Atlas MRI: Detecting post stroke lesions

Rosario Errazuriz
Stanford University
rerazu@stanford

Ayush Jain
Stanford University
ayush11@stanford.edu

Ray Zhong
Stanford University
rzhong0@stanford.edu

Abstract

This paper discusses the application of a Deep Learning algorithm that can segment post-stroke lesion based on a series MRI slices and their corresponding lesion masks. The algorithm has been trained using ATLAS (Anatomical Tracings of Lesions After Stroke), an open source dataset of 304 T1-weighted MRIs and 229 manually segmented lesions. The project tested whether using a Residual Neural Network Model with 20 units (using two convolutions each) can improve the accuracy (defined as the DICE coefficient) versus the provided 8-layer Convolutional Neural Network. We find that using the Residual Neural Network provides similar losses, however a weaker DICE coefficient. As future work we recommend adjusting the hyperparameters of the model for further optimization.

1. Introduction

Stroke is the leading cause of adult disability worldwide, with up to two-thirds of individuals experiencing long-term disabilities. Large-scale neuroimaging studies have shown promise in identifying robust biomarkers (e.g., measures of brain structure) of stroke recovery. However, analyzing large datasets is problematic due to barriers in accurate stroke lesion segmentation. Manually traced lesions are currently the gold standard for lesion segmentation, but are labor intensive and require anatomical expertise. While algorithms have been developed to automate this process, the results often lack accuracy. Newer algorithms that employ machine-learning techniques are promising, yet these require large training datasets to optimize performance. In this paper, we present an AI algorithm aiming to automate the process of segmenting post-stroke lesions. Specifically, using publicly available MRI images of stroke patients to train deep learning models, trying different neural network structures and various hyperparameters.

2. Previous work

The traditional gold standard for identifying stroke lesions is manual tracing, however, this is labor intensive and requires anatomical expertise, making it impractical to analyze large dataset. Many computer based algorithm have been proposed as well. For example, Jhimli or Mitraa tried to use random forest following ischemic stroke to improve segmentation effect. However, many of these algorithms still suffer from a low accuracy rate. In recent times, machine learning has been identified as a potential solution to this very challenging problem. However, till date there have been no available training datasets to build reliable algorithms in this domain. The release of the ATLAS 1.1 (Anatomical Tracings of Lesions After Stroke) dataset to the public a few months ago presented a new opportunity for deep learning application.

3. Dataset

In this paper, we utilized the newly released dataset: ATLAS. It is an open-source dataset of 304 T1-weighted MRIs with manually segmented lesions and metadata. This large, diverse dataset can be used to train and test lesion segmentation algorithms and provides a standardized dataset for comparing the performance of different segmentation methods.

The 304 MRI images from 11 cohorts worldwide were collected from research groups in the ENIGMA Stroke Recovery Working Group consortium. For each MRI, brain lesions were identified and masks were manually drawn on each individual brain in native space using MRIcron, an open-source tool for brain imaging visualization and defining volumes of interest. A minimum of one lesion mask was identified for each individual MRI. If additional, separate (non-contiguous) lesions were identified, they were traced as separate masks. An expert neuroradiologist reviewed all lesions to provide additional qualitative descriptions of the type of stroke, primary lesion location, vascular territory, and intensity of white matter disease. Finally, a separate tracer performed quality control on each lesion mask.

4. Baseline Model

A baseline Convolutional Neural Network with 8 layers has been provided in the beginning to serve as a foundation for further improvement. The image input size is [232,196,1].

The structure of this network (encoder) is shown in Figure 1. The model uses as loss a sigmoid cross entropy function and an Adam optimizer.

