# Capstone Project Proposal

Machine Learning techniques using AWS

HananAli Elmogey
Data scientist

Alzheimer MRI Prediction-Kaggle

For Udacity AWS-ML Nanodegree

## Project Overview

Advancement in technology has resulted in the generation of a prodigious amount of data from everywhere. Due to the increasing amounts of electronic data in the healthcare, life sciences, and bio-science industry, medical doctors and physicians are facing problems in analyzing the data using traditional diagnosing systems. Nevertheless, machine learning and deep learning techniques have aided doctors and experts in detecting deadly diseases in their early stages.

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 which is MRI images in grayscale we will be using them for our tutorial which already has augmented data with 4 labels which I displayed a random samples of each one through the visualization plots to check them out.
The data is not biased and augmented so we can go through modeling after just splitting our data.

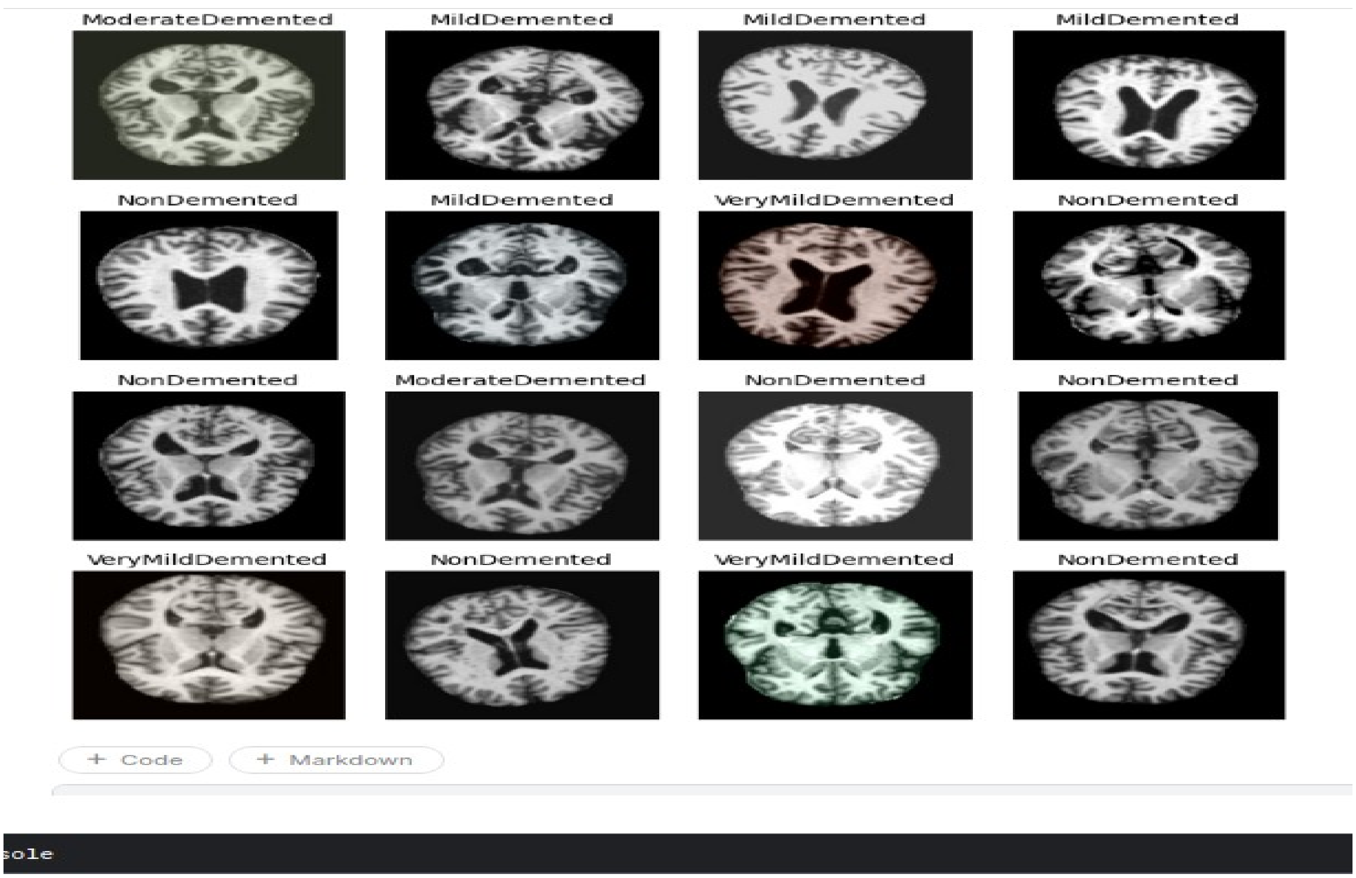
The data consists of four folders which are our classes for the different Alzheimer stages as following :

| 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".
We can use automated techniques on MRI Images for accurate detection of Alzheimer stages

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.



