Automatic Brain Tumor Segmentation

Michael Mernagh
mernagh@stanford.edu

Mihir Pendse
mpendse@stanford.edu

Abstract

We present a deep learning based algorithm for automatic segmentation of brain tumors from a set of MRI scans. The input is a set of four 3D scans (240x240x155) with different constrast settings (T1, T2, T1 with contrast enhancement, and FLAIR), and the output is a labeled 240x240x155 image where each voxel is classified as one of (a) healthy tissue, (b) necrosis or nonenhancing tumor, (c) edema, or (d) enhancing tumor.

We will explore three different architectures of increasing complexity: a simple 2D model which outputs a single prediction for the central pixel of the slice, a 3D triplanar model which takes three orthogonal views as an input and outputs a prediction for each voxel in a 3D central patch (which is significantly smaller in size than the input), and a U-net model which is similar to the triplanar model but includes down and up convolution layers and skip connections. We use the Dice score as our evaluation metric to assess the similarity between the predicted labels and the ground truth (manually labeled by a radiologist) and achieve a score of over 0.8 on the validation set with the U-net model.

1. Introduction

Gliomas are the most common form of malignant brain tumor and treatment usually requires image-guided surgery or radiation therapy [8]. Since gliomas are heterogeneous and contain subregions corresponding to different stages of the disease, accurate treatment requires a a labeled image of the patient's brain depcting these stages. Currently, labeling is done manually by a highly trained radiologist using a set of MRI scans. However, this procedure is both time-intensive, and subject to inter- and intra-labeler error.

This labeling process is time-consuming because it requires analyzing several different images (different contrasts and views) simultaneously which is difficult for humans, but not particularly difficult for a computer with sufficient processing power. We aim to speed up this task by automating the segmentation. The input to the system is the output from an individual MRI scan, which is a 3D image of the brain, with 4 channels each corresponding to different MRI contrasts (T1, T2, T1 with contrast enhancement, and FLAIR). We output a volume that is of the same dimensions as the input scan, where each voxel of the input is replaced with a label that is one of {tumor core, peritumoral edema, enhancing tumor, non-tumor}.

2. Related Work

Prior to the development of deep learning, most machine learning approaches to tackle the tumor segmentation task involved training a discriminative model that used predetermined features extracted from each of the input modalities followed by a classifier to label each voxel as a certain tissue type, usually using random forests [15]. Typically, Markov random fields or conditional random fields are used for regularization of the predicted class types [11].

Since 2014, deep learning methods have shown to have the best performance in multiple tumor segmentation challenges and are likely to replace traditional machine learning methods for segmentation [12]. The use of a CNN with inputs of patches is common [2] for the segmentation problem where the task is to classify the center pixel in each patch [13]. This has been attempted specifically in the brain tumor segmentation domain by Zikic et al. [18] and Dvorak and Menze [4]. Even though the MRI is 3D, most approaches are performed in 2D for computational tractability and because of the fact that MR images are usually anisotropic (different resolution in each orientation). In some cases patches in three orthognal orientations (axial, sagittal, and coronal) for the same central voxel are used as an input rather than a complete 3D volume [14]. In order to capture both local and global information from the image, one approach is to use two pathways: a local pathway that has more layers but a smaller receptive and a global pathway that has fewer layers but a larger receptive field [5].

3. Methods

3.1. Preprocessing

MRI scans suffer from bias field distortion due to variations in the field between different scanners, which causes voxel intensities to vary between images. Even within a sin-

gle image the voxel intensity can be distorted due to local and temporal effects. Additionally, the boundaries between tumor regions of the scan may be blurred.

To correct the bias, we apply the N4ITK method that predicts the bias field through B-spline fitting and uses this prediction to create uniform intensities [17]. This method is available with the Advanced Normalization Tools (ANTS) package. We applied this correction to all samples (both training and testing), before inputting them to the model.

3.2. Baseline

We started by building a simple CNN to label each voxel. The full architecture is show in Table 1 on page 3. Note that we use a CONV layer instead of a max pooling layer for layers 3 and 5. We apply batch normalization before each convolutional layer. For this baseline we use a ReLU activation layer.

