universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Deep Learning based Segmentation for Multi-Modal Medical MRI Images"

verfasst von / submitted by

Mustafa Arikan, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Diplom-Ingenieur (Dipl.-Ing.)

Wien, 2018 / Vienna 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 940
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Scientific Computing
Betreut von / Supervisor:	Univ.-Prof. Torsten Möller, PhD

Abstract

In this thesis we propose an algorithm for the automatic segmentation of multi-modal data sets using deep learning while considering the memory footprint/computational cost versus accuracy trade-off. Thus we can achieve faster deep learning architecture search and parameter analysis for multi-modal semantic segmentation models. Our method is based on the SegNet architecture. We adapted SegNet and applied visualizations for finding accurate and robust semantic segmentation models. We evaluate our approach on manually segmented MRI data sets and compare to a state-of-the-art algorithm.

Zusammenfassung

In dieser Masterarbeit stellen wir einen automatischen Algorithmus für Segmentierung von multi-modalen Datenstzen mittels Deep Learning vor. Unser Algorithmus basiert auf der SegNet Architektur. Wir haben SegNet erwietert, sodass dieses nicht nur die drei Kanle eines RGB-Bildes, sondern eine beliebige Anzahl von Kanlen von multi-modalen Bildern verarbeiten kann. Weiters haben wir Visualisierungen für das Untersuchen von genauen und robusten Modellen entwickelt. Wir evaluieren unseren Algorithmus auf manuell segmentierten MRI Datensätzen und vergleichen mit einem State-of-the-Art Algorithmus.

Contents

1 Introduction

In recent years, deep learning based algorithms in computer vision have shown astounding performances in several application domains. Deep learning or Convolutional Neural Networks (CNN) are now considered the state-of-the-art for tasks such as object classification[1], detection[2] and also for semantic segmentation since the introduction of fully convolutional layers[3]. Fully convolutional architectures are performing very well on benchmarks like Microsoft COCO[4], PASCAL VOC2012[5], CamVid[6] and the Cityscapes[7]. These datasets focus on everyday scenes, buildings, roads or common objects from a human point of view. In this thesis, we apply deep learning for medical images from MRI devices. We rely on imaging data provided by the Multimodal Brain Tumor Image Segmentation Challenge (BRATS) 2016[8]. With the advancement of medical imaging and the development of MRI scanners, brain tumors can be detected by different Magnetic Resonance sequences, such as T1-weighted (T1), contrast enhanced T1-weighted (T1c), T2-weighted (T2) and T2-weighted with fluid attenuated inversion recovery (Flair) scans. We built segmentation models using Flair, T1, T1c, and T2 scans using architectures based on SegNet[9]. These four modalities are used to train and segment a semantic map containing four regions of interest in brain tumors. These four regions of interest are: necrosis, edema, non-enhancing core, and enhancing core. As medical volumes of data exponentially grow, deep learning networks can be trained to understand huge amounts of images. The task of manually segmenting or annotating MRI scans would therefore be a lot easier and/or could be replaced by automatic and semi-automatic segmentation tools. However, there are some challenges. First, the images have different gray scale ranges for different MRI vendors. Second, every voxel in MRI images has a semantic meaning because of the higher granularity. This differs from data sets mentioned above with everyday objects. This is very challenging in bordering regions, where the automatic algorithm could mix-up two or multiple labels.

In this thesis, we show how to train semantic segmentation models of multi-modal MRI data. We consider all four modalities of BRATS data and rely on the SegNet architecture. Moreover, we build on this baseline approach and use 3D information from the previous and the next slice available in the volumes to extend the number of modalities and to improve the accuracy. We use the Mask R-CNN[10] algorithm for a first rough estimation of the area of the tumor regions within the MRI scans. Thus we encircle the area of interest for the segmentation task and prevent mislabeling in the rest of the volume. We also calculate the entropy[11] of the segmented volumes to incorporate uncertainty into the model evaluation and selection. Finally, we provide interactive visualizations to evaluate, understand, highlight and select models and to compare to a state-of-the-art algorithm.

The remainder of the thesis is organized as follows. Section 2 describes related work about semantic segmentation using deep learning and semantic segmentation in MRI images. Section 3 describes the algorithm in detail, including a novel algorithm for the segmentation of brain tumors from medical MRI images, and the overall pipeline. In Section 4 a novel visualization interface is presented that helps for the task of understanding and selecting accurate and robust models. Section 5 elaborates on the experiments as well as the achieved results. The thesis is concluded (Section 6) with a discussion and ideas for future work (Section 7).

2 Related Work

In computer vision, the goal of semantic segmentation is to assign a label to each pixel or voxel of an image or a volume. Ciresan et. al.[12] use image patches and CNN classification, where each pixel is separately classified into labels. Long et. al.[3] replaced the fully connected layers at the end of a CNN with convolutional layers with large receptive fields. Therefore, the output is a probability map instead of a probability vector. Many state-of-the-art algorithms follow the principles described there. The encoder-decoder architectures reduce the spatial dimension using pooling layers in the encoder part to gather a rich representation of a collection of feature vectors and recover object details and spatial dimension in the decoder part to produce an output with the input dimensions[13, 9]. Dilated convolutions are a different approach to convolution. Dilated convolutions use a parameter called dilation rate, which defines the spacing between the values in a convolutional kernel. This reduces the number of parameters for larger receptive fields[14]. Conditional random fields are used to smooth and improve results of underlying probabilistic results[15].

In medical image segmentation, deep learning models were developed to segment tumors by Pereira et al.[16] using very deep convolutional neural networks[17] with very small 3 x 3 kernels applied on

patches. Similar approaches were developed by Darko et al.[18], Havaei et al.[19] training CNNs using image patches. These approaches classify each patch into classes, such as healthy tissue, necrosis, edema, non-enhancing core, and enhancing core. The classification result is used to label the pixel in the center. The result is a semantic map representing the segmentation of the tumor. 3D-CNN based approaches were developed by Kamnitsas et al.[20] and Yi et al.[21]. These approaches could use all the available 3D volume data but the increased network sizes and the computational costs lead to the adoption of 2D-CNNs.

In visualization research, tools have been developed to help users to interpret results from deep learning models and to gain insight into deep learning in general. One common strategy is to visualize filters and weights, and this is done using heat maps since filters are two-dimensional convolution kernels. Zeiler and Fergus[22] introduced visualizations for feature activation in intermediate layers using deconvolutions to highlight the regions from images that are responsible for the firing of neurons of the layers. Another approach is retrieving and visualizing images and image regions that maximally activate neurons by Girshick et al.[23]. Kahng et al.[24] introduced an interactive visualization system for interpreting large-scale deep learning models and results. They developed an instance-based exploration by tracking how an example behaves inside the models and an feature- and subset-based exploration to enable users to better understand the relationships between data and models. Pezzotti et al.[25] presented DeepEyes, which is a visual analysis tool improving on ideas from [22] and [23] to support users during the training process through filter analysis and early feedback.