## Benchmark Model:

Many ML algorithms used to distinguish between different Alzheimer disease stages, here is a paper discussing how to use  CNN techniques through extracting the CNN-based and FreeSurfer-based image features to serve that exact purpose, you can get more information checking the following link.
“https://www.frontiersin.org/articles/10.3389/fnins.2018.00777/full”

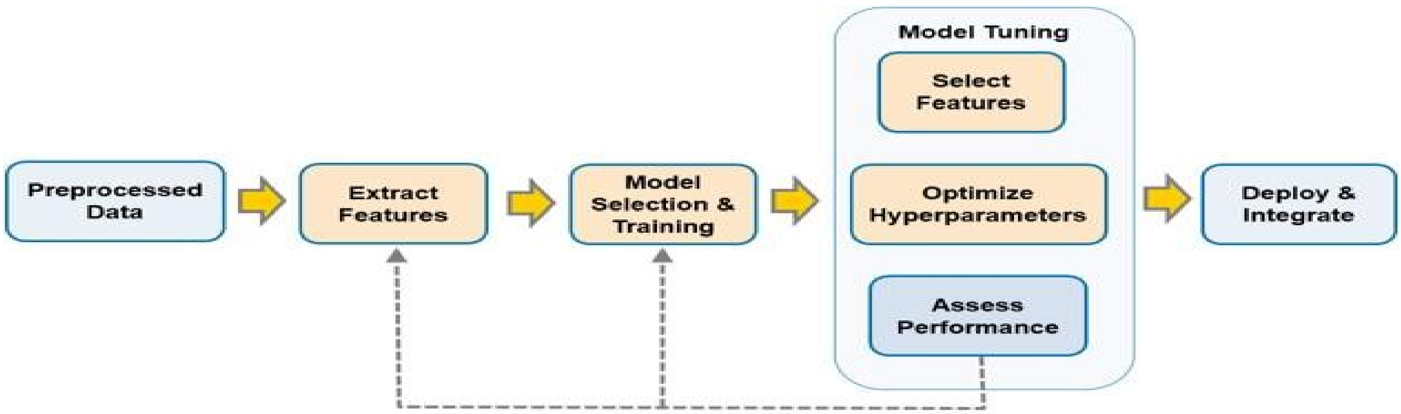
This approach achieved low accuracy at first then increased it’s accuracy to 73.4% when trained with AD/NC patches. This means that the CNN learned more useful information from AD/NC data than that from converters/non-converters data.

## Data Preprocessing:

I’ve created my data loader that will load images from each label and resizing image to (224*224) as my input layer in ResNet50 model used.

## Implementation

My solution Is to take my dataset provided by Kaggle luckily  it’s Augmented, explore it and I’ve already mentioned results of that exploration above, then made my prepossessing to the data, and trained my model with that data according to that work flow:




And hence the Algorithm and platform will be through the following sequence:

