Capsule Nets for Complex Medical Image Segmentation Tasks

Shanmugapriya Survarachakan[1], Jenny Stange Johansen[1], Mathias Aarseth Pedersen[1], Mahdi Amani[1,2], and Frank Lindseth[1]

[1] Norwegian University of Science and Technology, Department of Computer Science, Trondheim, Norway
{shanmugapriya.survarachakan, mahdi.amani, frankl}@ntnu.no
{jennysj, mathiaap}@stud.ntnu.no
[2] SimulaMET, Simula Research Laboratory, Oslo, Norway

Abstract. In recent years, automatic image segmentation using deep learning has shown great potential. U-Net is commonly applied to the medical image segmentation task. Further, a new architecture called SegCaps, based on capsules networks has been introduced. Even though medical image segmentation has made great strides, it is still a complex task and continued research in this area is important. In this paper, the method from the Segcaps paper was implemented and trained on the same dataset and achieved a similar result. In addition, a Multi-SegCaps model, an EM-routing SegCaps model and a U-Net model able to segment an arbitrary number of classes were developed. The models were implemented for the segmentation of single 2D slices using two of its neighboring slices on each side (five in total). These models were trained to perform left atrium and hippocampus (anterior & posterior) segmentation on the MSD datasets. The performance of the Multi-SegCaps model and the EM-routing SegCaps model were compared to the U-Net model. The 2.5D U-Net model had an overall better performance on all the datasets achieving a Dice score of 91.39% for the left atrium segmentation, and the Dice scores of 85.18% and 83.90% for the anterior and posterior hippocampus segmentation. The multi-class SegCaps model was also applied to two different datasets, showing the ability to segment several classes reasonably well. A Dice score of 68.2% was achieved for the left atrium and the Dice scores of 72.42% and 70.49% were achieved for the anterior and posterior hippocampus, respectively. The EM-SegCaps model was applied to the hippocampus dataset and it achieved a Dice score of 54.50% for the whole hippocampus and the Dice scores of 18.67% and 24.52% for the anterior and posterior hippocampus, respectively.

Keywords: Medical Image Segmentation · U-Net · SegCaps · Capsule Network.

1 Introduction

Medical image segmentation methods attempt to extract and locate the precise location of organs, tumors and other structures of interest, with the intention of aiding health professionals in making accurate diagnoses in a shorter amount of time. In recent years, convolutional neural networks (CNNs) have been commonly used for image analysis tasks. A successful application of CNNs to the task of image segmentation was shown by [9] when they introduced fully convolutional networks where the skip connections were introduced to directly connect the opposing convolutional layers in the contraction and expansion paths. The network can work regardless of the original image size, without requiring any fixed number of units. The performance was improved by the elegant fully convolutional U-Net architecture, published by [10]. The architecture is symmetric and the skip connections between the downsampling path and the upsampling path apply a concatenation operator instead of a sum. The network combines the location information from the downsampling path with the contextual information in the upsampling path and preserves the spatial resolution of the output. SegNet, an encoder-decoder network similar to FCN was introduced in [1]. It uses max unpooling in the upsampling path which eliminates the need for the network to learn the upsampling and provides a more efficient way to achieve segmentations than FCN. [6] created a new U-Net type architecture, showing state of the art performance on the task of multimodal brain tumor segmentation using the BraTS dataset. The architecture was residual and worked directly with 3D patches of medical images. Though CNNs are popularly used they have some limitations due to its properties of rotational invariance and lack of instantiation parameters. So, many images with different views are needed to train the network. To overcome this issue, [11] presented a brand new capsule neural network architecture. Unlike CNNs, capsule networks are equivariant which can detect the objects if they are rotated and how many degrees it is rotated. This reduces the number of images required to train the network. [11] showed that capsule networks gave state of the art performance on the task of classifying highly overlapping digits in the MNIST dataset, proving CapsNet to be an efficient architecture for learning this type of problem. In [3], the capsule network consisting of two convolutional layers, two capsule layers, one fully connected layer, and two de-convolutional layers employing dynamic routing mechanism was proposed to segment Left ventricle (LV). Building on the work of capsule networks, [8] presented a method for performing binary image segmentation. The number of parameters needed by the architecture was reduced substantially by constraining the dynamic routing to capsules only in a defined kernel. The network showed good results for lung segmentation using the LUNA16 dataset. [5] claimed that the EM-routing algorithm solves most of the issues with the routing algorithm from [11]. It reduces the number of parameters needed, as well as using a metric of agreement that does not saturate with highly confident predictions. It gave state of the art performance at detecting objects at novel viewpoints on the smallNORB dataset. In [2], a novel capsule is introduced which combines pose and appearance information encoded as capsules,

named Matwo-Caps. Additionally, a new routing mechanism, i.e. dual routing, combining this two information was proposed. The method was evaluated for the semantic segmentation of the JSRT chest X-ray dataset. In [4], Fourier analysis and the circular Hough transform methods were applied to indicate the approximate location of the LV and the capsule network with dynamic routing was used to precisely segment the LV. Thresholding and morphological processing were used as postprocessing methods to increase the accuracy of LV segmentation.

