CS230

Stroke Detection and Dating from FLAIR MRI Scans - Winter 2021

Alex Maruniak
Stanford University
alexemar@stanford.edu

Caroline Zanze
Stanford University
ckzanze@stanford.edu

Hayden Hofmann
Stanford University
haydenhofmann@stanford.edu

Abstract

No neural network MRI stroke classification precedents exist. Nor has there been any published work done in stroke dating. Being able to classify when a stroke occurs is extremely important for the treatment of the patient because treatment is based on the time of onset. Our model detects stroke with >96% accuracy. Furthermore, we got preliminary results in the stroke dating task. We distinguished between healthy brain scan, acute stroke, and subacute stroke with 88% accuracy.

1 Introduction

Stroke detection and dating are areas ripe for exploration through deep learning. Stroke dating involves determining the time of onset of the stroke. The four main categories for this task are hyperacute (<1 day), acute (1 - 7 days), subacute (1 week - 3 weeks) and chronic (>3 weeks). A patient's treatment is determined based on which of these categories they belong to.

Detecting a stroke and determining when it occurred can change the course of treatment for patients. For example, sometimes strokes inhibit a patients ability to tell a doctor when their symptoms began. In these cases, doctors must use rely solely on scans to diagnose patients.

Our ultimate vision is use deep learning to help doctors better detect and date strokes. Since there is no precedence for MRI stroke detection with neural networks, we must first achieve the task of reliable stroke detection.

2 Related work

The work we have found in the stroke imaging field all employ convolutional neural networks (CNNs)[1][2]. We found one stroke detection network [1]. It uses CT scans as inputs. In practice, doctors do use CT scans to diagnose strokes, but they also use MRI [3]. Since there is very little work on stroke detection, we looked into other diseases in our background research. There a precedence for using CNNs on MRI brain scans for a variety of diseases including Alzheimer's [4]. Other work in stroke imaging includes segmenting strokes using U-Nets [5].

We did not find any published neural networks attempting the task of stroke dating—the temporal classification of strokes.

3 Dataset and Features

We used a dataset of MRI scans provided to us by our mentor, Dr. Elizabeth Tong. The dataset contains MRI scans of 11 control patients, 90 acute stroke patients, 32 subacute stroke patients, and 71 chronic stroke patients. For each patient, there was data from 3 types of MRI scans: diffusion-weighted magnetic resonance imaging (DWI), apparent diffusion coefficient (ADC), and fluid-attenuated inversion recovery (FLAIR). Each scan is a dicom file corresponding to a 512x512 resolution image. The number of scans within and across the scan types differed from patient to patient. Each patient also has a csv file listing which FLAIR slices in which a stroke is visible.

For this specific project, we only used the FLAIR scans and csv files, but the vastness of this dataset allows for promising future exploration.

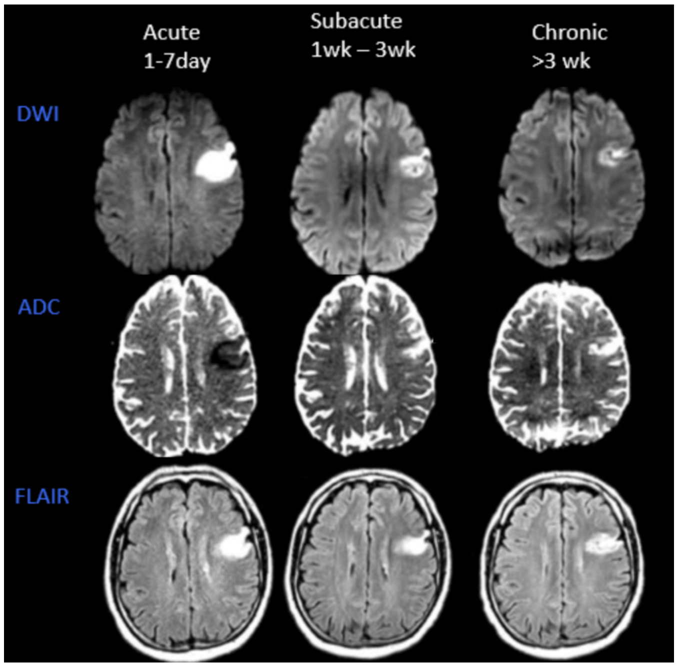


Figure 1: Appearance of acute, subacute, and chronic strokes in DWI, ADC, and FLAIR scans.

