Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Self-supervised Few-shot Learning for 3D Supervoxel-based Medical Image Segmentation

Fabian Zhang, Shang-Ching Liu

fabzhang@ethz.ch, shang-ching.liu@studium.uni-hamburg.de

October 20, 2022

Abstract

One of the main challenges in medical image segmentation is the scarcity in annotated images. As there are many different objects in the body to be segmented, it is impossible to find labeled 3D data on all different types of object classes in medical images. Also, the common challenging foreground-background imbalance problem persists. The unsupervised few-shot semantic segmentation approach, named SSL-APLNet, aims to solve these issues by using 2D slicewise generated superpixel-based pseudo-labels. In the means of improving this method, one essential step is to change the 2 dimensionality of this approach to keep the 3D image information intact. In this work we explore the approach to process the image in its entirety including depth, retaining volumetric contextual information using a 3D convolutional neural network. This 3D unsupervised few-shot learning framework has been implemented by using the abdominal MRI CHAOS dataset.

Contents

Chapter 1

Introduction

Manual medical image segmentation and labeling of 3D CT or MRI data is a highly labor intensive procedure requiring clinical expertise. Conventional fully supervised methods in image segmentation need unrealistically large amounts of labeled medical image data in order to perform well. Also, the number of possible biological segmentation targets in medical images is very high and it is impractical to cover them entirely by a single model. To solve these two problems, the self-supervised adaptive local prototypical network (SSL-ALPNet) [1] has been proposed to treat the annotated training data scarcity issue using self-supervision by superpixels and the issue of poor generalizability to unseen classes with few-shot learning. However, in the SSL-ALPNet, each 3D image is reformed as 2D slices to process, which causes losing some spatial information during the transformation.

Furthermore, manual slice-wise annotation of 3D volumes is tedious and inefficient due to similar information contained in neighboring slices. In our work, image representations are learned by extracting the 3D volumetric features in order to generalize to real semantic classes as prototypes. Our idea is utilizing a) the 3D U-Net structure [2] and b) the SSL-ALPNet pipeline to perform few-shot segmentation, which so far relies on an entirely 2D architecture, incapable of capturing contextual volumetric information. Our proposed network takes the 3D images directly as input, creates 3D pseudo-labels for self-supervised learning and processes the volumes with the 3D U-Net by its 3D operations.

Chapter 2

Related Work

Our research is motivated by the work "Self-supervised Learning for Few-shot Medical Image Segmentation" (SSL-APLNet) by Ouyang et al. [1], which provides the foundational pipeline for the self-supervised few-shot learning and the adaptive local prototype segmentation network. The idea of prototypical networks for few-shot learning stems from Snell et al. [3]. The proposed prototypical networks perform classification by computing metric distances to prototype representations, which are learned from only a small number of examples of new classes. This framework then has been extended to perform semantic segmentation by Wang et al. [4]. Their Prototype Alginment Network (PANet) matches pixel regions to learned prototypes of semantic classes and introduces prototype alignment regularization for better generalization and for making better use of the given data and is the foundation for the SSL-ALPNet.

In the discussion of Ouyang et al. [1], the benefits of switching from 2D to 3D networks have been described. To extract meaningful features in this setting, the 3D U-Net [2] by Çiçek et al. has been used, which itself is extending the architecture of the 2D U-Net [5] into the 3D space. The 2D U-Net has been proven to have good generalization from very few annotated samples and is trainable from end-to-end. Our work aims to combine the SSL-ALPNet framework with the benefits of utilizing 3D convolutional neural network and 3D prototypes to make use of the entire spatial information available.

Chapter 3

Methods

Our self-supervised few-shot learning framework for 3D segmentation can be split into two stages, consisting of an offline unsupervised pseudo-labels generation phase and an online episode based training phase of the ALP module, as in Fig. 3.1. First, in Section 3.1 the specific few-shot learning problem is introduced, which is using the supervoxels generated after Section 3.2 as training data. Finally, the means of performing segmentation and generating 3D prototypes using the modified ALPNet are then being presented in Section 3.3.