We propose a new brain tumor segmentation method by using SegNet[9] and Mask R-CNN[10] in a unified framework. We train the basic SegNet and the standard SegNet using all four modalities and the 3D information from the previous and the next slice in the volumes. This allows us to take full advantage of 3D information of the MRI data in a 2D encoder-decoder architecture while limiting the computational cost. We choose SegNet, since it provides a good balance between accuracy and inference time and memory usage. And we use Mask R-CNN as a preprocessing step to detect the overall region of the tumor and to limit the region, where the segmentation should be applied.

3 Method

In this section, the pre-processing including normalization is addressed. We also describe the used architectures and the modifications to the architectures. We describe the SegNet architecture for the segmentation task and the extension to be able to process multi-modal images. We then describe the application of the Masked R-CNN to find the regions of interest for the segmentation task. At the of the section we describe the method for class balancing for the different labels.

3.1 Preprocessing

The different modalities of the MRI volumes are already registered. Therefore, there is no need to apply any registration methods. We normalize the intensity values of the voxels of every modality by subtracting the mean and dividing by the standard deviation. Thus our intensity values will have mean zero and variance/standard deviation of 1. As next step, we crop the areas of the tumors from the volumes and create LMDB[26] files to store the training images and the modalities. The cropping is done using the ground truth images, where we find smallest rectangle area encompassing the voxels with the unhealthy labels. The data sets consist of 274 volumes, where 220 are used for the training and 54 for validation and testing. For the purpose of training, we prepare different parameter settings to train multiple models for the evaluation using the SegNet and SegNet basic. The SegNet architecture itself is based on VGG-16[17] and has 13 convolutional layers. The SegNet basic is a smaller version of SegNet with 4 convolutional layers. The size of the images is 166 x 130. Given current GPU memory limitations, the small image size is ideal to pass through multiple images to GPU at every iteration. The original SegNet is applied to RGB images with 3 channels. The input layer of SegNet is taking RGB images as input. We adapt SegNet to use 4 or more channels using the LMDB input module.

3.2 Architectures

We use both the SegNet and the SegNet basic architecture. SegNet has an encoder-decoder architecture based on the convolutional layers of VGG-16 by Simonyan et. al.[17]. The used architecture can be seen in figure 1. The network used for the semantic segmentation takes four images (T1c, T1, T2 and Flair) as input data, and have a label, which is a 5 channel image, where 5 is the number of labels involved. The labels are healthy tissue or background (0), necrosis (1), edema (2), non-enhancing core (3), and enhancing core (4). The architecture has two important parts, namely encoder and the decoder. The architecture has the typical bulding blocks of CNNs, such as convolutional layer with a number of filters, pooling layers to reduce the spatial dimensions, the batch normalization layers, the ReLU layers and the softmax layer as the final output layer. The encoder applies convolutions, followed by pooling operations, batch normalization and rectified linear units. The decoder part has the same number of convolution layers and the same number of blocks. Instead of pooling, the decoder part applies upsampling using unpooling layers.

- Convolutional Layer: a layer with a defined number convolution filters, where the size of the filters are defined and the parameters or weights are learned through back-propagation. The output of the convolution layer is the activation or the feature map.
- Batch normalization: a layer to estimate the mean and standard deviation of each feature at training time to center and normalize the features of the minibatch. The minibatch are the images processed at once every iteration.
- ReLU: is an activation layer to introduce nonlinearity to the network after the convolution layers.
- Pooling Layer: a layer used to reduce the spatial size of the activation maps, to reduce the amount of parameters and computational cost in the network and control overfitting using e.g. max or average pooling.
- Upsampling: the upsampling is achieved using deconvolution or transposed convolution.

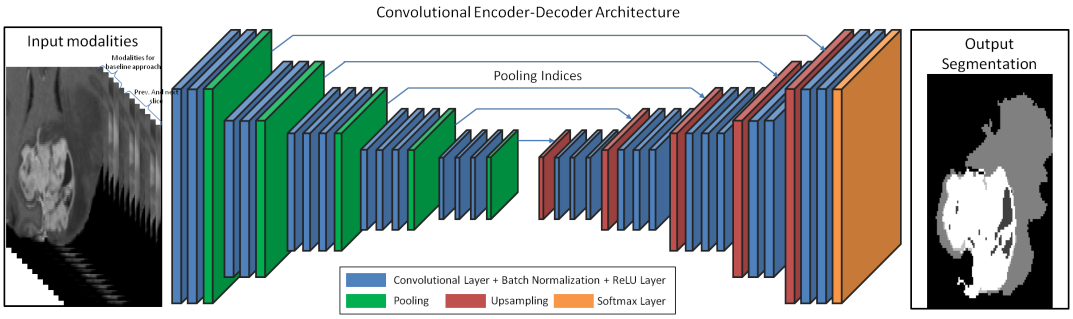

Figure 1: The segmentation architecture with four or twelve input modalities with the T1c, T1, T2 and Flair modalities from the current, previous and the next slice of the MRI volume. The architecture is based on the encoder-decoder architecture from SegNet.

We changed the input layer from an RGB input to a multi-modal input with four or more input channels using Lightning Memory-Mapped Database (LMDB). Thus, we can not only process the four modalities of MRI volumes but also the modalities from the neighboured slices of the volumes as well. Therefore, we can take full advantage of the 3D information in a 2D-CNN architecture with a light-weight memory footprint.

Listing 1: Configuration for Multi-modal Input

```
name: "segnet"

layer {
  name: "data"
  type: "Data"
  top: "data"
  data_param {
    source: "/SegNet/tif_training_brats_not_normalized_3D_3_slices"
```

```
    batch_size: 6
    backend: LMDB
  }
}

layer {
  name: "data"
  type: "Data"
  top: "label"
  data_param {
    source: "/SegNet/tif_training_brats_not_normalized_label"
    batch_size: 6
    backend: LMDB
  }
}
```

The Masked R-CNN is used as a preprocessing step to the segmentation architecture. The Masked R-CNN is an approach to solve the problem of instance segmentation. In instance segmentation, the goal is to not only make pixel- or voxel-wise predictions for regions of interest but also to identify each object entity separately as part of that object class. Masked R-CNN detects objects while simultaneously generating mask for each instance. It divides the task of object detection in sub-tasks: finding objects (objects proposal) and understanding them (region classification). The variants or predecessors of Masked R-CNN are called R-CNN[23], Fast R-CNN[27] and Faster R-CNN[28], where the speed was improved gradually. Masked R-CNN gets regions of interest of an image and must classify them. Regions of interest are rectangles of different shapes. These are transformed into the input shape of a particular ConvNet (e.g. AlexNet[1]). The regions are processed through the classification network and the values obtained on the last feature map are saved. These feature maps are transformed to vectors and used for classification. A one-vs-rest support vector machines strategy is applied on all feature vectors of the regions. At the end a bound box regression is applied to center the area on the object.

3.3 Pipeline

The segmentation pipeline consists of two important parts. The first part is the tumor detection module. This is a deep learning based object detection algorithm using Mask-RCNN. Mask-RCNN is a deep neural network aimed to find instances of objects in images. It can find and separate different regions in an image. The output is a bounding box of found objects. First, proposals are calculated about the regions where there might be objects. Second, it predicts the probability for the found object, and refines the bounding box and generates a mask of the object based on the proposals from the first stage. This is a preliminary step for the segmentation. The detection module is trained on the same modalities as the segmentation module, only the labels are different. All tumor labels are set to the same value. Therefore, the detection module is used to detect the overall region of the tumor as whole. The output of the detection module is a subvolume of the original volume. This is used to prepare the modalities for the segmentation task. After the detection of the tumor, the output is forwarded to the input of the segmentation module. The segmentation is applied on the subvolume as described in section 3.2.

