CS230

Post-Stroke Lesion Segmentation using Cascaded Convolutional Neural Networks

David Zhou
Department of Computer Science
Stanford University
davidiot@stanford.edu

Abstract

Lesion segmentation is a time-consuming and labor-intensive process, and there currently exists no accurate algorithm for automating this task. In this work, we present a cascaded convolutional neural network trained using the ATLAS dataset that represents the first steps toward the construction of such an algorithm. The cascade is designed to split the task into a pipeline of simpler tasks, where each of the individual components can be trained in parallel. The model is able to attain a Sørensen-Dice coefficient of at least 0.40 on most images for the second step.

1 Introduction

Stroke is one of the leading causes of death and disability worldwide, and large scale neuroimaging studies have shown great promise in identifying markers for stroke recovery and improving the lives of stroke victims. One of the biggest barriers for these studies is the difficulty of obtaining accurate segmentation data (identifying the lesions that form in the brain after stroke), which is a time-consuming and labor-intensive process. Developing an accurate automated segmentation algorithm would reduce the need for manual segmentation. If an algorithm is produced, it would facilitate the construction of larger neuroimaging datasets and assist radiologists in the reading room.

In this work, we outline the first steps toward creating such an algorithm. Formally, the input to this algorithm is an image slice of the brain, and the output is a lesion mask, which indicates the pixels where the lesion is located. We use a *cascaded neural network*, or two pipelined neural networks operating in series, where the output of the first neural network is the input to the second. The first neural network takes the input slice and is responsible for outputting a bounding box around the lesion, while the second network takes the cropped bounding box as input and is responsible for outputting the final lesion mask. The performance of the algorithm is measured by the Sørensen-Dice coefficient, which is commonly used for segmentation tasks and is given by

$$\text{DICE} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{1}$$

where TP, FP, and FN are respectively the true positive, false positive, and false negative pixel counts of the lesion mask predicted by the algorithm compared to the ground-truth lesion mask.

2 Related work

To train our model, we used the ATLAS dataset [Liew et al., 2017], which was released in February 2018 of this year. To our knowledge, there are no published research papers that have established a benchmark on this task, and the gold standard for accuracy is still manual segmentation by hand.

However, a similar task is the segmentation of brain tumors using the BraTS 2017 dataset [Menze et al., 2015, Bakas et al., 2017]. For this dataset, cascaded convolutional neural networks have been trained that achieve Dice scores of more than 0.70 [Wang et al., 2017, Chen et al., 2018]. The cascaded approach is especially effective for the BraTS dataset because it is a multi-class segmentation problem with three classes, which becomes three binary segmentation problems after being decomposed into three pipelined networks. Although lesion segmentation is already a binary problem to begin with, we think that the cascaded approach is still be useful for our task, because it decomposes the network into simpler networks that can be trained in parallel. These approaches use volumetric input data, which is likely to result in a better model because only using two dimensional slices as input cannot model the relationships between adjacent image slices (we defer the use of volumetric input data in ATLAS for future work).

Another network architecture that has been very effective at biomedical imaging segmentation is the U-Net architecture, which has been shown to work very well with relatively few training images with the help of data augmentation for learning invariances [Ronneberger et al., 2015]. We will use the U-Net architecture for the second step in our cascaded neural network, to be described later.

3 Dataset and Features

ATLAS contains 229 brain scans and lesion masks from 220 patients. The scans are volumetric, and presented as two dimensional 232 by 196 pixel input slices, as shown in Figure 1.

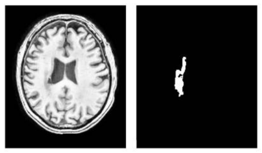
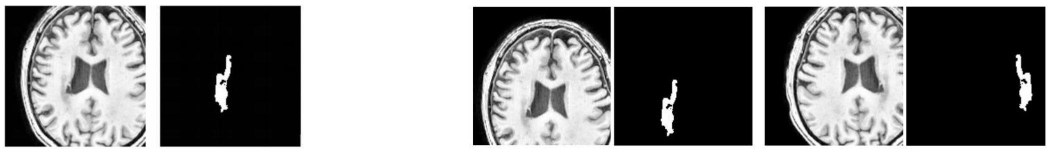


Figure 1: A brain image slice and its corresponding lesion mask

There are 43281 of these slices in ATLAS. Each slice can have one or more masks, depending on the number of lesions that appear in the slice, and there are 11678 masks for all of the slices. We consider each slice a data point for these experiments, and randomly split the data into training and dev sets of size 38953 and 4328 respectively (approximately 9:1). [1].

