Sardar Vallabhbhai National Institute of Technology
Surat

MRI Classification

Members:

Deval Pandya
Mansi Sampat
Insiyah Hajoori
Sayooj Balakrishnan
Dharmik Bhatt
Ojas Ramwala

Introduction

The aim of this project is to build a deep learning model which should be able to correctly classify the MRI brain scans into the respective four categories of Alzheimer's disease:-

1. Mild-Demented
2. Moderate-Demented
3. Non-Demented
4. Very Mild-Demented

This model could further be used by medical professionals to reduce the time required to determine what level of severity the patient is facing and start the treatment as soon as possible.

Softwares used

- Python 3.7
- Jupyter notebook
- pyTorch (Deep learning library)
- Fast-ai (Deep learning library built on top of pyTorch)

The learning phase

This project requires in-depth knowledge of Neural Networks and Convolutional Neural Networks which comes under the domain of deep learning and thus a major portion of the project timeline was devoted for the research phase wherein we studied various concepts of Machine Learning and Deep Learning most important being:-

1. Linear Regression
2. Logistic Regression
3. Gradient Descent
4. Neural Networks
5. Convolutional Neural Networks

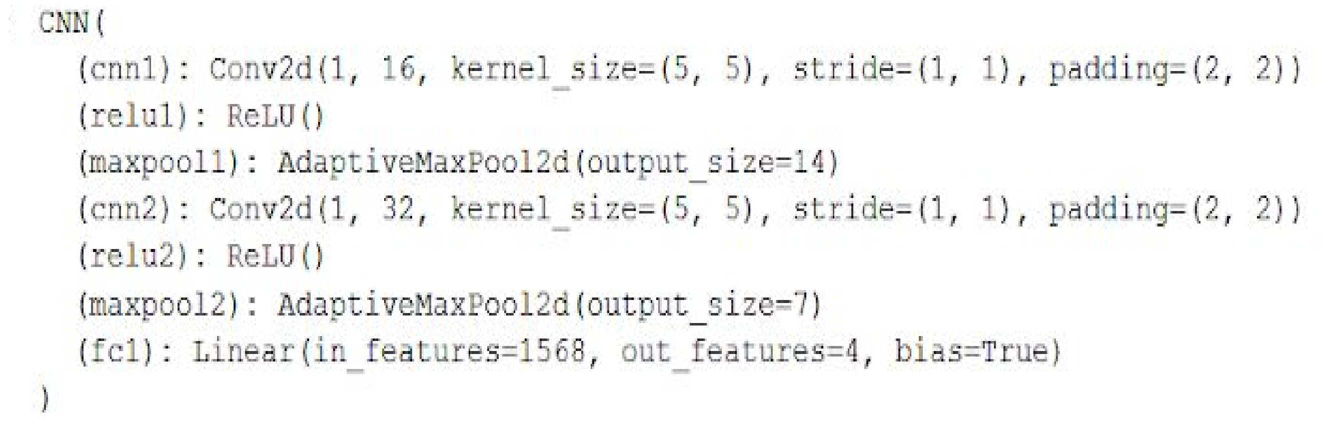
Various tutorials and blogs were skimmed and the following proved to be the most useful:-

1. Introduction to Machine Learning by Andrew Ng (https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN)
2. Introduction to Deep learning by Andrew Ng (https://www.youtube.com/channel/UCcIXc5mJsHVYTZR1maL5l9w)
3. Stanford's CS231n (https://www.youtube.com/playlist?list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk)
4. Deep Learning Wizard (https://www.udemy.com/user/deep-learning-wizard)
5. Fast.ai - Practical Deep learning for coders (https://www.fast.ai)

The implementation phase

The dataset was obtained from OASIS (https://www.oasis-brains.org) which contained around 5000 MRI brain scans of Alzheimer's patients. The extract file was uploaded on drive and the drive was mounted on Colab-Google's online notebook (https://colab.research.google.com). Colab was selected as the platform to code because it allows the user to use libraries like pytorch, tensorflow, keras without installing them on the local machine and also allows free GPU access. The images were extracted in the notebook and as the data was already clean less processing was required.

Once the required processing was done, a deep convolutional network was coded and was allowed to learn the data. As the data was less, chances of overfitting were more and thus less number of epochs were used.
The model which we defined was



```
CNN(
  (cnn1): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (relu1): ReLU()
  (maxpool1): AdaptiveMaxPool2d(output_size=14)
  (cnn2): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (relu2): ReLU()
  (maxpool2): AdaptiveMaxPool2d(output_size=7)
  (fc1): Linear(in_features=1568, out_features=4, bias=True)
)
```

This model could only give accuracy of 54 - 56 % which was way less than expected. This was mainly because of low data availability.

So to overcome this we decided to use transfer learning. **Transfer learning** allows us to use layers of pre-trained models. These models are pre-trained on large datasets and each layer extracts some specific features like edges, lines, colour, etc. from any given input image.

