**Project 1 Report**

**Name: Chidozie Jeffrey Arukwe**

**MavID: 1001894300**

The project attempts to build a Convolutional Neural Network (CNN) which performs classification task on a dataset containing 10 MRI (Magnetic Resonance Imaging) scans of Alzheimer’s disease patients and 10 MRI scans of normal patients.

This work uses a 3D CNN approach. This is motivated by the fact that dealing with the individual slices independently in 2D CNNs deliberately discards the depth information which results in poor performance for the intended task [1].

This report will show the steps needed to build a 3D convolutional neural network (CNN) to predict the presence of Alzheimer’s disease (AD) in MRI scans. 2D CNNs are commonly used to process RGB images (3 channels). A 3D CNN is simply the 3D equivalent: it takes as input a 3D volume or a sequence of 2D frames (e.g., slices in an MRI scan).

**The dataset**

It contains 10 T1 MRI scans of Alzheimer’s disease patients and 10 T1 MRI scans of normal patients.

The [dataset](https://mavsuta-my.sharepoint.com/:f:/r/personal/xxy1302_mavs_uta_edu/Documents/Courses/6389DataIntroduction/Data_and_Code?csf=1&web=1&e=lw3WC6) can be found:

https://github.com/Jeffreyarukwe/CN/raw/main/CN.zip

https://github.com/Jeffreyarukwe/AD/raw/main/AD.zip

**Objectives**

There are some goals set forth with exploring this dataset, they include:

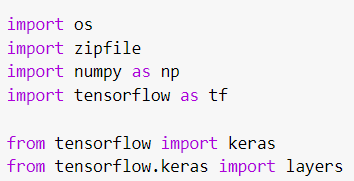
1. Train a CNN model

2. Using the trained model to do the classification, i.e., given new data points, how accurately can the model predict what class it belongs

3. Report some performance metrics

**Setup**

Tensorflow Keras library was used for this project.



**Loading data and preprocessing**

The files are provided in **Nifti** format with the extension .nii. To read the scans, we use the **Nibabel** package. A threshold between 0.0 and 1055409.2 was used to normalize MRI scans.

To process the data, we do the following:

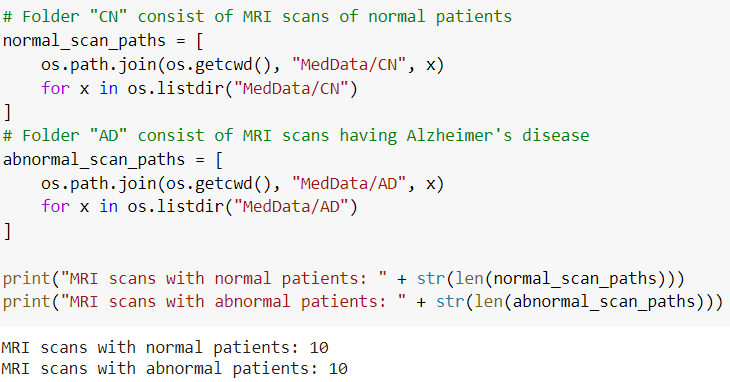
We first rotate the volumes by 90 degrees, so the orientation is fixed

We scale the values to be between 0 and 1.

We resize width, height and depth.

Here we define several helper functions to process the data. These functions will be used when building training and validation datasets.

Then we read the paths of the MRI scans from the class directories.



**Build train and validation datasets**

Read the scans from the class directories and assign labels. Downsample the scans to have shape of 128x128x64. Rescale the raw voxel values to the range 0 to 1. Lastly, split the dataset into train and validation (test) subsets.

Text

Description automatically generated

**Data augmentation**

Data augmentation is a strategy that enables us to significantly increase the diversity of data available for training models, without collecting new data. Data augmentation techniques such as cropping, padding, flipping, rotation, translation, brightness, contrast, color Augmentation, and saturation are commonly used to train large neural networks.

The MRI scans also augmented by rotating at random angles during training. Since the data is stored in rank-3 tensors of shape (samples, height, width, depth), we add a dimension of size 1 at axis 4 to be able to perform 3D convolutions on the data. The new shape is thus (samples, height, width, depth, 1).

Graphical user interface, text

Description automatically generated

While defining the train and validation data loader, the training data is passed through and augmentation function which randomly rotates volume at different angles. Note that both training and validation data are already rescaled to have values between 0 and 1.

Text

Description automatically generated

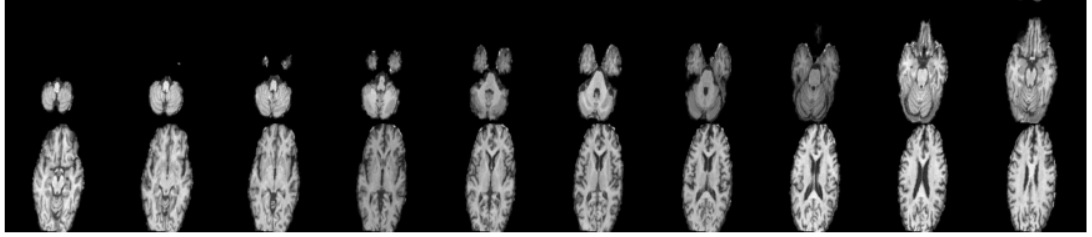
**Visualization**

We can now visualize an augmented MRI scan.

A picture containing text

Description automatically generated

Or Multiple of them:



**Define a 3D convolutional neural network**

To make the model easier to understand, we structure it into blocks. The architecture of the 3D CNN used in this example is based on this paper:

Diagram

Description automatically generated with medium confidence

The design is that of a 17-layer 3D CNN which comprises four 3D convolutional (CONV) layers with two layers consisting of 64 filters followed by 128 and 256 filters all with a kernel size of 3×3×3. Each CONV layer is followed by a max-pooling (MAXPOOL) layer with a stride of 2 and ReLU activation which ends with batch normalization (BN) layer. Essentially, the feature extraction block consists of four CONV-MAXPOOL-BN modules. The final output from the feature extraction block is flattened and passed to a fully connected layer with 512 neurons with a dropout set to 30% and fed to a softmax layer for a binary classification.

Below is a summary of the model.

Table

Description automatically generated

**Train the model**

Then we compile and train the model for 25 Epochs.

I experimented with the different hyperparameters (e.g., The learning rate of 0.0001 was chosen. Batch size = 2)

Graphical user interface, text, application

Description automatically generated

**Visualizing Model Performance**

Here the model accuracy and loss for the training and the validation sets are plotted. Since the validation set is class-balanced, accuracy provides an unbiased representation of the model's performance. The graphs below show the accuracy and model loss for both training and test sets.

Chart

Description automatically generated

**Limitations:**

* I did not have a GPU to train for longer epochs to improve the accuracy performance on out of sample data.
* Future work can train for longer epochs (e.g., 100 epochs) for a better performance.

**Appendix**

Below is the view of the dataset uploaded to their respective folders (Google Colab).

Graphical user interface, text, application, email

Description automatically generated

**Reference**

1. Zunair, H., Rahman, A., Mohammed, N., & Cohen, J. P. (2020, October). Uniformizing techniques to process CT scans with 3D CNNs for tuberculosis prediction. In *International Workshop on PRedictive Intelligence In MEdicine* (pp. 156-168). Springer, Cham.