Capstone Project Proposal

Machine Learning techniques using AWS

Alzheimer MRI Prediction-Kaggle

HananAli Elmogey
Data scientist

For Udacity AWS-ML Nanodegree

Project Overview

Machine learning has a phenomenal range of application in the health sciences. This project proposal will go over the complete pipeline to build a model that can determine the dementia level of an Alzheimer's patient from their MRI image.

This tutorial highlights the prediction of MRI Alzheimer insights through building a CNN Model using Some **Python** libraries trying to get the highest accuracy of prediction.

Dataset Background

We'll be using a https://www.kaggle.com/datasets/uraninjo/augmented-alzheimer-mri-dataset for our tutorial which already has augmented data with 4 labels and not biased that we can go through modeling after just splitting our data.

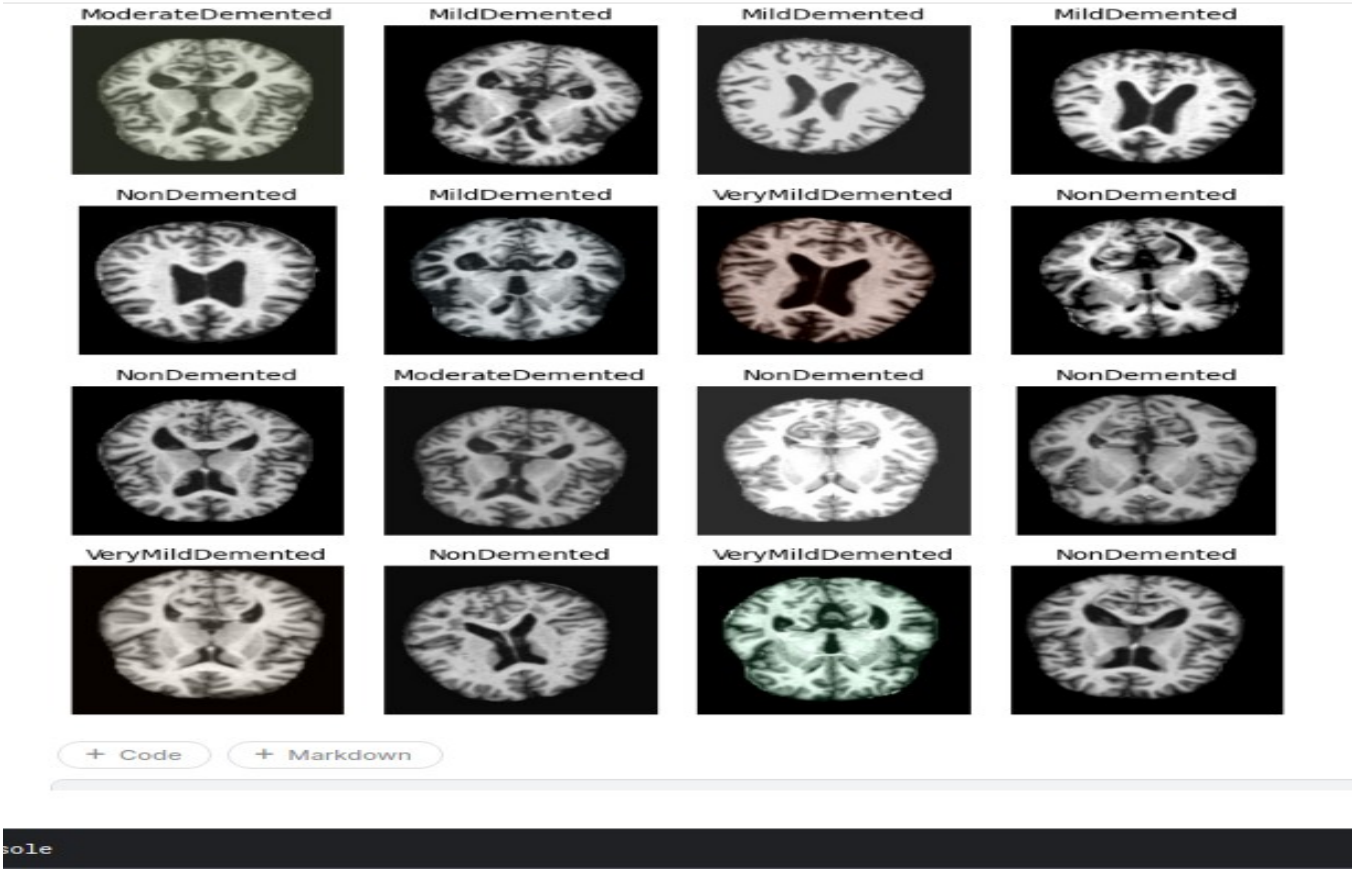
The data consists of four folders which are our classes for the different Alzheimer stages as following :

Class	No.Records
MildDemented	8960
ModerateDemented	6464
NonDemented	9600
VeryMildDemented	8960
Total Records:	**33984**