Transfer Learning Models:-

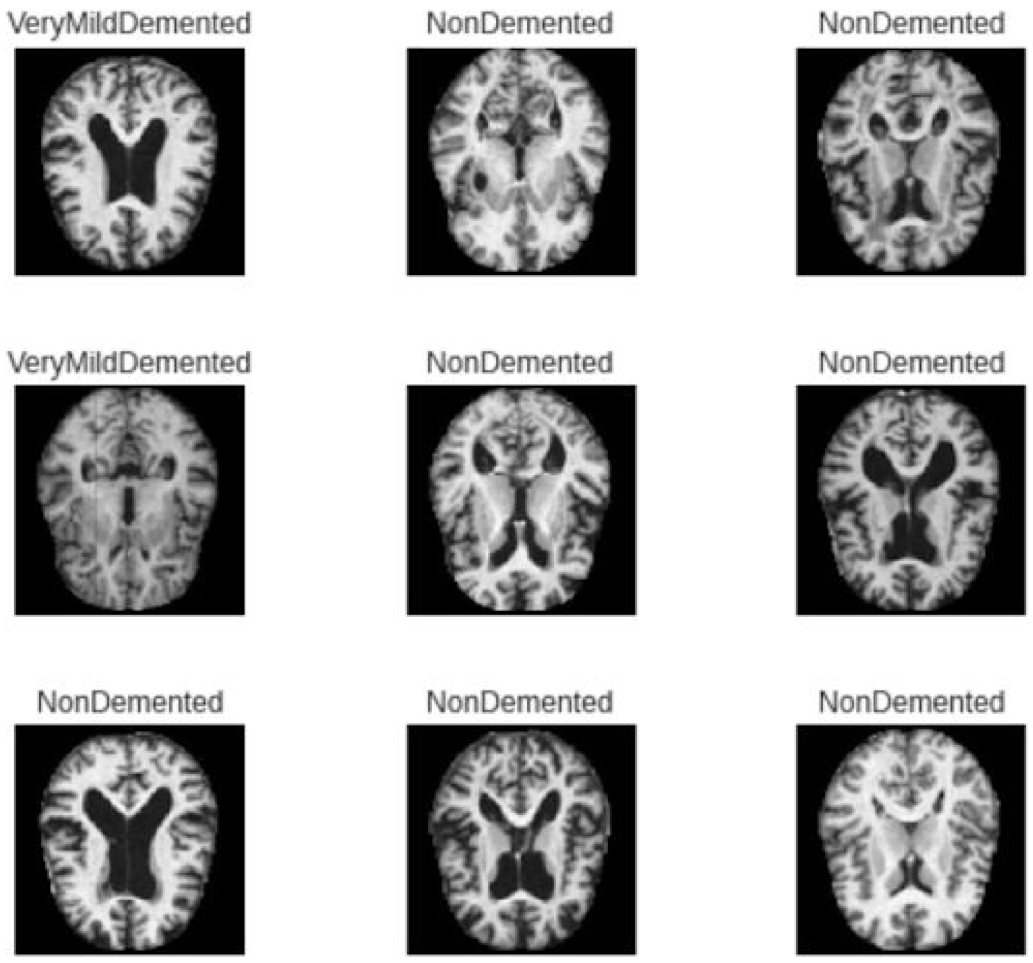
These two major transfer learning scenarios look as follows:

1. Fine Tuning the convnet: Instead of random initialization, we initialize the network with a pretrained network, like the one that is trained on imagenet 1000 dataset. Rest of the training looks as usual.
2. ConvNet as fixed feature extractor: Here, we will freeze the weights for all of the network except that of the final fully connected layer. This last fully connected layer is replaced with a new one with random weights and only this layer is trained.

