Finding Traumatic Brain Injury in CT Scans

Amanda Breton, Alex Bzdel, Mackenzie Looney, Zhenye Lu
ECE 588, Fall 2022

Duke PRATT SCHOOL of ENGINEERING

Background

Traumatic brain injury (TBI) is a multifaceted condition that affects brain function and quality of life. Mild TBIs can lead to temporary complications but often can result in serious long-term complications and death [2]. Speed and accuracy are the most important factors in managing TBI.

Computerized Tomography (CT) obtains detailed internal images of the body. Most TBI patients undergo CT scans, but noise and artifacts make it difficult for clinicians to identify regions of TBI. In this project, we aim to speed up the diagnoses process segmenting out and highlighting areas of TBI interest and flagging splices for a professional to take a closer look.

Methods

A. Preprocessing

1. Histogram equalization: The first step in the preprocessing pipeline was to perform histogram equalization on the CT scan. CT scans are in one spectrum from black to white, making it difficult to tell similar gray areas apart from each other. Applying histogram equalization helps us see regions we wouldn't be able to see before. It works by spreading out the high-intensity values in the histogram, which can help increase the contrast of the image. In this project, histogram equalization is done block by block.

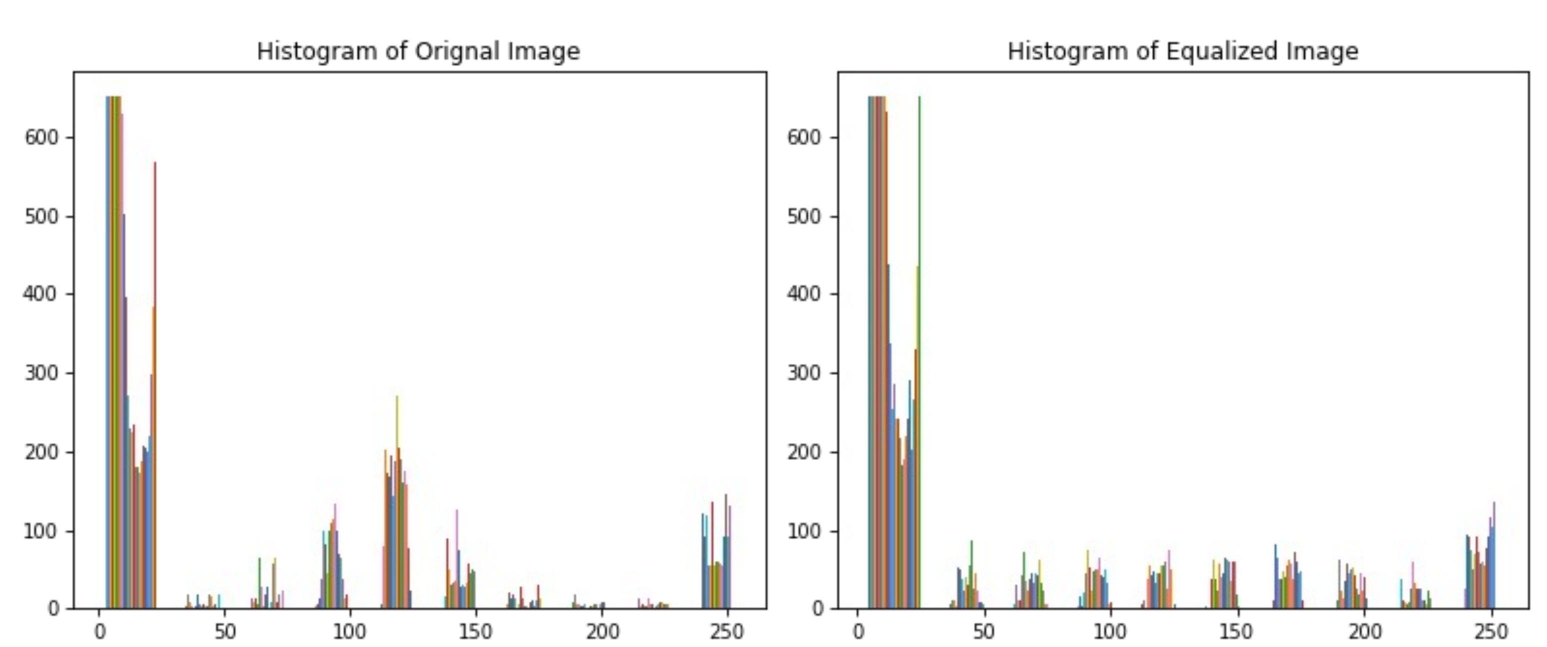


Figure 1A. Histogram distribution of original CT scan.
Figure 1B. Histogram distribution of equalized CT scan.