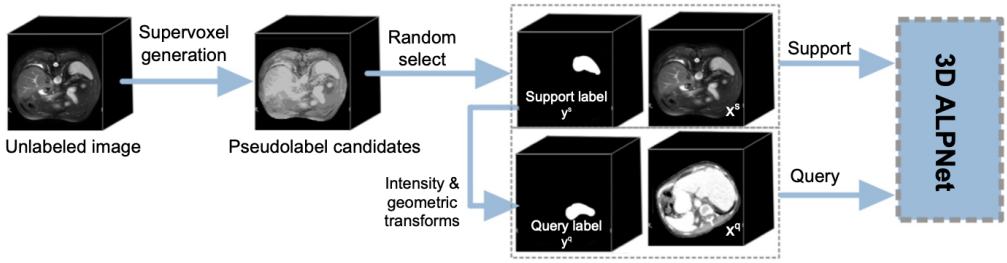

Figure 3.1: Workflow of the proposed supervoxel based self-supervised learning method.

3.1 Few-shot Learning

The concept of few-shot learning originates from image classification by segmenting and representing previously unseen classes. Few-shot learning is inspired by human learning in the sense of gathering knowledge from a few observations and using prior knowledge to learn. This is especially important in the case of the amount of annotated training data being scarce. In classic few-shot learning for image classification the task is to learn image class representations from support images, and assign the unseen query images to a set of N classes with K examples each, which is called N-way-K-shot classification. Compared to classical image

classification methods, the support set we use for training our network has very few examples to learn from. The problem of few-shot learning can be changed to a segmentation problem, by classifying each pixel to an object class. For this work, we train with one support image, and differentiate between foreground or background, a 1-way-1-shot configuration. This however can be adapted to have a different number of support images and number of classes. For segmenting these unseen query images, no further training or fine tuning is necessary. During the training phase of the network, generalizable image representations are picked up by the feature extractor network.

3.2 Pseudo-labels generation

The idea to bypass the reliance on manually annotated training images in this self-supervised framework is based on using generated 3D supervoxels as pseudo-labels to train the network for one-shot learning. This way, with only a couple images, as much support data as there are supervoxels is available, and as much query data as needed by using the augmentation methods in Section 3.2.2.

3.2.1 Supervoxels

Superpixels are groupings of pixels that naturally share image properties with real semantic objects in a 2D plane and have been used as clues for grouping features in supervised and unsupervised settings. The original paper uses the Felzenszwalb graph-based approach [6]. When processing a 3D image, it has to be viewed in 2D slices. The superpixels then are generated by the minimum spanning tree of their constituent pixels. While the 2D slice-wise generated superpixels make sense on the axial plane, viewing the superpixels from the coronal and sagittal view does not provide a coherent segmentation into superpixels, as seen in Fig. 3.2.

To capture the 3D information, simple linear iterative clustering (SLIC) [7] is used to generate supervoxels. SLIC is an iterative gradient ascent based algorithm based on k-means clustering, which is faster and more memory efficient. The SLIC supervoxels provide good 3D depth context coherent with real biological features and are in itself already an improvement to the self-supervised learning of the model. The parameter k can be set to the approximate number of segments wanted, for this work k has been set to 10. A comparison of superpixels in different views generated by [6] and using 3D SLIC can be seen in Fig. 3.2.

