CSE572-DATA MINING PORTFOLIO REPORT

Siva Rama Pavan Kumar Buddi
School Of Computing and Augmented Intelligence
Arizona State University
Tempe, Arizona , USA
sbuddi3@asu.edu

***Abstract*— Brain functionality detection gained importance in the past few years when machine learning came into the picture. The proposed system in this paper uses rs-fMRI (functional magnetic resonance imaging) to detect the functionality of the brain network when it is in a resting state. This detection is done by analyzing the clusters of blood consumption activations. The dataset consists of images of IC images of a patient's fMRI scan. These IC images are used to detect the functionality of the brain using machine learning.**

Keywords—Machine Learning, Supervised Learning, Unsupervised Learning, DBSCAN, Template Matching, TensorFlow, Keras, Accuracy, Cluster

I. Introduction

With the advancement in technology detecting the diseases and curing them has become widely popular. One such advancement is in the field of medical sector which is detecting the functionality of brain to identify any disease affecting the functionality of the brain. Most the advancements in medical field are crucial in the present world due the factors that are affecting the human kind (pollution etc.). In the past few years, machine learning is used in every sector to find the accuracy and identify the problems which humans cannot do. In the present world computers are overtaking humans to identify the issue in no time and resolve them before it gets tough. Working with the brain is quite impossible without proper identification of the effected part and its functionality. Researchers are working on identifying the functionalities rather than developing a system to detect the factors effecting the functionalities. Using machine learning techniques this paper describes the detection of brain activity using patient's data. Definitely there is a necessity to develop a detection system which can be used to detect the disease fast so that it can be cured initially.

II. Project Background and Motivation

Since the last decade advancement in technology has shown a major spike in its use for enhancing human life. Till date, rs-fMRI scans of the brain functionality in a resting state are mainly analyzed by unsupervised learning. Though unsupervised learning deals well with image data, supervised learning has overcome its importance in dealing with image data by training and testing the data. These algorithms built helped researchers to find a better way to analyze the functionality of the brain in its resting state. This requirement in detecting brain functionality has paved way for the new development. Better solutions were developed and analyzed to overcome the current existing project.

The main motivation for this project came from the excitement of building an unknown system. There are techniques to detect brain functionality using rs-fMRI scans of a patient. But these techniques could not meet future necessities.

III. Proposed System

This system uses fMRI scans of patients to detect the functionality of the brain in a resting state using blood consumption activations. One main challenge is to get the clusters (red-colored structures in the images) and count the clusters to perform analysis on the values. Here the patient's data is analyzed by both unsupervised and supervised machine learning techniques to classify the images and find the accuracy. This paper gives the best solution to overcome the challenges and produce great results.

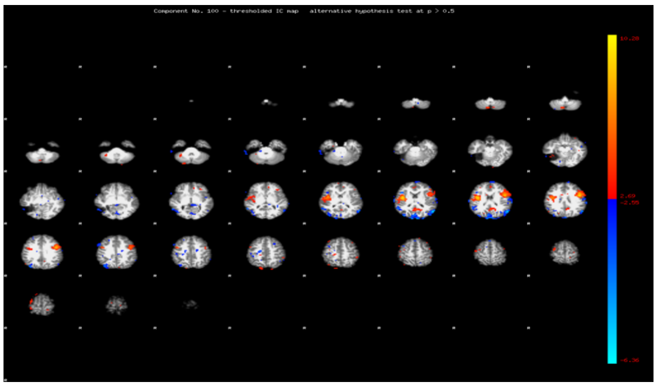

Component No. 100 - thresholded IC map alternative hypothesis test at p > 0.5

Fig.1. Sample IC thresh scan of patient

A. Data Preprocessing and folders generations

The dataset consists of images of a patient's rs-fRMI scans. For this system, only IC images that end with the word "thresh are needed. These images (that end with thresh) are filtered and used for brain slice extraction and brain boundary extraction which are then used to get the clusters and finally accuracy.

For every process the system stores images of patient data in respective folders. For this, a method is implemented to generate the folders dynamically which are used in future implementation.

