OSS-Net: Memory Efficient High Resolution Semantic Segmentation of 3D Medical Data

Christoph Reich
christoph.reich@bcs.tu-darmstadt.de
Tim Prangemeier
tim.prangemeier@bcs.tu-darmstadt.de
Özdemir Cetin
oezdemir.cetin@bcs.tu-darmstadt.de
Heinz Koeppl
heinz.koeppl@bcs.tu-darmstadt.de

Centre for Synthetic Biology,
Department of Electrical Engineering and Information Technology,
Department of Biology,
Technische Universität Darmstadt

Abstract

Convolutional neural networks (CNNs) are the current state-of-the-art meta-algorithm for volumetric segmentation of medical data, for example, to localize COVID-19 infected tissue on computer tomography scans or the detection of tumour volumes in magnetic resonance imaging. A key limitation of 3D CNNs on voxelised data is that the memory consumption grows cubically with the training data resolution. Occupancy networks (O-Nets) are an alternative for which the data is represented continuously in a function space and 3D shapes are learned as a continuous decision boundary. While O-Nets are significantly more memory efficient than 3D CNNs, they are limited to simple shapes, are relatively slow at inference, and have not yet been adapted for 3D semantic segmentation of medical data. Here, we propose Occupancy Networks for Semantic Segmentation (OSS-Nets) to accurately and memory-efficiently segment 3D medical data. We build upon the original O-Net with modifications for increased expressiveness leading to improved segmentation performance comparable to 3D CNNs, as well as modifications for faster inference. We leverage local observations to represent complex shapes and prior encoder predictions to expedite inference. We showcase OSS-Net's performance on 3D brain tumour and liver segmentation against a function space baseline (O-Net), a performance baseline (3D residual U-Net), and an efficiency baseline (2D residual U-Net). OSS-Net yields segmentation results similar to the performance baseline and superior to the function space and efficiency baselines. In terms of memory efficiency, OSS-Net consumes comparable amounts of memory as the function space baseline, somewhat more memory than the efficiency baseline and significantly less than the performance baseline. As such, OSS-Net enables memory-efficient and accurate 3D semantic segmentation that can scale to high resolutions.

Code and trained models are available at https://github.com/ChristophReich1996/OSS-Net.

1 Introduction

Transferring established 2D deep learning solutions, such as CNNs, to dense 3D data entails various issues. A significant limitation of this approach, is the increased computational complexity and memory consumption for processing dense 3D volumes. Nonetheless, state-of-the-art approaches for segmenting 3D medical data rely on 3D CNNs [4, 5, 9, 18, 25, 27]. The main drawback of training 3D CNNs, to learn a voxelised mapping, is the cubic growth of the memory and computational complexity with the training voxel resolution [22]. In practice, the available GPU memory limits the voxel resolution, resulting in trade-offs between memory usage, segmentation resolution, model capacity, and inference speed [19, 27, 41].

Deep learning approaches that leverage 3D representations beyond dense voxelized volumes are available to overcome the cubical complexity of 3D CNNs [13, 22, 33, 36]. Implicit neural representations, such as occupancy networks [22], have been proposed for various 3D learning tasks [6, 23, 29]. O-Nets learn a continuous decision boundary in a function space to represent 3D shapes. Predicting occupancy values in training, instead of a dense voxelised representation, enables O-Nets to be significantly more memory efficient than CNNs on 3D data [22]. Despite the memory efficiency, O-Nets are relatively slow at inference and their expressiveness is limited in comparison to CNNs [22, 30]. To the best of the author's knowledge, O-Nets have not yet been modified to jointly overcome these shortcomings in 3D segmentation, or to segment 3D medical data.

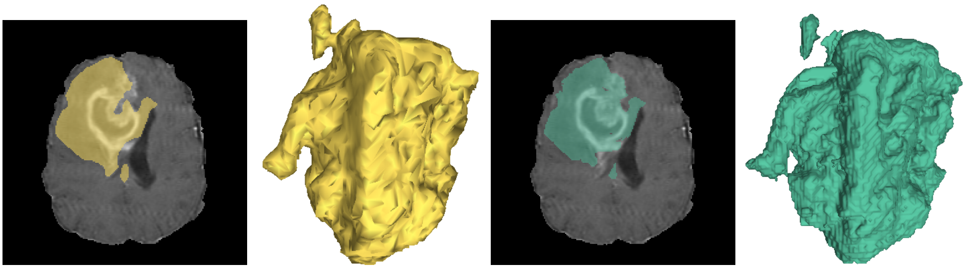

Figure 1: Brain tumour segmentation results of OSS-Net (config. C) on the BraTS 2020 dataset [2]. Brain tumour prediction in yellow 🟨 and label in green 🟩. 2D MRI slice (Tc1 modality) overlaid with the corresponding voxelized prediction or label on the left and the corresponding extracted mesh on the right. Best viewed in color.