We also convert the traditional fully-connected layers at the end to convolutional layers. Finally we use a global averaging layer as the last layer.

We apply the softmax function to the final scores to obtain estimates for the probability of labeling the center voxel of the input patch for each of the possible classes.

To train the network, we minimize cross-entropy loss between the prediction and the true label. We use the RMSProp algorithm to update the gradients. We also add the L2 loss of the weights to the loss. The input to this baseline model is 2D patches from the MRI scans, of 33 x 33 voxels. We selected the 2D patches at random from the input samples, in any of the 3 orthogonal planes. This amounted to a total of almost 18 million possible training samples. From these we randomly selected samples according to a uniform distribution among the possible voxel labels.

The original distribution of labels among the training voxels is show in Table 2 on page 3.

3.3. Triplanar Network

We first improved on the baseline model by using 3D input patches instead of 2D patches. To do so, a naive approach would be to use actual 3D cubes of voxels as patches. However, as was proposed by Lai [9], we used a tri-planar approach, which greatly reduces the memory and computations required to train and predict using the model, without sacrificing network accuracy.

Around each voxel we extracted patches in each of the 3 orthogonal planes (XY, XZ, YZ).

According to [9], this method is essentially as effective as training on an entire cubic volume around the center voxel, but it obviously allows a much smaller number of parameters, and a faster training time.

We made several assumptions about the 3 planar regions. First we assumed that the parameters of the model are independent of orientation–in other words, rotating an input tri-

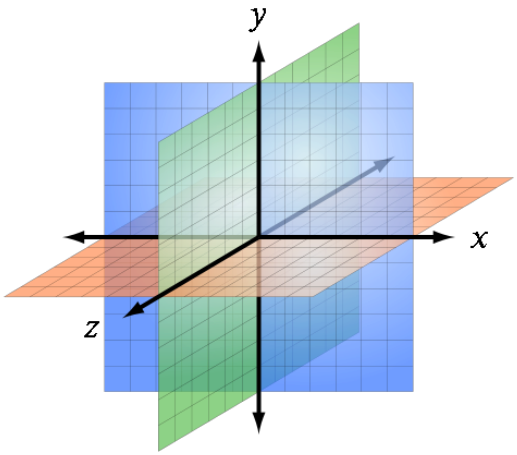

Figure 1. The model input was three orthogonal planar patches around the voxel whose label was predicted.

planar patch does not affect the predicted label of the center voxel. This seems corroborated by the anatomy of a tumor, which is not aligned along an particular axis a priori. This assumption significantly reduced the number of parameters, enabling larger minibatches of samples to fit in memory, and training / prediction to proceed much more quickly.

As part of this assumption, we also assumed that the intermediate predictions based on an individual plane of the input patch are weighted equally across each of the three planes. While the weighting of each plane could conceivably be a learned parameter, any deviation from equal weighting would indicate overfitting to the training data. This assumption likewise reduced the number of parameters required.

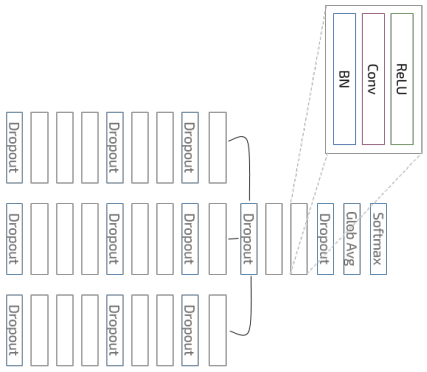

Figure 2. The layers of the tri-planar model.

The layers of the tri-planar model are shown in Table 3 on page 3. The first 6 layers are independently applied to each of the 2D planes, and the output are combined as input to the final layers. The activation for each layer was ReLU.