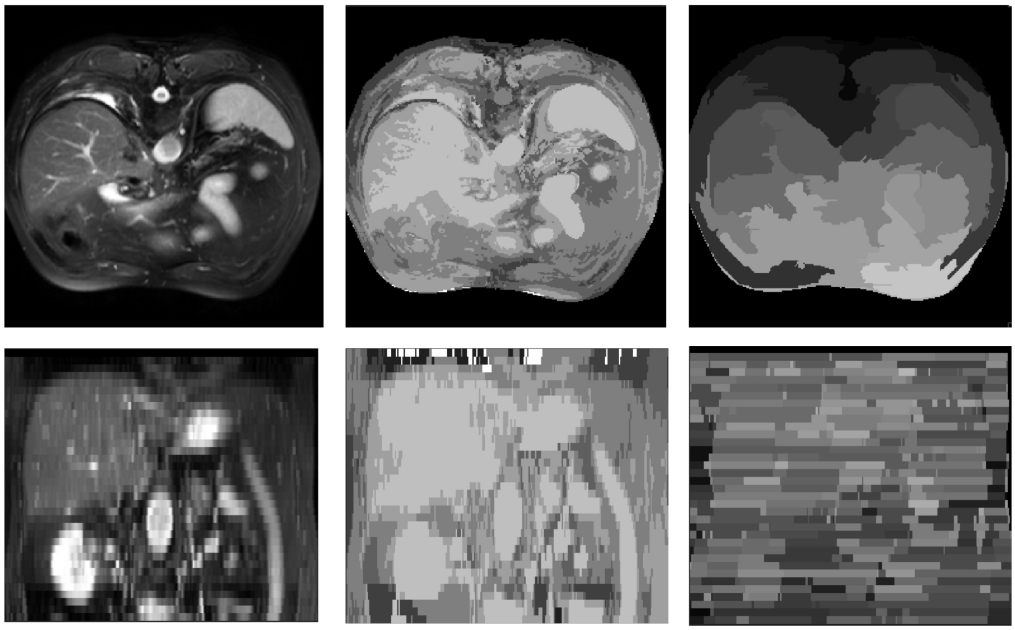

Figure 3.2: The same slice of the same image in axial view (top) and corronal view (bottom), original (left), SLIC (middle), Felzenszwalb (right).

3.2.2 Pseudo-labels transformations

To obtain the query set, geometric and intensity transformations in 3D from the Torch-IO data augmentation functions[1] are being applied to the corresponding support image. The geometric transformations include a random affine transformation and a random elastic deformation. For intensity transformation a random gamma correction is applied.

[1] https://torchio.readthedocs.io/transforms/augmentation.html

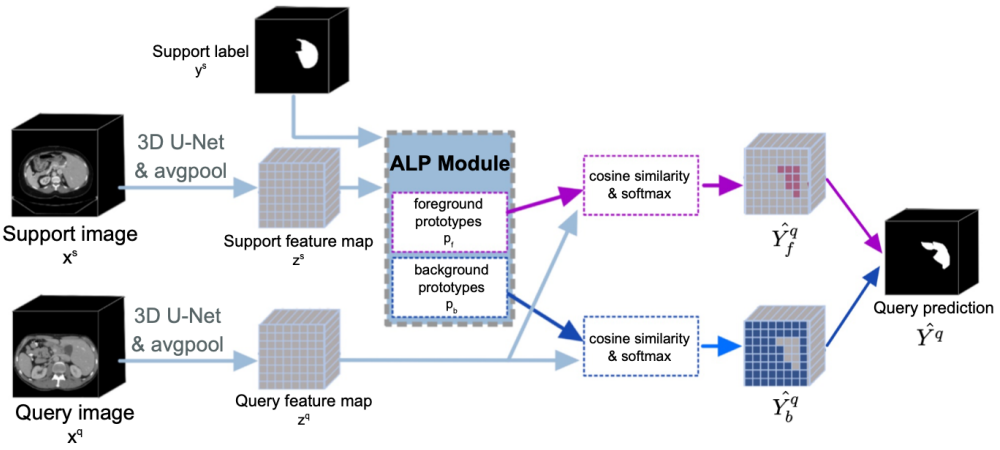

Figure 3.3: The 3D ALP-Net workflow. Support and query image both go through the 3D U-Net and support feature maps are formed after an average pooling operation. With the support feature map and the support label, foreground and background prototypes are calculated using the ALP module. A prediction each is formed with a grid-wise cosine similarity and softmax calculation, with the final prediction being a fusion between the foreground prediction and the background prediction.

3.3 3D ALPNet

The ALPNet takes a query and support image with its corresponding label as input and gives out the query segmentation prediction. As the code[2] from Ouyang et al. [1] was written entirely according to 2D slice-wise segmentation prediction, the entire code for the ALPNet to process 3D data had to be written from scratch. The 3D ALPNet can be divided into three main compontents:

n

[2] https://github.com/cheng-01037/Self-supervised-Fewshot-Medical-Image-Segmentation

3.3.1 3D U-Net Architecture

In the orignal ALPNet [8], a generic feedforward convolutional neural network, more specifically the deeplabv3 based on a Res-Net 101 backbone [9] was used. The deeplabv3 is pre-trained on the MS-COCO dataset, which in itself already gives the network a good segmentation performance. However, this network only has been trained and is capable of processing 2D images, which implies that in the 3D medical image segmentation case the image needs to be processed in 2D slices.

