Resting-State Functional MRI(rs-fMRI) analysis using Data Mining techniques

Sai Likhith Yadav Seera
Computer Science
School of Computing and Augmented Intelligence
Arizona State University
Tempe, United States of America
sseera@asu.edu

***Abstract*—When compared to traditional medical specialties, the use of magnetic resonance imaging (MRI) in healthcare and the emergence of radiology as a practice are both relatively new. Since its revival in the 1970s and subsequent adoption in the 1980s, the use of MRI has grown exponentially, spawning exciting new areas of research. One such advancement is the use of computational techniques to analyze MRI images like those of a radiologist. With the introduction of low-cost, high-performance computing hardware and parallel advances in computer vision, MRI image analysis has grown at an unprecedented rate. Because of the interdisciplinary and complex nature of this sub field, it is critical to survey the current landscape and examine current approaches for analysis as well as future trending trends [1].**

***Index Terms*—magnetic resonance imaging, computer vision, pixel, contour, template matching, clustering, classification**

I. Introduction

In this project, I applied image processing techniques to resting-state functional magnetic resonance imaging (rs-fMRI) scans of the brain. These include extracting brain boundaries, classifying independent component (IC) images as noise and resting-state networks (RSN) using supervised learning techniques, and detecting the number of clusters in extracted brain slices using unsupervised learning techniques.

II. Description of Solution

In this project, I will be working with rs-fMRI data. rs-fMRI is functional magnetic resonance imaging that is used to assess functional connectivity in brain networks while the patient is at rest. The patient's brain is evaluated based on blood consumption activations (shown as red clusters in **Figure 1**) in various regions of the brain.

These clusters reflect changes in brain activity caused by active areas of the human body. fMRI data is typically 4D data that is further decomposed into spatial independent components (ICs) using MELODIC software. In this project, I will use the spatial independent components (ICs) of the patient's rs-fMRI scan, which are similar to the image in **Figure 1** (except the blood activation (red/blue clusters) part), to implement the model. These images will depict the evolution of the brain over time. A patient typically has 100 such images, and the location of the red cluster and blue cluster in each image varies depending on the active part of the body.

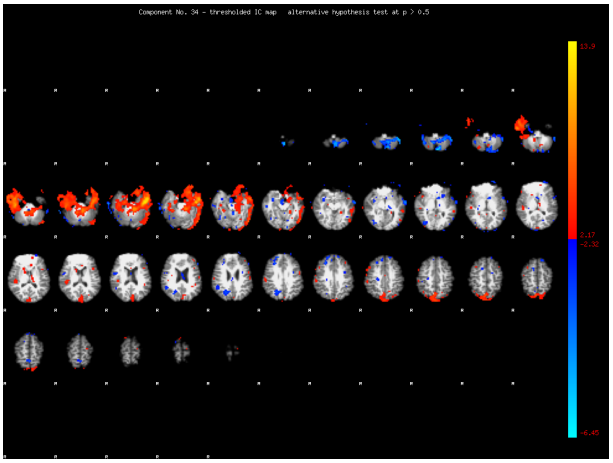

Fig. 1. Sample Brain rs-MRI scan image.

A. Extracting Brain boundaries from rs-fMRI data

In this section, there are two tasks performed. The first step is to extract brain slices. The second is to retrieve brain boundaries from the derived brain slices.

The brain slices are extracted from the IC images using template matching. The letter "R" is considered the template matcher in this case, and it maps the coordinates where this is located [2]. The non-black pixel image is sliced along these coordinates. Then, the brain boundary is identified using contour edge detection on these brain slices [3].

B. Detecting brain clusters using unsupervised learning

In this section, there are two tasks performed. The first step is to extract brain slices. The second is to retrieve the number of clusters from the derived brain slices.

The brain slices are extracted from the IC images using template matching. The letter "R" is considered the template matcher in this case, and it maps the coordinates where this is located. The non-black pixel image is sliced along these coordinates. Image masking is then used to separate the brain blood activation cells (red and blue clusters) from the brain image [4]. The DBSCAN (Density-based spatial clustering

of applications with noise) algorithm is used to calculate the number of clusters using the masked cells [5]. Clusters with pixel values greater than 135 pixels were taken into account.

C. Classifying IC images using supervised learning

In this section, three tasks are performed. The first step is to create a single-label list of IC images for binary classification. The second is to train the model using these IC images. Finally, test the data with the trained model to predict the labels for the test data and calculate the performance metrics of the trained model on the test data.

A single-label type of IC image is created where only two labels are present: ”0” for noise and ”1” for RSN. RSN ICs have different labels (1, 2, and 3); these labels are changed to 1 when the existing labels are greater than 0. Using this image data and its corresponding label, the Convolutional Neural Network (ConvNet/CNN) algorithm with ReLU and Softmax activation functions is used to create a supervised learning model [6]. Using this model, I have predicted the label for the test data and calculated several performance metrics like accuracy, precision, sensitivity, and specificity of the predicted labels compared to the test data labels [7].

III. Results

I have implemented in Python using several libraries, which include computer vision (cv2), pandas, sklearn, TensorFlow, Keras, and PIL.

A. Extracting Brain boundaries using Contour detection

The following **figures 2 and 3** show the brain image slices and the corresponding brain boundary images for an instance of IC image present in the image data.