Table 1. Baseline CNN

	Type	Filter Size	Stride	# Filters	Input
Layer 1	Conv	3 x 3	1 x 1	64	33 x 33 x 4
Layer 2	Conv	3 x 3	1 x 1	64	33 x 33 x 64
Layer 3	Conv	2 x 2	2 x 2	64	33 x 33 x 64
Layer 4	Conv	3 x 3	1 x 1	128	16 x 16 x 64
Layer 5	Conv	2 x 2	2 x 2	128	16 x 16 x 128
Layer 6	Conv	1 x 1	1 x 1	256	8 x 8 x 128
Layer 7	Conv	1 x 1	1 x 1	4	8 x 8 x 256
Layer 8	Glob Avg				

Table 2. Distribution of Training Data Labels

Non-tumor	Necrotic and Non-enhancing Tumor	Peritumoral Edema	GD-enhancing Tumor
0.4148	0.2005	0.3231	0.0617

Table 3. Triplanar CNN

	Type	Filter Size	Stride	# Filters
	Dropout			
Layer 1	Conv	3 x 3	1 x 1	64
Layer 2	Conv	3 x 3	1 x 1	64
Layer 3	Conv	3 x 3	1 x 1	64
	Dropout			
Layer 4	Conv	2 x 2	1 x 1	64
Layer 5	Conv	3 x 3	1 x 1	128
	Dropout			
Layer 6	Conv	2 x 2	1 x 1	128
	Averaging			
	Dropout			
Layer 7	Conv	1 x 1	1 x 1	256
Layer 8	Conv	1 x 1	1 x 1	4
Layer 9	Glob Avg	8 x 8		

Spatial batch normalization was applied before each convolutional layer. We apply the softmax function to the final scores to obtain estimates for the probability of labeling the voxels.

To train the network, we minimize cross-entropy loss between the prediction and the true label. We use the RMSProp algorithm to update the gradients. We also add the L2 loss of the weights to the loss.

For the tri-planar model, a padding of 15 voxels was used around each voxel whose label was predicted. Therefore around the entire input sample we applied a 15 voxel SAME padding that extended the size of the original sample. Due to memory constraints, we were unable to predict labels for an entire image, so we split the input image into 320 equal sized patches, each of which contained a 15 voxel padding around the center volume whose voxel labels were predicted.

3.4. U-Net Model

Memory constraints in the triplanar network prohibited us from developing a deeper network. For this reason, we decided to experiment with a network that downsamples and then upsamples, which would reduce the number of parameters required. A similar network was described by Ronneberger et al. [16].

We also experimented with skip-layers, as proposed by He et al. [6]. In our model, we downsample twice (using max pool layers), but also pass the original uncompressed output to the corresponding upsampled layers. Our architecture is show in Figure 3 on page 5, and described in detail in Table 4 on page 5.

The input to the model are 2D planes, each of which contains a 20 voxel padding. The convolutional layers of the network do not add any padding, so the height and width dimensions of their output layers decrease by 2. We use max pooling 2 x 2 layers to downsample the data. When upsampling, we apply a 2 x 2 convolutional layer immediately after the upsample.

The skip layers (from layer 2 to layer 15, and from layer 5 to layer 11), are cropped to match the smaller size of the destination layer. For example, the output of layer 2, which is 64 x 64 x 64 is trimmed to 32 x 32 x 64, to be concatenated with the output from layer 12, which is also 32 x 32 x 64. A fully connected layer terminates the model, producing a 28 x 28 x 4 output.

As the activation for each convolutional layer we use ReLU. We apply the softmax function to the final scores to obtain estimates for the probability of labeling the voxels. To train the network, we minimize cross-entropy loss between the prediction and the true label. We use the RMSProp algorithm to update the gradients. We also add the L2 loss of the weights to the loss.

For all of our models, we modified code for training from the course assignment 2 [10].

3.5. Evaluation

For assessing the quality of prediction it is necessary to use more sophisticated metrics than just the accuracy since the imbalanced classes make it possible for a model to have high accuracy overall but still be inaccurate over the tumor voxels. We chose the Dice score [3] which measures the fraction of true positives with respect to the total (true positives, false negatives, and false positives).

4. Dataset

We have collected 285 labeled samples from [1], of both histological diagnosis: astrocytomas or oligoastrocytomas, LGG, and anaplastic astrocytomas, and glioblastoma multiforme tumors, HGG. There are 210 HGG samples and 75 LGG samples. The samples are manually labeled.

