BRAIN fMRI CLASSIFIER WITH 3D Convolutional Neural Network

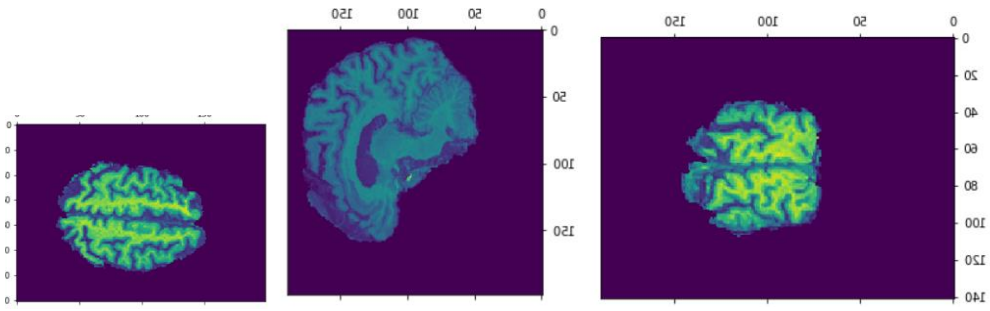
Flyn Sequeira, M.S. Computer Science in University of Texas at Arlington | ID: 1001778678 Email: flyn.sequeira@mavs.uta.edu

ABSTRACT

In this project we test out various methodologies of classifying fMRI (Functional Magnetic Resonance Imaging) brain scan of a patient, or a healthy individual. With the lack of domain knowledge, and meta-data on the dataset, it becomes tricky to be able to come to the right solution. Methodologies such as identifying Region Of Interest(ROI) and performing 3D slicing in those regions, becomes less effective with no meta data, about the disease, and domain knowledge. We investigate a few techniques to perform 2D slicing of the 3D fMRI data, to identify the pitfalls of the techniques, and further, experiment with a 3D Convolutional Neural Network (CNN) to test out the performance.

INTRODUCTION

fMRI dataset fMRI datasets are complicated in various ways, as the datasets usually come in an extremely small amount and the data itself is usually noisy. But they are usually better than CT scans in terms of being less invasive and gives a better 3D map of the brain. In our dataset, the 3D map from different axis, of an individual slides looks like the image given below.



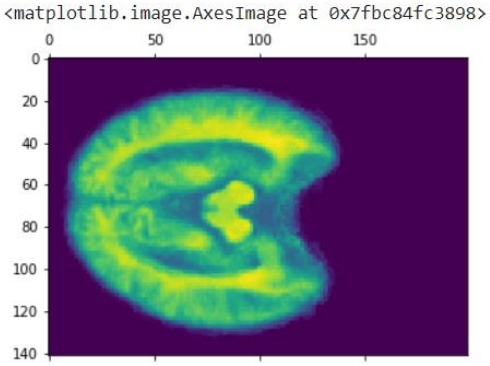
Modern neural network algorithms such as CNN have shown great promise towards building classifiers that can detect the presence of various diseases such as Alzheimer's Disease, by mapping the part of the brain that has a difference in grey matter. CNN has also been used to detect to identify, classify and analyze brain network (1). Most research papers require domain knowledge, to be able to give better results with the CNN models. In this project we try to approach the same problem without having much of a domain knowledge. The biggest problem that comes into play is being able to segment data, identify regions of interest or slice the right frame of the 3D brain scan.

METHODOLOGY

PREPROCESSING

To reduce the size of the image, the entire set of train data was summed up, to give a new image, that was the summation of the rest of the training image. A slice of this resultant matrix looks like the following. This was done to crop the image to the edges, without cropping any essential segments of

other images.



Further the image size was reduced to (80,130,100)

MODEL

Failed Attempt: 2D Convolution Network with Slicing of the 3D image into 2D counter parts

The first approach that was used to classify was to use a simple 2D Convolutional Neural Network, by slicing the 3D image into it's 2D components and classifying the components. On the first run, every slice was trained with the label corresponding to the label of the brain it belongs to. This brings about the first problem. TensorFlow gave a result of +90% consistently while training, but the data was heavily biased and overfitted, resulting in a 50% accuracy during testing phase. The biggest issue was the slices that were not affected by the disease. This were falsely categorized as "Patient" when they were healthy, giving a very random result to the entire dataset.

To resolve this, I approached the problem by taking the Euclidean distance between the aggregated of all the healthy, and aggregated slices of all the patient subjects. Then identified which slices have the biggest difference between them. The problem this gave rise to is the fact that some slices didn't have much difference and some slices had a big difference solely because of the fact that they did not utilize much of the scan region.

3D Convolution Network with Slicing of the 3D image into 2D counter parts

The biggest problem with 3D convolutional neural network was the fact that it requires a lot of GPU resource to run, such that Google Collab would often not be able to allocate every parameters of the network to the GPU, resulting in a Resource Allocation error. As the data is already noisy, it is necessary to make sure a lot of data isn't lost, and the parameters were maintained at around 10 million, and as the result of this, a model looked like the following.

a. **Conv3D** – With Kernal size 5, and filter size of 32.
b. **Batch Normalization**
   Batch Normalization Becomes extremely critical in making sure the network learns, without making the weight matrix values distance themselves far beyond necessity. (2)
c. **Dense Layers** – Two dense layers of 32 and 16 nodes were created, followed by 2 layers to represent "Health" or "Patient"

Model: "sequential_25"

Layer (type)	Output Shape	Param #
conv3d_37 (Conv3D)	(None, 76, 126, 96, 32)	4032
max_pooling3d_37 (MaxPooling	(None, 12, 21, 16, 32)	0
batch_normalization_35 (Batc	(None, 12, 21, 16, 32)	128
flatten_25 (Flatten)	(None, 129024)	0
dense_70 (Dense)	(None, 32)	4128800
dense_71 (Dense)	(None, 16)	528
dense_72 (Dense)	(None, 2)	34

Total params: 4,133,522
Trainable params: 4,133,458
Non-trainable params: 64

TRAINING

Loss Function: Binary Cross Entropy

Optimizer: Adam with Learning Rate of 0.01 with epsilon decay of 0.6 to reduce the learning rate over each steps

Training: The data was trained with an epoch of 20 and batch size of 3 to make sure it's stable but not take away most of the data in each batch.

EVALUATION

During training, the model gave an accuracy of above 0.8 to 1 but fluctuating very frequently. After training, the resulting model was tested against test data to consistently obtain an accuracy score of above 80% and hitting an accuracy score of 100% most times.

CONCLUSION AND FUTURE OPPORTUNITIES

The model was able to classify the dataset easily as the data was already aligned, cleaned, and mostly cropped up in the right way and given. The summed dataset of all the brain images showed that the images almost perfectly aligned with each other. Identifying better ways to preprocess a messy data, would a good approach to solving this problem

Further the slicing technique's issue, regarding the Euclidean distance between the corresponding slices of healthy and patient, can be resolved by finding the average Euclidian distance, after ignoring the pixels that were valued at 0.

As a conclusion, given a clean dataset, and very few data, a 3D Convolutional Neural Network is effectively capable of detecting between brain scans of healthy and unhealthy subjects of an unknown brain disease.

REFERENCES

(1) Brain Network Analysis and Classification Based on Convolutional Neural Network https://www.frontiersin.org/articles/10.3389/fncom.2018.00095/full

(2) Batch Normalization - https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/