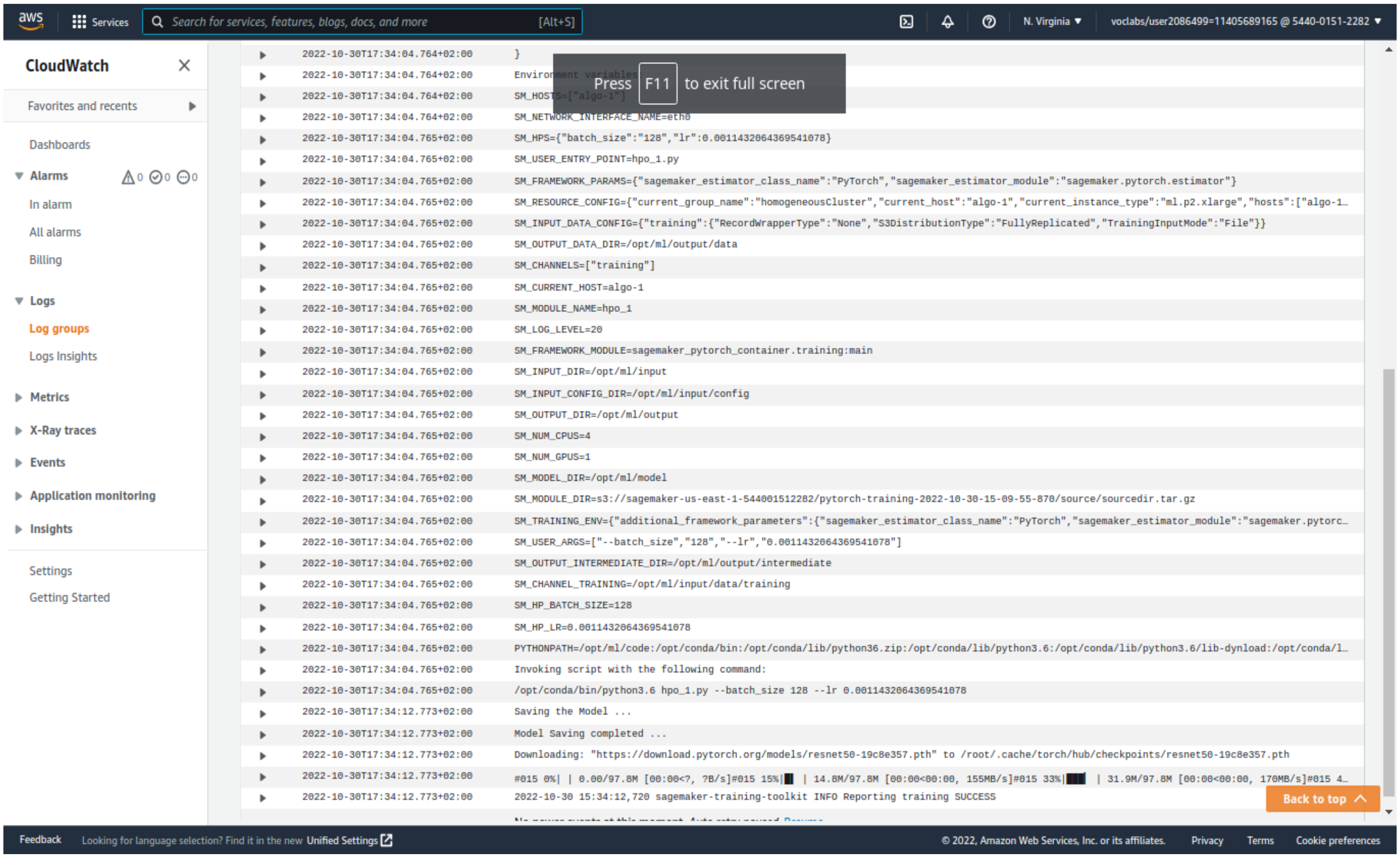
1. The Algorithm will be a Convolution Neural Network (CNN) architecture (using ResNet50 pretrained model in Sagemaker).
2. Deep Learning Framework  is PyTorch.
3. A corresponding Sage Maker instance(type: ml.p2.xlarge) will be created and data will be fed from a storage bucket.
4. The model will also be tuned to find out the best hyper-parameters (Learning rate, Bach size).
5. Will be using Amazon Web Services as my platform
6. And Sage Maker Studio to train, tune and deploy the model.
7. S3 as my Storage Bucket.

## Evaluation & Validation metrics:

As it's obvious we are dealing with a classification problem, so my aim is to get the best accuracy for the model and test it's ability to predict the right stage of Alzheimer from the input MRI image.