In this study, we propose Occupancy Networks for Semantic Segmentation to combine the performance of 3D CNNs with the memory efficiency of O-Nets while avoiding the drawbacks of the respective methods. We modify the original O-Net architecture by leveraging local observations. This enables OSS-Net to represent more complex 3D shapes, such as the fine tumour structure depicted in Figure 1. Inference speed is improved by utilising a prior prediction of the OSS-Net encoder as an initial state of the inference approach. We demonstrate the performance on the BraTS 2020 dataset for 3D brain segmentation [1, 2, 21] and the LiTS dataset for 3D liver segmentation [3]. On both datasets, OSS-Net significantly surpass the original O-Net's segmentation performance, while being similarly memory efficient. OSS-Net performs on par in segmentation accuracy with the voxelised baseline (3D residual U-Net [9, 27]) for brain tumour segmentation while falling slightly short of this baseline on the smaller LiTS dataset. OSS-Net makes it feasible to scale to high voxel resolutions (512^3) while the memory consumption remains manageable.

2 Related Work

Recent approaches towards memory efficient processing of three-dimensional data with deep neural networks can be clustered into two categories. The first category is based around making existing 3D CNNs architectures more efficient. RevNet by Gomez *et al.* [14] is an example of this and employs mathematically reversible building blocks, avoiding the need to store all activations during training. Brugger *et al.* employed this approach for semantic segmentation of 3D medical data, achieving a 30% reduction in memory cost in comparison to a baseline 3D U-Net [4]. Another avenue towards increased efficiency of existing CNNs is to utilise sparse convolutions, as proposed by [8, 15]. This approach is of limited applicability to medical data, where the required spare data is seldom given [3, 21]. A practical approach for increasing the memory efficiency of 3D biomedical segmentation with CNNs is, to reduce the 3D segmentation problem to a 2D problem by slicing and utilising a 2D CNN, for example, a 2D U-Net [31, 32, 42, 43]. While these approaches can achieve high memory efficiency, they inherit the strong limiting assumption that a certain 3D segmentation task can be solved in 2D [12, 31], neglecting 3D relations and limiting their applicability to general 3D problems [34]. Recent refined slicing-based approaches also include an additional hand-designed fusion step, adding complexity to the segmentation pipeline [31, 42].

The second category is to consider alternative approaches to represent 3D data beyond voxelised volumes, such as 3D point clouds or 3D meshes. Both representations have recently been used to perform 3D semantic segmentation [13, 16, 33, 36]. 3D point cloud methods typically require heavy post-processing, and the available GPU memory limits 3D mesh approaches [22]. Furthermore, both 3D point cloud and 3D mesh approaches are challenging to apply to dense and high-resolution 3D medical data.

An alternative representation that is potentially significantly more efficient was recently proposed by Mescheder *et al.* [22]. The proposed O-Net learns a continuous decision boundary in a 3D function space to reconstruct a 3D shape. The resulting continuous decision boundary can be extracted at an arbitrary resolution. Compared to voxelised architectures, O-Nets have a vastly higher memory efficiency since the typically used CNN decoder is replaced with a fully-connected occupancy decoder [22].

Occupancy networks and related implicit representation approaches, such as [6, 23, 29], are memory efficient yet limited to represent simple 3D shapes and do not generalize well to unseen shapes [30]. This issue is mainly caused by the lack of local features in the occupancy decoder [30]. To overcome this limitation, multiple extensions have been proposed [7, 11, 24, 30, 35, 38]. These extensions are either not applicable to the task of 3D semantic segmentation, require highly complex architectures, or sacrifice memory efficiency.

3 Occupancy Networks for Semantic Segmentation

An occupancy network is described by the learnable mapping $f_\theta : \mathbb{R}^3 \times \mathcal{X} \rightarrow [0,1]$ [22]. The mapping is parameterised by θ. The inputs are a 3D location $p \in \mathbb{R}^3$ and an observation $x \in \mathcal{X}$. The output is the occupancy probability $o \in [0,1]$ that describes whether the input location lies within or outside a continuous decision boundary. In the case of semantic segmentation, the observation x is a 3D volume that is encoded by a 3D CNN to a global latent representation x_l. Since only a global representation x_l of the observation x is produced, the occupancy network $f_\theta\left(p,x\right)$ cannot capture small details in practice [30].

To overcome the lack of local information in our OSS-Net occupancy encoder, we extend the original learnable mapping f_θ to

$$f_\theta : \mathbb{R}^3 \times \mathcal{X} \times \mathcal{Z} \to [0,1], \tag{1}$$

with a local observation $z \in \mathcal{Z}$ as an additional input. The local observation z is a local 3D patch sampled from the global observation x centered at the 3D location p which is encoded to a local latent representation. Employing a local latent representation for each location as the input to the occupancy encoder solves the issue of missing local information. Equation 1 can readily be adapted to multi-class segmentation by chaining the last layer to predict a softmax vector instead of a scalar occupancy probability.

3.1 Architecture

We implemented OSS-Net, described by Equation 1 as a deep neural network composed of three main components. An overview of the OSS-Net architecture is provided in Figure 2. First, the 3D CNN encoder maps a downscaled input volume to a global latent vector. Second, the patch encoder, consisting of two 3D convolutions, encodes n 3D patches and produces n local latent vectors. Third, based on the global and the n corresponding local latent vectors, the fully connected occupancy decoder produces an occupancy probability prediction for each of the n input coordinates.