3.4 Class Balancing

By detecting the overall tumor area, we keep the area for the segmentation task small. This is important for the segmentation part in order to keep both the training runtime and the inference runtime low and to avoid attention to areas, where the algorithm might learn no relevant information. This is also good for faster convergence of the algorithm as well. A second hurdle is the class imbalance. The number of voxels for individual labels in the ground truth images are unbalanced. There is a high imbalance between background and healthy voxels and the remaining labels and also between the remaining tumor labels. We apply class balancing for our models by weighting each class for the loss function according to Eigen and Fergus[29]. We weight each pixel by $a_c = median\_freq/freq(c)$ where $freq(c)$ is the number of voxels of class c divided by the total number of voxels in images where c is present, and $medianfreq$ is the median of these frequencies.

Listing 2: Code for Class Balancing

```
counter = [0,0,0,0,0];
for i in range (n_label):
    for j in range (166):
        for k in range (130):
            if (final_label[i][0][j][k] < 5):
                counter[final_label[i][0][j][k]] = counter[final_label
                    [i][0][j][k]] + 1;
counter2 = [0,0,0,0,0];
for i in range (n_label):
    isThere = [False, False, False, False, False];
    for j in range (166):
        for k in range (130):
            if (final_label[i][0][j][k] < 5):
                isThere[final_label[i][0][j][k]] = True;
    for l in range(0, 5):
        if(isThere[l] == True):
            counter2[l] = counter2[l] + 1;
for i in range (0,5):
    counter3.append(counter2[i]*130*166);
freq = []
for i in range (0,5):
    freq.append(float(float(counter[i])/float(counter3[i])));
ac = []
for i in range (0,5):
    ac.append(float(np.median(freq))/float(freq[i]));
```

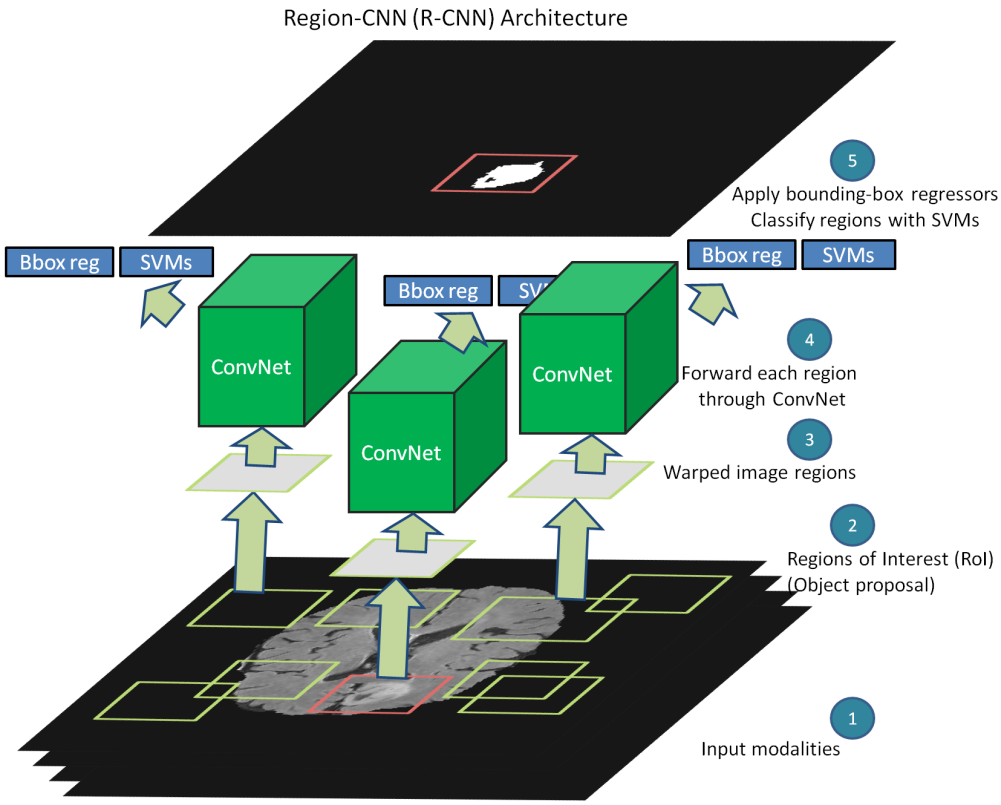

Figure 2: The pre-processing architecture for finding the region of the tumor. The architecture is based on the Masked R-CNN approach. The input modalities are the same as for the segmentation architecture. Object proposals are extracted from the input modalities, transformed into the right shape for the ConvNet for classification. The output of the last layer of the classification network is used as input for SVM classification and bounding-box regression.

4 Evaluation

To compare our method with a current state-of-the-art method, we trained several models using data from BRATS 2016 with different hyperparameter settings such as learning rate or number of epochs. For the evaluation, the dice coefficient was calculated for the whole tumor region as well as for individual labels. The model that we tested against is the one by Wang et al.[30]. This algorithm is a cascaded system trained on particular labels instead of on all labels at once. Therefore, multiple models are trained for individual labels or for the whole tumor. It also utilizes all three planes (axial, coronal and sagittal) for improving the accuracy by training models for each plane with the overall drawback of longer training process.

4.1 Visualization of Results

To provide useful visualizations for analyzing models for accuracy and robustness, we identify the following tasks. The tasks are related to the results of the segmentation in terms of accuracy or uncertainty.

- (Task 1) Provide an overview over all trained models: an overview for accuracy or uncertainty for getting a quick understanding in order to identify interesting models.
- (Task 2) Finding interesting data sets: visualizations to delve into specific models for analyzing and finding interesting data sets and results thereof.
- (Task 3) Monitoring the training process: to evaluate the progress, to spot overfitting on data set level and compare different stages within one model.
- (Task 4) Visualize detailed results for accuracy and uncertainty: to compare individual models on data set level and slice level.

For the training process, we experimented with different hyperparameter settings to create multiple models to evaluate. We trained 8 different models (see table 1), where we set different learning rates, used different architectures (SegNet or SegNet basic) or used different normalization techniques (including batch normalization) as a starting point. We follow the visual information-seeking mantra *"overview first, zoom and filter, then details-on-demand"* by Shneiderman[31]. We visualized results for median accuracy for the whole model or at individual data set level or per slice of the volume. We begin with an overall visualization, where we show results for all models in an interactive visualization (figure 3). This visualization provides information regarding parameters of individual models and accuracy using the median dice score (Task 1). As next step, we visualize the dice scores for every validation or test data set in an interactive visualization, where one can select particular data

Table 1: Summary of trained models:

- Name: name defined for the trained model.
- Architecture: the architecture used for this particular model in training.
- Learning rate: learning rate used for this particular model in training.
- No. Modality: number of modalities; four if only 2-D data is used or 12 if 3-D information from the previous and the next slice is used as well.
- Normalization: this shows if global normmalization is applied, and BN if batch normalization applied.