In our work, the method and the results from [8] were reproduced and trained to adapt to other medical datasets and different modalities. The architecture was, as reported in the paper, capable of segmenting images of lungs. However, the experiment also revealed that the architecture struggles to adapt to another medical dataset. A SegCaps architecture is further developed and evaluated for multi-label image segmentation. It was experimentally shown to be able to perform segmentation of datasets with multiple label classes but struggles to segment structures in imbalanced datasets. Additionally, a framework for image segmentation using capsule networks and the EM-routing algorithm was developed and evaluated. As far as the authors know, an attempt of this has never been reported in the literature so far. Although the model did not give very good results, it showed that it was capable of segmenting some simple structures from the hippocampus dataset. A 2.5D architecture based on a U-Net variant was implemented and compared with the performance of the SegCaps, the Multi-SegCaps and the EM-SegCaps models on the cardiac and the hippocampus datasets from the Medical Segmentation Decathlon (MSD) datasets.

This paper consists of five sections. The first section gives an introduction to the paper and the motivation behind it. The various methods and the datasets used in the paper are presented in the second section. In the third section, the experiments comparing different methods and their corresponding results are presented. The fourth section summarizes the discussion of the results and the fifth section presents the conclusion and suggests future work.

2 Methodology

This section presents the different models developed and the datasets used in the experiments.

2.1 SegCaps

The SegCaps architecture presented in [8] was used to perform binary segmentation of medical images (see Fig. 1). The architecture used is called SegCaps R3, referring to the use of three iterations of the dynamic routing algorithm in each capsule.

A regular 2D convolution is performed on an input image to produce 16 feature maps. The tensor consisting of 16 feature maps is reshaped in order to give it a new dimension of length one, such that the reshaped tensor now represents a single 16D capsule. The 16D capsule is forwarded to the primary capsule layer,

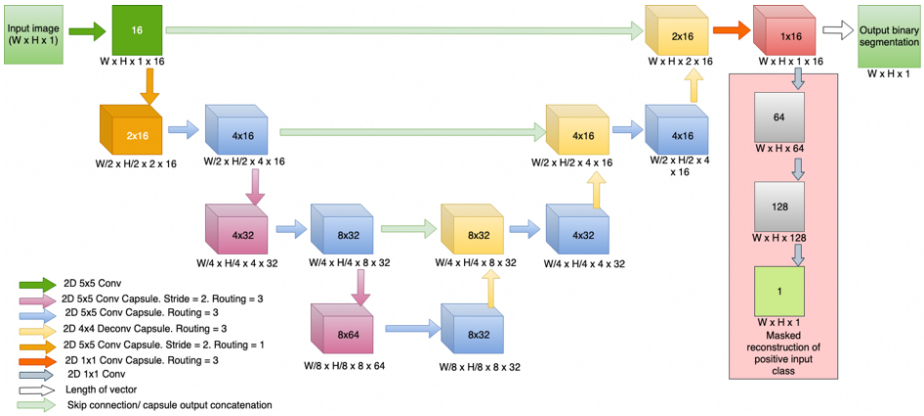

Fig. 1. SegCaps architecture

which is a regular convolutional capsule layer with one routing iteration, which returns predictions of two 16D capsules. Following the primary capsule layer, there are sets of two convolutional capsule layers at every layer for the remainder of the contracting path. The first of the two operations is a 5x5 convolutional capsule layer with stride two, which also doubles the number of feature maps that are output. The second layer consists of 5x5 convolutional capsules without a stride. Output features from the latter operation being forwarded to the next level of the contracting path, as well as stored for concatenation with capsules in the expanding path. After predictions are downsampled in the contracting path, they are then upsampled in three levels of the expanding path. Every level of the expanding path contains two capsule layers. The first layer is a 4x4 deconvolutional capsule, which upsamples the image by a factor of two using transposed convolution. The output vectors from deconvolutional capsules are then concatenated with capsule output from the corresponding level in the contracting path before they are used to predict the second layer of capsules.The second capsule layer uses the concatenated predictions from both the contracting and expanding path, to predict a layer of 5x5 convolutional capsules. Instead of using 5x5 convolutional capsules in the last layer, they are replaced with 1x1 convolutional capsules, which produces the final segmentation vectors. The segmentation vectors are converted into actual predictions by calculating the Euclidean length of them. For each pixel in the image, it is classified as the target class if its length is longer than a given threshold. To perform reconstruction of the pixels belonging to the target class,the output from the last layer is reshaped from capsule form into a convolutional form. This involves flattening the 16D capsule into 16 feature maps. Using ground truth labels, the pixels that do not belong to the target class are masked out. This prevents the reconstruction head from considering irrelevant background pixels. Three layers of 2D convolutions with a 1x1 kernel is applied to the masked feature maps. The last of the three convolutions output a single filter giving a reconstruction of the target class of the

input image. A mean square error loss between pixels from the input image and the reconstructed image is used in combination with the segmentation loss, to train the network. To train the segmentation part of the network, a weighted cross-entropy loss is calculated using the Euclidean length of the output capsule as prediction and a ground truth label that is 0 for background and 1 for the target class.