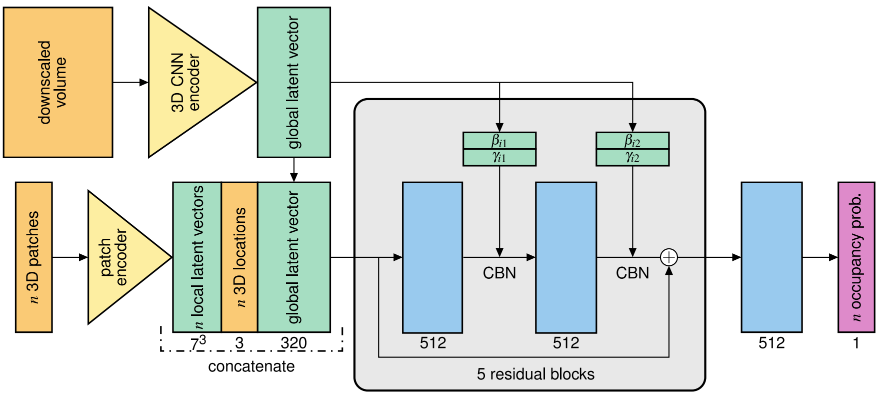

Figure 2: Architecture of the OSS-Net, with downscaled global volume, n local 3D patches, and n 3D locations (in orange 🟧) as inputs. The 3D CNN encoder (in yellow 🟨) extracts a global latent vector (in green 🟩). The patch encoder produces n local latent vectors (in green 🟩). The global and local latent vectors as well as the n 3D locations are concatenated and fed into five residual fully connected blocks with conditional batch normalization (β & γ predicted parameters) to produce n occupancy probability predictions (in purple 🟪). Best viewed in color.

The task-specific 3D CNN encoder (Fig. 3) exhibits a ResNet [17] like architecture. We utilise output skip-connections in the encoder, as introduced by [38]. These incorporate features of each encoder stage directly into the global latent vector. The output of the lowest encoder stage predicts a low-resolution voxelized segmentation of the input volume. This segmentation is used in an auxiliary loss during training (Sec. 3.2). At inference the segmentation is used to speed up the dense segmentation extraction approach (Sec. 3.3).

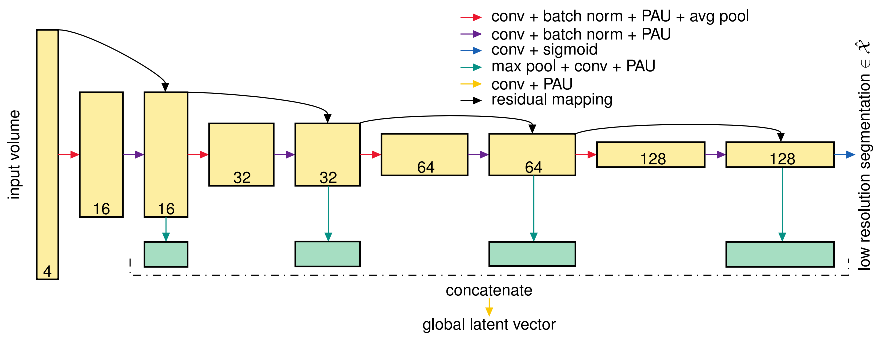

Figure 3: Residual 3D CNN encoder with output skip-connections. Operations indicated with an arrow, feature tensors colored in yellow 🟨 and intermediated latent tensors visualized in green 🟩. Channel dimensions shown in feature tensors are corresponding to BraTS network setting. Best viewed in color.

3.2 Training and Evaluation

Training the neural network, $f_\theta(p,x,z)$ (Eq. 1), was approached by sampling n 3D locations p and corresponding local patches z from the input volume x to be segmented. The predicted occupancy probabilities were supervised by a binary cross-entropy loss (first term)

$$\mathcal{L} = -\frac{1}{kn}\sum_{i=1}^{k}\sum_{j=1}^{n}\left[o_{ij}\log(f_\theta(p_{ij},x_i,z_{ij})) + (1-o_{ij})\log(1-f_\theta(p_{ij},x_i,z_{ij}))\right]$$
$$-\alpha\frac{1}{kw}\sum_{i=1}^{k}\sum_{m=1}^{w}\left[y_{im}\log(f_\theta^e(x_i)_m)) + (1-y_{im})\log(1-f_\theta^e(x_i)_m)\right]. \quad (2)$$

The main loss (first term) is computed over n sampled locations, with it's corresponding ground truth label o, in a mini-batch of size k.

Sampling n 3D locations from the global volume is performed by uniform sampling and border region sampling. In the border region sampling, one half of the locations are sampled uniformly from the 25 voxel wide border of the ground truth segmentation hull. The other half of the location is sampled uniformly over the whole volume.

To guide the encoder to learn relevant features for the segmentation, we employed and auxiliary endocer loss in addition the main occupancy loss (Equation 2 second term). The auxiliary loss (Equation 2 second term) is also a binary cross-entropy loss, computed between the low-resolution voxelized segmentation (voxels indexed by m) prediction of the encoder segmentation mapping $f_\theta^e : \mathcal{X} \to \hat{\mathcal{X}}$ and the downsampled ground truth voxelised label y. The weighting factor $\alpha \in \mathbb{R}$ was 0.1.