Name	Architecture	Learning rate	No. Modality	Normalization
Model 1	SegNet	0.01	4	No normalization
Model 2	SegNet	0.001	4	No normalization
Model 3	SegNet	0.001	4	Normalization
Model 3 in 3D	SegNet	0.001	12	Normalization
Bas. Mod. 1	SegNet Basic	0.01	4	No normalization
Bas. Mod. 2	SegNet Basic	0.01	4	No normalization + BN
Bas. Mod. 3	SegNet Basic	0.01	4	Normalization + BN
Bas. Mod. 1 in 3D	SegNet Basic	0.01	12	No normalization

sets and corresponding dice scores for every model (figure 4). This visualization is used to identify interesting data sets (Task 2). And finally, we visualize the details for individual slices from the volumes, where we can compare the ground truth to segmentation result of individual data sets (figure 5). The visualization of individual slices and results help to analyze qualitative and quantitative results (dice scores per slice) in order to solve Task 4.

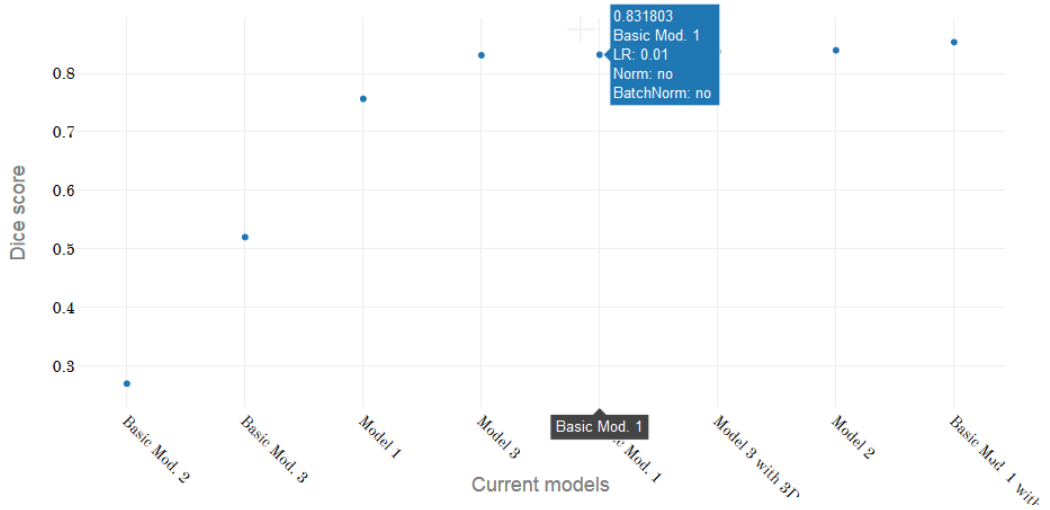

Figure 3: Overview visualization to show all trained models (x-axis) from low median accuracy to high median accuracy (y-axis).

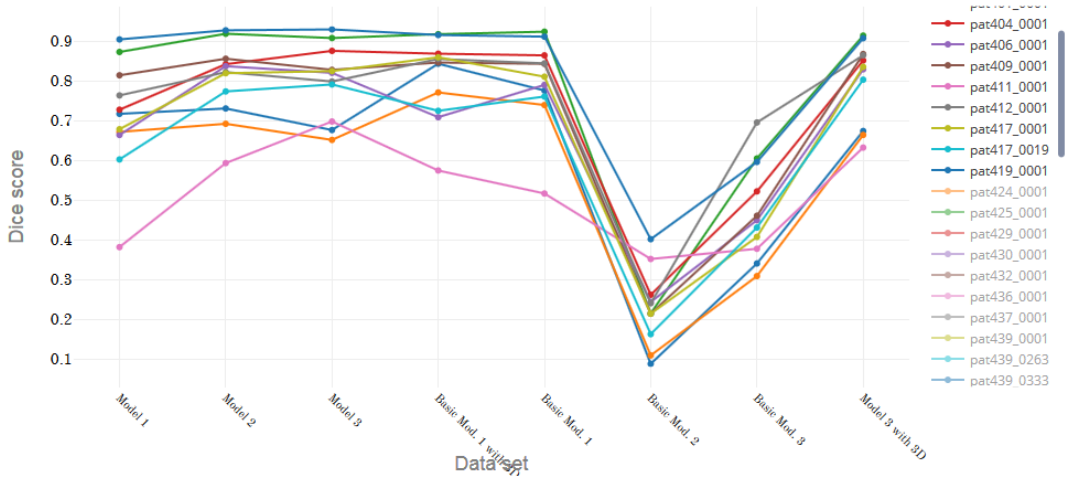

Figure 4: Distribution of dice scores (y-axis) for individual data sets (right) and models (x-axis).

4.1.1 Epoch Visualization

While training our models we create checkpoint models at certain epochs. To evaluate and understand the progress and the accuracy of the trained model we visualize the median dice score for the checkpoint models, meaning we show the dice score for individual training epochs. The visualization helps us to see the progress of the training process up to the model with the best median accuracy (Task 1 and Task 3). In this visualization, one can compare one epoch to the previous or next epochs. This provides information to monitor progress and spot overfitting on training data (Task 3). In figure 6 you can see this visualization for SegNet basic with 3D information incorporated into training.

Furthermore, the epoch visualization can be switched to show more detail on the individual data set level. The individual data sets can be selected to highlight the progress across different epochs

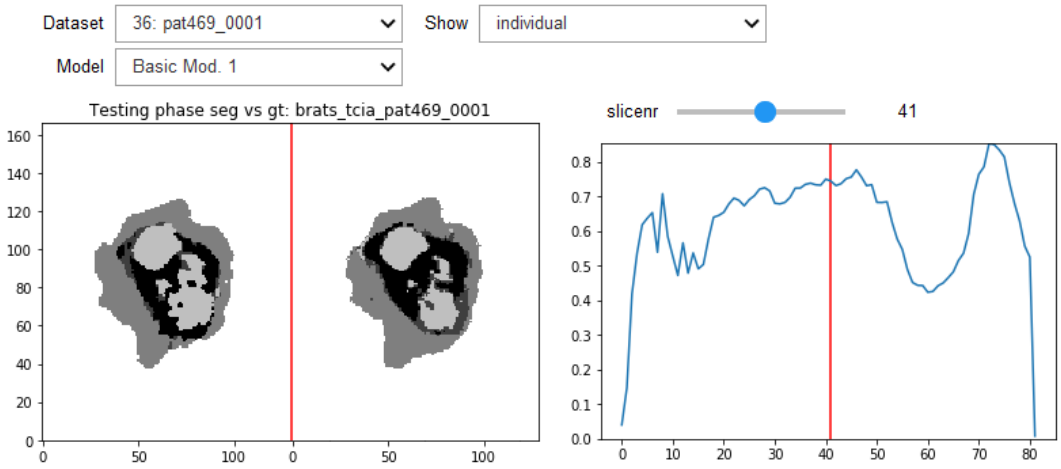

Figure 5: Individual data set and slice visualization (left), with a visualization of the segmentation (left) vs. ground truth (right) and the dice score of individual slices.