We found that by extracting patches for training, the number of available samples in our dataset far exceeded the amount that we could realistically train on for this project and therefore there was no need for any data augmentation steps.

Each sample has four MRI sequences : T1-weighted, T1 with gadolinium enhancing contrast, T2-weighted and FLAIR. The samples have already been aligned.

5. Experiments

We trained the baseline ("simple") CNN on 1600 patches and validated on 200 patches with the same distribution as the training set (150 HGG patients). The size of each patch was 32x32 in the axial orientation and the output was a single prediction of the class of the center voxel. The patches were not selected uniformly, but were chosen so that the class labels were nearly uniformly distributed. This means that the number of healthy tissue labels in the training set was much less than the number in the overall population. When sampling we discarded any samples that were less than 16 pixels from the border of the image since this would prevent using a 32x32 patch. Additionally, we discarded any samples that were located in air since this can be trivially classified as non-tumor since the image intensity at these locations is zero. During training, we used a learning rate of 0.05. a regularization weight of 1e-7, batch size of 300, and 100 epochs. We achieved near perfect training accuracy but less than 50% validation accuracy.

For the triplanar model, we found that the lack of any max pooling made the size very large. We had to make a compromise and reduce the size of the patch in order to perform meaningful training. We used a 90x90x90 input and a 2x2x2 patch. As with the simple model, we used 1600 patches for training and 200 patches for validation. In order to address the discrepancy between training and validation accuracy in the baseline model, we implemented dropout which helps prevent coadapation of the feautures and helps improve generalization error [7]. With a learning rate of 1e-2, a regulrization weight of 1e-5, a batch size of 1, and 40 epochs we were able to achieve approximately 70% validation accuracy.

When examining the segmentation maps of the original two models, we saw that there were a large number of false positives (ie. there were many healthy tissue voxels that were being labeled as one of the three tumor classes). This is because our sampling scheme ws biased towards tumor classes since the overwhelming majority of the labels are healthy tissue and thus we needed a biased sampling scheme. We found that weighting the loss function by the relative frequency of each of the classes helped account for this class imbalance. We also decided to use the Dice score to monitor training instead of accuracy since the raw accuracy metric will be artificially high because of the large number of healthy tissue voxels.

For the U-net model, we were able to increase the total number of layers because we used max pooling to decrease the height and width and thus memory requirements. We found that despite the larger number of layers it was possible to train this model without extensive tuning of hyperparameters. This may be due in part to the skip connections which have been shown to make training easier [6]. As before training was done on 1600 patches and validation on 200 patches. With a learning rate of 1e-2, a regularization weight of 1e-5, dropout of .15, batch size of 1, and 40 epochs, we achieved a Dice score of 0.88 on the training set and 0.81 on the validation set. By visualizing examining the segmented patches we confirmed that the predicted segmentation was quite similar to the ground truth.

6. Conclusion

Designing various architectures to solve the labelling problem was a rewarding challenge. We learned that the depth of the architecture was important, and even worth the tradeoff of losing some precision by downsampling the data.

We find that the U-net architecture has the best performance and this confirms the findings made in prior literature [16]. One desirable aspect of this architecture is that high resolution information is passed on to the upconvolution network in the form of the skip connections. Another desirable property of the U-net is that there are a large number of features after downsampling and these features can be used to reconstruct details in the upsampling step. This may be one reason why the U-net is better able to resolve boundaries and results in more contiguous segmentations than using a simpler architecture.

There are several ways that this project could be further improved. We think that an even deeper architecture, perhaps with entire layers that are stochastically dropped during training, would probably perform better. An architec-