We followed the original O-Net approach, uniformly sampling 2^{17} locations and patches from the global volume, to validate the OSS-Net during training [22]. The resulting occupancy predictions and corresponding labels were utilised to compute the Intersection over Union (IoU) $\frac{|\mathbf{o}\cap\hat{\mathbf{o}}|}{|\mathbf{o}\cup\hat{\mathbf{o}}|}$ and the Dice score $\frac{2|\mathbf{o}\cap\hat{\mathbf{o}}|}{|\mathbf{o}|+|\hat{\mathbf{o}}|}$ between the sampled ground truth labels $\mathbf{o}$ and the corresponding predictions $\hat{\mathbf{o}}$.

3.3 Inference

At inference time, the goal is to extract the predicted decision boundary of the OSS-Net. This can be done by utilizing the Multiresolution IsoSurface Extraction (MISE) [22] algorithm proposed with the original O-Net. We built upon this approach with the aim to reduce the runtime while preserving the prediction accuracy.

MISE starts by building an octree and incrementally evaluates the continuous decision

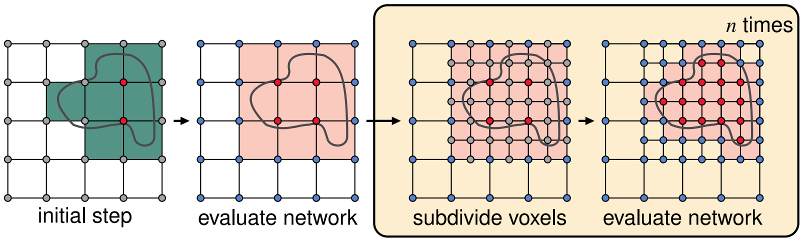

Figure 4: Our improved dense segmentation extraction approach in 2D. Initial upsampled and thresholded segmentation of the 3D CNN encoder in green 🟩, occupied coordinates in red 🟥, unoccupied coordinates in blue 🟦, coordinates to be evaluated in gray ⬜, and the current voxelized segmentation in pink 🟫. Best viewed in color.

boundary [22]. In the next step, the produced dense prediction is used to extract a mesh which is subsequently smoothed [22]. An example of a mesh extracted with this approach is shown in Figure 1. Medical use-cases typically only require dense voxelised predictions and we omitted the mesh extraction and smoothing in practice. We refer to MISE without mesh extraction and smoothing as the original O-Net inference approach.

Our modified inference approach (Fig. 4) utilises the low-resolution dense prediction of the encoder as the initial state of the octree. This results in faster inference since fewer locations have to be evaluated by the OSS-Net decoder. The low-resolution encoder segmentation is thresholded and used to determine which locations are occupied in the initial octree. All unoccupied locations are evaluated by the OSS-Net decoder to complete the first evaluation stage. In the following stages, the voxels of the current segmentation are subdivided and then refined by additional network evaluations. This process is repeated until the desired resolution is reached. If the desired resolution surpasses the resolution of the data sample, patches can be queried from the sample using trilinear interpolation. A 2D visualization of OSS-Net's expedited inference approach is presented in Figure 4.

4 Experiments

We analyse the segmentation performance (Tab. 2 & 1), memory consumption (Tab. 3 & Fig. 6), and inference speed (Fig. 5) of the proposed OSS-Net in binary semantic segmentation experiments on the BraTS 2020 and the LiTS challenge dataset, and compare these to an original O-Net and a voxelised CNN baseline. We investigate the efficacy of the proposed modifications by adding these to the original O-Net incrementally. All segmentation results are based on 3 training runs with different random seeds, with the best result reported.

4.1 Baselines and Implementation Details

An original O-Net [22] serves as our function space baseline. We employed a basic 3D ResNet encoder without output skip-connection in the original O-Net. Five residual fully connected blocks with 512 features were utilised as the occupancy encoder. Conditional batch normalization [10] between the encoder and decoder was used alongside a concatenation of the global latent vector with the input of the occupancy encoder as in [28] (Fig. 2).

All function space models were trained for 50 epochs with a combination of the Lookahead [39] and RAdam [20] optimiser. The learning rate was 3×10^{-4} except for the weights of the Padé Activation Units [26], where a learning rate of 10^{-2} was employed. Each learning rate was decreased by an order of magnitude after 20 and 30 epochs. The Lookahead optimiser parameters k and α were 5 and 0.8, respectively.

A 3D residual U-Net [9, 25, 27, 40] trained on the binary cross-entropy loss served as our voxelised performance baseline. The U-Net encoder is comprised of four residual blocks with 8, 32, 128, and 384 filters, respectively. In the decoder three residual blocks with 128, 32, and 8 respectively are utilised. Group normalization was employed in both the encoder and decoder [37]. The voxelised baseline was trained for 20 epochs on the same optimiser configuration as the function space approaches with a learning rate of 10^{-3}.