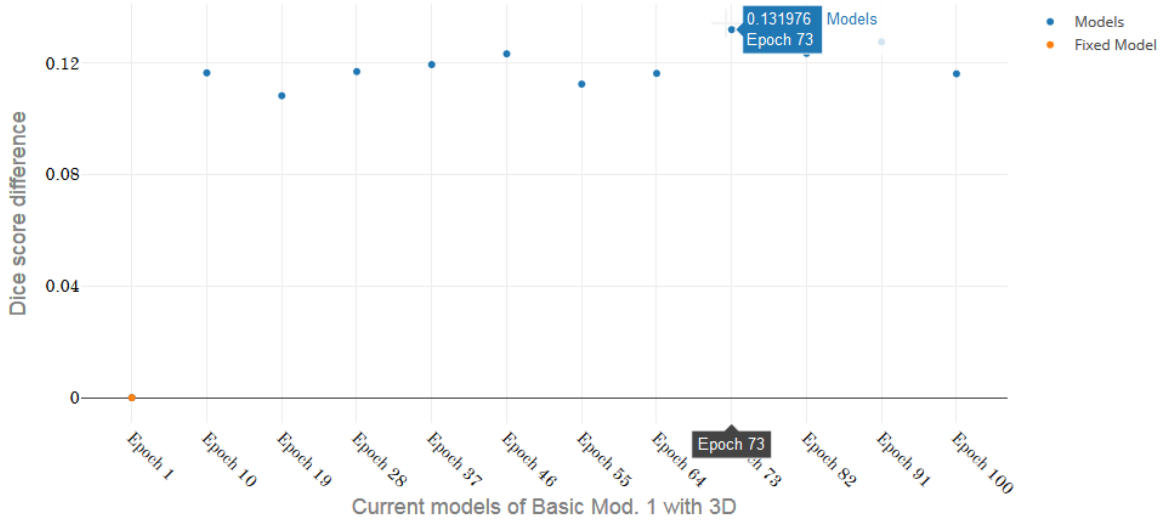

Figure 6: Epoch visualization for single model: the fixed checkpoint model is selected as a reference. In this example the reference is set at epoch 1 with a dice score of 0.7258. And one can see the accuracy going up until epoch 73 (dice score 0.86).

(figure 7). The epoch visualization is used to analyze the convergence of the neural network, and is an indicator for parameter search for the learning in cases when e.g. the convergence is either too slow or too fast.

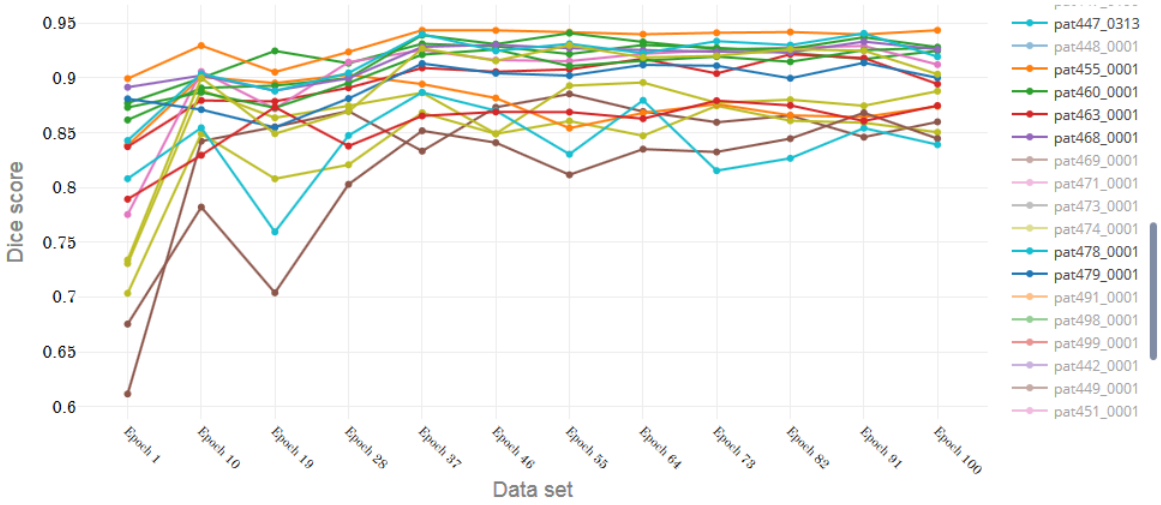

Figure 7: Epoch visualization at individual data set level. The accuracy of selected data sets for all the trained checkpoint models are shown. The data sets can be analyzed individually by selecting.

4.1.2 Comparative Visualization

We extended our visualizations to be able to compare multiple results from different epochs or models. Since there are multiple models and multiple checkpoints, it is useful to compare checkpoints and models against each other for the purpose of model understanding and selection. We show results from selected models together with input modalities in the individual slice visualization with the corresponding dice scores (figure 8). This visualization shows detailed results, to compare results from different models side-by-side and analyze individual slices of the volumes (Task 4).

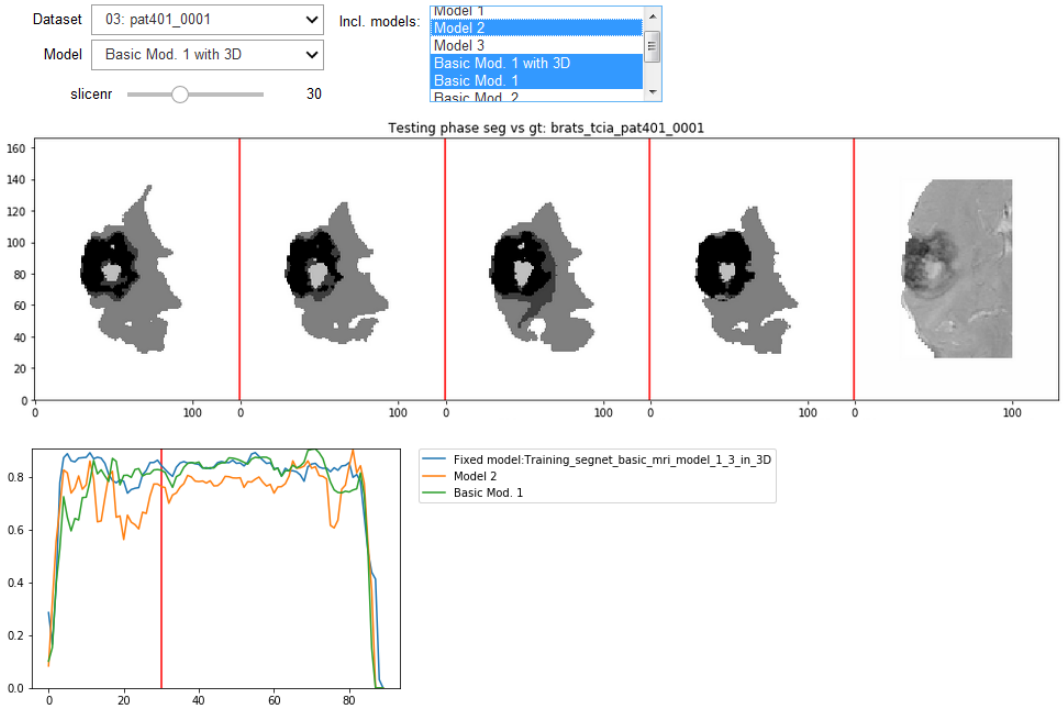

Figure 8: Comparative visualization of three selected models and the ground truth. The accuracy for individual slices is visualized at the bottom. One input modality can be selected to compare to, in order to understand the potential importance or influence of the selected modality.