Problem Statement

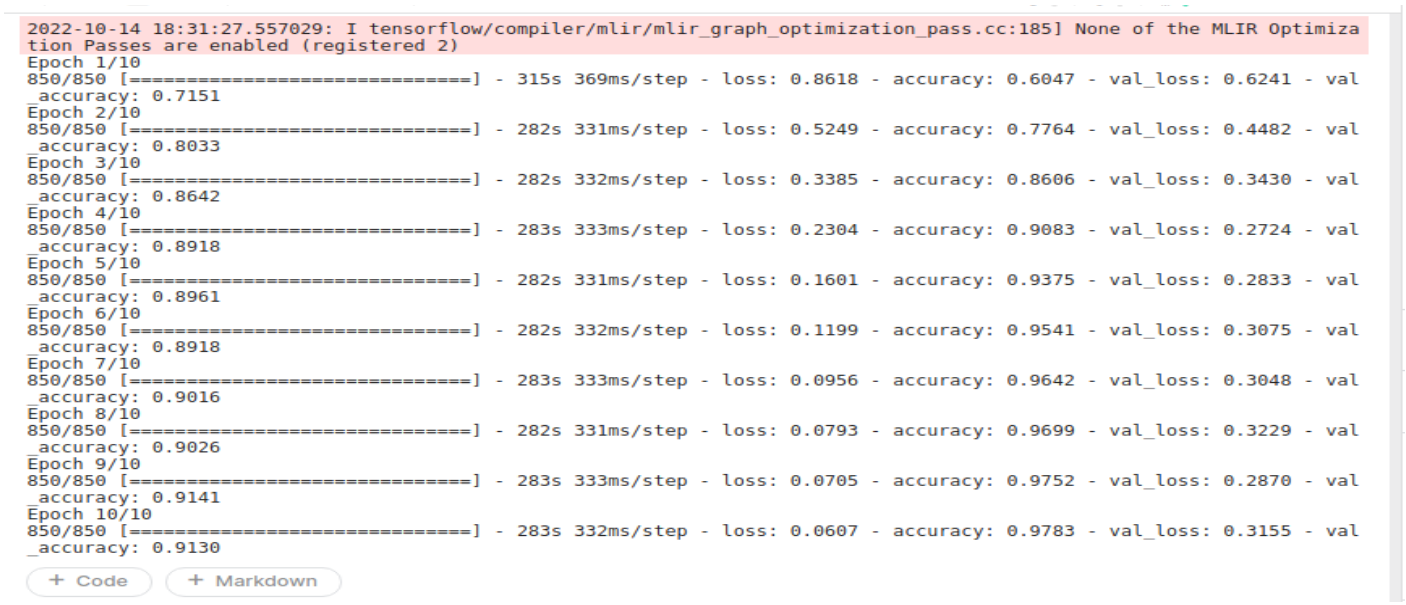
The brain is the most important organ in the human's body, it regulates all processes and when it experience any damage it can lead to losing memory for a short time or even long lasting memory loss "Alzheimer".

The project's objective is to build a CNN model with high accuracy to predict the stages of Alzheimer from the MRI scanned images to help physicians distinguish between different stages easily with high performance to prescribe the suitable medication for cure.



Implementation

My solution Is to train and deploy a ML model to my data after uploading it into S3 bucket, then through AWS SagemakerStudio will be using suitable notebook instances, I might use “ml.m5.2xlarge” and creating the Endpoint calling it through Lambda function to get the best accuracy result for prediction of images.



```
2022-10-14 18:31:27.557029: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimiza
tion Passes are enabled (registered 2)
Epoch 1/10
850/850 [==============================] - 315s 369ms/step - loss: 0.8618 - accuracy: 0.6047 - val_loss: 0.6241 - val
_accuracy: 0.7151
Epoch 2/10
850/850 [==============================] - 282s 331ms/step - loss: 0.5249 - accuracy: 0.7764 - val_loss: 0.4482 - val
_accuracy: 0.8033
Epoch 3/10
850/850 [==============================] - 282s 332ms/step - loss: 0.3385 - accuracy: 0.8606 - val_loss: 0.3430 - val
_accuracy: 0.8642
Epoch 4/10
850/850 [==============================] - 283s 333ms/step - loss: 0.2304 - accuracy: 0.9083 - val_loss: 0.2724 - val
_accuracy: 0.8918
Epoch 5/10
850/850 [==============================] - 282s 331ms/step - loss: 0.1601 - accuracy: 0.9375 - val_loss: 0.2833 - val
_accuracy: 0.8961
Epoch 6/10
850/850 [==============================] - 282s 332ms/step - loss: 0.1199 - accuracy: 0.9541 - val_loss: 0.3075 - val
_accuracy: 0.8918
Epoch 7/10
850/850 [==============================] - 283s 333ms/step - loss: 0.0956 - accuracy: 0.9642 - val_loss: 0.3048 - val
_accuracy: 0.9016
Epoch 8/10
850/850 [==============================] - 282s 331ms/step - loss: 0.0793 - accuracy: 0.9699 - val_loss: 0.3229 - val
_accuracy: 0.9026
Epoch 9/10
850/850 [==============================] - 283s 333ms/step - loss: 0.0705 - accuracy: 0.9752 - val_loss: 0.2870 - val
_accuracy: 0.9141
Epoch 10/10
850/850 [==============================] - 283s 332ms/step - loss: 0.0607 - accuracy: 0.9783 - val_loss: 0.3155 - val
_accuracy: 0.9130
```

+ Code + Markdown

For further information, contact:
Hanan Ali
nana.elmogey@gmail.com