We employed a 2D residual U-Net [40] with group normalization [37] as a memory-efficient slicing baseline. Five residual encoder blocks (128, 256, 384, 512, and 640 filters) and four residual encoder blocks (512, 384, 256, and 128 filters) were used. We used the same loss and optimiser configuration (learning rate 10^{-4}) as for the voxelised baseline for training (50 epochs). On each dataset, we trained three networks for three separate slice orientations, to investigate and demonstrate their effect on segmentation performance. We slice along each spatial dimension (height, width, depth) and choose the best performing orientation, for a fair comparison.

All training runs were performed on two Nvidia V100 16GB. For all function space models, we employed a batch size of 8 on the BraTS 2020 dataset experiments and a batch size of 4 on the LiTS dataset. The voxelised baseline was trained with a batch size of 2. The slicing baseline used batch size of 32 and 12 for the BraTS 2020 and LiTS datasets, respectively. All inference tests were performed on a single consumer-grade Nvidia RTX 2080 Ti.

4.2 Datasets and Data Augmentation

The BraTS 2020 [1, 2, 21] and the LiTS [3] dataset were utilised for binary semantic segmentation. We trained the networks for segmenting whole brain tumours and full livers on the the BraTS 2020 and LiTS datasets, respectively. The BraTS 2020 dataset was utilised in its native resolution of $240 \times 240 \times 155$, which is zero-padded to $256 \times 256 \times 160$. The variably sized CT scans and pixel-wise labels of the LiTS dataset were resized to a fixed voxel resolution of 512^3. The voxelised baseline was trained on downscaled CT scans with 224^3 voxels, to match the limited available GPU memory of 16GB. At inference time the predictions are trilinearly upsampled to the native training resolution. For reproducibility we only employed the publicly available samples of the BraTS 2020 and LiTS dataset. We split the BraTS 2020 dataset randomly into 320 training and 45 validation samples, as well as the LiTS dataset into 111 training and 20 validation samples. Flipping, brightness adjustment, and Gaussian noise injection was used for data augmentation. Each augmentation was applied with a probability of 0.5 for all training runs.

4.3 Experimental Results

We observed a high impact of the sampling strategy on the segmentation performance, in preliminary experiments. Therefore, we conducted experiments to investigate the effect of different sampling strategies and the number of sampled locations, summarised in Table 1.

On the BraTS 2020 dataset, the uniform sampling outperforms the border sampling approach, while results on the LiTS dataset are in favor of the border sampling approach. The best segmentation results were achieved for 2^{14} sampled location, regardless of the dataset. The segmentation performence dropped slighlty for 2^{13} sampled locations, and merely a single training run converged to a competitive result. The worst LiTS training run with 2^{13} samples and border sampling converged to an IoU of 0.5365. Based on these preliminary findings, we employed uniform sampling for the experiments on the BraTS 2020 dataset and border sampling for the LiTS dataset experiments, both with 2^{14} sampled locations.

Table 1: Numerical comparison of different sampling strategies on the BraTS 2020 and LiTS dataset. OSS-Net variant C (Tab. 2) utilised.

		BraTS 2020		LiTS	
Sampling strategy	Samples	Dice ↑	IoU ↑	Dice ↑	IoU ↑
Uniform	2^{13}	0.8816	0.7941	0.6844	0.5248
Uniform	2^{14}	**0.8842**	**0.7991**	0.6317	0.4709
Uniform	2^{15}	0.8820	0.7944	0.6997	0.5427
Border	2^{13}	0.8380	0.7367	0.7596	0.6168
Border	2^{14}	0.8353	0.7333	**0.7616**	**0.6201**
Border	2^{15}	0.8341	0.7312	0.7559	0.6105

The segmentation results achieved for various configurations of the proposed OSS-Net, the voxelised and slicing CNN baselines, as well as the function space baseline are summarised in Table 2. All proposed variants of the OSS-Net outperform the original O-Net in segmentation performance, on both the BraTS 2020 and the LiTS datasets. The OSS-Net variant A improves the IoU by 0.203 over the original O-Net on the BraTS 2020 dataset. Employing the proposed encoder with output skip-connections and the auxiliary loss (variant C & D) increases the Dice score further to 0.8842 and 0.8774, respectively. Using larger patches with a size of 14^3 (variants B and D) lead to no significant change in performance.

The best performing OSS-Net variant C performs on par to the voxelised baseline on the BraTS 2020 dataset, which was better on the smaller ($\sim$ 100 training samples) LiTS dataset. In comparison to the slicing baseline, OSS-Net variant C achieves better segmentation performance on both datasets. The performance gap, between the explicit 3D methods (3D residual U-Net, OSS-Net variant C) and the slicing-baseline may be caused by the inherently limited spatial context of the 2D residual U-Net [12, 34]. The slicing baseline is sensitive to slice orientation with significantly different performance for the different slice orientations (Tab. 1 footnote).

The memory consumption is presented in Table 3. OSS-Net C is 5 times more memory efficient than the voxelised baseline during training on the BraTS 2020 dataset, and 10 times more efficient for inference with 2^{14} locations. On the LiTS dataset, OSS-Net, operating at the native resolution (512^3), remains significantly more memory efficient than the voxelised baseline on a reduced resolution (224^3). In comparison to the function space baseline, O-Net, the proposed OSS-Net variant C requires a maximum of 10% more memory at training.