4.1.3 Uncertainty Visualization

Uncertainty plays a major role to fully understand data and algorithm characteristics. This is also true for deep learning networks and for semantic segmentation results. Therefore, we compute the entropy according to Potter et al.[11]. This also relates to the cross-entropy used in the loss function. The entropy is defined as $H[x] = -\sum_{i=1}^{n} p(x_i) * log(p(x_i))$. Consider the following two examples for the entropy. Since we have five different labels, we have five probabilistic results per voxel. The range for the entropy lies between 0 and 2. The two examples show these two limits.

- $p_{min} = \{1.0, 0.0, 0.0, 0.0, 0.0\} \Rightarrow H[X] = 0.00$
- $p_{max} = \{0.2, 0.2, 0.2, 0.2, 0.2\} \Rightarrow H[X] = 2.00$

The first example, p_{min}, indicates high confidence of the algorithm in its result. The algorithm sets the value for this example to label 0 and the corresponding entropy is 0. The second example on the other hand, is the maximal-entropy case, therefore the algorithm can't decide on a label. We calculate the entropy with:

Listing 3: Code for Computing the Entropy

```
entropy = np.negative(np.add(np.add(np.add(np.add(np.multiply(
problabel0, np.log2(problabel0))), np.multiply(
problabel1, np.log2(problabel1))), np.multiply(
problabel2, np.log2(problabel2))), np.multiply(
problabel3, np.log2(problabel3))), np.multiply(
problabel4, np.log2(problabel4))));
```

We visualize the uncertainty within the segmentations using box plots. It is not only important to see the progress of the learning process along several epochs regarding accuracy. Uncertainty should be considered as well. Therefore, we integrate uncertainty information into the model selection. We show results for the checkpoint models, where uncertainty is integrated and visualized as shown in figure (figure 9). This visualization shows uncertainty for specific models (box plots) and data sets (scatter plot) to solve Task 4. A more detailed slice visualization for entropy is calculated as well (see figure 10).

5 Experiments

5.1 Experimental Results

To compare our method and our trained models with the current state-of-the-art, we train a model based on the U-Net architecture described by Wang et al.[30] using the same training data sets. This method won 2nd place at BRATS 2017. Our best models, without any post-processing applied, achieve results, which are approaching the state-of-the-art results with room for further improvement (see table 2). We also present example segmentation results in figure 11.

Table 2: Dice score for the whole volume

Data set	State-of-the-art result	Our result
brats_tcia_pat399_0595	0.958	0.864
brats_tcia_pat399_0815	0.897	0.830
brats_tcia_pat401_0001	0.913	0.803
brats_tcia_pat404_0001	0.943	0.873
brats_tcia_pat406_0001	0.967	0.903
brats_tcia_pat409_0001	0.931	0.856
brats_tcia_pat411_0001	0.965	0.864
brats_tcia_pat412_0001	0.841	0.757
brats_tcia_pat417_0001	0.847	0.791
brats_tcia_pat417_0019	0.801	0.771

The inference time for the state-of-the-art model is on average about 40 seconds per volume for the whole tumor region only. We have an average inference time of 28 seconds for all regions of interest.

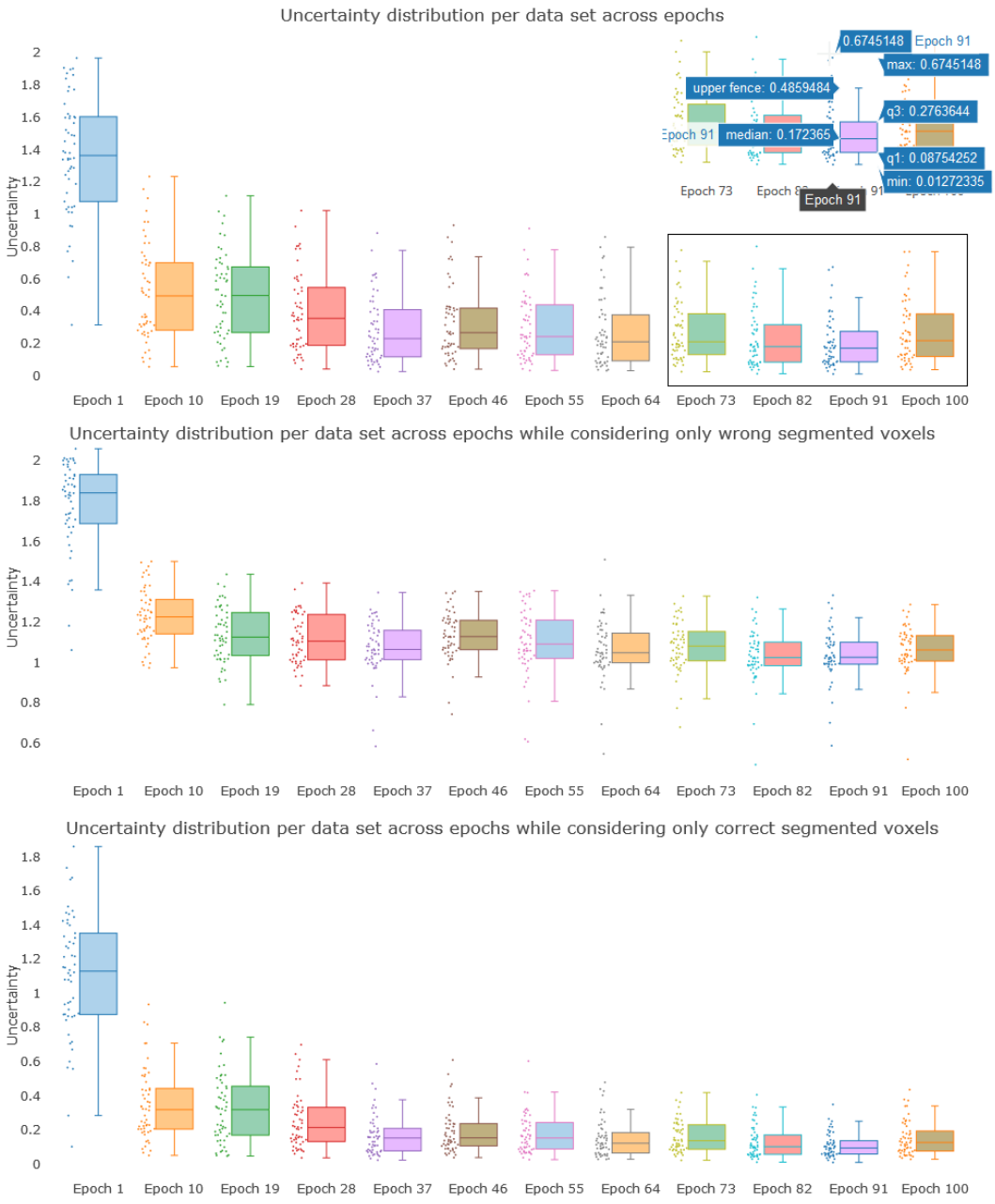

Figure 9: Visualization of uncertainty (y-axis) using box plots for models and scatter plots for individual data sets and checkpoint models (x-axis). We calculate three different uncertainty distributions, overall uncertainty (top), uncertainty regarding wrong segmented voxels (middle) and uncertainty regarding correctly segmented voxels for a particular model.