We performed experiments on three subsets of the greater dataset. The first contained 2 classes: negative and positive. The negative class contained all 1318 FLAIR slices from control patients. The positive class contained the FLAIR slices with visible stroke from acute, subacute, and chronic patients. This class contained 2168 scans. Next, we constructed a similar dataset with a positive and negative class. The positive class was the same as the previous experiement, but the negative class also contained "negative slices" from acute and subacute stroke patients—slices which did not show a stroke. This negative class contained a total of 13618 dicom files. However, in our preprocessing code, we shuffled these files and randomly selected from them so that the classes were balanced. In our third and last experiment, we created three classes: negative, positive acute, and positive subacute. The negative class is the same as that from the second experiment. The positive acute class and positive subacute class consist of all 1173 and 489 FLAIR slices containing stroke for the acute and subacute patients respectively.

In each of these data sets, our training, validation, and test splits comprised 64%, 16%, and 20% of the data, respectively.

4 Methods

Due to the sparsity of work done in the field of stroke imaging, we could not find an existing model compatible with or task. Instead, we found and employed a tumor classification model. As illustrated in the below diagram, the model uses a series of convolutional layers, batch normalization, ReLu activations, and MaxPools. At the end of the archicecture, the activation is flattened and passed through two consecutive dense layers. These layers include dropout if specified. The last layer is the softmax. The first two experiments are binary classification problems; they output whether the scan is positive or negative. The third experiment had three classes: negative, acute, and subacute.

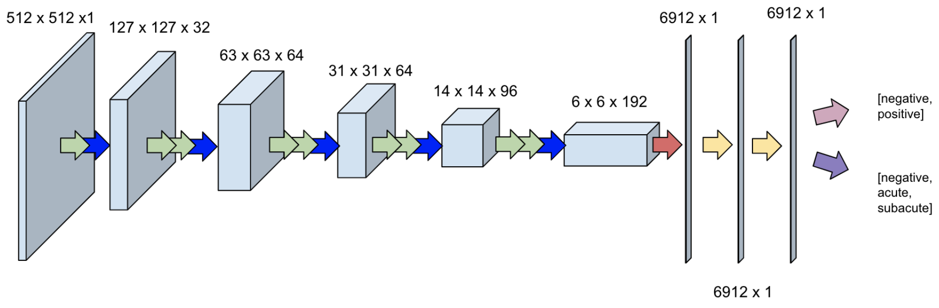


Figure 2: Model Architecture

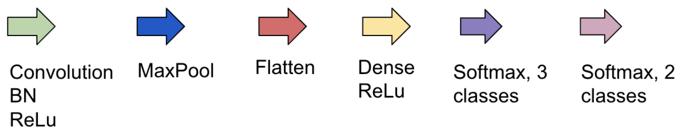


Figure 3: Model Architecture Key

We used stochastic gradient descent with Nesterov momentum, also known as Nesterov Accelerated Descent. This smooth out the steps of gradient descent and uses a look-ahead term in order to ensure that the algorithm slows down when approaching a minimum.

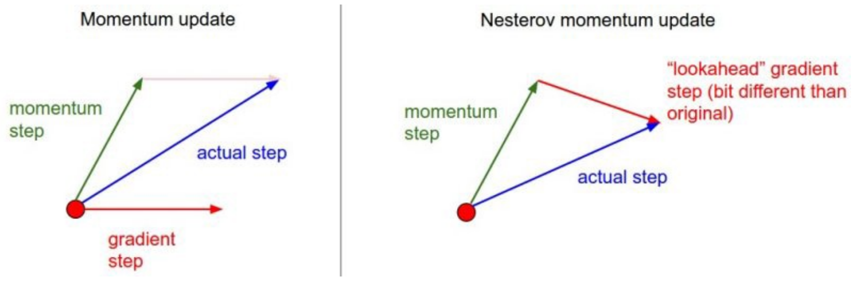


Figure 4: Diagram demostrating Nesterov momentum [6]

For this, we will refer to parameters W and b as θ, since we apply the same formulas to each. We use the formulas
$v_t = \beta v_{t-1} + \nabla J_\theta(\theta - \beta v_{t-1})$
$\theta = \theta - \alpha v_t$
Where $\theta - \beta v_{t-1}$ gives an approximation of the next position our parameters will be [7].