In my approach here I have chosen the best Hyperparameters which were:

- Learning rate: "0.00020751076671493 92"
- Bach size: "128"



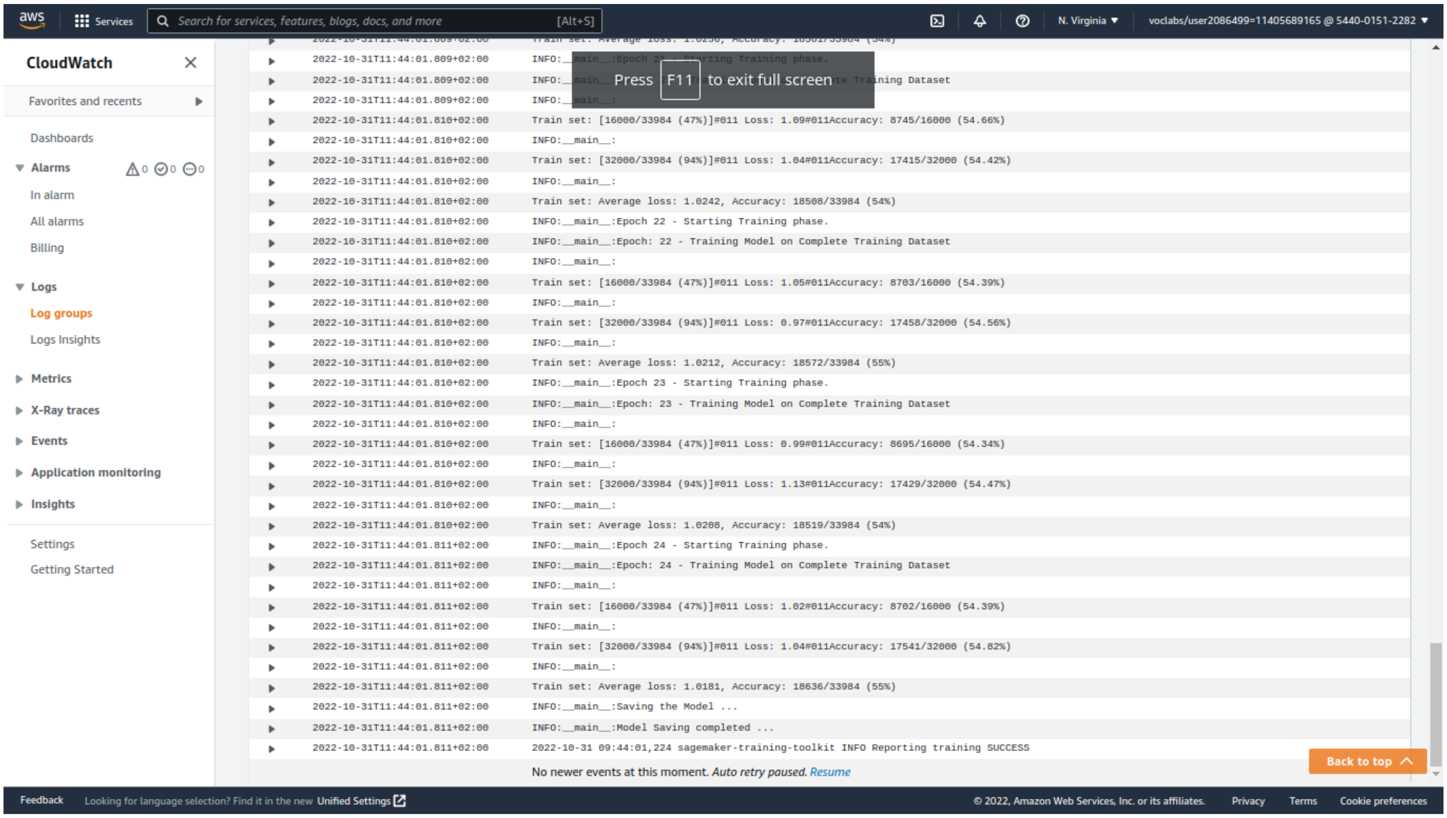

This approach had 55% accuracy in which I suggested some refinement to get a better score. While the "**Benchmark Model"** scored 73.3% according to some parameters as shown in the following figure:

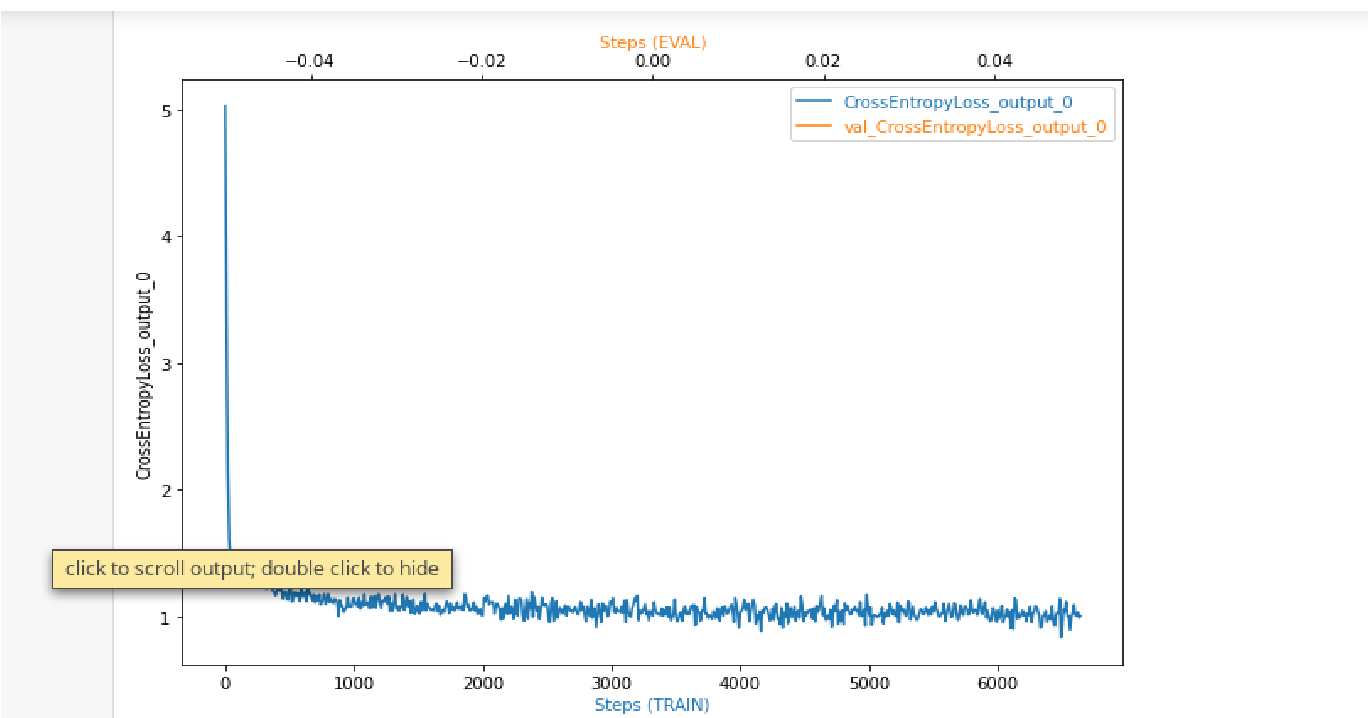
| | Classifying: AD/NC Trained with: AD/NC | Classifying: MCIc/MCInc Trained with: MCIc/MCInc | Classifying: MCIc/MCInc Trained with: AD/NC | Different patch Sampling |
|---|---|---|---|---|
| Accuracy | 88.79% | 68.68% | 73.04% | 72.75% |
| Standard deviation | 0.61% | 1.63% | 1.31% | 1.20% |
| Confidence interval | [0.8862, 0.8897] | [0.6821, 0.6914] | [0.7265, 0.7343] | [0.7252, 0.7299] |

*MCIc means MCI converters. MCInc means MCI non-converters. The results were obtained with 10-fold cross validations, and averaged over 50 runs.*

# Model Training Evaluation metrics:

Training the pre-trained ResNet model ended up with the following results



## Refinement:

In my opinion making some improvements to enhance model accuracy would be as follow:

- Increasing no. of instances instead of 4, may be 10 for example could make my data more accurate.
- Using more Hyperparameters such as "Scheduler" will use learning rates in more flexible ways which could help choosing the best lr for the job, also could increase the range of Bach sizes used (may be adding 512, 64 bach sizes to the original ones used).
- Also using another pretrained sagemaker model would be a good choice.

## Justification:

- Using already augmented data for my job made it easier to work with during the training process.
- Using Sagemaker pre-trained model (ResNet50) was helpful and I think other pre-trained models could be more useful too for a classification task like this.
- Choosing a faster instance type (ml.p2.xlarge) was a great choice it saved a lot of time but I guess using less fast types could save more cost.
- Using a range of Hyperparameters, then choosing the best of them to train with was a privilege.

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