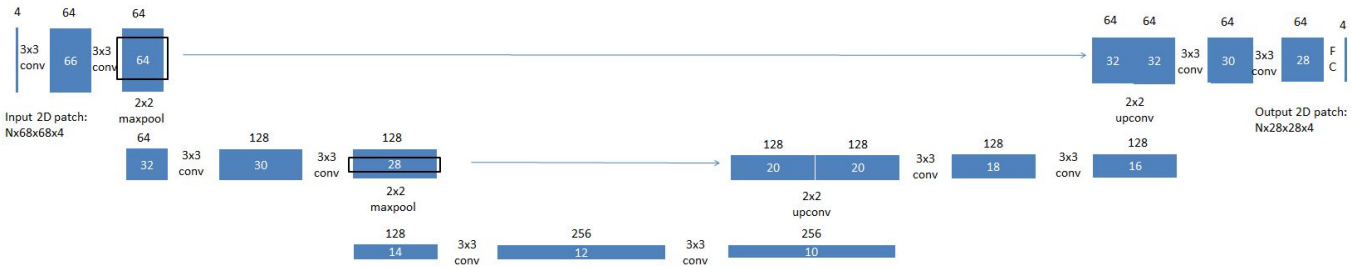

Figure 3. The layers of the U-net model. There filter size is 3x3 and the stride is 1 so the height and width decreases by 2 after each convolution step. After two convolutional steps a maxpool operation is applied to decrease the height and width by a factor of 2. At each level of the u-net the depth is doubled. The smallest height and width after the down convolutions is 10 and a depth of 256. Two stages of upconvolutions increase the height and width to 28. Tthe output of each upconvolution step is concatenated with a cropped version of the input to the corresponding maxpool.

Table 4. U-Net CNN

	Type	Filter Size	Stride	# Filters	Input
Layer 1	Conv	3 x 3	1 x 1	64	68 x 68 x 4
Layer 2	Conv	3 x 3	1 x 1	64	66 x 66 x 64
Layer 3	Max Pool	2 x 2	2 x 2		64 x 64 x 64
Layer 4	Conv	3 x 3	1 x 1	128	32 x 32 x 64
Layer 5	Conv	3 x 3	1 x 1	128	30 x 30 x 128
Layer 6	Max Pool	2 x 2	2 x 2		28 x 28 x 128
Layer 7	Conv	3 x 3	1 x 1	256	14 x 14 x 128
Layer 8	Conv	3 x 3	1 x 1	256	12 x 12 x 256
Layer 9	Upsample	2 x 2	1 x 1		10 x 10 x 256
	Concatenate cropped Layer 5 out				
Layer 10	Conv	3 x 3	1 x 1	128	20 x 20 x 256
Layer 11	Conv	3 x 3	1 x 1	128	18 x 18 x 128
Layer 12	Upsample	2 x 2	1 x 1		16 x 16 x 128
	Concatenate cropped Layer 2 out				
Layer 13	Conv	3 x 3	1 x 1	64	32 x 32 x 128
Layer 14	Conv	3 x 3	1 x 1	64	30 x 30 x 64
Layer 15	FC			4	28 x 28 x 64

ture that included an even wider input size would probably also be slightly more performant. Perhaps a very large input size could be first downsampled, and then concatenated with the original input data, in order to include a larger area as the input to the prediction.

A final cleanup step could also be added (e.g., to disallow segmentations whose size is smaller than some threshold).

Finally, we would need to implement a method for translating the patches to cover the entire brain volume in order achieve a whole brain segmentation.

This work could be easily extended to similarly label other input types, e.g., cell types, or other organic masses [3].

In terms of impact, the availability of an automated segmentation could eliminate a very time-consuming task for radiologists. A computer is able to analyze multiple modalities, as was done in this project, much more easily than a human since it is only possible to view a single image at a time.

6.1. Acknowledgement

We wish to thank all of teaching staff of cs231n, and especially Agrim Gupta, for their suggestions, availability, and support.

References

[1] Multimodal Brain Tumor Segmentation Challenge 2017. `https://www.med.upenn.edu/sbia/brats2017/data.html`, 2017. [Online; accessed 8-May-2017].

[2] D Ciresan. Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in neural information processing systems.*, 2013.

