Detecting Brain Tumors in Low Quality Magnetic Resonance Images

Computer Vision – Healthcare
Venkat Yerramsetti (venkaty) - https://github.com/vypr213/c230_final

Abstract

This project explores how the 2D U-Net, 3D U-Net and V-Net models perform at the brain tumor segmentation task, when the input image quality is degraded. These models were implemented from scratch and trained on a combination of high and low quality images, respectively, taken from the 2018 BraTS dataset [1] [2] and a low quality dataset generated from the original BraTS dataset. Results were inconclusive due to bad ground truth labels for the low quality dataset.

Introduction:

Neuroimaging has been extensively used for such brain tumor detection and also to evaluate the chosen treatment and the progression of the disease [1]. Since manual annotation of neuroimages is a very tedious and error prone task, deep convolutional neural networks have been designed to automatically and quickly detect tumors in MRIs using segmentation techniques [3] [4] [5]. However, one problem with using neuroimages for automatic tumor detection is that these images could be suffering from quality degradation due to device calibration errors, subject movement, etc. In this project, I explored whether the existing models, that produced very good segmentation results, perform just as well when the input image quality is degraded. I implemented 2D and 3D U-Net and V-Net models, and trained them on a mix of high quality and low quality images obtained from the BraTS 2018 dataset and a dataset of degraded images obtained by applying random deformations to the images in the BraTS 2018 dataset. Initial results show poor performance for all the models. Doing error analysis, showed that the labels for about 50% of the generated, low quality images were faulty, which resulted in underfitting. Due to lack of time, the models could not be retrained with corrected ground truth labels, so the models' performance on low quality images could not be conclusively determined.

Related Work:

Deep convolutional networks have dominated visual recognition tasks in the past few years. Consequently, several convolutional models were developed for the task of general image segmentation [5] [6] et. al. which used fully convolutional networks and were usually trained on relatively large datasets. However, for biomedical image segmentation, datasets tend to be very small, so newer convolutional networks were developed to cope with such small datasets. In [7], Ciresan et. al proposed a convolutional network using the sliding-window technique to segment neuronal structures. This network has the advantage of increasing the training data with multiple patches, but it is very inefficient due to repeated calculations on overlapping areas. By using a fully convolutional network and skip connections, the U-Net model [8] demonstrated superior segmentation results. Cicek et. al [9] extended the 2D U-Net model to 3D volumetric data which constitutes majority of biomedical data. Furthermore, in [10], Milletari et. al. developed a similar volumetric network with additional skip connections within each layer for faster convergence. While all of these models produced very good results, they were trained with high quality image data. In reality, biomedical images might not be always available in high quality. This project explores the question of whether these models could produce equally superior results even with the low quality images.

Data:
I used the BraTS 2018 dataset for training the models. This dataset contains MRI scans for 285 patients. For each patient, four MRI scan are provided, which correspond to four modalities, namely 1) native (T1), 2) post-contrast T1-weighted (T1Gd), 3) T2-weighted, and 4) T2 Fluid Attenuated Inversion Recovery (FLAIR). Besides these images, a single image containing segmentation labels is also provided. The labels are 4 for GD-enhancing tumor, 2 for peritumoral edema, and 1 necrotic and non-enhancing tumor core, and were generated by one to four annotators, and were approved by experienced neuro-radiologists. The different labels correspond to different sections of the tumor, and all of the labels together segment the whole tumor region. Different combinations of the four modalities, stated above, are used to predict the different labels. For simplicity, this project focused on predicting only the whole tumor.

Data Preprocessing:
In order to simulate the low quality MRI scans obtained in the field, a new dataset was generated by applying small, random deformations to each MRI scan in the original BraTS dataset. The same deformations were also applied to the corresponding ground truth labels.

Figure 1 Example slice from the original image (left) and the corresponding deformed image (right). Tumor labels are show in blue. Notice that the deformation is also applied to the label for the deformed image

Then, the data was randomly separated into train/dev/test sets containing 200/55/30 examples respectively. Each of these sets contains equal number of images from the original BraTS dataset and the generated, low quality dataset. In order to use the same train, dev, and test sets for the entire duration of the project, the IDs (directory path) of the examples () were stored in text files.

Each MRI scan for a given example is a (240, 240, 155) volume. In order to work well with the 3D models, the input was padded to get the shape of (240, 240, 160). As stated above, the dataset, consists of four MRI scans for each patient. During preliminary training, all four MRI scans were combined into a single image with four channels to produce an input of shape (4, 240, 240, 160) using the "channels_first" convention. However, whole tumor can be identified using just the FLAIR MRI scan, so speedup training and reduce memory usage, only the FLAIR scan was used in later training rounds. This had brought down the input shape to (1, 240, 240, 160).