The slice is the input to the cascaded network as a whole, and the mask is the output. However, we also needed to generate data for the bounding boxes, which would be used as the training labels for the first neural network and the training input data for the second neural network. As we will describe later, there are four possible allowed bounding box sizes. A program was written to process the data and determine the smallest box size that would encapsulate the entire mask. The "ground-truth" bounding boxes have the mask in the center. They are used to train the first network and are shown in Figure 2.



(a) (b)

Figure 2: Data preprocessing and augmentation. a) The "ground-truth" bounding boxes. b) Random crops (left) and flips (right) used to augment the input examples to the second neural network.

In addition to the "ground-truth" bounding boxes that were generated for each of the 11678 masks, random crops and flips were used to augment the bounding box input dataset for training the second

[1]It would better ensure that slices from the same scan were placed into the same set. However, this also would have made the problem harder (doing it this way means that adjacent slices, which are likely to be similar, can appear separately in the training and dev sets), and this work only represents a first attempt at this task

neural network so that it would learn translational and left-right flip invariance. Because most random crops would not contain any lesions, the data processing was done to ensure that a substantial number of crops contained an entire lesion mask by randomly sampling the boxes within the region of the mask. Ultimately, the bounding box data is summarized in Table 1

Table 1: Bounding Box Dataset

Size	Dimensions	"Ground-Truth" Boxes	Training Boxes	Positive Training Boxes
S	12 x 12	3416	501554	159275
M	32 x 32	3690	681605	337991
L	64 x 64	2644	613743	275410
XL	160 x 160	1928	486551	152895

Training Boxes refers to the sum of the number of "ground-truth" and augmented boxes, while *Positive Training Boxes* refers to the subset of *Training Boxes* that are guaranteed to contain the entire lesion mask. Boxes are split between the training and dev sets based on the slices they correspond to.

4 Methods

The neural networks were constructed using TensorFlow [Abadi et al., 2015], and built on starter code that was supplied by David Eng. The starter code provided included a neural baseline that achieved a Dice score of 0.25 on the dataset when randomly split by slice.

In order to make the neural network cascaded, we originally planned to have two neural networks in series. However, upon processing the dataset, we found that there was a wide range of lesion sizes, from only a couple of pixels wide to occupying half the brain. Because the filters used in the UNet architecture place constraints on the input image sizes, and we did not want to use a large bounding box size for even the small lesions, we decided to instead train four separate U-Nets, which were each responsible for a different size of lesion. The overall network architecture is shown in Figure 3

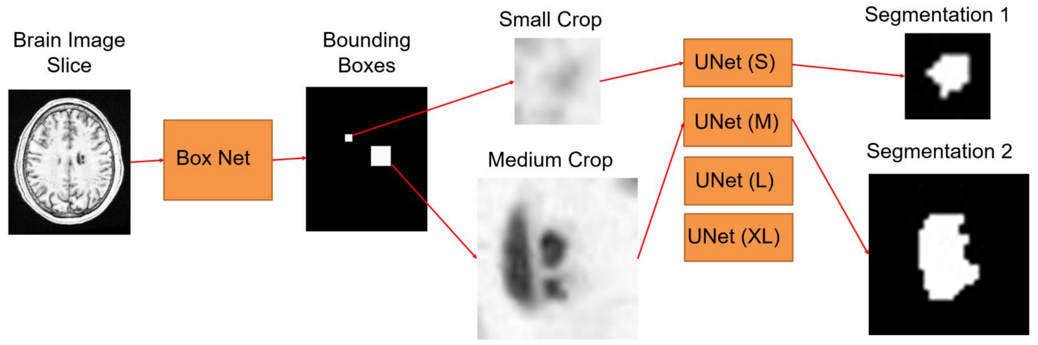


Figure 3: The overall network architecture.

As described earlier, the first neural network (BoxNet) is responsible for determining if there is a lesion, how large it is, and its approximate location. Its architecture is similar to the convolutional neural networks we described in lecture: it has two convolutional layers, two fully connected layers, and an output layer that encodes the positions and probabilities associated with the bounding boxes.

The neural networks responsible for performing the segmentations used U-Net architectures, which generally look like the network presented in in Figure 4.