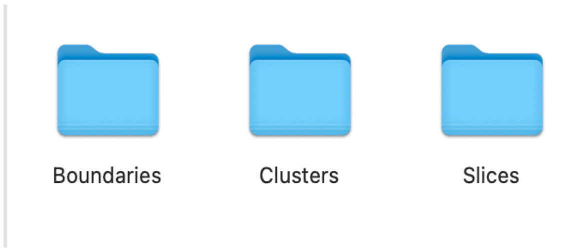


Fig.2. Folders generated

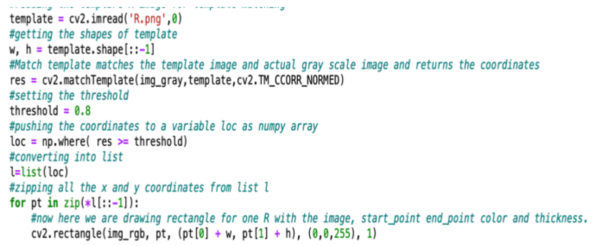
B. Brain Slice Extraction

Every IC scan image of a patient consists of a letter 'R' surrounding every slice of the brain over the time. Brain slice extraction is used to get every slice over period of time and use it for brain boundary extraction which will then be used to find clusters and apply machine learning techniques to find the accuracy. This brain slice extraction is done by using “Template Matching”.

Template matching is moving the template image over the desired image and getting if there is a matching with the template. This system uses a template (letter ‘R’ in this scenario) and matches with the IC image to get the coordinates of the ‘R’ in the image.

After getting the coordinates of ‘R’ in the IC scan image of a patient, a method is used to get each slice by cropping the image based on coordinates by leaving empty slices and storing it in a separate folder named “Slices” which is dynamically created by the method before the implementation of brain slice extraction function.

Fig.3. Template used for template matching



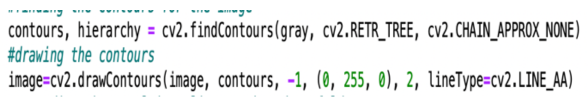
```
template = cv2.imread('R.png',0)
#getting the shapes of template
w, h = template.shape[::-1]
#Match template matches the template image and actual gray scale image and returns the coordinates
res = cv2.matchTemplate(img_gray,template,cv2.TM_CCORR_NORMED)
#setting the threshold
threshold = 0.8
#pushing the coordinates to a variable loc as numpy array
loc = np.where( res >= threshold)
#converting into list
l=list(loc)
#zipping all the x and y coordinates from list l
for pt in zip(*l[::-1]):
    #now here we are drawing rectangle for one R with the image, start_point end_point color and thickness.
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 1)
```

Fig.4. Template Matching code snippet

C. *Brain Boundary Extraction*

Once the slices of each IC images that end with thresh are filtered and saved in a folder, the boundary of the brain is extracted and saved in the other folder named “boundaries”. Firstly, each slice of a single IC scan image of a patient is stored in respective folder. In this step, the slices of specific IC scan image are imported and brain boundary extraction is performed. This boundary extraction is done by using “contour detection”.

Contour detection is an image processing technique used to detect the boundaries of the image and localize them. This technique is used in many image-processing applications and is considered the best method for border detection. Since brain data should be precise, this method is used for the system in this paper. Inbuilt methods like findContours() and drawContours() were used to detect and draw the boundary for the brain slices. After getting the boundaries of each slice of the respective IC scan image, boundaries are stored in respective IC scan image folder.



```
contours, hierarchy = cv2.findContours(gray, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
#drawing the contours
image=cv2.drawContours(image, contours, -1, (0, 255, 0), 2, lineType=cv2.LINE_AA)
```

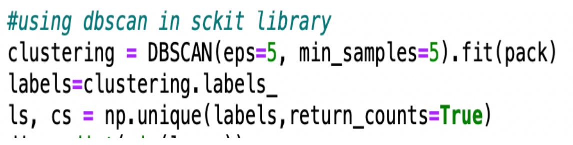
Fig.5. Contour Detection code snippet

D. *Unsupervised Learning Technique*

A machine learning algorithm that deals with unlabeled data, which is used to detect patterns and form clusters. This technique in this system helps to find the number of clusters in a brain slice of a patient’s fMRI scan. In this part, brain slices extracted from the patient’s IC scan image were used to detect the clusters. The cluster detection for each slice of the patient’s IC scan is done using a clustering technique named “DBSCAN”. DBSCAN (Density-based spatial clustering of applications with noise) is a clustering technique that groups points that are close to each other and form a cluster. In this project, DBSCAN is used to find clusters in each slice.

Firstly, the slices stored in the brain slice extraction process are retrieved and applied with the DBSCAN clustering technique. Only clusters with pixel values greater than 135 are counted to maintain big clusters. DBSCAN is imported from sklearn library and is used to find clusters of IC scan images. For detecting the clusters only eps, min_samples parameters are used where eps is the maximum distance and min_samples is the number of samples close to each other. The number of clusters are counted and stored in a list. Once the cluster detection is done resultant clusters of a brain slice are stored in the respective IC scan image cluster folder which is under main “Clusters” folder.