Because of the huge size of a single example, a data generator was implemented to feed one example at a time into the model. When feeding the input to the 2D U-Net model, the data generator splits each example into 155 2D images of shape (?, 240, 240), where ? corresponds to the number of channels. Notice that the image doesn't need to be padded for the 2D model. Finally, before feeding the input into a model, the data generator normalizes it to zero mean and a standard deviation of one.

Models:
I implemented three models namely 2D U-Net [8], 3D U-Net [9], and V-Net [10]. All models are based on the original architectures, but with a few changes to adapt them to this project's data and to satisfy computing resource constraints. All of the models have a "contracting path", where the input features are gradually down sampled, followed by an "expansive path", where the features are gradually up sampled to the original input size.

For the U-Net models, the "contracting path" consists of four down convolutional blocks, where each block contains two 3x3 convolutions with ReLU activations, followed by a 2x2 max pooling layer with a stride of 2. The "expansive path" consists of 4 up convolutional blocks where each block consists of a 2x2 up-sampling convolution, followed by a concatenation with the activations of the corresponding layer in the contracting path, which is, then, followed by two 3x3 convolutions with ReLU activations. The down-convolution segment and the up-convolution segment are joined by a sequence of two 3x3 convolutions with ReLU activations. Finally, the output layer consists of a 1x1 convolutional layer, with sigmoid activation. Both 2D and 3D U-Net models follow this same structure except that with the difference that the 2D model uses 2D convolutional operations whereas the 3D model uses corresponding 3D convolutional operations.

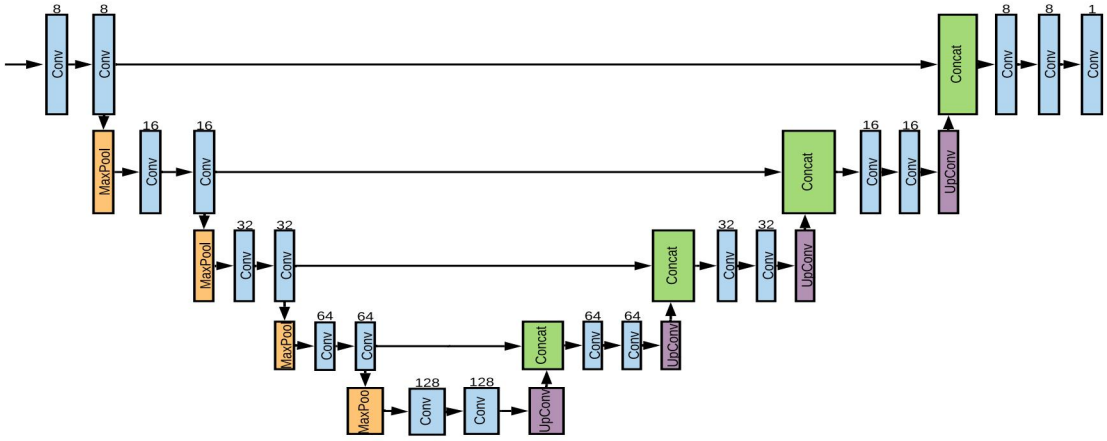

Figure 2 Structure of U-Net models. Each "Conv" layer (blue) uses a kernel of size 3 with the number of kernels shown above each block. Each MaxPool (orange) halves the size of input to the layer. Each UpConv (purple) doubles the size of input to the layer

For the V-Net model, the contracting path consists of four down convolutional blocks, where each block consists of one or two or three 5x5x5 convolutions with PReLU activations. Each down convolutional block also implements a skip connection using elementwise sum of the block's input and the output of the last convolution of the block. Unlike U-Net, the V-Net model down samples the input by using a 2x2x2 convolution with stride 2, which provides an opportunity to double the number of feature channels and could potentially reduce memory usage by avoiding the switches needed for MaxPool layers. The "expansive path" consists of four up convolutional blocks, where each block starts off with concatenating (except the first block) its input with the activations of the corresponding layer in the contracting path followed by one or two or three 5x5x5 convolutions with PReLU activations. Similar to the contracting path, each block implements an elementwise sum based skip connection between its input and the output of the last convolution of the block. The final layer is a 1x1x1 convolution with sigmoid activation.