2. Median filtering: After the histogram equalization, the median filter with the size of 3 is used to help remove noise from images.
3. Gray matter subtraction: In this step, we first get the maximum intensity level of the histogram using the mode function. Then, subtract it from the image to remove gray matter intensity.
4. Total variation: Total variation denoising is performed using the function denoise_tv_chambolle with weight of 0.25. All other parameters are by default.

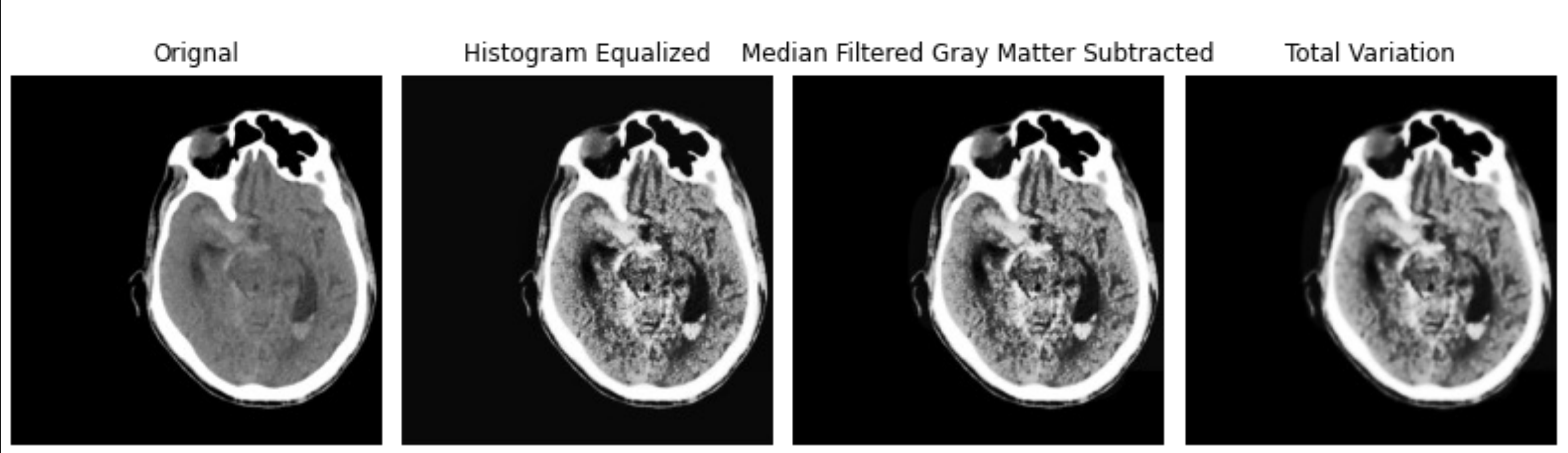


Figure 2. Progression of image augmentation during preprocessing

B. Segmentation

A few different segmentation methods were experimented with before settling on using Multi-Otsu thresholding as the segmentation method for this project. To understand Multi-Otsu thresholding, one needs to understand Otsu thresholding, the basis for Multi-Otsu thresholding.
1. Otsu thresholding: Segmentation at its core is an impulse problem and tries to make each class you're trying to segment as compact as possible. Otsu's method assumes the histogram of the image is bimodal, and finds a threshold where the pixel values are separated into two groups, thus being segmented. The threshold is found by minimizing the weighted within class variance, whose equation is seen below:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

2. Multi-Otsu thresholding: Otsu's method can be adjusted to achieve multiple segmentations by performing Otsu's method in a hierarchical way to achieve multiple thresholds. This is what was used to achieve the segmented results.