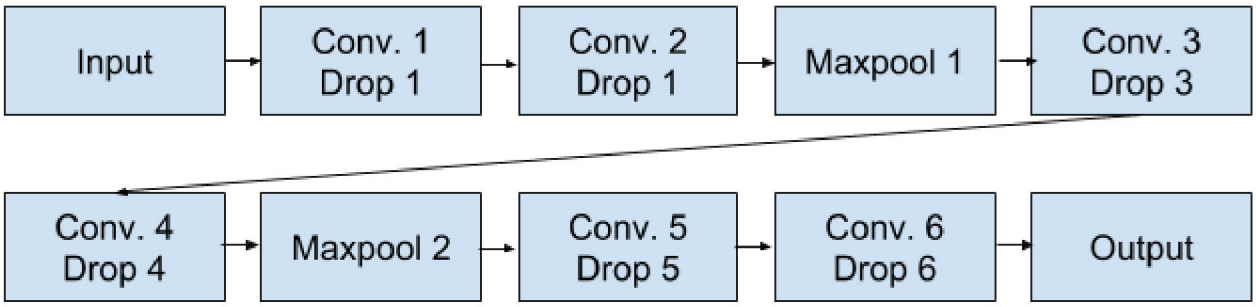

Figure 1: Structure of the baseline Convolutional Neural Network Encoder

- Conv1: Convolves the input with 64 filters of size [3,3,1], with same padding and a stride of [1,1,1]. The weights are initialized with a Xavier initializer. The output dimensions are [232,196,64]. The output then goes through a ReLU activation. Finally, a dropout regularization is applied to the activations.
- Conv2: Convolves the input with 64 filters of size [3,3,64], with same padding and a stride of [1,1,1]. The weights are initialized with a Xavier initializer. The output dimensions are [232,196,64]. The output then goes through a ReLU activation. Finally, a dropout regularization is applied to the activations.
- Maxpool 1: Performs Maxpooling on the input, with shape [2,2,1] and stride [2,2,1]. The .output dimensions are [116,98,64].
- Conv3: Convolves the input with 128 filters of size [3,3,64], with same padding and a stride of [1,1,1]. The weights are initialized with a Xavier initializer. The output dimensions are

[116,98,128]. The output then goes through a ReLU activation. Finally, a dropout regularization is applied to the activations.

- Conv4: Convolves the input with 128 filters of size [3,3,64], with same padding and a stride of [1,1,1]. The weights are initialized with a Xavier initializer. The output dimensions are [116,98,128]. The output then goes through a ReLU activation. Finally, a dropout regularization is applied to the activations.
- Maxpool 2: Performs Maxpooling on the input, with shape [2,2,1] and stride [2,2,1]. The .output dimensions are [58,49,128].
- Conv5: Convolves the input with 256 filters of size [3,3,128], with same padding and a stride of [1,1,1]. The weights are initialized with a Xavier initializer. The output dimensions are [58,49,256]. The output then goes through a ReLU activation. Finally, a dropout regularization is applied to the activations.
- Conv6: Convolves the input with 256 filters of size [3,3,256], with same padding and a stride of [1,1,1]. The weights are initialized with a Xavier initializer. The output dimensions are [58,49,256]. The output then goes through a ReLU activation. Finally, a dropout regularization is applied to the activations.

4.1 Other Hyperparameters

Besides the structure of the Neural Network described on the previous section, other relevant hyperparameters of the model are:

- **Learning rate** = 0.001
- **Proportion of units randomly dropped in dropout regularization** = 0.15
- **Mini-batch size** = 100

5. Methods

5.1 Residual Neural Networks

The most salient feature of Residual Neural Networks is their depth. Their architecture enables the training of much deeper networks, avoiding the problem of exploding or vanishing gradients. The main innovation is that in these networks the input of a lower layer is made available to a neuron in a higher layer.

These networks currently have a state-of-the-art performance for multiple deep learning use cases, such as image recognition, object detection and semantic segmentation.

5.2 Replacing the encoder

A pre-built Residual Neural Network was adapted for this project[1]. The baseline encoder was replaced with a new network, the structure of which is explained below.