The answer is to look for a 3D convolutional image processing network, however as the training image data scarcity problem prevails, currently the pre-trained 3D CNNs that fit our use case don't exist. In comes the 3D U-Net from Çiçek et al. [2], which can be trained end-to-end from scratch on very few annotated volumes. The 2D U-Net [5] has been proven to have good generalization from very few annotated samples. For biomedical images a small number of image samples is required to train the network. The U-Net architecture benefits from the fact that properly applied rigid transformations and slight elastic deformations still yield biologically plausible images to be used for training.

Fig. 3.4 illustrates the network architecture. The network consists of a contracting encoder, which analyzes the whole image (analysis path), and an expanding decoder to upscale back to a segmentation in original resolution (synthesis path). In contrast to the 2D U-Net, which takes 2D slices of the image, the 3D implementation[3] takes the 3D volumes directly as input.

The network consists of encoder and decoder blocks, with the encoder block consisting of two 3D $3 \times 3 \times 3$ convolution layers, a 2×2 3D max pooling layer, batch normalization for faster convergence, followed by a leaky rectified linear unit (ReLu). The decoder blocks have the same structure, but instead of the max pooling layer they use a 3D up-convolution layer with strides 2 in each dimension to get back to a higher resolution. With the max pooling, going down the encoding path, the features get a lower resolution but with deeper features. Shortcut connections from layers of same resolution provide essential high resolution features to the synthesis path in form of concatenation.

The output of the network yields a feature map of the same size, but with number of feature maps $f_{maps} = 32$ channels. The number of network parameters is in linear relation to f_{maps}, and with $f_{maps} = 32$ we get 4.1M parameters in total.

[3] https://github.com/wolny/pytorch-3dunet

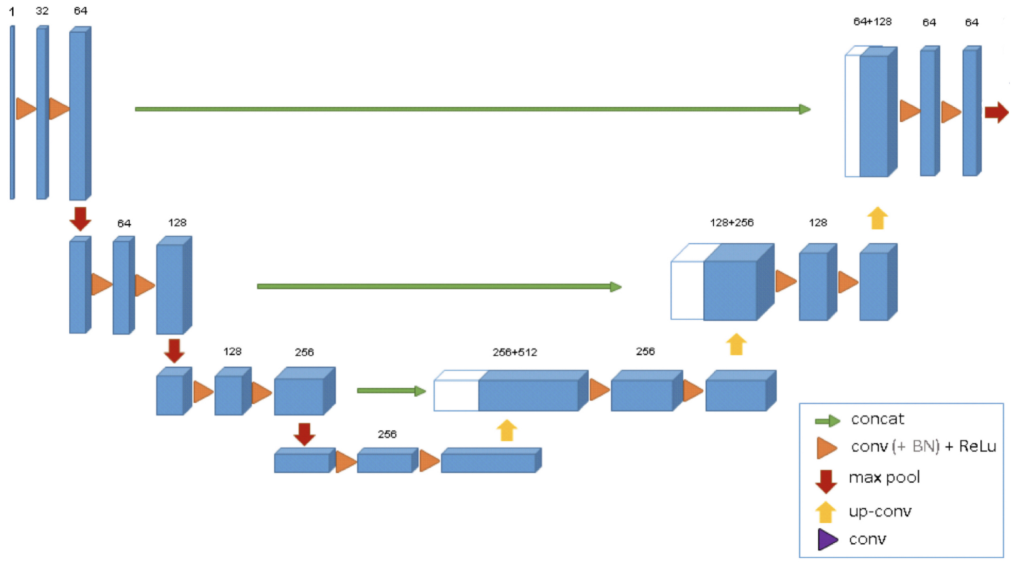

Figure 3.4: The adapted 3D U-Net architecture [2]. During the whole path, a 3D volume is processed under different resolutions with a different number of channels, which is denoted above each step.