The training time for the state-of-the-art model is about 40 hours for the whole tumor region only. The training time four our model is about 13 hours for all regions of interest.

6 Conclusion

We have developed a unified framework for tumor detection and segmentation using a deep learning architecture with a light-weight memory footprint. Our tool enables to train tumor detection and

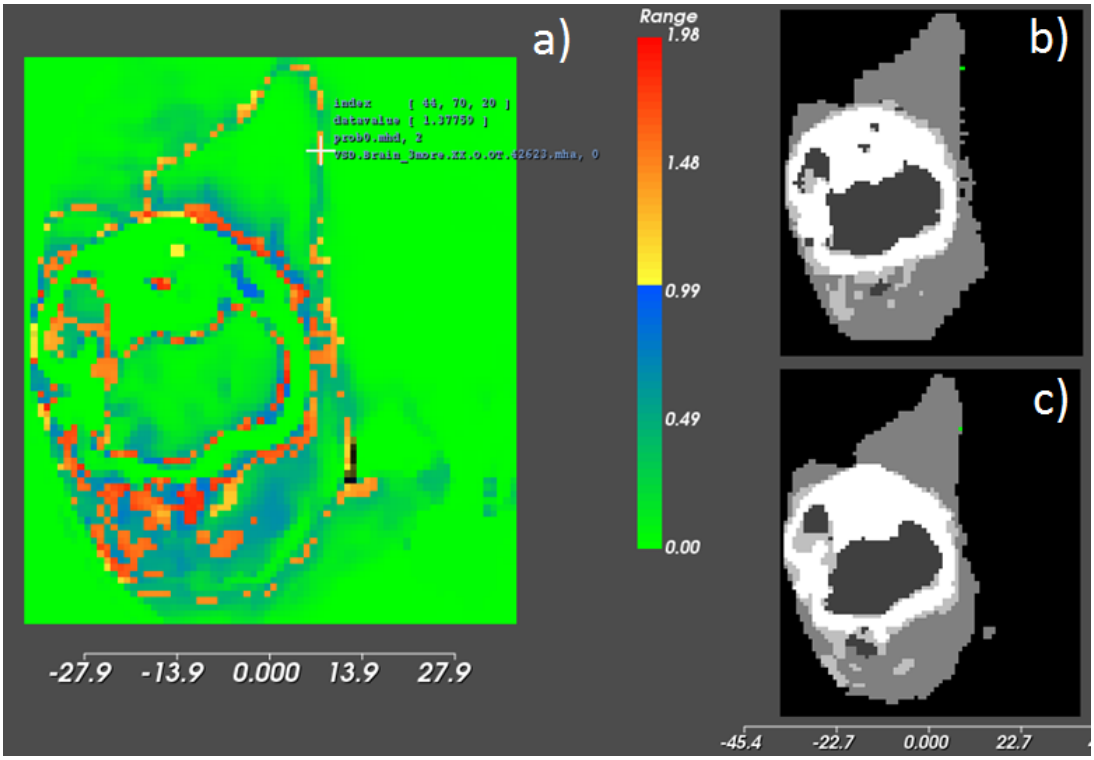

Figure 10: Visualization of uncertainty for a single slice, where two different color maps show (normalized) uncertainty for areas with correct segmentation (green to blue or 0.0 to 1.0) and areas, where the segmentation failed (yellow to red or 1.0 to 2.0). In addition, we show the ground truth (b) and the segmentation (c).

segmentation models for multi-modal MRI data sets. Our starting point was the use of multi-modal data sets in different application domains. Our tool is developed in a modular fashion and can be modified and applied to other use cases as well, where multi-modal data sets are available, such as hyperspectral imaging or RGB-D scene understanding. Our algorithm has potential as an alternative to segmentation frameworks with much more complex architectures, higher memory use, longer training process, and higher inference times. The experimental results are promising, even though we only need one model for the prediction of all labels instead of specific models for the regions with different labels. We also train using axial plane slices only instead of training multiple models for axial, coronal, and the sagittal plane slices. Therefore, our algorithm is kept simple and still has potential for further improvements.

7 Future Work

We will apply our tool to different multi-modal data sets. We will experiment with applying conditional random fields on the output results in order to get smoother segmentations. We would like to analyze not only the influence of hyperparemeters such as learning rate or number of epochs, but also the parameters of individual layers of the network as well. We will experiment with different kernel sizes for the convolutional layers (e.g. smaller kernels for all convolutional layers or combinations of small and large kernels) and with different types of convolution (e.g. dilated convolutions) and pooling strategies (e.g. pyramid pooling). We also would like to see the benefit of extensive data augmentation. We will include uncertainty into the segmentation architecture and/or adapt Bayesian SegNet for multi-modal data sets. These steps will bring us a step further to understanding the complex interior of the deep learning networks for multi-modal semantic segmentation with the overall goal of being able to easily do debugging and to do visual neural architecture search on these algorithms.