The slicing baseline requires less memory during training than the OSS-Nets. On the BraTS 2020 dataset, the 2D residual U-Net consumes half as much memory as the OSS-Net variant C at training. On the LiTS dataset with a resolution of 512^3, the difference in training

Table 2: Semantic segmentation results of our approaches and baselines on validation data.

	BraTS 2020		LiTS	
Model/OSS-Net Configuration	Dice ↑	IoU ↑	Dice ↑	IoU ↑
3D residual U-Net (voxelised baseline)	*0.8827*	**0.7995**	**0.7888**	**0.6558**
2D residual U-Net (slicing baseline)	0.8589*	0.7658*	0.6674†	0.5233†
O-Net (function space baseline) [22]	0.7016	0.5615	0.6506	0.4842
+ patch encoder w/ small patches 7^3 (OSS-Net A)	0.8592	0.7644	0.7127	0.5579
+ avg. pooled intermediate patches 14^3 (OSS-Net B)	0.8541	0.7572	0.7585	0.6154
A + encoder skip-con. & aux. loss (OSS-Net C)	**0.8842**	*0.7991*	*0.7616*	*0.6201*
B + encoder skip-con. & aux. loss (OSS-Net D)	0.8774	0.7876	0.7566	0.6150

* Best slice depth orientated (height slice IoU 0.7269, Dice 0.8360; width slice IoU 0.7300, Dice 0.8369).
† Best slice width orientated (height slice IoU 0.2732, Dice 0.1673; depth slice did not converge).

memory usage between the slicing baseline and OSS-Net variant C reduces to 0.9GB. At inference, the memory for the OSS-Net variant C and the slicing baseline is roughly similar.

Processing a single slice per forward pass of the 2D residual U-Net results in a runtime of 5s for a full prediction on the BraTS 2020 dataset. When Utilizing multiple slices per forward pass this is reduced to 2.5s, similar to the inference runtime of the OSS-Net (characteristically 2.6s, see Fig. 5). Processing multiple slices, per forward pass of the 2D U-Net, yields a linear increase in memory usage relative to the number of processed slices, resulting in a trade-off between inference speed and memory consumption, which OSS-Net avoids.

We analysed the effect of the number of locations on the inference runtime (Fig. 5), as

Table 3: GPU memory consumption of our networks and baselines. Inference GPU memory usage of the network evaluation step (Fig. 4) for different number of sampled locations.

Model/	BraTS 2020			LiTS		
OSS-Net	Training	Inference	Inference	Training	Inference	Inference
Configuration	2^{14} loc.	2^{14} locations	2^{17} locations	2^{14} loc.	2^{14} locations	2^{17} locations
3D res. U-Net	14.41GB	3.57GB (dense pred.)		14.41GB*	3.57GB (dense pred.)†	
2d res. U-Net	1.16GB‡	0.46GB (slice pred.)‡		4.29GB‡	1.20GB (slice pred.)‡	
O-Net [22]	2.35GB	0.29GB	1.93GB	5.07GB	1.47GB	1.99GB
OSS-Net A	2.58GB	0.39GB	2.73GB	5.18GB	1.50GB	2.35GB
OSS-Net B	2.76GB	0.48GB	3.45GB	5.21GB	1.53GB	2.53GB
OSS-Net C	2.59GB	0.39GB	2.73GB	5.19GB	1.51GB	2.35GB
OSS-Net D	2.76GB	0.48GB	3.46GB	5.22GB	1.53GB	2.53GB

* Memory usage when trained on reduced resolution (224^3). Memory usage at native resolution $>$ 110GB.
† Memory usage on downsampled resolution (224^3) with prediction upsampled to native res. (512^3).
‡ Memory usage for processing a single slice (native res.), memory increases linearly for multiple slices.

the maximum number of locations evaluated per forward pass affects the memory consumption significantly (Tab. 3 & Fig. 6). We also measured the runtime of the original O-Net inference approach in conjunction with OSS-Net variant C and compared this to the modified inference approach. Runtimes were averaged over the whole BraTS 2020 validation dataset, with an initial octree resolution of one-quarter of the BraTS 2020 resolution.

The improved inference approach achieves a two times faster inference than the original approach, regardless of the maximum number of locations processed. The runtime for both inference approaches remained constant for 2^{12} and more maximum locations processed in a single forward pass (Fig. 5). Taking both inference speed and memory consumption (Fig. 6) into account, an optimal maximum number of locations is in the range of 2^{12} to 2^{14}. We evaluated the disagreement between voxel predictions for both inference approaches to be merely $5.9658 \cdot 10^{-4}\%$ on the BraTS 2020 validation set. This demonstrates that the proposed inference approach retains segmentation performance, while begin two times faster.

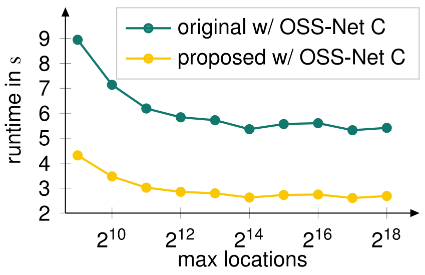

Figure 5: Inference runtimes for different maximum locations per encoder forward pass on the BraTS 2020 validation set. Standard O-Net inference approach in green 🟩, our improved inference approach in yellow 🟨. OSS-Net variant C employed.