To display the precise values, the Slice Number and number of clusters in that slice are reported in a .csv file. Each folder contains a .csv file which has the Slice Number of all slices in that folder and the respective count of clusters. This machine learning algorithm is used to find the clusters(patterns) while the brain is in a resting state. With this, the researchers can detect the functionality of the brain when it is affected and act accordingly.



```
#using dbscan in sckit library
clustering = DBSCAN(eps=5, min_samples=5).fit(pack)
labels=clustering.labels_
ls, cs = np.unique(labels,return_counts=True)
```

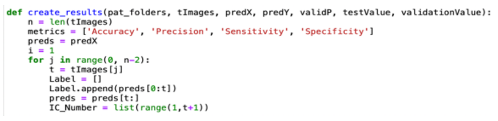
Fig.6. DBSCAN code snippet

E. *Supervised Learning Technique*

This technique is completely based on labeling the images. In this, the IC scan images are classified as noise and resting states. In dataset 5 patient data is provided to train the model to classify between noisy and resting states. The TensorFlow library is used to load the model and train, and test the data.

TensorFlow is a free library that is used for many machine learning and artificial intelligence applications. Keras was used to train and test the data. Based on these factors, accuracy is obtained by generating a confusing matrix for the train and test data values. Initially, a separate python function is developed to create a model based on values from the values. This model is stored in a .h5 which is then used to compare with train and test data values. Training and Testing of values are done with a standard machine learning ratio (70:30). Inbuilt methods in Keras are used to train and test the data. Once training and testing is done, accuracy is determined by using confusion matrix.

After training and testing, the accuracy and other factors are found and are stored in two .csv files. (metrics.csv and results.csv). Metrics.csv consists of metric, and score columns. Metric columns consist of precision, accuracy, sensitivity, and specificity of a patient IC image and results.csv consists of IC image number, and label (0 or 1).



```
def create_results(pat_folders, tImages, predX, predY, validP, testValue, validationValue):
    n = len(tImages)
    metrics = ['Accuracy', 'Precision', 'Sensitivity', 'Specificity']
    preds = predX
    i = 1
    for j in range(0, n-2):
        t = tImages[j]
        Label = []
        Label.append(preds[0:t])
        preds = preds[t:]
        IC_Number = list(range(1,t+1))
```

Fig.7. Code snippet for Generating Accuracy and other parameters

IV. Result Analysis

At the very first folder generation is done followed by brain slice extraction and brain boundary detection. Once the above process is done machine learning techniques are used to find the clusters and calculate the accuracy.

A. *Brain Slice Extraction and Brain Boundary Detection*

Brain Slice extraction is done by template matching and boundary detection is done by contour detection. The Below images display the slice of one IC scan image and its detected boundary.

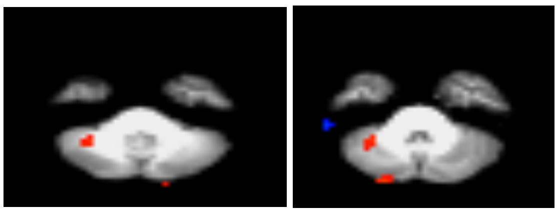


Fig.8. Sample slices of Fig,1 IC image

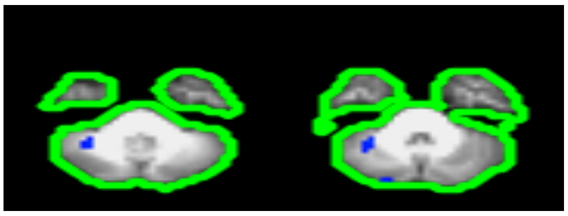


Fig.9. Sample boundaries of Fig9 slices

B. *Clustering*

Using the slices formed the system detects the clusters using DBSCAN, counts the number of clusters in each slice and displays it in a .csv file.