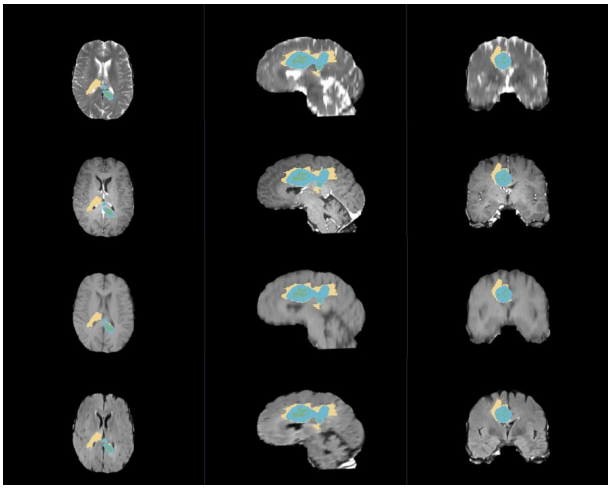

Figure 4. Example of dataset for one subject. Orientations: 1st column: axial slice, 2nd column: sagittal slice, 3rd column: coronal slice. Modalities: 1st row: T2, 2nd row: T1 with contrast, 3rd row: T1, 4th row: FLAIR. Colors: yellow: whole tumor, blue: enhancing tumor, green: necrosis

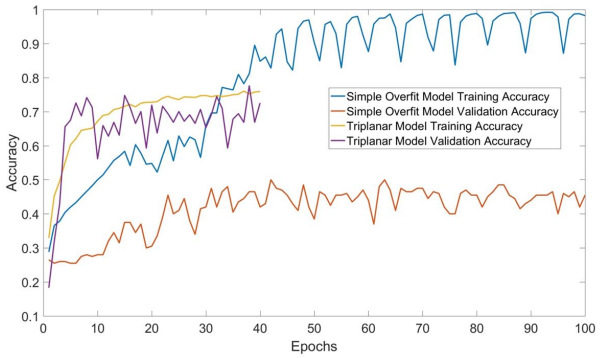

Figure 5. Training and validation accuracies for the baseline (simple) and triplanar models. The baseline model has significant overfitting but with the triplanar model the training and validation accuracies are comparable.

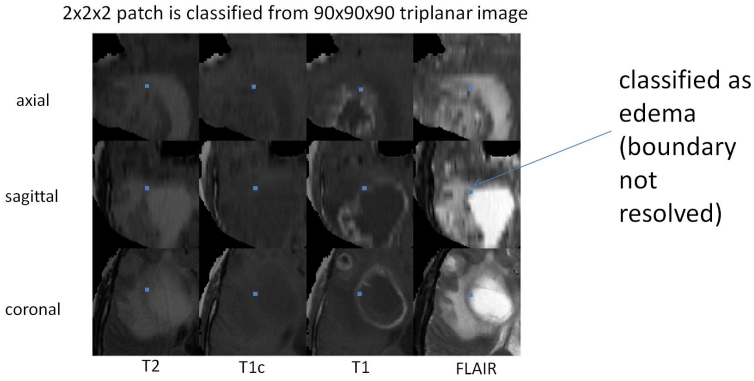

Figure 6. Using the triplanar model, the maximum patch size for a 90x90x90 input without exceeding the available GPU memory was 2x2x2, which is clearly very small. Even though the patch is on the boundary of edema and enhancing tumor the entire patch is labeled as edema showing that this architecture does not do well at resolving boundaries

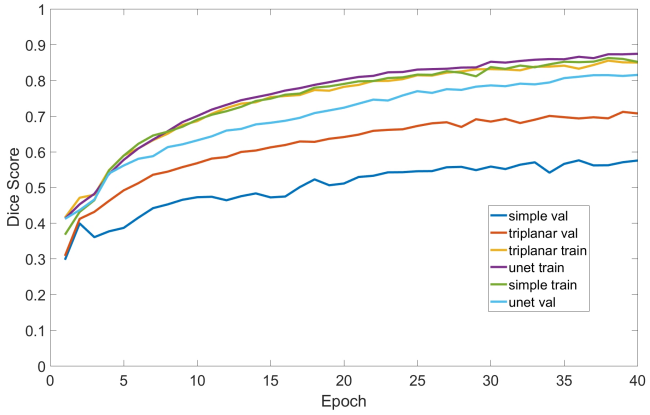

Figure 7. Values of the dice coefficient on both the validation and training sets after each epoch of training. The U-net results in the best validation Dice coefficient.