2.2 Multi-SegCaps

The architecture proposed in [8] was extended to support end-to-end segmentation of different datasets containing an arbitrary number of output classes from different input modalities. Due to the support for multiple target classes, the architecture was called Multi-SegCaps. The output capsule layer was modified to output N 16D output capsules, where N is the number of classes in the dataset, including background, and the predicted class is the one represented by the capsule with the longest euclidean length.The network is trained using the predicted capsule lengths and one-hot encoded ground truth labels. Instead of only trying to reconstruct a single target class, Multi-SegCaps attempts to reconstruct the pixels belonging to all classes, except for the background class. The segmentation capsule's output is masked with a ground-truth mask for all these classes. The N 16D masked capsules are then flattened into N x 16 convolutional feature maps that are forwarded to the reconstruction head. The reconstruction head consists of three convolutional layers, each with a kernel size of 1. The last of the three convolutional layers will output as many filters as there are input modalities. This incentivizes the network to store properties about more than one input channel, if available. The total reconstruction loss is the mean-squared error of pairwise output reconstruction and positive masked input modalities.

2.3 EM-SegCaps

The classification method from [5] was extended to perform semantic image segmentation. The network has a U-Net-style encoder-decoder topology, similar to the original SegCaps architecture. The architecture uses matrix capsules with EM-routing and is shown in Fig. 2. A regular 2D convolution with kernel size 3x3 is applied to an input image, producing 16 feature maps that are forwarded to the primary capsule layer. The primary capsule layer is responsible for transforming the output from the first convolution into an estimated pose and activation of a single capsule. The primary capsule layer performs two sets of 2D convolutions with a 1x1 kernel. The first convolution transforms the 16 input feature maps into 16 predictions for each receptive field. The 16 prediction values represent the current pose of the entity that is detected and can later be reshaped into a 4x4 matrix. The second convolution in the primary capsule layer is also given the same 16 feature maps as the previous convolution. The difference between them is that the second convolution only calculates a single capsule activation for each receptive field. In other words, for every pose matrix that is calculated, a single scalar is also detected. The scalar is treated as the activation of the

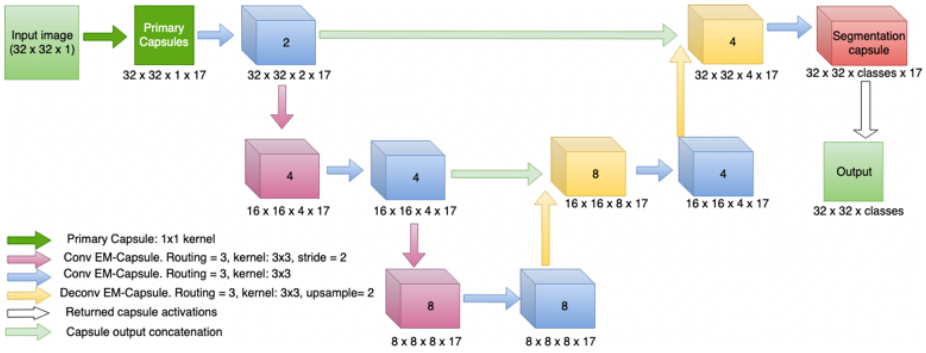

Fig. 2. EM-SegCaps architecture

capsule and works similarly to how convolutional kernels calculate activations in a convolutional neural network. The difference between the activation value of a capsule in a capsule network and a neuron in convolutional networks, is that capsule networks do not use the activation directly when calculating the output. It is rather used as a coefficient that influences the amount of information that is forwarded from a capsule to different capsules in the next layer. A sigmoid non-linearity is applied to the activation values in order to scale them between zero and one. The primary capsule layer concatenates the pose values and activation into 17D vectors and returns it as its output. Apart from the first primary capsule layer, the capsule network architecture also contains convolutional capsule layers, strided convolutional capsule layers, and deconvolutional capsule layers. Convolutional capsule layers accept the poses and activations from capsules in the previous layer and output new poses and activations for the capsules in the next. This is accomplished using the Expectation- maximization routing algorithm (EM-routing). Before performing EM-routing, all child capsules cast an initial vote of the output for every capsule in the next layer, using its own pose matrices. Casting this vote involves multiplying the pose matrix by a trained transformation matrix going into the parent capsule, which is shared by all child capsules. This effectively translates into performing a modified version of 2D convolution. Instead of performing pointwise multiplication between a convolution kernel and the input image, each element in the kernel is a transformation matrix that is multiplied by the pose matrix located at a specific location in the receptive field. After the predictions are calculated, they are forwarded to the EM-routing algorithm, along with the activations from the previous layer. The EM-routing algorithm is run for three iterations before it returns the final pose and activations for all capsules in the current layer. Strided convolutional capsules are almost identical to regular convolutional capsules. The only difference is that a stride of 2 is used during matrix convolution to reduce the feature maps in half along the height and width axes. Deconvolutional capsules are also similar to regular capsules, as they perform regular capsule convolution after upscaling by a factor of two, using nearest-neighbor interpolation. They are technically