3.3.2 3D Prototypes

The 3D adaptive local prototype pooling module creates local and class level prototypes using the support feature map z^s and the binary mask label y^s. It builds on top of the PANet [4], which uses class level masked average pooling. The ALP module additionally uses local prototypes to capture location sensitive information according to the LSNet [10]. The ALP module differentiates between foreground class level prototypes p_f, which are focused on the area to be segmented, and local background prototypes p_b. The binary mask label y^s is average pooled to the same spatial size as the network output with window sizes L_h, L_w and L_d in each dimension. The class of each prototype is then assigned after the threshold T, which has been set to 0.95.

Local Background Prototypes

The location sensitive LSNet uses local information within the corresponding grid in the image to calculate the background object prototypes. This works under the assumption support and query images have similar spatial layout. Another benefit is dividing the difficult task of entire image segmentation into easier ones to calculate sub-problems. The grid window size is a critical parameter that determines the location sensitivity of the prototypes to be further experimented

on. As output, a set of prototypes along the grid are calculated as follows:

$$p_b(l,m,n) = avgpool(z^s)(l,m,n) = \frac{1}{L_h L_w L_d} \sum_l \sum_m \sum_n z^s(l,m,n) \tag{3.1}$$

where

$hL_H < l < (h+1)L_H$ with $h = 0, 1, 2, ..., H/L_H$,

$wL_W < m < (w+1)L_W$ with $w = 0, 1, 2, ..., H/L_W$,

$dL_D < n < (d+1)L_D$ with $d = 0, 1, 2, ..., H/L_D$.

Class-level Foreground Prototypes

The class-level foreground prototypes are an ensemble between local global prototypes as in [4]. Averaging the support feature map z with local pooling windows, giving us $l \times m \times n$ local prototypes as by the equation above.

The global foreground prototype features are calculated using masked average pooling, by spatially averaging the support feature map z under the binary foreground mask. The local and global prototypes are concatenated into prototype ensembles:

$$p^g = \frac{\sum_l \sum_m \sum_n y^s(l,m,n) z^s(l,m,n)}{\sum_l \sum_m \sum_n y^s(l,m,n)} \tag{3.2}$$

3.3.3 Similarity-based Segmentation Prediction

The segmentation is performed by a similarity based prediction process, where the distance between prototypes and query features are being computed locally. Each spatial location in the prediction grid (l,m,n) corresponds to similarity $S_c(l,m,n)$ between prototype and query feature map as follows:

$$S_c(l,m,n) = \alpha sim(p_k, z^q(l,m,n)), \tag{3.3}$$

where sim computes the cosine similarity of inputs.

During training, α controls the significance for local similarity and has been set to $\alpha = 20$ to be in line with [4]. The 3D prediction probability grid score is given by a Softmax operation between both prototypes of the cosine similarity.

$$\hat{Y^q_{f,b}}(l,m,n) = \underset{c}{softmax}(S_c(l,m,n)) \tag{3.4}$$

The final prediction is a fusion between the foreground and background prototype prediction, and then interpolated back to original input image size. β is the weighting of foreground and background, set to 0.5.

$$\hat{Y^q} = \frac{\hat{Y_f^q} + (1-\beta)\hat{Y_b^q}}{2} \tag{3.5}$$

3.3.4 Loss Function

As the evaluation metric, the commonly used Dice score is employed. It measures the overlap between the predicted segmentation and the ground truth label and works seamlessly in 3D. Derived from the Dice score, the Dice loss[4] between the predicted query mask M and query ground truth is calculated as in (3.6).

$$L_{dice}(\hat{Y}, Y) = 1 - \frac{2Y\hat{Y} + 1}{\mathrm{Y} + \hat{Y} + 1} \tag{3.6}$$

To further improve generalization and boost performance, prototypical alignment regularization as in [4] has been employed. After the prediction of the query image using the support prototypes, the query image and its prediction is used again as a support image itself to predict the original support image. This new loss is added as the regularization loss L_{reg} to the training objective and the regularization weight λ has been set to 1.

$$L = L_q(\hat{Y_q}, Y_q) + \lambda L_{reg}(\hat{Y_s}, Y_s) \tag{3.7}$$

[4] https://github.com/wulalago/DAF3D/blob/master/Utils.py

Chapter 4

Experiments

4.1 Datasets

To test and train our proposed method, the abdominal MRI dataset from the CHAOS challenge[1] [11] was used. The dataset contains 30 3D MRI scans as well as the ground truth labels for liver, left kidney, right kidney and spleen in order to calculate the validation score. Depending on to the memory capability of the GPU, the experiments need to run on correspondingly down-sampled versions of the original image resolution, $256 \times 256 \times 30$ in x, y, z dimensions.