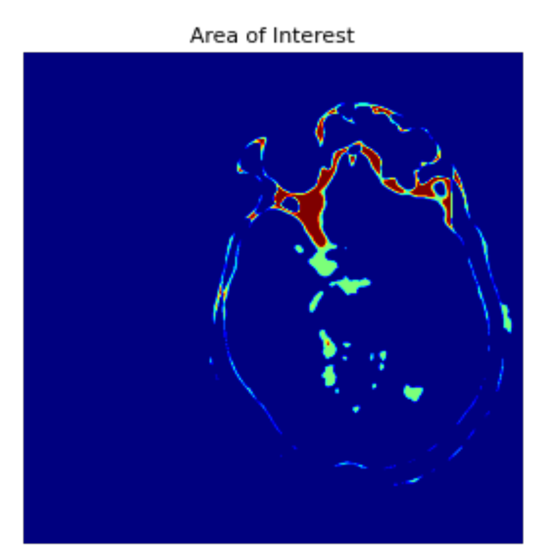


Figure 3. Segmented brain, green indicates areas of interest

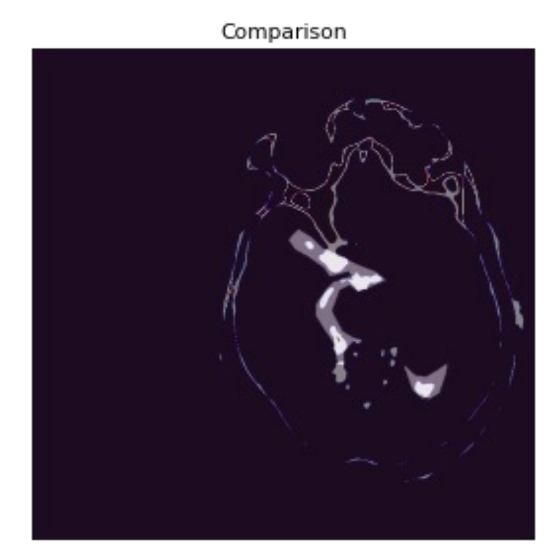


Figure 4. Segmented brain with ground truth overlay

C. Postprocessing Analysis

To quantitatively assess the success of the algorithm, recall, precision and accuracy calculations were performed with the equations below:

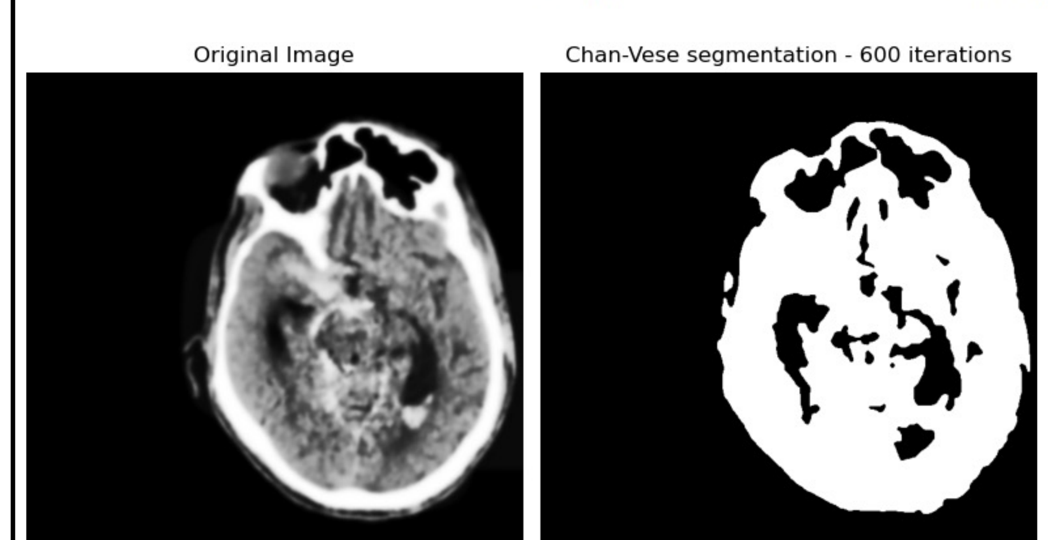
$$recall = \frac{TP}{TP+FN} \qquad precision = \frac{TP}{TP+FP} \qquad accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Confusion matrices were created to visualize the number of true positive, true negative, false positive, and false negative predictions.