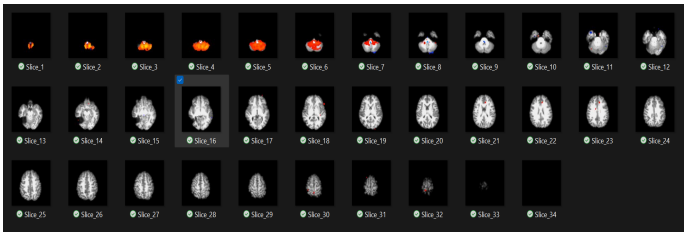

Fig. 2. Brain slice extraction from rs-MRI scan image.

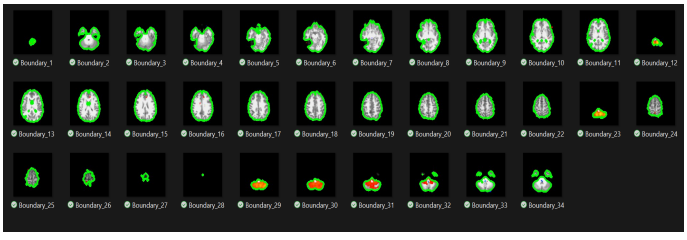

Fig. 3. Brain boundary detection from brain image slices.

B. Detecting brain clusters using DBSCAN

Figure 4 and table 1 show the clusters of brain image slices and the corresponding number of clusters for an instance of an IC image present in the image data.

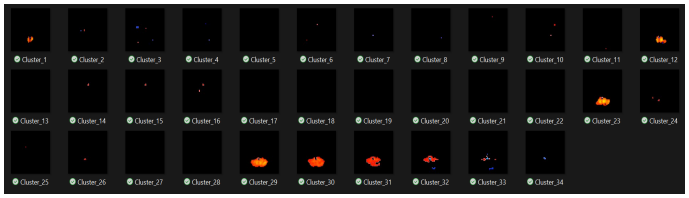

Fig. 4. Brain cluster detection from brain image slices.

TABLE I
NUMBER OF CLUSTERS IN EVERY BRAIN SLICE OF AN IC IMAGE

IC_1_thresh.png	
SliceNumber	***ClusterCount***
1 to 10	0
11 to 20	0
21 to 28	0
29	1
30	2
31	2
32	2
33	1
34	0

C. Binary Classification of IC images using CNN

Tables 2 and 3 show the predicted classification of IC images as noise or RSN, as well as the corresponding performance metrics of the trained model on the predicted over the test data.

TABLE II
LABELS OF PREDICTED CLASSIFICATON OF IC IMAGES AS NOISE(0) OR RSN(1)

testPatient Data (107 IC images)		
IC_Number Range	***Label***	***IC_Numbers***
1 to 10	0	2,3,5,6,7,9,10
	1	1,4,8
11 to 20	0	12,13,16,17,19
	1	11,14,15,18,20
21 to 30	0	26,28
	1	21,22,23,24,25,27,29,30
31 to 40	0	34,36,39
	1	31,32,33,35,37,38,40
41 to 50	0	42,43,44,45,46
	1	41,47,48,49,50
51 to 60	0	51,53,57,60
	1	52,54,55,56,58,59
61 to 70	0	61,63,64,67
	1	62,65,66,68,69,70
71 to 80	0	71,72,74,76,77,78,79
	1	73,75,80
81 to 90	0	81.84.85.86.87.88.89.90
	1	82,83
91 to 100	0	94,95,96,97,98,99,100
	1	91,92,93
101 to 107	0	101,102,104,105,106,107
	1	103

TABLE III
PERFORMANCE METRICS OF THE TRAINED MODEL ON THE PREDICTED OVER THE TEST DATA

testPatient Data (107 IC images)	
Metrics	***Value (in terms of %)***
Accuracy	79%
Misclassification-Error	21%
Precision	67%
Recall	85%
F1-Score	75%
Specificity	76%
Sensitivity	85%

IV. CONTRIBUTION

The contributions that I have made to this project are as follows:

- Implemented slicing of brain imaged from the IC images using template matching in Python.
- Implemented boundary detection of the brain images using contour-edge detection in Python.
- Implemented image masking to retrieve the red and blue activation cells from the brain image and then used DBSCAN in Python to identify and calculate the number of clusters formed.
- Implemented a binary classification model using CNN to identify the IC image as noise or RSN, and then used a confusion matrix in Python to calculate performance metrics on the predicted over the test data.

REFERENCES

[1] V.Vadmal, G.Junno, C.Badve, W.Huang, K.A. Waite, J.S. Barnholtz-Sloan, "MRI image analysis methods and applications: an algorithmic perspective using brain tumors as an exemplar," PNeuro-Oncology Advances, Volume 2, Issue 1, January-December 2020, vdaa049, https://doi.org/10.1093/noajnl/vdaa049 .

[2] For template matching in Python, https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html

[3] For contour detection or edge detection in Python, https://learnopencv.com/contour-detection-using-opencv-python-c

[4] For image masking or image segmentation in Python, https://machinelearningknowledge.ai/image-segmentation-in-python-opencv/

[5] For DBSCAN clustering in Python, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

[6] For CNN binary classification model, https://www.datacamp.com/tutorial/convolutional-neural-networks-python .

[7] For confusion matrix and performance metrics calculation, https://www.w3schools.com/python/python_ml_confusion_matrix.asp .