[3] Lee Dice. Measures of the amount of ecologic association between species. *Ecology*, pages 297–302, 1945.

[4] Pavel Dvorak and Bjoern Menze. Structured prediction with convolutional neural networks for multimodal brain tumor segmentation. *Proceeding of the Multimodal Brain Tumor Image Segmentation Challenge*, pages 13–24, 2015.

[5] M Havaei. Brain tumor segmentation with deep neural networks. *Medical Image Analysis.*, 2016.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[7] GE Hinton. mproving neural networks by preventing co-adaptation of feature detectors. *arXiv*, 2012.

[8] Dongjin Kwon. Combining generative models for multifocal glioma segmentation and registration. *MICCA*, pages 763–770, 2014.

[9] Matthew Lai. Deep learning for medical image segmentation. *arXiv preprint arXiv:1505.02000*, 2015.

[10] Fei Fei Li, Justin Johnson, and Serena Yeung. cs231n Assignment 2, Spring 2017. `http://cs231n.github.io/assignments2017/assignment2/`, 2017. [Online; accessed 20-May-2017].

[11] R Meier. A hybrid model for multimodal brain tumor segmentation. *Multimodal Brain Tumor Segmentation.*, 2013.

[12] 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

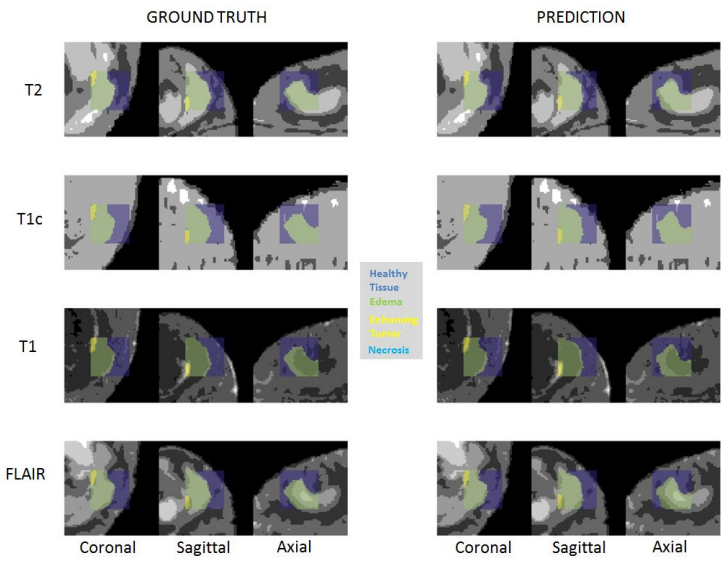

Figure 8. An example of a segmented patch overlayed on the input data (one of four contrasts in one of three views) using the U-net model. There is very good match between the ground truth and the prediction. The four tissue types (healthy tissue, edema, enhancing tumor, and necrosis) are represented by separate colors.

benchmark (brats). *IEEE transactions on medical imaging*, 34(10):1993–2024, 2015.

[13] Sérgio Pereira, Adriano Pinto, Victor Alves, and Carlos A Silva. Brain tumor segmentation using convolutional neural networks in mri images. *IEEE transactions on medical imaging*, 35(5):1240–1251, 2016.

[14] V Rao. Brain tumor segmentation with deep learning. *MICCAI BraTS (Brain Tumor Segmentation) Challenge.*, 2014.

[15] S Reza. Multi-class abnormal brain tissue segmentation using texture features. *proc of BRATS Challenge - MICCAI*, 2013.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[17] Ghazaleh Tabatabai, Roger Stupp, Martin J van den Bent, Monika E Hegi, Jörg C Tonn, Wolfgang Wick, and Michael Weller. Molecular diagnostics of gliomas: the clinical perspective. *Acta neuropathologica*, 120(5):585–592, 2010.

[18] Darko Zikic, Yani Ioannou, Matthew Brown, and Antonio Criminisi. Segmentation of brain tumor tissues with convolutional neural networks. *Proceedings MICCAI-BRATS*, pages 36–39, 2014.