not deconvolutional capsules as they don't perform transposed convolution, due to the technical challenges of implementing transposed matrix convolutions.

2.4 U-Net

The U-Net model was based on the architecture by [6], which is a 3D model. In our implementation, the 3D operations were replaced with 2D operations. To get the volumetric information, we used five neighboring slices as input(the slice that is supposed to be segmented in addition to two slices on each side). The replaced operations were convolutions, spatial dropout and nearest neighbor upsampling. Both the Multi-SegCaps model and the EM-SegCaps model were trained and their performances were compared with the U-Net model.

2.5 Dataset

The LUNA16 dataset was used in experiment 1 and the images were normalized similar to how [8] did in their work with SegCaps. In experiment 2 and 3, the cardiac dataset and hippocampus dataset from the MSD [13] were used. Normalization and augmentation were applied to the images before training. The datasets were normalized using their minimum and maximum values. While normalization was only done once, augmentation was applied repeatedly, with different parameters for every training iteration. Augmentations techniques like rotation(max degrees=45), flipping(axis=0,1), shifting(max horizontal=0.2, max vertical=0.2), shearing(amount=16), zooming(max zoom range=0.75, 0.75), elastic deformations(alpha=1000, sigma=80, alpha affine=50) and salt and pepper noise(salt=0.2, amount=0.04) were applied to the training data randomly. The goal of the cardiac dataset is to segment the left atrium from MRI images. It contains 30 3D volumes of which includes 20 training and 10 testing volumes. The hippocampus dataset has two classes, anterior and posterior hippocampus. The hippocampus dataset has 394 3D volumes which include 263 training volumes and 131 testing volumes.

2.6 Training

All experiments were performed using four splits. Each split was made up of 75% for training and 25% for validation and testing. The slices containing atleast one pixel from the positive class were added to the batch while slices containing only background class were discarded to reduce the effect of class imbalance.

The training process of SegCaps was similar to the one used by [8]. It lasted up to 200 epochs, but was terminated when the validation Dice score had not improved for 25 epochs. One epoch is 10000 training steps with a batch size of 1. Weighted cross-entropy loss and MSE reconstruction loss were minimized with equal weight. The Adam optimizer was used for stochastic gradient descent with an initial learning rate of 0.0001 and a Beta 1 of 0.99. The learning rate was decayed by a factor of 0.05 if the validation Dice score did not improve the last 5 epochs.

For Multi-SegCaps, the training was performed using a batch size of 1, due to memory limitations. One epoch had 1000 steps, and the model early stopped if it did not improve for 100 epochs. These parameters were selected to match the ones used for 2.5D U-Net. After early stopping, the model with the best validation score was stored, and used for testing. For every validation, 500 steps were used. The initial learning rate was set to 0.0005 and was reduced to 70% every time the model did not improve over 5 epochs. The Adam Beta 1 parameter was set to 0.99, due to the low batch size. Augmentation and reconstruction weighting was used for regularization with a reconstruction weight of 0.01. Five consecutive slices were used as input for making a prediction for the middle slice.

For EM-SegCaps, the architecture requires that all the images have the same dimensions during training, so every slice in all the hippocampus images and masks were cropped into 32x32 pixels. During inference, the hippocampus images were zero-padded to fit entire image slices into a batch, but the padded voxels were removed before calculating performance metrics.

2.7 Environment

The experiments were carried out using two different clusters (cluster1/cluster2) and a local computer. SegCaps models were trained on cluster1 using P100 GPUs, while some of the 2.5D U-Net models were run on V100 GPUs. Both GPUs had 16 GB of video memory. To be able to train more models in parallel, some of the 2.5D U-Net models were trained on cluster2. It has V100 GPUs with 32GB of video memory. EM-SegCaps models were trained on a local computer having a single GTX1080 GPU with 11GB of memory. This was done because of an incompatibility between the way EM-SegCaps performs multiplications of large batches of matrices and the version of CUDA installed on the clusters.