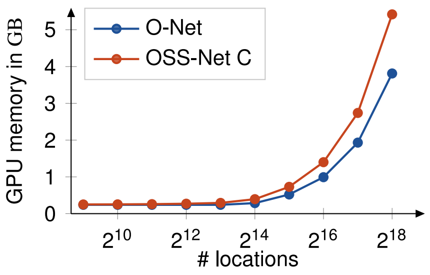

Figure 6: GPU memory consumption of a full inference forward pass for different number of sampled locations and patches (only OSS-Net C). O-Net baseline in blue 🟦, our OSS-Net C in orange 🟧. BraTS 2020 resolution utilised.

5 Conclusion

We propose OSS-Net for memory-efficient and high-resolution semantic segmentation of 3D medical data in function space. OSS-Net employs local observations to overcome the limited expressiveness of the original occupancy networks, achieving segmentation performance comparable to state-of-the-art 3D CNN approaches. We experimentally demonstrated the subsequent significantly increased segmentation accuracy in comparison to the original O-Net, as well as in comparison to an efficiency baseline (slicing based approach, 2D residual U-Net). Compared to a voxelised CNN performance baseline (3D residual U-Net), OSS-Net performs on par in segmentation accuracy in the brain tumour segmentation on the BraTS 2020 dataset, while falling slightly short on a dataset with limited training data ($\sim$ 100 liver segmentation samples, LiTS dataset). OSS-Net vastly outperforms the function space and efficient baselines on the latter more challenging dataset.

In terms of training memory consumption (for a given resolution), OSS-Net is similarly efficient to the O-Net and in excess of 5 times more memory efficient than the voxelised CNN baseline (and 10 times more efficient at inference). In comparison to the efficient baseline, OSS-Net requires more memory at training (2.6GB in comparison to 1.2GB) and a similar amount during inference. Both memory requirements are feasible with mid-range consumer GPUs, and OSS-Net has the additional advantage of being able to capture 3D relations. The proposed OSS-Net inference approach, that leverages prior encoder predictions, improves the inference speed by a factor of two in comparison to the original O-Net. In summary, OSS-Net combines the strong segmentation performance of the voxelised CNN performance baseline with the memory efficiency of the original O-Net, enabling accurate, fast and memory efficient 3D semantic segmentation that can scale to high resolutions.

Future work may consider a more principled approach to choosing the best sampling strategies, beyond the ablation study presented here. For example, using the network's confidence of past predictions could be employed to achieve an adaptive sampling strategy.

Acknowledgements We thank Marius Memmel and Nicolas Wagner for the insightful discussions, Alexander Christ and Tim Kircher for giving feedback on the first draft, and Markus Baier as well as Bastian Alt for aid with the computational setup.
This work was supported by the Landesoffensive für wissenschaftliche Exzellenz as part of the LOEWE Schwerpunkt CompuGene. H.K. acknowledges support from the European Research Council (ERC) with the consolidator grant CONSYN (nr. 773196). O.C. is supported by the Alexander von Humboldt Foundation Philipp Schwartz Initiative.

References

[1] Spyridon Bakas, Hamed Akbari, Aristeidis Sotiras, Michel Bilello, Martin Rozycki, Justin S Kirby, John B Freymann, Keyvan Farahani, and Christos Davatzikos. Data Descriptor: Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. *Scientific Data*, 4:170117, 2017.

[2] Spyridon Bakas, Mauricio Reyes, Andras Jakab, Stefan Bauer, Markus Rempfler, Alessandro Crimi, Russell Takeshi Shinohara, Christoph Berger, Sung Min Ha, Martin Rozycki, et al. Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge. *arXiv:1811.02629*, 2018.

[3] Patrick Bilic, Patrick Ferdinand Christ, Eugene Vorontsov, Grzegorz Chlebus, Hao Chen, Qi Dou, Chi-Wing Fu, Xiao Han, Pheng-Ann Heng, Jürgen Hesser, et al. The Liver Tumor Segmentation Benchmark (LiTS). *arXiv:1901.04056*, 2019.

[4] Robin Brügger, Christian F Baumgartner, and Ender Konukoglu. A Partially Reversible U-Net for Memory-Efficient Volumetric Image Segmentation. In *MICCAI*, pages 429–437, 2019.

[5] Cheng Chen, Kangneng Zhou, Muxi Zha, Xiangyan Qu, Xiaoyu Guo, Hongyu Chen, Zhiliang Wang, and Ruoxiu Xiao. An Effective Deep Neural Network for Lung Lesions Segmentation From COVID-19 CT Images. *IEEE Transactions on Industrial Informatics*, 17(9):6528–6538, 2021.

[6] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. In *CVPR*, pages 5939–5948, 2019.

[7] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion. In *CVPR*, pages 6970–6981, 2020.

[8] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*, pages 3075–3084, 2019.