4.2 Training

Due to GPU memory limitations we use, the images had to be down-sampled to a size of $128 \times 128 \times 24$ voxels. The grid size for prototype training has been set to adaptively be $8 \times 8 \times 4$, making the local pooling windows (L_H, L_W, L_D) sizes $16 \times 16 \times 6$. 20 images were randomly split into 16 for the training set and 4 for the validation set. 5 fold cross validation was applied, as well as an AdamW optimizer with batch size 1, and one cycle scheduler policy with a maximum learning rate of 0.0001. The training ran for 2000 epochs, 16 iterations per epoch, and took 20h on a single Nvidia RTX 3090 GPU, consuming 20.7 GBs of memory. The best model was saved as well as checkpoints for every 200 epochs.

[1] https://chaos.grand-challenge.org/Combined_Healthy_Abdominal_Organ_Segmentation/

4.3 Results

The training and validation losses can be seen in Fig. 4.1 and Fig. 4.2. It shows that the model starts to learn the pseudo-labels representations after 900 epochs. The support loss is the alignment regularization loss in Equation (3.7) and not expected to be at the same level as the query loss L_q, but also shows steady decrease. After a correction of the validation loss the model continues to run as in Fig. 4.2.

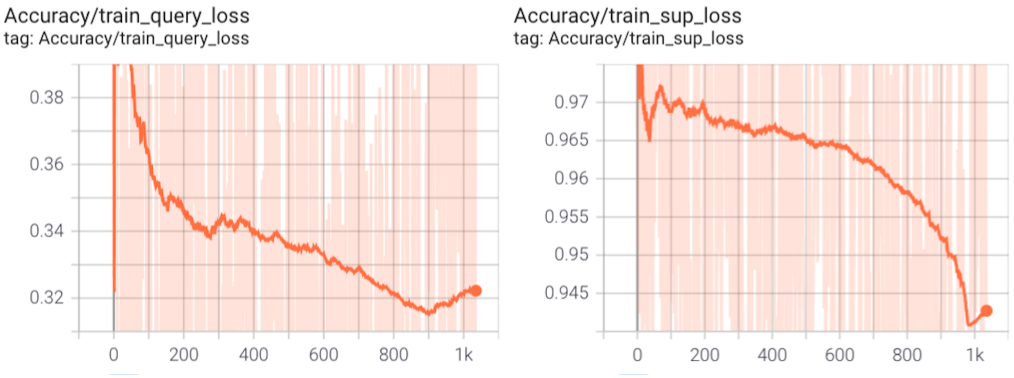

Figure 4.1: Training losses until the 1000th epoch.

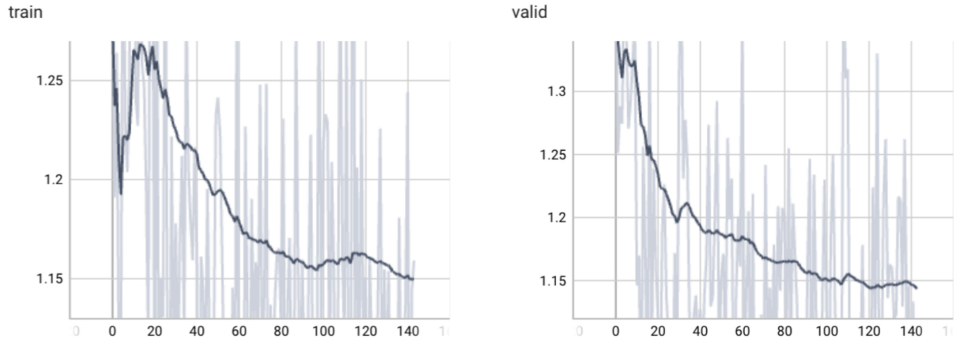

Figure 4.2: Training and validation loss from the checkpoint after 1000 epochs.