3 Experiments and Results

This section presents the three experiments conducted and their corresponding results. In experiment 1, we applied the SegCaps model and reproduced the results using the LUNA 16 dataset. In experiment 2, the SegCaps model, the Multi-SegCaps model and the 2.5D U-Net model were trained on the cardiac dataset and the results were compared. In experiment 3, the Multi-SegCaps model, the EM-SegsCaps model, and 2.5D U-Net were trained on the hippocampus dataset and compared. In addition, to test the performance of EM-SegCaps on binary class segmentation, the two classes of the hippocampus were merged to a single class and the model's performance was explored as part of the experiment.

3.1 Reproduce the SegCaps results using the LUNA16 dataset

The goal was to reproduce the results presented in [8] where the proposed architecture was tested on the LUNA16 dataset [12]. In this experiment, the same

Table 1. Comparison of Dice score of our results and the results presented in [8] on LUNA16 dataset

	Split-0	Split-1	Split-2	Split-3	Average
Results presented in [8]	98.499	98.523	98.455	98.474	98.479
Our results	98.467	98.189	98.072	98.238	98.242

architecture and parameters were applied to test the validity of the results reported by [8]. Table. 1 shows both our scores and the scores reported by [8] for the LUNA16 dataset. In [8], which volumes the four splits were made of was not reported. So the scores for each split are not directly comparable. However, the average Dice score achieved in this experiment is comparable to the one reported in the literature.

3.2 Comparison of SegCaps, Multi-SegCaps and 2.5D U-Net on the cardiac dataset

The segmentation results and the Dice scores achieved on the cardiac dataset using different architectures are shown in Fig. 3 and Table. 2, respectively. In the first part of this experiment, the cardiac dataset was used to test if SegCaps would be able to generalize to new datasets without modifying the network topology. The cardiac dataset contains fewer images than LUNA16 making it

Table 2. Comparison of SegCaps, Multi-SegCaps and 2.5D U-Net on the cardiac dataset.

	Recall	Precision	Dice
U-Net	90.690	92.114	91.396
SegCaps	96.348	43.962	60.376
Multi-SegCaps	86.892	54.467	66.960

faster to train, but the cardiac dataset has more background voxels which result in a higher class imbalance, giving the network a different challenge.

The segmentation performance for the cardiac dataset is significantly lower than for the LUNA16 dataset. The Dice score is 30% lower for the cardiac dataset when compared to the one obtained by the U-Net model on the same datasets. The SegCaps model shows the ability to achieve high recall, but the precision is poor. This shows that the model captures the left atrium well, but wrongly segments some of the background as the target class. To test the feasibility of performing multi-label semantic segmentation, the SegCaps architecture was modified to handle multiple classes. The cardiac dataset showed the performance impact of using a multi-class architecture on a problem that could otherwise be solved using a plain binary segmentation architecture. This would address whether multi-class SegCaps would be able to perform segmentation with the same performance as SegCaps. As part of this experiment, we compared the

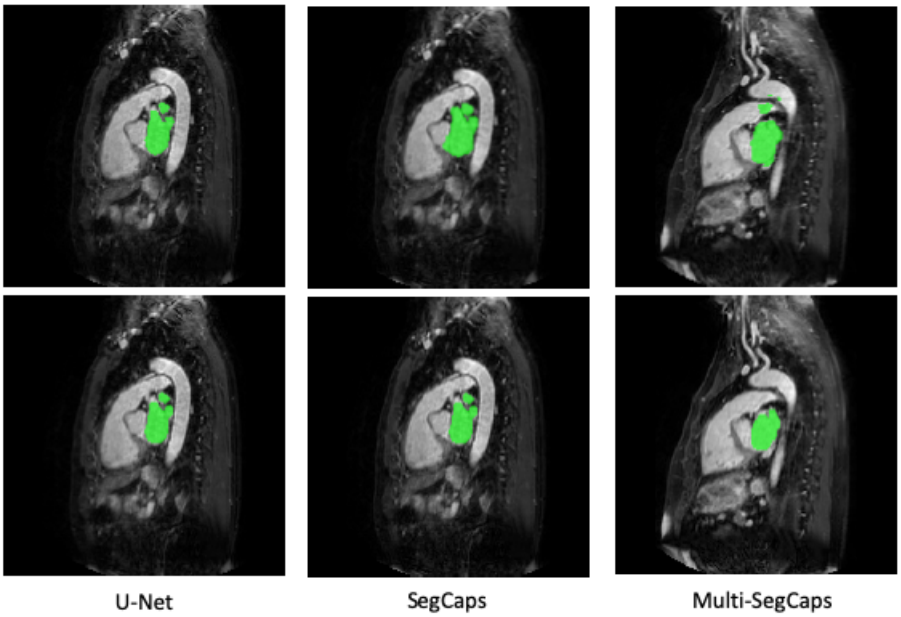

Fig. 3. Left Atrium Segmentation(upper row-prediction, lower row-groundtruth)

performance of Multi-SegCaps using dice loss and weighted cross-entropy loss functions. The motivation behind this comparison was that the Dice loss had shown good results on the 2.5D U-Net, while the original SegCaps used weighted binary cross-entropy. By applying Multi-SegCaps to datasets with multiple target classes, the ability of SegCaps adapting to multi-class segmentation was addressed further. By using a similar training process as for 2.5D U-Net, the results for Multi-SegCaps would be comparable to the U-Net based model. Training Multi-SegCaps on the cardiac dataset gave higher Dice scores than those achieved by the original SegCaps architecture (Table. 2).