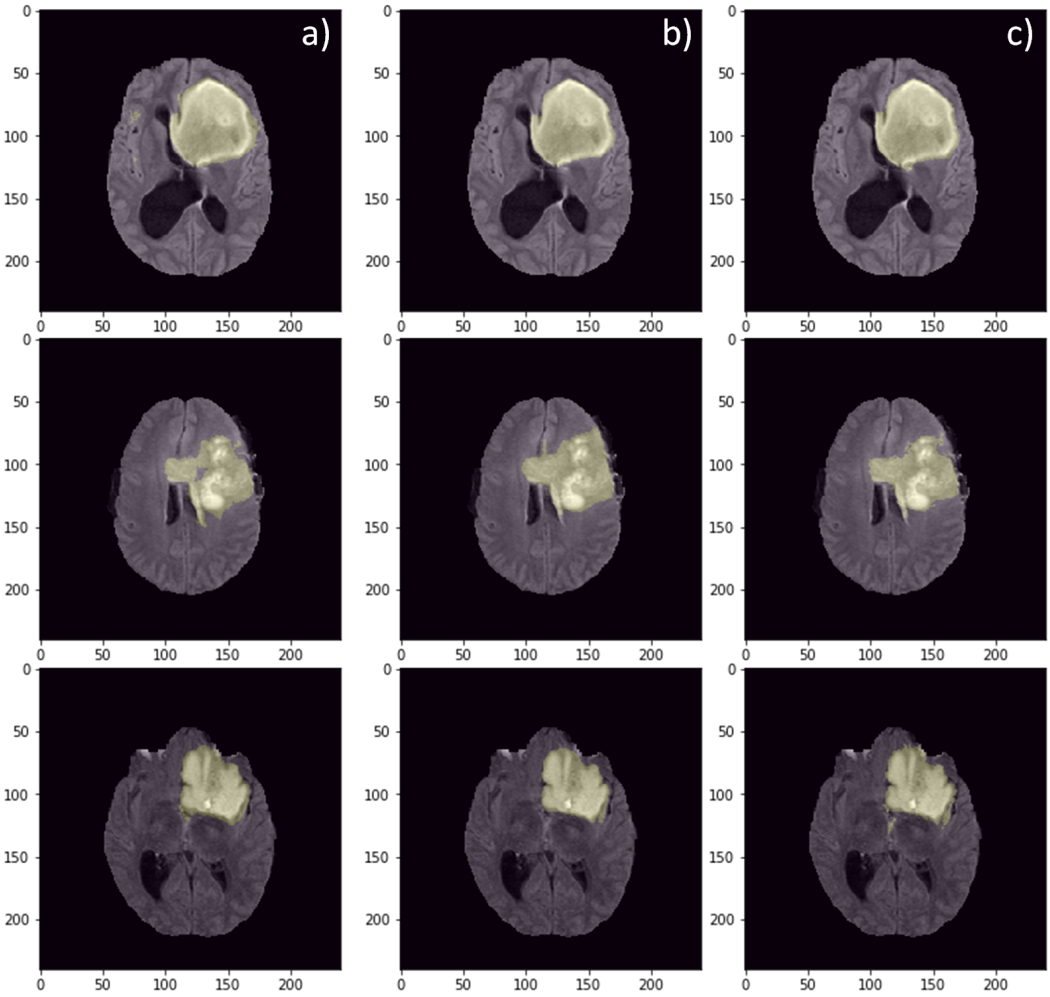

Figure 11: Example segmentations using our algorithm (a) compared with the state-of-the-art results using the model from [30] (b) and the expert ground truth segmentation (c).

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105. URL: http://dl.acm.org/citation.cfm?id=2999134.2999257.

[2] Pierre Sermanet et al. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *CoRR* abs/1312.6229 (2013). arXiv: 1312.6229. URL: http://arxiv.org/abs/1312.6229.

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *CoRR* abs/1411.4038 (2014). arXiv: 1411.4038. URL: http://arxiv.org/abs/1411.4038.

[4] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[5] M. Everingham et al. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136.

[6] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. "Semantic Object Classes in Video: A High-definition Ground Truth Database". In: *Pattern Recogn. Lett.* 30.2 (Jan. 2009), pp. 88–97. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2008.04.005. URL: http://dx.doi.org/10.1016/j.patrec.2008.04.005.

[7] Marius Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *CoRR* abs/1604.01685 (2016). arXiv: 1604.01685. URL: http://arxiv.org/abs/1604.01685.

[8] Bjoern H. Menze et al. “The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS).” In: *IEEE Trans. Med. Imaging* 34.10 (2015), pp. 1993–2024. URL: http://dblp.uni-trier.de/db/journals/tmi/tmi34.html#MenzeJBKFKBPSWL15.

[9] V. Badrinarayanan, A. Kendall, and R. Cipolla. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”. In: *IEEE Transactions on Pattern Analysis Machine Intelligence* 39.12 (Dec. 2017), pp. 2481–2495. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2644615. URL: doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2644615.

[10] Kaiming He et al. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2980–2988.

[11] K. Potter, S. Gerber, and E. W. Anderson. “Visualization of Uncertainty without a Mean”. In: *IEEE Computer Graphics and Applications* 33.1 (Jan. 2013), pp. 75–79. ISSN: 0272-1716. DOI: 10.1109/MCG.2013.14. URL: doi.ieeecomputersociety.org/10.1109/MCG.2013.14.

[12] Dan Ciresan et al. “Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 2843–2851. URL: http://papers.nips.cc/paper/4741-deep-neural-networks-segment-neuronal-membranes-in-electron-microscopy-images.pdf.

[13] O. Ronneberger, P.Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, 2015, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

[14] Fisher Yu and Vladlen Koltun. “Multi-Scale Context Aggregation by Dilated Convolutions”. In: *CoRR* abs/1511.07122 (2015). arXiv: 1511.07122. URL: http://arxiv.org/abs/1511.07122.

[15] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. ISBN: 1-55860-778-1. URL: http://dl.acm.org/citation.cfm?id=645530.655813.

[16] Sérgio Pereira et al. “Deep Convolutional Neural Networks for the Segmentation of Gliomas in Multi-sequence MRI”. In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Ed. by Alessandro Crimi et al. Cham: Springer International Publishing, 2016, pp. 131–143. ISBN: 978-3-319-30858-6.

[17] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.

[18] Darko Zikic et al. *Segmentation of Brain Tumor Tissues with Convolutional Neural Networks*. Sept. 2014.

[19] Mohammad Havaei et al. “Brain tumor segmentation with Deep Neural Networks”. In: *Medical Image Analysis* 35 (2017), pp. 18–31. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2016.05.004. URL: http://www.sciencedirect.com/science/article/pii/S1361841516300330.

[20] Konstantinos Kamnitsas et al. “Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation”. In: *Medical Image Analysis* 36 (2017), pp. 61–78. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2016.10.004. URL: http://www.sciencedirect.com/science/article/pii/S1361841516301839.

[21] Darvin Yi et al. “3-D Convolutional Neural Networks for Glioblastoma Segmentation”. In: *CoRR* abs/1611.04534 (2016). arXiv: 1611.04534. URL: http://arxiv.org/abs/1611.04534.

[22] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *CoRR* abs/1311.2901 (2013). arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901.

[23] Ross B. Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *CoRR* abs/1311.2524 (2013). arXiv: 1311.2524. URL: http://arxiv.org/abs/1311.2524.

[24] Minsuk Kahng et al. “ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models”. In: *CoRR* abs/1704.01942 (2017). arXiv: 1704.01942. URL: http://arxiv.org/abs/1704.01942.

[25] Nicola Pezzotti et al. “DeepEyes: Progressive Visual Analytics for Designing Deep Neural Networks”. In: *IEEE Trans. Vis. Comput. Graph.* 24.1 (2018), pp. 98–108. DOI: 10.1109/TVCG.2017.2744358. URL: https://doi.org/10.1109/TVCG.2017.2744358.

[26] H. Chu. “Lightning memory-mapped database”. In: *https://en.wikipedia.org/wiki/Lightning_Memory-Mapped_Database* (2018). [Online; accessed 31-October-2018].

[27] Ross B. Girshick. “Fast R-CNN”. In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[28] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[29] David Eigen and Rob Fergus. “Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture”. In: *CoRR* abs/1411.4734 (2014). arXiv: 1411.4734. URL: http://arxiv.org/abs/1411.4734.

[30] Guotai Wang et al. “Automatic Brain Tumor Segmentation using Cascaded Anisotropic Convolutional Neural Networks”. In: *CoRR* abs/1709.00382 (2017). arXiv: 1709.00382. URL: http://arxiv.org/abs/1709.00382.

[31] Ben Shneiderman. “The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations”. In: *Proceedings of IEEE Symposium on Visual Languages*. IEEE Computer Society, 1996, pp. 336–. ISBN: 0-8186-7508-X. URL: http://dl.acm.org/citation.cfm?id=832277.834354.