Loss Functions:
During preliminary training, pixelwise cross-entropy loss was used as a baseline loss. However, the primary loss function throughout the entire training was the dice loss proposed in [10].

$$Dice\ Loss = -\frac{2 * \sum(\hat{y} * y)}{\sum\hat{y} + \sum y}$$

I also experimented with a new compound loss function incorporating the feature loss based on a pretrained VGG19 network as proposed in [11]. Here, the feature loss is calculated between the predicted segmentation labels and the ground truth labels.

$$Compund\ Loss = \text{vgg}_{\text{loss}} + dice_{loss}$$

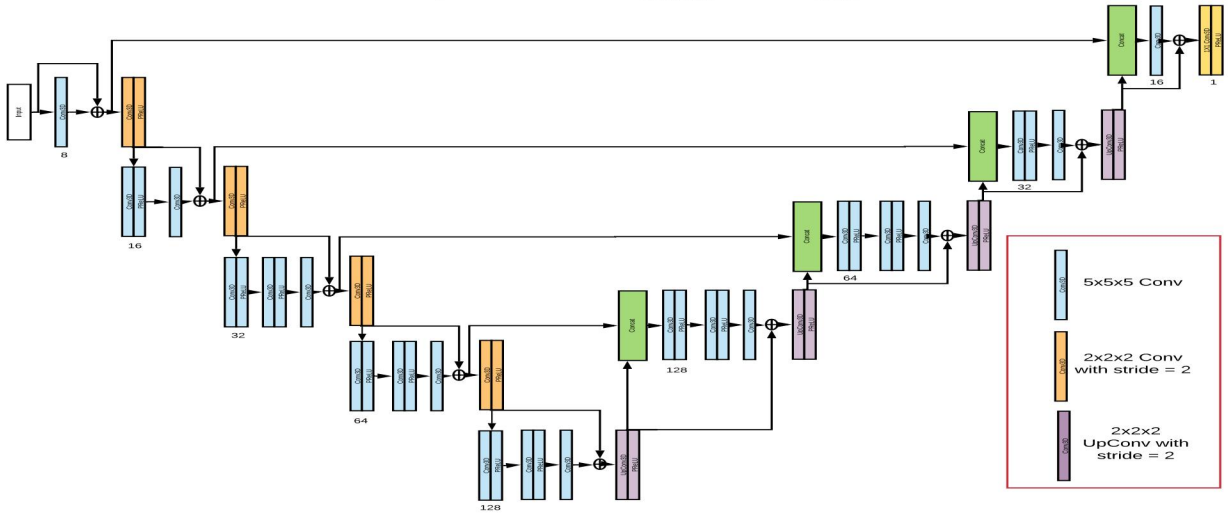

Figure 3 Structure of V-Net model. Number of kernels for each level are shown at the bottom of the "Conv" layers (blue)

Training:

All models were initially trained for 50 epochs with a batch size of 1 for 3D models and 155 for 2D models. This batch size was chosen to avoid memory exhaustion. Using a learning rate of $1e^{-3}$ caused the models to diverge after a few epochs. Upon tuning, $1e^{-4}$ seemed to work well for faster convergence and avoiding plateaus. For the optimizer, I initially used the stochastic gradient descent with momentum, and then switched to Adam, which caused a huge speed up in the training.

Results and Analysis:

During initial training, I used the "accuracy" metric, which in retrospect, is ill suited for the dataset at hand because the dataset is highly imbalanced. Majority of the ground truth label data is comprised of background, which is annotated as 0s. Because of this model was able to achieve very high accuracy by predicting the background labels most of the time. Subsequently, the metric was switched to dice coefficient.

Model	Train (N = 200)	Dev (N = 55)	Test (N = 30)
Unet2D	0.7499	0.6438	0.6860
Unet3D	0.7683	0.6322	0.6882
Vnet3D	0.8515	0.6225	0.6492
Unet2D (w/ vgg)	0.6885	0.6345	0.6785

Figure 4 Dice scores after training the models for 150 epochs

Training on the original BraTS dataset, the models achieved about 85% dice score. However, when the generated, low quality images were added to the dataset, the models plateaued at about 73% dice score, and couldn't improve even on the train set loss. The was caused due to data loader erroneously producing different training examples for each epoch, thus causing an input covariate shift. Fixing this problem, the models suffered from high bias and high variance problem. Using L2 regularization with a lambda value of 0.001 resolved the high variance problem, but the bias problem persisted as we can see below.