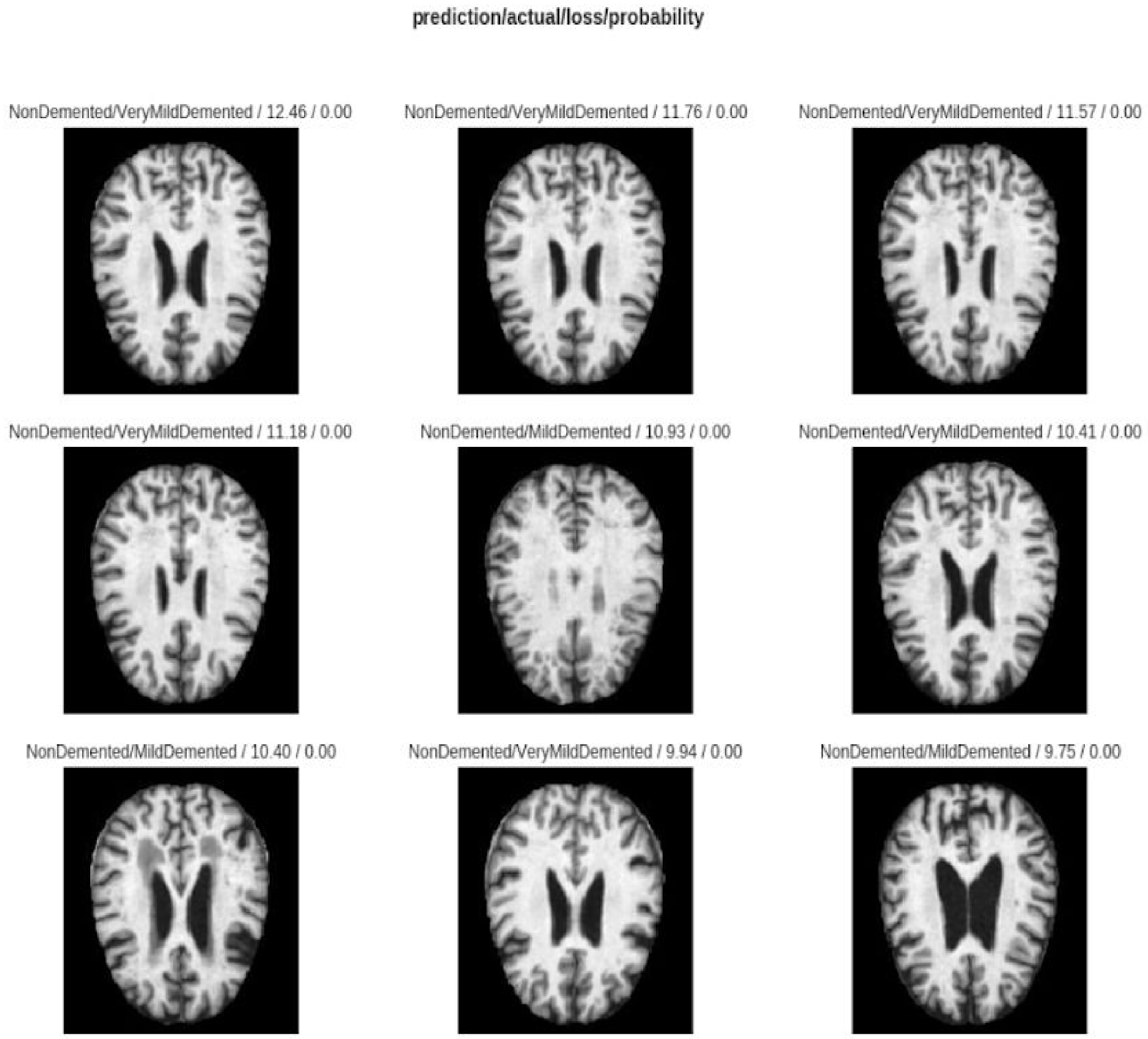
We used 2 pre-trained model just to check if the accuracy of the model changed or not.

- ResNet 18 - Train accuracy - 63.89%
  - Test accuracy - 60.13%
  - Validation accuracy - 61.37 %
- ResNet 34 - This model gave an accuracy of 75.68% which was a significant rise in accuracy when compared to our defined model.
- ResNet 50 - The performance of this model was not up to the mark when compared to that of ResNet 34. It gave an accuracy of 66-67 % accuracy.

Data:



One thing was observed that the models confuses between Non Demented & Very Mild Demented which seems pretty reasonable.


prediction/actual/loss/probability
NonDemented/VeryMildDemented / 12.46 / 0.00
NonDemented/VeryMildDemented / 11.76 / 0.00
NonDemented/VeryMildDemented / 11.57 / 0.00
NonDemented/VeryMildDemented / 11.18 / 0.00
NonDemented/MildDemented / 10.93 / 0.00
NonDemented/VeryMildDemented / 10.41 / 0.00
NonDemented/MildDemented / 10.40 / 0.00
NonDemented/VeryMildDemented / 9.94 / 0.00
NonDemented/MildDemented / 9.75 / 0.00

Problems faced and their solutions:-

- **Installing pyTorch on local system** - pyTorch is a deep learning library and has a lot of dependencies. Installing it on our local machine was a challenge which was faced by many. Normal **pip installation** seemed to give error everytime. Instead **conda installation** was used. Conda takes care of all the dependencies and installs the package properly.
- **Uploading data on colab** - Those who don't have cuda support on their system can use colab. It provides an online GPU access. The only challenge is uploading data. The entire dataset needs to be uploaded on the drive and the drive needs to be mounted in the particular notebook. The larger the dataset more time-consuming this process is.
- **Less data**- The dataset was pretty small for effective classification. The dataset can be increased by using another deep learning approach called Generative Adversarial Networks a.k.a GANs. GANs generates new images when trained on the types of images we need it to generate. Training a GAN is a lengthy process and this procedure will be more time consuming as there are 4 classes whose images need to be generated.

Future aspects:-

Once the dataset is big enough this classification problem can be converted into segmentation problem. Segmentation requires a pretty huge dataset but produces even more specific results eg. It will highlight the region where the targeted disease is present (will be more effective in tumor detection).