3.3 Comparison of Multi-SegCaps, EM-SegCaps and 2.5D U-Net on the hippocampus dataset

The performance of Multi-SegCaps, EM-SegCaps and 2.5D U-Net on the hippocampus dataset are shown in Table. 3. When applying Multi-SegCaps the Dice scores for anterior and posterior hippocampus are fairly good, showing that the architecture was able to learn a multi-class problem. An example of a segmentation is shown in Fig. 4. The model is able to capture the main structures of the hippocampus, but struggles to determine the class membership of the pixels at the borders. It also wrongly predicts some of the background as the anterior hippocampus.

Concerning the possibilities and limitations of using matrix capsules with EM-routing for segmentation, a new type of capsule network architecture was

Table 3. Comparison of Multi-SegCaps, EM-SegCaps and 2.5D U-Net on hippocampus dataset.

	Recall	Precision	Dice
U-Net(Anterior hippocampus)	84.546	85.818	85.177
U-Net(Posterior hippocampus)	81.433	86.541	83.909
Multi-SegCaps(Anterior hippocampus)	80.764	65.645	72.424
Multi-SegCaps(Posterior hippocampus)	84.455	60.493	70.494
EM-SegCaps(Anterior hippocampus)	17.508	20.006	18.674
EM-SegCaps(Posterior hippocampus)	19.008	34.548	24.523

designed for this project. The encoder-decoder style EM-SegCaps network was compared with Multi-SegCaps on the segmentation of hippocampus images. The experiment would show if SegCaps with EM-routing would be able to learn the tasks using only 30,166 trainable parameters, compared to the 1,436,769 parameters used by Multi-SegCaps. The architecture was trained to perform binary and multi-class segmentation of the hippocampus. Fig. 4 shows that EM-SegCaps achieves mixed results when performing multi-class segmentation of volumes from the hippocampus dataset. The results of EM-SegCaps on the multi-class hippocampus dataset are significantly worse than the ones obtained using the multi-class SegCaps and the 2.5D U-Net model (Table. 3).

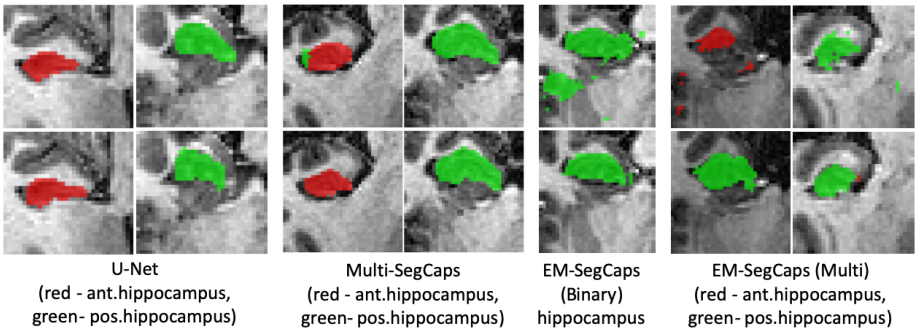

Fig. 4. Hippocampus Segmentation(upper row-prediction, lower row-groundtruth)

Table. 4 shows that EM-Segcaps achieves a Dice score of about 54.50% when performing binary segmentation (both the anterior and posterior hippocampus are merged as a single class) of volumes from the hippocampus dataset. The model correctly classifies most pixels belonging to the hippocampus class, but also incorrectly classifies some irrelevant structures as being the hippocampus. The reason why EM-SegCaps was only tested on hippocampus data, is because slices of volumes from that dataset are only around 32x32 pixels in size. The implementation of EM-SegCaps does not use any native Tensorflow operations for the EM-routing algorithm, nor the matrix convolution operation. Because

Table 4. Dice score of the whole hippocampus using EM-SegCaps.

	Recall	Precision	Dice
EM-SegCaps(Hippocampus)	71.264	44.122	54.500

these operations are not implemented and optimized in a low- level programming language, they have to be emulated in Python code using other Tensorflow operations. This made the calculations in this experiment extremely expensive both in terms of computation and memory.