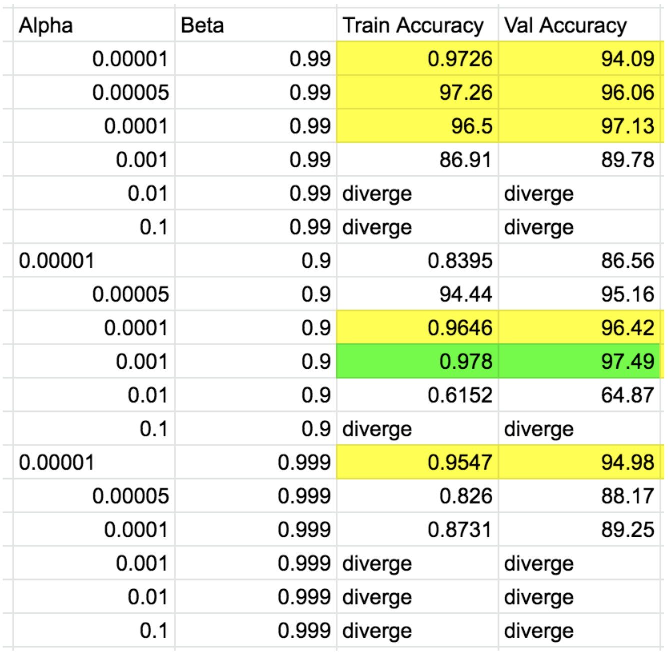
We used categorical cross-entropy loss, as is used often with binary classifiers and softmax outputs. The equation for this loss is

$$Loss(y, \hat{y}) = -\sum_{i=0}^{n} y^{(i)} * log(\hat{y}^{(i)})$$

Where there are n test examples, and $y^{(i)}$ and $\hat{y}^{(i)}$ are the true value of the output and our models predicted value of the output for example i, respectively.

5 Experiments/Results/Discussion

As a group, we followed more of a "Panda approach" where we babysat one model. The reason for this was that the model we found online worked well with minimal changes. We decided that we could best create an accurate model by working hard to optimize it with different hyperparameters and datasets modifications. The hyperparameter tuning was critical in improving both the train and validation accuracy of our model. The main hyperparameters we tuned were the learning rate (alpha) and the momentum (beta). We decided to run a grid search on these two hyperparameters in order to find the optimal combination. We ran the grid search testing alpha at intervals of 10^{-r} where r ranged from 1 to 5 and beta at intervals of $1 - 10^{-k}$ where k ranged from 1 to 3. Table 1A below outlines the hyperparameter grid search results. In the end, we found that a learning rate of 0.0001 and a momentum of 0.9 provided both the highest train and validation accuracy (highlighted in green).



Alpha	Beta	Train Accuracy	Val Accuracy
0.00001	0.99	0.9726	94.09
0.00005	0.99	97.26	96.06
0.0001	0.99	96.5	97.13
0.001	0.99	86.91	89.78
0.01	0.99	diverge	diverge
0.1	0.99	diverge	diverge
0.00001	0.9	0.8395	86.56
0.00005	0.9	94.44	95.16
0.0001	0.9	0.9646	96.42
0.001	0.9	0.978	97.49
0.01	0.9	0.6152	64.87
0.1	0.9	diverge	diverge
0.00001	0.999	0.9547	94.98
0.00005	0.999	0.826	88.17
0.0001	0.999	0.8731	89.25
0.001	0.999	diverge	diverge
0.01	0.999	diverge	diverge
0.1	0.999	diverge	diverge

Table 1A: Grid Search Results

Additionally we chose a mini-batch size of 10 to help speed up our training during our hyperparameter search. Increasing the batch size led to marginally better results but took significantly longer to train. The mini-batch size of 10 was the perfect balance between efficiency and accuracy.

In our first experiment, we used positives from scans from all of our stroke patients and negatives scans just from control patients who didn't have a stroke. Our confusion matrix in Table 1 highlights that this model was quite effective.

Confusion Matrix		
	Predict 0	Predict 1
Actual 0	247	17
Actual 1	7	399

Table 1: Classification for Control and Positives Data

In the second experiment, we used the same positive and negative examples but this time added negative scans from patients who had a stroke. This more accurately reflects how our model would be deployed. In looking at Table 2, it is clear that our model remained equally effective.

We used accuracy as our metric because ultimately the application of stroke classification is to help doctors accurately decide which MR scans have a stroke present. In Table 4, we see that the recall is higher than the precision meaning that we have more false positives than false negatives. This is important for our application because it is patients' lives depend on diagnosis. Therefore flagging an

Confusion Matrix		
	Predict 0	Predict 1
Actual 0	246	18
Actual 1	7	399

Table 2: Classification for Negatives/Positives Same Patient Data

image as a stroke is much better than missing a possible diagnosis. We did not overfit to the training set because the train and validation were always very similar. Additionally the test accuracy was close to our train accuracy.