D. Other Segmentation Methods

Chan-Vese Segmentation was also explored and is based on active contours. The goal is to minimize the following energy function:

$$\underset{c_1,c_2,C}{\arg\min}\ \mu\,\mathrm{Length}(C) + \nu\,\mathrm{Area}(inside(C)) + \lambda_1 \int_{inside(C)} |f(x)-c_1|^2\,dx + \lambda_2 \int_{outside(C)} |f(x)-c_2|^2\,dx.$$



This algorithm did not provide the desired segmentations and the computation time was much longer than other methods, especially with higher iteration limits.

Figure 5. Results of Chan-Vese segmentation

Results

A performance analysis was done by identifying the images that contained a large number (>1.5% of total pixels) of flagged pixels. If an image was identified and had a corresponding professional segmentation, it counted as a true positive.

Recall = 0.735
Precision = 0.299
Accuracy = 0.561

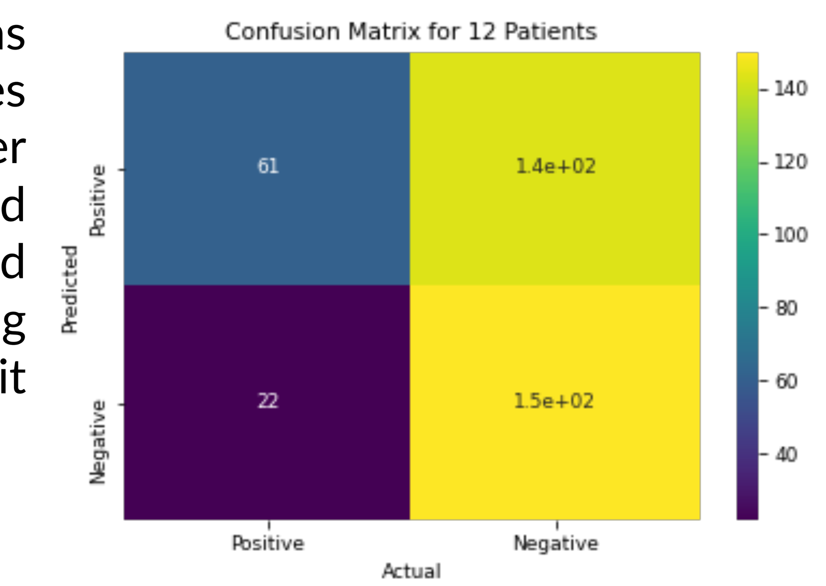


Figure 6. Image accuracy confusion matrix for 12 patients, some containing TBI, some not.

There are some false negatives, but in comparison to the true positives and negatives, they are not a significant concern. The main goal of trying to detect TBI within images as a whole was to decrease the load of CT scans doctors had to view, so this algorithm successfully eliminates slices without areas of TBI. The thresholding value could be adjusted depending on user goals for strictness of image flagging.

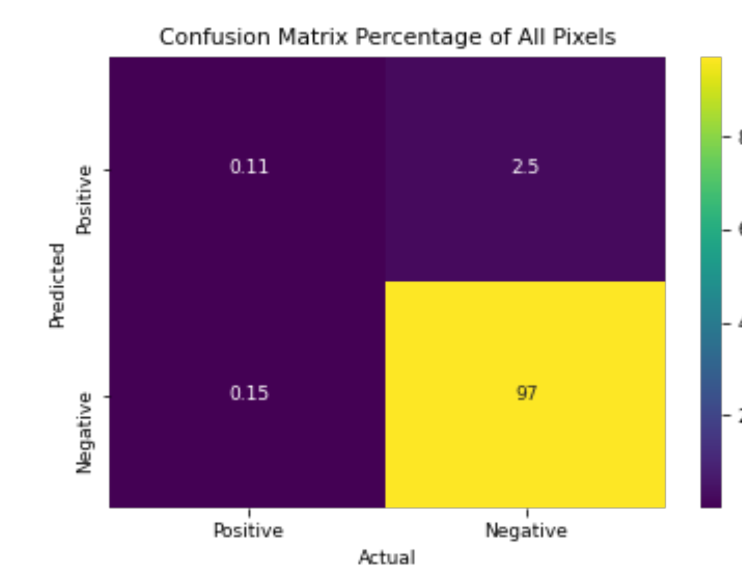


Figure 7. Pixel accuracy confusion matrix for 6 images, all containing TBI.