4 Discussion

In the first experiment on reproducing the results from [8], the average Dice score achieved when the SegCaps model trained using the LUNA16 dataset was 98.242%, which is comparable to the average Dice score of 98.479% that was presented in the literature. The exact same SegCaps architecture was also trained on the cardiac dataset in the second experiment. The cardiac dataset got a score of 62.091% when averaging the median per-image Dice score of all four splits. The result is significantly lower than the results achieved by 2.5D U-Net. The result was surprising, as SegCaps had slightly outperformed a U-Net based model on the LUNA16 dataset when compared by [8]. Applying SegCaps to the cardiac datasets gave disappointing results. There could be several reasons why SegCaps performed much better on the LUNA16 dataset. Our initial hypothesis was that SegCaps could be extremely sensitive to hyperparameters, which possibly needed to be carefully tuned for the task at hand. However, we did experiment with some changes to learning rate, learning rate decay and reconstruction weighting but did not observe that these variations had a significant impact on the performance. This could indicate that the ability to learn was highly dependent on some random phenomena such as sampling order or weight initialization. The application of SegCaps to a new dataset discloses some challenges with the architecture. In the same experiment by applying Multi-SegCaps to the cardiac dataset it was possible to compare the architecture to the original SegCaps. The second part was to compare Multi-SegCaps to a U-Net based model. While still keeping most configurations as in the experiment with the original SegCaps, some parameters in this experiment were changed to match the ones used in the 2.5D U-Net for having comparable results. Dice loss is commonly used for training U-Net based architectures. Testing whether a Dice loss function could be used for training a SegCaps model was therefore of interest. A Multi-SegCaps architecture was developed and trained on the cardiac dataset using Dice loss. Surprisingly, it was incapable of learning how to segment anything of the target class, and all predictions were solely background; giving a Dice score of zero. The reason seems to be that the model starts to output very confident predictions for the background class and when this was combined with the non-linearity function of the output capsule, we observed that the error gradients were vanishing. As no results of interest were accomplished,

weighted cross-entropy was used for the Multi-SegCaps experiment. When using Multi-SegCaps with weighted cross-entropy, the cardiac dataset discloses a Dice score of 67.0%. The score was a bit higher than when using regular SegCaps. There are some factors that could have influenced the results. When using the multi-class version of the network, the final layer consists of more capsules. The added weights in the network could possibly add some assistance in separating the positive class from the background class. A more likely explanation for the increased performance is that the training routines are different. While the routine in the previous SegCaps experiments was identical to [8], the routine in this experiment was chosen to match the experiment with 2.5D U-Net, which favors shorter epochs and a smoother learning rate decay. Another factor that could have influenced the gap is the number of consecutive slices provided to the model during training. While SegCaps was only given one slice, Multi-SegCaps was given five slices. Using five slices gives more spatial information compared to a single slice. This could partly explain the higher performance achieved by the Multi-SegCaps model.

For addressing whether Multi-SegCaps would be able to segment several classes, it was also applied to dataset with multiple target classes. Multi-SegCaps was further tested on the hippocampus dataset in the third experiment. The Dice scores of about 72% and 70% were achieved for anterior and posterior hippocampus, respectively. The hippocampus images are relatively small, making the task fast to learn. Inspecting the hippocampus predictions revealed some confusion between the two classes, a similar issue as experienced with the 2.5D U-Net. A reason could be that classes lay next to each other and the boundaries are not clear. On the binary segmentation of the hippocampus using EM-SegCaps, the architecture was only able to segment the rough structure of the hippocampus. The network has only 30,166 trainable parameters compared to the 1,436,769 parameters used by Multi-SegCaps, which might be too few to learn segmentation of complex structures. Although the low number of parameters may be partly to blame for the low performance, another reason could be that the capsule network itself is not able to accurately reason about the precise location of detected entities at different capsule layers. Due to resource demands, EM-SegCaps has relatively few layers in the network and small receptive fields of 3x3 in its convolutional capsule kernels. Hence, the model might be too simple to reason about patterns observed at one location in the image using information from other regions. The predictions contain multiple separate structures, as well as individual pixels, labelled as the hippocampus. The explanation could be that the network is either not able to take global feature information into account fully, or that the EM-routing algorithm fails at routing information in the way we would expect, when used for segmentation. The modified Jaccard Dice function was used in this experiment. The original Jaccard Dice loss function [7] calculates a total Dice score for all classes. The modified version calculates separate Dice loss scores for all classes, which are then averaged into a single total Dice score. The loss function appears to be even less sensitive to class imbalance than

Jaccard Dice. Thus, the choice of loss function was more or less arbitrary and is subject to improvements in the future.

On multi-class EM-SegCaps, the model got a Dice score of 18.674% for anterior and 24.523% for the posterior hippocampus classes. The model did not seem to be capable of separating the two hippocampus classes from each other. For some images, we observed that the model had correct predictions for only the posterior hippocampus class and in some images, predicted only the anterior class. A reason might be that there are too few capsules in the later layers, which are used for the final segmentation. This could cause an information bottleneck that prevents useful information from being used in the final evaluation.