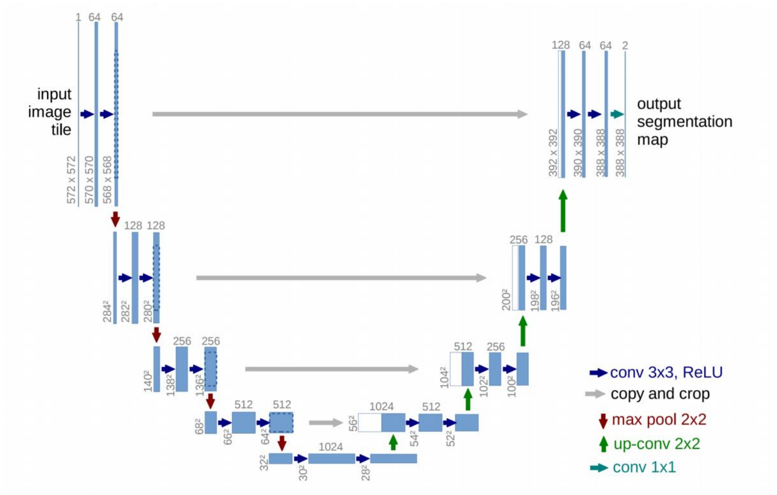


Figure 4: U-Net architecture, reproduced from Ronneberger et al. [2015]

These left side of this network is similar to a typical convolutional neural network, with convolutional layers and pooling layers. However, instead of adding fully connected layers, the UNet supplements the contracting network with successive layers where pooling operations are replaced by upsampling to increase the resolution of the output. The layers form a "U", where each layer on the same vertical level has the twice the number of filters and half the image dimensions as the layer above it. The output of the UNet has the same dimensions as the input, so it is ideal for segmentation. Because of constraints with GPU and storage memory, the UNets were not constructed with enough layers to make the image dimensions go all the way down to 1x1 in the last layer. Instead, the depth of the UNet (how many levels) was varied so that we could select the best one for each lesion size. In addition, the number of filters used in the convolutional layers was varied as well. We found that a UNet of depth 2 worked best for the small lesions, and a UNet of depth 3 worked best for the others. The total number of trainable parameters for each of the UNets and the BoxNet is summarized in table 2.

Table 2: Number of Variables by Network

Network	BoxNet	UNet-S	UNet-M	UNet-L	UNet-XL
Number of Variables	4683312	402625	1861697	1861697	1861697

The loss function used was was TensorFlow's built in weighted cross entropy loss with logits, which is given by

$$\text{Loss} = \text{targets} \times -\log(\sigma(\text{logits})) \times \text{weights} + (1 - \text{targets}) \times -\log(1 - \sigma(\text{logits})) \qquad (2)$$

Here, the logits are the inputs to the final sigmoid layers of the neural networks, and targets are the target masks (both are matrices in this equation). We used a value of 100 for the weights. For BoxNet, the bounding box is considered the target mask.

5 Experiments, Results, and Discussion

Training was performed using the Adam optimization algorithm with gradient clipping by the norm and run with a default learning rate of 0.001. Learning rate decay was applied by manually lowering the learning rate when the loss started to stall. The minibatch size was 100 for all models, which was selected because it was a compromise between the training speed and GPU memory limitations.

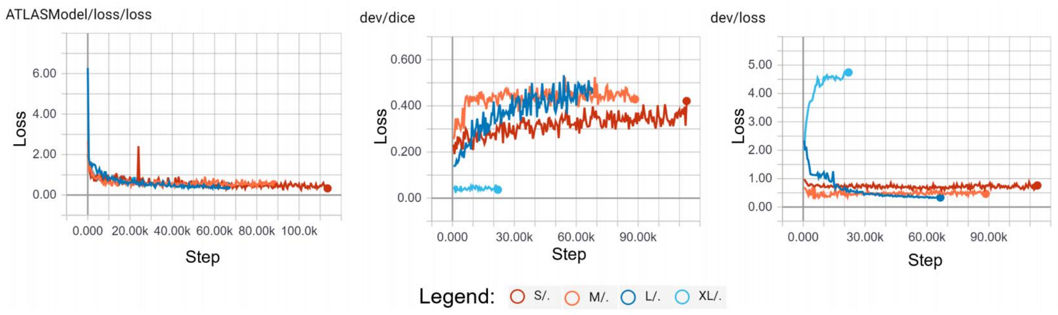


Figure 5: The loss on the training set (left), and the dev set Dice score and loss (right).