After testing our method on segmenting different classes (left kidney, right kidney, spleen, liver) of abdominal MRI images, we get following results as in Table 4.1. Due to our model being completely unsupervised until this point, we introduce a few labeled images as training data as real ground truth support. After the unsupervised training phase of the model, the training process is continued with 4 images with ground truth annotations. This improves the Dice segmentation score on the real classes significantly, denoted in Table 4.1 as 3D SSL-ALPNet with 20% annotated images.

Method	Mean Dice score
SE-NET [12]	50.66
Vanilla PANet [4]	46.33
ALPNet-init [1]	23.93
ALPNet	50.43
SSL-PANet [4]	54.85
SSL-ALPNet [4]	73.02
3D SSL-ALPNet self-supervised	**62.7**

Table 4.1: Mean segmentation experiment results of kidneys, spleen and liver on abdominal MRI images.

Chapter 5

Conclusion and Future Work

In this project, an end-to-end self-supervised learning method for few-shot segmentation using 3D methods is introduced. It outlines the concept of how to process the 3D volume information and learn without any manual labels during training. Because the 3D U-Net is trained from scratch, a long running time is required. Çiçek et al. took 3 days (30000 iterations) of training to get feasible results. The 3D U-Net requires considerable more training time than fine-tuning the SSL-APLNet, which uses a pre-trained image segmentation network. Although the training process for our model remains highly resource intensive, our idea of running the self-supervised few-shot learning using 3D image processing has been validated.

This work introduces the concept of processing 3D volumes and capturing its contextual. Further optimizations regarding network parameters, prototyping grid sizes, weights, and methods as well as other methods of augmentation should be tested. Regarding the supervoxel-pseudo-labels, the SLIC algorithm generates many noisy pseudo-labels that disrupt training. New methods to create more biologically feasible groupings can be further explored.

Concerning local prototypes, the strict rectangular prototyping grid can break the contextual semantics of the background texture and should be studied for improvement. A very limiting factor during training remains to be the image resolution, due to the bottleneck of GPU memory. To work with and capture higher resolution image volumes, patch-wise segmentation can be considered. With these adjustments, we look forward to keep improving the self-supervised 3D ALPNet framework further.

Bibliography

[1] C. Ouyang, C. Biffi, C. Chen, T. Kart, H. Qiu, and D. Rueckert, "Self-supervision with superpixels: Training few-shot medical image segmentation without annotation," *IEEE Trans Med Imaging*, vol. Jul;41(7), pp. 1837–1848, 2022.

[2] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, *CoRR*, vol. abs/1606.06650, 2016. [Online]. Available: http://arxiv.org/abs/1606.06650

[3] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *CoRR*, vol. abs/1703.05175, 2017. [Online]. Available: http://arxiv.org/abs/1703.05175

[4] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, "PANet: Few-shot image semantic segmentation with prototype alignment," *CoRR*, vol. abs/1908.06391, 2019. [Online]. Available: http://arxiv.org/abs/1908.06391

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[6] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, Sep 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000022288.19776.77

[7] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[8] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[10] Q. Yu, K. Dang, N. Tajbakhsh, D. Terzopoulos, and X. Ding, "A location-sensitive local prototype network for few-shot medical image segmentation," *CoRR*, vol. abs/2103.10178, 2021. [Online]. Available: https://arxiv.org/abs/2103.10178

[11] A. E. Kavur, N. S. Gezer, M. Barış, S. Aslan, P.-H. Conze, V. Groza, D. D. Pham, S. Chatterjee, P. Ernst, S. Özkan, B. Baydar, D. Lachinov, S. Han, J. Pauli, F. Isensee, M. Perkonigg, R. Sathish, R. Rajan, D. Sheet, G. Dovletov, O. Speck, A. Nürnberger, K. H. Maier-Hein, G. B. Akar, G. Ünal, O. Dicle, and M. A. Selver, "CHAOS challenge - combined (CT-MR) healthy abdominal organ segmentation," *Medical Image Analysis*, vol. 69, p. 101950, apr 2021. [Online]. Available: https://doi.org/10.1016%2Fj.media.2020.101950

[12] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "'squeeze & excite' guided few-shot segmentation of volumetric images," *CoRR*, vol. abs/1902.01314, 2019. [Online]. Available: http://arxiv.org/abs/1902.01314