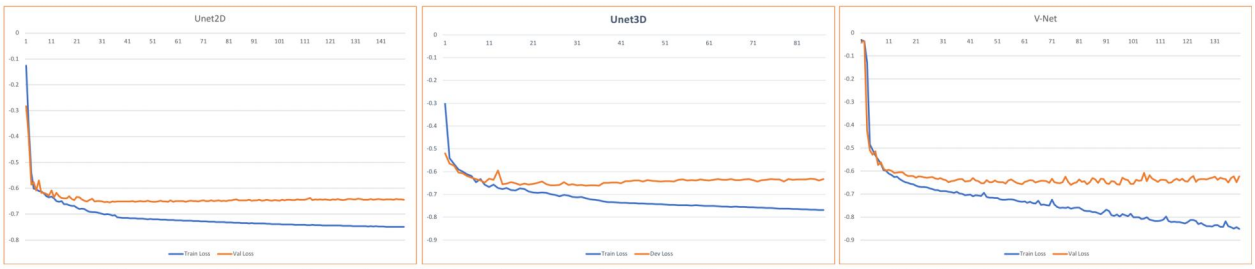

Figure 5 Train (blue) and Dev (orange) losses for 2D U-Net, 3D U-Net, and V-Net

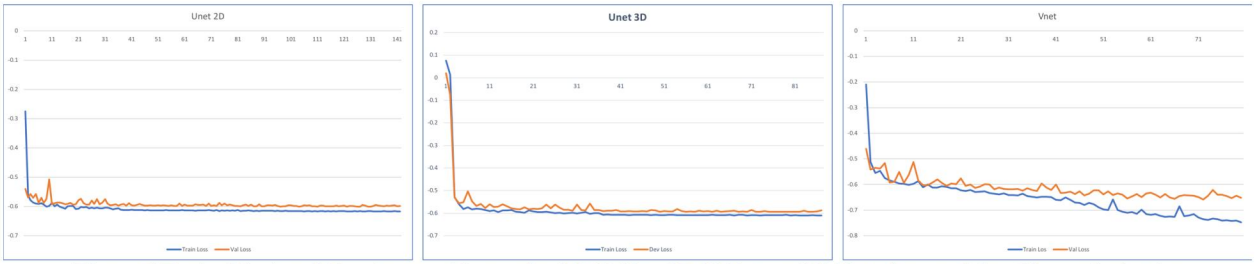

Figure 6 Train (blue) and Dev (orange) losses for 2D U-Net, 3D U-Net, and V-Net after applying L2 regularization

Performing error analysis and visualizing several examples revealed that the ground truth labels for many of the generated low quality images were missing or were shifted. I believe, this was caused due to a faulty random translation applied to some images. Moreover, visualizing test set examples show that the ground truth labels for majority of them were not affected by the random translation. I believe, this was the reason for the dice scores that were better than the train and dev sets in Figure 4. While I was able to fix the generated dataset with the correct labels, I couldn't retrain all of the models on the corrected dataset due to lack of time. I did train the 3D U-Net for about 10 epochs, and the train and dev set dice scores appeared to be improving to over 75%. However, more training and analysis is required to make a definitive conclusion.

Conclusion:

Having a non-faulty data pipeline and accurate data and ground truth labels is very crucial for successfully evaluating models. In this project, due to the faulty data, the obtained results were inconclusive about the performance of the models on low quality MRI. However, during a few epochs of training, the 3D U-Net model appeared to be improving suggesting a positive result. Given more time, I would validate the generated dataset more carefully and retrain the models on the corrected dataset. Eventually, in order to make a concreate conclusion, I would also train/evaluate the model's on a real dataset of low quality images instead of the simulated dataset.

Acknowledgement:
I would like to thank Dr. Olivier Keunen and Dr. Wintermark's lab for providing access to computing resources for training my models.

Bibliography

[1] J. A. B. S. K.-C. J. F. K. K. J. B. Y. P. N. S. J. W. R. L. L. G. E. W. M. A. T. A. B. A. N. B. P. C. D. C. N. C. J. C. A. D. T. D. H. D. Menze BH, The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)..

[2] H. A. A. S. M. B. M. R. J. S. K. J. B. F. K. F. C. D. Spyridon Bakas, Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features.

[3] W. L. S. O. T. V. Guotai Wang, Automatic Brain Tumor Segmentation using Cascaded Anisotropic Convolutional Neural Networks.

[4] 2018 International MICCAI BraTS Challenge Proceedings, 2018.

[5] E. S. T. D. Jonathan Lon, Fully Convolutional Networks for Semantic Segmentation.

[6] S. H. B. H. Hyeonwoo Noh, Learning Deconvolution Network for Semantic Segmentation.

[7] A. G. L. M. G. J¨. S. Dan C. Ciresan, Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images.

[8] P. F. T. B. Olaf Ronneberger, U-Net: Convolutional Networks for Biomedical Image Segmentation.

[9] A. A. S. S. L. T. B. O. R. Özgün Çiçek, 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation.

[10] N. N. S.-A. A. Fausto Milletari, V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation.

[11] L. T. F. H. J. C. A. C. A. A. A. A. A. T. J. T. Z. W. W. S. Christian Ledig, Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.

[12] B. B. Dwarikanath Mahapatra, Progressive Generative Adversarial Networks for Medical Image Super resolution.

[13] H. D. Z. L. G. S. Bingzhe Wu, SRPGAN: Perceptual Generative Adversarial Network for Single Image Super Resolution.

[14] G. S. N. U. Muhammad Haris, Task-Driven Super Resolution: Object Detection in Low-resolution Images.

[15] V. V. S. I. J. S. Z. W. Christian Szegedy, Rethinking the Inception Architecture for Computer Vision.