Since the models were trained on machines with different configurations, the time taken to train each model are not directly comparable.

5 Conclusion

Medical imaging is a common way of detecting diseases by examining the structures inside the body. Developing an automatic method for analyzing arbitrary medical images are therefore of great value to the medical community. In this project, we tested and further developed segmentation architectures based on capsule networks. Capsule networks show potential, despite requiring optimizations and continued research before being able to challenge U-Net.

In the future, researchers should look into decreasing the memory consumption of SegCaps. We do not believe that the architecture will be able to outperform U-Net as it is, keeping in mind that the training process is a lot slower and requires way more memory. Furthermore, loss functions should be researched to speed up the learning process and increase the stability of learning. Other network topologies should be explored further, which are more carefully tuned to work with capsules. Additionally, it could be of interest to investigate 3D capsule networks.

To improve EM-routing, we suggest looking at decreasing memory usage. The resource requirements did not allow us to use larger receptive fields than 3x3 in the convolutional capsule kernels. With lower memory requirements, the complexity of the network could have been increased and the effect on segmentation accuracy could have been studied.

Acknowledgement

This work is supported by H2020-MSCA-ITN Marie Sklodowska-Curie Actions, Innovative Training Networks (ITN)-H2020 MSCA ITN 2016 GA EU project number 722068 High Performance Soft Tissue Navigation (HiPerNav).

References

1. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(12), 2481–2495 (Dec 2017)

2. Bonheur, S., Stern, D., Payer, C., Pienn, M., Olschewski, H., Urschler, M.: Matwo-capsnet: A multi-label semantic segmentation capsules network. In: Medical Image Computing and Computer Assisted Intervention – MICCAI 2019. pp. 664–672. Lecture Notes in Computer Science, Springer (2019)
3. Dong, J., Liu, C., Yang, C., Lin, N., Cao, Y.: Robust segmentation of the left ventricle from cardiac mri via capsule neural network. In: Proceedings of the 2nd International Symposium on Image Computing and Digital Medicine. p. 88–91. ISICDM 2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3285996.3286016
4. He, Y., Qin, W., Wu, Y., Zhang, M., Yang, Y., Liu, X., Zheng, H., Liang, D., Hu, Z.: Automatic left ventricle segmentation from cardiac magnetic resonance images using a capsule network. Journal of X-Ray Science and Technology pp. 1–13 (03 2020). https://doi.org/10.3233/XST-190621
5. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with em routing. In: ICLR (2018)
6. Isensee, F., Kickingereder, P., Wick, W., Bendszus, M., Maier-Hein, K.H.: Brain tumor segmentation and radiomics survival prediction: Contribution to the brats 2017 challenge. In: BrainLes@MICCAI (2017)
7. Jaccard, P.: Etude comparative de la distribution florale dans une portion des alpes et des jura. Bull Soc Vaudoise Sci Nat **37**, 547–579 (1901)
8. LaLonde, R., Bagci, U.: Capsules for object segmentation. In: MIDL (2018)
9. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015)
10. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. pp. 234–241. Springer International Publishing, Cham (2015)
11. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 3856–3866. Curran Associates, Inc. (2017)
12. Setio, A.A.A., Traverso, A., de Bel, T., Berens, M.S., van den Bogaard, C., Cerello, P., Chen, H., Dou, Q., Fantacci, M.E., Geurts, B., van der Gugten, R., Heng, P.A., Jansen, B., de Kaste, M.M., Kotov, V., Lin, J.Y.H., Manders, J.T., Sóñora-Mengana, A., García-Naranjo, J.C., Papavasileiou, E., Prokop, M., Saletta, M., Schaefer-Prokop, C.M., Scholten, E.T., Scholten, L., Snoeren, M.M., Torres, E.L., Vandemeulebroucke, J., Walasek, N., Zuidhof, G.C., van Ginneken, B., Jacobs, C.: Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The luna16 challenge. Medical Image Analysis **42**, 1 – 13 (2017). https://doi.org/10.1016/j.media.2017.06.015
13. Simpson, A.L., Antonelli, M., Bakas, S., Bilello, M., Farahani, K., van Ginneken, B., Kopp-Schneider, A., Landman, B.A., Litjens, G.J.S., Menze, B.H., Ronneberger, O., Summers, R.M., Bilic, P., Christ, P.F., Do, R.K.G., Gollub, M., Golia-Pernicka, J., Heckers, S., Jarnagin, W.R., McHugo, M., Napel, S., Vorontsov, E., Maier-Hein, L., Cardoso, M.J.: A large annotated medical image dataset for the development and evaluation of segmentation algorithms. CoRR **abs/1902.09063** (2019)