We trained with and without dropout regularization for our model, and while the training accuracy's were similar, the validation accuracy without dropout was slightly worse than it was with dropout. Therefore, we decided to leave in dropout regularization for all of our trials.

Once we optimized our parameters, we tested our model. The results of the first and second tasks are summarized in the following table:

Model Summary					
	Precision	Recall	Train Accuracy	Validation Accuracy	Test Accuracy
Control vs. Positive	95.9	98.3	97.8	97.5	96.4
Negative from Positive	95.7	98.3	97.1	97.3	96.3

Table 3: Classification Summary for Different Datasets

Given our promising results and high accuracy, we took some preliminary steps in stroke dating problem. We modified our model to perform softmax on three classes instead of two. The three classes were subacute positive, acute positive, and negative. Using the same hyperparameters we optimized for our main problem, we got a train accuracy of 89.58%, a validation accuracy of 85.31%, and a test accuracy of 87.94%.

6 Conclusion/Future Work

Ultimately, our model was very effective in stroke detection regardless of the distribution of the negative data. Although we believed we only had time to train a stroke detection model, we had early success. After some parameter tuning, we had a model with >96% train and test accuracy. We were therefore able to start exploring stroke detection as mentioned at the end of the previous section. The results were promising, and we are optimistic about continuing to work on it.

Further steps include adding the positive chronic data, testing out more batch sizes, balancing the datasets via upsampling, and experimenting with the model archetecture. We could also split the acute class into hyperacute and acute. Adding another class would increase the specificity and usefulness of our model.

Looking even further ahead, once we have built a strong model to date strokes from individual MRI FLAIR slices, we can expand our inputs. Each input could represent a single patient, as it would include all their scans. We would try initially with only the FLAIR scans, but we anticipate that the model wouldn't differentiate well between the 5 classes. Therefore we would probably add the DWI and ADC scans as well. Although this would make training and performance much slower, we anticipate that it would yield the best results.

7 Contributions

Much of the time, Hayden, Caroline and Alex worked together closely on the project, changing the code and debugging together. This was mainly accomplished through weekly zoom meetings. Hayden completed the grid search, and Alex and Caroline did more data labeling and uploading.

References

Code: [https://github.com/21Vipin/Medical-Image-Classification-using-deep-learning]

Acknowledgement: Dr. Elizabeth Tong was our mentor, who provided us with the idea for the project as well as all of the data.

Libraries Used: Tensorflow, Keras, pydicom, pickel, h5py, pandas, scikit-learn

[1] D. R. Pereira, P. P. R. Filho, G. H. de Rosa, J. P. Papa and V. H. C. de Albuquerque, "Stroke Lesion Detection Using Convolutional Neural Networks," 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 2018, pp. 1-6, doi: 10.1109/IJCNN.2018.8489199.

[2] Soun, J. E., Chow, D. S., Nagamine, M., Takhtawala, R. S., Filippi, C. G., Yu, W., Chang, P. D. (2021). Artificial Intelligence and Acute Stroke Imaging. American Journal of Neuroradiology, 42(1), 2-11.

[3] Allen, L. M., Hasso, A. N., Handwerker, J., Farid, H. (2012). Sequence-specific MR imaging findings that are useful in dating ischemic stroke. Radiographics, 32(5), 1285-1297.

[4] Lin, W., Tong, T., Gao, Q., Guo, D., Du, X., Yang, Y., ... Alzheimer's Disease Neuroimaging Initiative. (2018). Convolutional neural networks-based MRI image analysis for the Alzheimer's disease prediction from mild cognitive impairment. Frontiers in neuroscience, 12, 777.

[5] Zhou, Y., Huang, W., Dong, P., Xia, Y., Wang, S. (2019). D-UNet: a dimension-fusion U shape network for chronic stroke lesion segmentation. IEEE/ACM transactions on computational biology and bioinformatics.

[6] Bushaev, Vitaly. "Stochastic Gradient Descent with Momentum." Medium, Towards Data Science, 5 Dec. 2017, towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d.

[7] Sebastian Ruder. "An Overview of Gradient Descent Optimization Algorithms." Sebastian Ruder, Sebastian Ruder, 20 Mar. 2020, ruder.io/optimizing-gradient-descent/.