Another performance analysis was done, but by comparing segmented regions to ground truth annotations on a pixel basis. Accuracy was high because the images were mostly identified as true negatives.

Recall = 0.412
Precision = 0.0410
Accuracy = 0.974

Discussion

In a traditional setting, image processing in the medical field is used to give clinicians better visibility of their patients' scans. This can be pivotal in improving patient outcomes. Image segmentation for TBI is a rapidly growing area of research, with a plethora of approaches being explored.

Researchers at the University of Michigan developed an automated system to identify, localize and quantify the imaging features of TBI using Gaussian mixture model for segmentation. Researchers at the Kochi University of Technology developed a CNN to increase the efficiency of 3D image reconstruction from MRI brain scans [6?]. The researchers found a way to decrease the dimensionality, leading to decreased computational costs. Neural networks can be applied to audio recordings of an athlete speaking after an injury. Researchers at North Umbria University increased the reliability and efficiency of these tests [7?]. An intersection of image and audio processing could increase the performance of these tests further.

References

[1] G. T. Manley, and A. I. R. Mass, "Traumatic Brain Injury: An International Knowledge-Based Approach," *JAMA*, vol. 310, no. 5, 2013, pp. 473-474.
[2] "Traumatic brain injury," Mayo Clinic, https://www.cdc.gov/traumaticbraininjury/index.html (accessed Nov. 2, 2022).
[3] "Histograms - 2: Histogram equalization," *OpenCV*, Available at: https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html. (Accessed: Dec. 2, 2022).
[4] A. Burkov, "Model Performance Assessment," in *The Hundred-Page Machine Learning Book*, Quebec, Canada: Andriy Burkov, 2019, ch. 5, pp. 13-19.
[5] S. M. R. Soroushmehr, A. Bafna, S. Schlosser, K. Ward, H. Derksen, and K. Najarian, "CT Image Segementation in Traumatic Brain Injury*" in *Conference of the IEEE EMBC*, Milan, Italy, 25-29 Aug. 2015, pp. 2973-2976.
[6] H. Zhang, Y. Shinomiya, and S. Yoshida, "3D MRI Reconstruction Based on 2D Generative Adversarial Network Super-Resolution," *Sensors*, vol. 21, no. 9, 2021, pp. 2978.
[7] C. Wall, et al., "A deep learning-based approach to diagnose mild traumatic brain injury using audio classification," *PLOS One*, vol. 17, no. 9, 2022.
[8] P. Getreuer, "Chan-Vese Segmentation," *Image Processing On Line*, vol. 2, 2012, pp. 214–224.
[9] P. Virtanen, et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, no. 3, 2020, pp. 261-272.
[10] C.R. Harris, et al., "Array programming with NumPy," *Nature*, vol. 585, 2020, pp. 357–362.
[11] W. McKinney, "Data Structures for Statistical Computing in Python," *Proceedings of the 9th Python in Science Conference*, Austin, Texas, 2010, pp. 56–61.
[12] A. Clark, 2015. "Pillow (PIL Fork) Documentation", *readthedocs*, 2015. Available at: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.
[13] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
[14] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", *Computing in Science & Engineering*, vol. 9, no. 3, 2007, pp. 90-95.
[15] S. van der Walt, et al., "scikit-image: Image processing in Python," *PeerJ*, vol. 2, ed. 453, 2014.
[16] F. Pedregosa, et al, "Scikit-learn: Machine Learning in Python," *JMLR*, vol. 12, 2011, pp. 2825-2830.
[17] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, 2021, pp. 3021