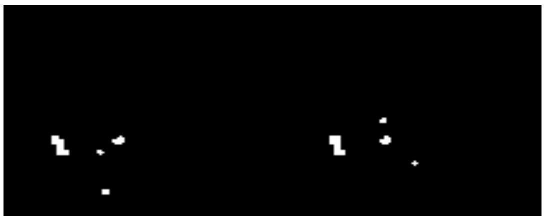
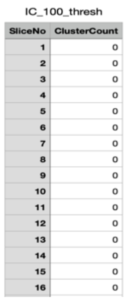


Fig.10. Clusters formed for fig 9 slices

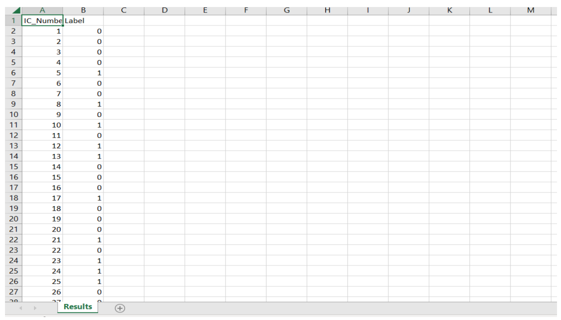

IC_100_thresh

SliceNo	ClusterCount
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0

Fig.11. Number of clusters for a sample IC thresh image

C. *Labeling and Accuracy of patient's data*

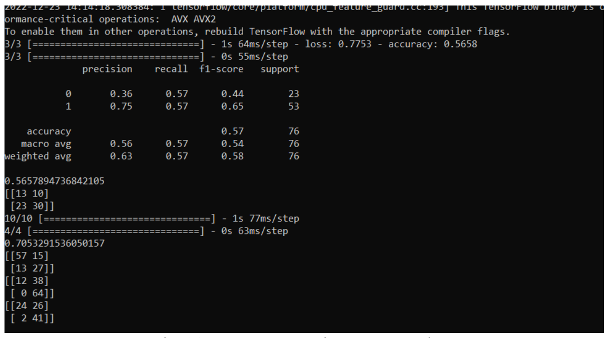
By using TensorFlow accuracy is determined by creating a model, training and testing the data. The Below figures shows the accuracy which is stored in a .csv file of a patient's data.



	A	B
1	IC_Numbe	Label
2	1	0
3	2	0
4	3	0
5	4	0
6	5	1
7	6	0
8	7	0
9	8	1
10	9	0
11	10	1
12	11	0
13	12	1
14	13	1
15	14	0
16	15	0
17	16	0
18	17	1
19	18	0
20	19	0
21	20	0
22	21	1
23	22	0
24	23	1
25	24	1
26	25	1
27	26	0

Results

Fig.12. Labelling the state of brain as noisy or resting and storing in metric.csv



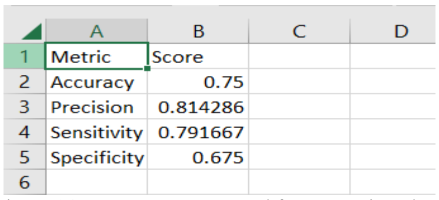
```
2022-12-23 14:14:18.308384: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is o
ormance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
3/3 [==============================] - 1s 64ms/step - loss: 0.7753 - accuracy: 0.5658
3/3 [==============================] - 0s 55ms/step
              precision    recall  f1-score   support

           0       0.36      0.57      0.44        23
           1       0.75      0.57      0.65        53

    accuracy                           0.57        76
   macro avg       0.56      0.57      0.54        76
weighted avg       0.63      0.57      0.58        76

0.5657894736842105
[[13 10]
 [23 30]]
10/10 [==============================] - 1s 77ms/step
4/4 [==============================] - 0s 63ms/step
0.7053291536050157
[[57 15]
 [13 27]]
[[12 38]
 [ 0 64]]
[[24 26]
 [ 2 41]]
```

Fig.13. Accuracy using TensorFlow



	A	B	C	D
1	Metric	Score		
2	Accuracy	0.75		
3	Precision	0.814286		
4	Sensitivity	0.791667		
5	Specificity	0.675		
6				

Figure 14. Parameters generated for one patient data.

V. Scope For Future Work

The machine learning algorithms used at present are producing the best results in detecting brain functionality by a patient's IC scan images. The scope of improvement could be done in getting the brain boundaries in a more precise way by analyzing each point on the boundary and getting the boundary image. Another such improvement could be done if there is a new machine learning algorithm that deals with the best image data to find the accuracy more than the current system's accuracy. Apart from using template matching for getting the brain slices, advancement can be made in the algorithm itself to break the slices and find the accuracy.

VI. References

[1] https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html

[2] https://learnopencv.com/contour-detection-using-opencv-python-c/

[3] https://www.geeksforgeeks.org/python-pil-image-crop-method/

[4] https://en.wikipedia.org/wiki/Resting_state_fMRI

[5] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

[6] https://www.tensorflow.org/api_docs/python/tf/keras

[7] https://keras.io/about/