[9] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *MICCAI*, pages 424–432. Springer, 2016.

[10] Harm de Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *NeurIPS*, volume 30, pages 6594–6604, 2017.

[11] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local Deep Implicit Functions for 3D Shape. In *CVPR*, pages 4857–4866, 2020.

[12] Eli Gibson, Francesco Giganti, Yipeng Hu, Ester Bonmati, Steve Bandula, Kurinchi Gurusamy, Brian Davidson, Stephen P Pereira, Matthew J Clarkson, and Dean C Barratt. Automatic Multi-Organ Segmentation on Abdominal CT With Dense V-Networks. *IEEE Transactions on Medical Imaging*, 37(8):1822–1834, 2018.

[13] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *CVPR*, pages 9785–9795, 2019.

[14] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The Reversible Residual Network: Backpropagation Without Storing Activations. In *NeurIPS*, pages 2214–2224, 2017.

[15] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *CVPR*, pages 9224–9232, 2018.

[16] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016.

[18] Yuanfeng Ji, Ruimao Zhang, Zhen Li, Jiamin Ren, Shaoting Zhang, and Ping Luo. UXNet: Searching Multi-level Feature Aggregation for 3D Medical Image Segmentation. In *MICCAI*, pages 346–356, 2020.

[19] Konstantinos Kamnitsas, Enzo Ferrante, Sarah Parisot, Christian Ledig, Aditya V Nori, Antonio Criminisi, Daniel Rueckert, and Ben Glocker. DeepMedic for Brain Tumor Segmentation. In *MICCAI Brainlesion Workshop*, pages 138–149. Springer, 2016.

[20] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the Variance of the Adaptive Learning Rate and Beyond. In *ICLR*, 2020.

[21] Bjoern H Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE transactions on medical imaging*, 34(10):1993–2024, 2014.

[22] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *CVPR*, pages 4460–4470, 2019.

[23] Mateusz Michalkiewicz, Jhony K. Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit Surface Representations As Layers in Neural Networks. In *ICCV*, pages 4743–4752, 2019.

[24] Marko Mihajlovic, Yan Zhang, Michael J. Black, and Siyu Tang. LEAP: Learning Articulated Occupancy of People. In *CVPR*, pages 10461–10471, 2021.

[25] 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.

[26] Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé Activation Units: End-to-end Learning of Flexible Activation Functions in Deep Networks. In *ICLR*, 2020.

[27] Andriy Myronenko. 3D MRI brain tumor segmentation using autoencoder regularization. In *MICCAI Brainlesion Workshop*, pages 311–320. Springer, 2018.

[28] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics. In *ICCV*, pages 5379–5389, 2019.

[29] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *CVPR*, pages 165–174, 2019.

[30] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020.

[31] Mathias Perslev, Erik Bjørnager Dam, Akshay Pai, and Christian Igel. One Network To Segment Them All: A General, Lightweight System for Accurate 3D Medical Image Segmentation. In *MICCAI*, pages 30–38. Springer, 2019.

[32] Tim Prangemeier, Christian Wildner, André O Françani, Christoph Reich, and Heinz Koeppl. Yeast cell segmentation in microstructured environments with deep learning. *Biosystems*, page 104557, 2021.

[33] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *NeurIPS*, pages 5099–5108, 2017.

[34] Holger R. Roth, Hirohisa Oda, Xiangrong Zhou, Natsuki Shimizu, Ying Yang, Yuichiro Hayashi, Masahiro Oda, Michitaka Fujiwara, Kazunari Misawa, and Kensaku Mori. An application of cascaded 3d fully convolutional networks for medical image segmentation. *Computerized Medical Imaging and Graphics*, 66:90–99, 2018.

[35] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization. In *CVPR*, pages 84–93, 2020.

[36] Pengyu Wang, Yuan Gan, Panpan Shui, Fenggen Yu, Yan Zhang, Songle Chen, and Zhengxing Sun. 3D Shape Segmentation via Shape Fully Convolutional Networks. *Computers & Graphics*, 70:128–139, 2018.

[37] Yuxin Wu and Kaiming He. Group Normalization. In *ECCV*, pages 3–19, 2018.

[38] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction. In *NeurIPS*, pages 492–502, 2019.

[39] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead Optimizer: *k* steps forward, 1 step back. In *NeurIPS*, pages 9597–9608, 2019.

[40] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

[41] Tongxue Zhou, Su Ruan, and Stéphane Canu. A review: Deep learning for medical image segmentation using multi-modality fusion. *Array*, 3:100004, 2019.

[42] Xiangrong Zhou, Takaaki Ito, Ryosuke Takayama, Song Wang, Takeshi Hara, and Hiroshi Fujita. Three-Dimensional CT Image Segmentation by Combining 2D Fully Convolutional Network with 3D Majority Voting. In *Deep Learning and Data Labeling for Medical Applications*, pages 111–120. Springer, 2016.

[43] Yuyin Zhou, Lingxi Xie, Wei Shen, Yan Wang, Elliot K Fishman, and Alan L Yuille. A Fixed-Point Model for Pancreas Segmentation in Abdominal CT Scans. In *MICCAI*, pages 693–701. Springer, 2017.