* (2016).
4. A. Fetzer et al., "An interactive 3D segmentation for the medical imaging interaction toolkit (MITK)," in *Proc. MICCAI Interact. Med. Image Comput.*, pp. 1–11 (2014).
5. O. Ronneberger, P. Fischer, and T. Brox, "U-Net: convolutional networks for biomedical image segmentation," *Lect. Notes Comput. Sci.* **9351**, 234–241 (2015).
6. L.-C. Chen et al., "Encoder–decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, pp. 801–818 (2018).
7. T. Fan et al., "MA-Net: a multi-scale attention network for liver and tumor segmentation," *IEEE Access* **8**, 179656–179665 (2020).
8. B. Settles, "Active learning literature survey," Computer Sciences Technical Report, University of Wisconsin–Madison (2009).
9. M. Belkin et al., "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proc. Natl. Acad. Sci. U. S. A.* **116**(32), 15849–15854 (2019).
10. R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Compute and energy consumption trends in deep learning inference," https://doi.org/10.48550/arXiv.2109.05472 (2021).
11. V. Lempitsky et al., "Image segmentation with a bounding box prior," in *IEEE 12th Int. Conf. Comput. Vision*, pp. 277–284 (2009).
12. H. Kervadec et al., "Bounding boxes for weakly supervised segmentation: global constraints get close to full supervision," in *Proc. Third Conf. Med. Imaging with Deep Learn., Proc. Mach. Learn. Res., PMLR*, T. Arbel et al., Eds., Vol. 121, pp. 365–381 (2020).
13. C.-C. Hsu et al., "Weakly supervised instance segmentation using the bounding box tightness prior," in *Adv. Neural Inf. Process. Syst.* 32, pp. 6586–6597 (2019).
14. A. Bearman et al., "What's the point: semantic segmentation with point supervision," *Lect. Notes Comput. Sci.* **9911**, 549–565 (2016).
15. V. V. Valindria et al., "Reverse classification accuracy: predicting segmentation performance in the absence of ground truth," *IEEE Trans. Med. Imaging* **36**, 1597–1606 (2017).
16. A. Top, G. Hamarneh, and R. Abugharbieh, "Active learning for interactive 3D image segmentation," *Lect. Notes Comput. Sci.* **6893**, 603–610 (2011).
17. N. Tajbakhsh et al., "Embracing imperfect datasets: a review of deep learning solutions for medical image segmentation," *Med. Image Anal.* **63**, 101693 (2020).
18. D. Mahapatra et al., "Efficient active learning for image classification and segmentation using a sample selection and conditional generative adversarial network," *Lect. Notes Comput. Sci.* **11071**, 580–588 (2018).
19. A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Adv. Neural Inf. Process. Syst.* 30 (2017).
20. Z. Zhang et al., "A sparse annotation strategy based on attention-guided active learning for 3d medical image segmentation," https://doi.org/10.48550/arXiv.1906.07367 (2019).
21. H. Zheng et al., "An annotation sparsification strategy for 3D medical image segmentation via representative selection and self-training," *Proc. AAAI Conf. Artifi. Intell.* **34**, 6925–6932 (2020).
22. T. Sugino et al., "Label cleaning and propagation for improved segmentation performance using fully convolutional networks," *Int. J. Comput. Assist. Radiol. Surg.* **16**(3), 349–361 (2021).
23. R. Zhang et al., "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR* (2018).

24. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Adv. Neural Inf. Process. Syst.*, F. Pereira et al., Eds., Vol. 25, Curran Associates, Inc. (2012).
25. A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *20th Int. Conf. Pattern Recognit.*, IEEE, pp. 2366–2369 (2010).
26. M. Nolden et al., "The medical imaging interaction toolkit: challenges and advances," *Int. J. Comput. Assist. Radiol. Surg.* **8**(4), 607–620 (2013).
27. G. Herman, J. Zheng, and C. Bucholtz, "Shape-based interpolation," *IEEE Comput. Graphics Appl.* **12**(3), 69–79 (1992).
28. A. Fedorov et al., "3D Slicer as an image computing platform for the quantitative imaging network," *Magn. Reson. Imaging* **30**(9), 1323–1341 (2012).
29. A. B. Albu, T. Beugeling, and D. Laurendeau, "A morphology-based approach for interslice interpolation of anatomical slices from volumetric images," *IEEE Trans. Biomed. Eng.* **55**(8), 2022–2038 (2008).
30. J. Deng et al., "ImageNet: a large-scale hierarchical image database," in *IEEE Conf. Comput. Vision and Pattern Recognition*, pp. 248–255 (2009).
31. P. Yakubovskiy, "Segmentation models pytorch," 2020, https://github.com/qubvel/segmentation_models.pytorch.
32. D. Kern et al., "2D vs. 3D U-Net abdominal organ segmentation in CT data using organ bounds," *Proc. SPIE* **11601**, 1160114 (2021).
33. MetaAI, "Segmentation data sets," 2021, https://paperswithcode.com/datasets?task=semantic-segmentationpage=1 (accessed 2022-03-01).
34. K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 770–778 (2016).
35. S. Xie et al., "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 1492–1500 (2017).
36. B. H. Menze et al., "The multimodal brain tumor image segmentation benchmark (BRATS)," *IEEE Trans. Med. Imaging* **34**(10), 1993–2024 (2014).
37. M. Antonelli et al., "The medical segmentation decathlon," *Nat. Commun.* **13**(1), 4128 (2022).
38. J. Ma et al., "Abdomenct-1K: is abdominal organ segmentation a solved problem," *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 6695–6714 (2021).
39. O. U. Aydin et al., "On the usage of average Hausdorff distance for segmentation performance assessment: hidden error when used for ranking," *Eur. Radiol. Exp.* **5**, 4 (2021).

Tristan Payer works as a research associate in the Visual Computing Group at Ulm University. He received his master's as well as his bachelor's degree from Radboud University Nijmegen in the field of artificial intelligence.

Faraz Nizamani received his master's degree from Otto von Guericke University in medical systems engineering. He later joined the Visual Computing Group at Ulm University.

Meinrad Beer is the medical director of the Department of Diagnostic and Interventional Radiology at Ulm University. After his professorships in clinical radiology at the University of Wuerzburg and pediatric radiology at the Medical University of Graz, Austria, he has been appointed as a chairman of the Radiology Department at Ulm University in 2013.

Michael Götz is the head of the Experimental Radiology Section at Ulm University Hospital. Before moving to Ulm, he worked as postdoctoral researcher at German Cancer Research Center, where he also received his PhD in 2017.

Timo Ropinski is heading the Visual Computing Group at Ulm University. Before moving to Ulm, he was a full professor at Linköping University, Sweden. He obtained his PhD in computer science from the University of Münster, where he has also completed his habilitation.