1. Initial convolution
2. 20 units that have the same structure. Differences only occur for the size of the filters. The structure of each unit is shown in Figure 2 and explained below:
    a. Initiates with a Leaky ReLU activation
    b. Normalizes (batch-normalization) and performs a convolution
    c. Normalizes (batch-normalization) and performs a second convolution
    d. Adds the output with the activations of the first Leaky ReLU (step a.)
3. Leaky ReLU activation
4. Final fully connected layer with 64 neurons

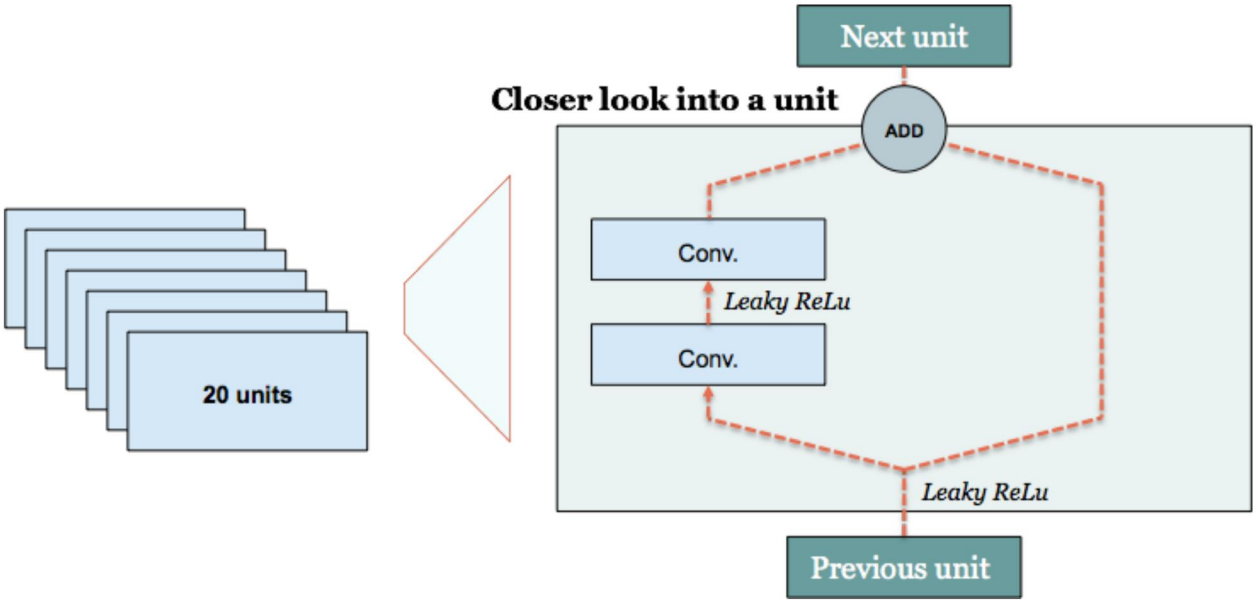

Figure : Structure of the Residual Neural Network used to encode the image

5.3 Hyperparameter tuning

The model tested various learning rates to determine its impact on the model accuracy. The set of tested learning rates each varied by one magnitude. We found that adjusting the rate from $\alpha = 0.001$ (as in the baseline) to $\alpha = 0.0001$ optimized the loss function and DICE coefficient of the model.

A further adjustment needed to be made on the mini-batch size. While the Convolutional Neural Network was operating on a mini-batch size of 100, this proved to be excessively resource-consuming for the purpose of our model. Adjusting the mini-batch size to 50 greatly improved the model performance.

[1] https://github.com/tensorflow/models/tree/master/research/resnet

6. Results

7. Conclusion / Future Work

We conclude that using the depth of the Residual Neural Network Model did not improve the accuracy as compared to the 8-layer Convolutional Neural Network. While the loss function yielded similar results, the RNN was unable to achieve a DICE coefficient that could match the Convolutional Neural Network's accuracy.

We expect that tuning the hyperparameters (e.g. number of hidden layers, numbers of neurons in final fully connected layer of encoder) could be a helpful lever to improve the model's accuracy.

References

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Identity mappings in deep residual networks.* In European Conference on Computer Vision (pp. 630-645). Springer, Cham.

[3] Zagoruyko, S., & Komodakis, N. (2016). *Wide residual networks.* arXiv preprint arXiv:1605.07146.