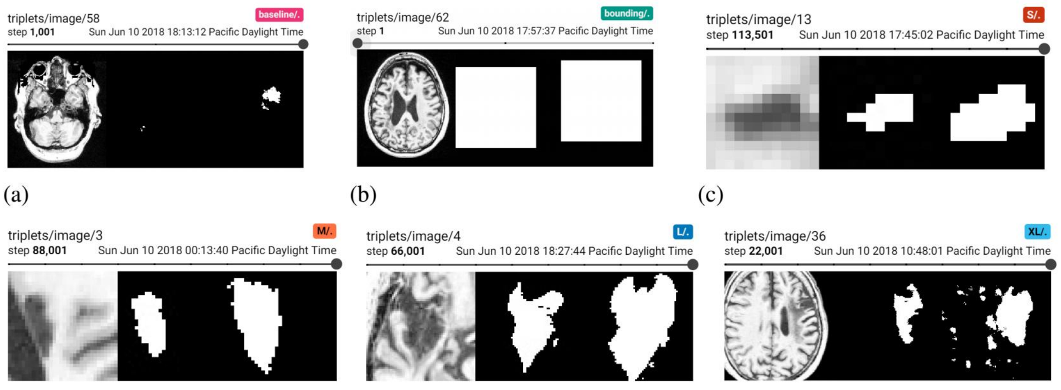


Figure 6: A few samples from Tensorboard containing input cropped images (left), target lesion masks (center), and predictions (right) for the baseline model (a), BoxNet (b), UNet-S (c), UNet-M (d), UNet-L (e), and UNet-XL (f)

The medium, and large UNets were all able to achieve Dice scores of more than 0.5, while the small UNet was able to achieve a dice score of 0.4. They were able to achieve some good segmentations, as shown in Figure 6b, 6c, and 6d. Unfortunately, the extra large UNet, which also took the longest to train, tended to overfit the training set for all of the architectures that we attempted to use, and never attained a Dice score of more than 0.1. This can be clearly seen in Figure 5, where the development set error increases with the number of iterations, and Figure 6f, where the predicted mask shows signs of overfitting. Unfortunately, we were also unable to optimize BoxNet, and its Dice score on the Dev set was too low for Tensorflow to calculate. The few images it was able to predict bounding boxes for were of the largest size, as shown in Figure 6b, because these were the easiest to determine.

6 Conclusion and Future Work

For this project, we were able to separate the segmentation process into two discrete steps, and train both of them in parallel. Three of the five networks we trained produced modestly good results, while the other two could be improved with more architecture and hyperparameter tuning, which we will defer to later work. Interestingly, the designs used in our approach favor the smaller lesions, while the baseline end-to-end network seems to perform better on the larger lesions. As shown in Figure 6a, the baseline struggled with smaller lesions, which makes sense, since the larger lesions have a far greater effect on end-to-end loss. In future work, we would also like trying to use volumetric data, since it seemed to work well for other segmentation problems.

7 Contributions

This is a one person team, so all work presented here can be attributed to me. All given starter code was developed by TA David Eng. Special thanks to him for setting all of this up!

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, J. B. Freymann, K. Farahani, and C. Davatzikos. Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. *Sci Data*, 4:170117, 09 2017.

L. Chen, Y. Wu, A. M. DSouza, A. Z. Abidin, A. Wismuller, and C. Xu. MRI Tumor Segmentation with Densely Connected 3D CNN. *ArXiv e-prints*, Jan. 2018.

S.-L. Liew, J. M. Anglin, N. W. Banks, M. Sondag, K. L. Ito, H. Kim, J. Chan, J. Ito, C. Jung, S. Lefebvre, W. Nakamura, D. Saldana, A. Schmiesing, C. Tran, D. Vo, T. Ard, P. Heydari, B. Kim, L. Aziz-Zadeh, S. C. Cramer, J. Liu, S. Soekadar, J.-E. Nordvik, L. T. Westlye, J. Wang, C. Winstein, C. Yu, L. Ai, B. Koo, R. C. Craddock, M. Miham, M. Lakich, A. Pienta, and A. Stroud. The anatomical tracings of lesions after stroke (atlas) dataset - release 1.1. *bioRxiv*, 2017. doi: 10.1101/179614. URL `https://www.biorxiv.org/content/early/2017/08/26/179614`.

B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Gerstner, M. A. Weber, T. Arbel, B. B. Avants, N. Ayache, P. Buendia, D. L. Collins, N. Cordier, J. J. Corso, A. Criminisi, T. Das, H. Delingette, C. Demiralp, C. R. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. M. Iftekharuddin, R. Jena, N. M. John, E. Konukoglu, D. Lashkari, J. A. Mariz, R. Meier, S. Pereira, D. Precup, S. J. Price, T. R. Raviv, S. M. Reza, M. Ryan, D. Sarikaya, L. Schwartz, H. C. Shin, J. Shotton, C. A. Silva, N. Sousa, N. K. Subbanna, G. Szekely, T. J. Taylor, O. M. Thomas, N. J. Tustison, G. Unal, F. Vasseur, M. Wintermark, D. H. Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes, and K. Van Leemput. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Trans Med Imaging*, 34(10):1993–2024, Oct 2015.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL `http://arxiv.org/abs/1505.04597`.

G. Wang, W. Li, S. Ourselin, and T. Vercauteren. Automatic brain tumor segmentation using cascaded anisotropic convolutional neural networks. *CoRR*, abs/1709.00382, 2017. URL `http://arxiv.org/abs/1709.00382`.