Project 1: Binary image classification for MRI Images using CNN

Anirudh Parashar
Email: axp0814@mavs.uta.edu
Course: CSE 6389
UTA ID: 1001720814
Session: Fall 2021

Proect Overview/Introduction:

MRI brain scans use a strong, permanent, static magnetic field to align nuclei in the brain region being studied. Another magnetic field, the gradient field, is then applied to spatially locate different nuclei. Finally, a radiofrequency (RF) pulse is played to kick the nuclei to higher magnetization levels, with the effect now depending on where they are located. When the RF field is removed, the nuclei go back to their original states, and the energy they emit is measured with a coil to recreate the positions of the nuclei. (Source wikipedia)

MRI images are captured using a magnetic resonance device either at hospital or a research facility usually with a magnetic field of 1.5T upto 10T. MRI scans are helpful in diagnosing many brain related diseases and abnormalities, to properly distinguish which type of abnormality is present in a brain we need a method to say if a patient has that particular disease. For this problem the image classification is very helpful as there is no direct and proper way to distinguish and diagnose the subject. In this particular project there is a dataset of normal and Alzheimer's patients. We use CNN to train the model and predict if the person has Alzheimer's or not. A typical CNN consists of a Convolution layer, a pooling layer and a dense layer. For this project we have 3d images of 10 normal and Alzheimer's patients.

2. Problem Definition and Algorithm

2.1 Task definition:

To build a binary classifier to classify normal subjects from Alzheimer's.

The task involves the following:

- Splitting data into training and validation
- Data augmentation
    - -preprocessing
    - -Data downsampling
- Building model
- Training model
- Predict

The model is expected to classify the 3d images as Normal or Alzheimer's.

2.2 Algorithm Definition

The algorithms we use for the 3d image classification is CNN(Convolutional Neural Network)

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix

for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.(Source Wikipedia)

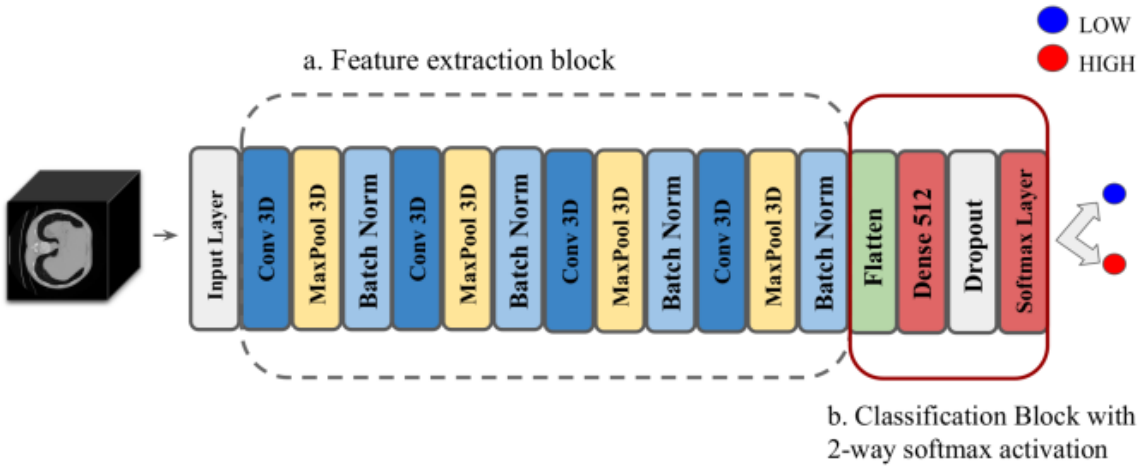


Fig. 2: Our proposed 17-layer 3D convolutional neural network architecture which consist of several modules of 3D conv, maxpool and batch normalization layers.

3. Experimental Evaluation

3.1 Methodology

A binary classifier is implemented through CNN using the Tensorflow library keras. The dataset provided for the project consists of 10 Normal and 10 Alzheimer's subjects. The steps in the task definition are executed to make the prediction.

The dataset dimensions are 166 by 256 by 256, so these are high resolution images, running CNN on these much bigger dimension images leads to resource exhaustion, so we need to downsample the data. The data is downsampled to 64 by 128 by 128, so it effectively is the same data but it can be trained by using Google Colab or local system.

Scanning the data:
The first step is to load the data from the Nifti files, nibabel package is used for this task. Secondly the volume is normalized. The third step is resizing or downsampling the data to the lower dimension as stated above.

Splitting the dataset:
For our model we need the training and validation dataset, so from the given dataset of 20 subjects we split the data into the ratio of 70/30, 70 for training and 30 for validation. 14 for Training and 6 for Validation(equally from each AD/CN)

Data augmentation and preprocessing:
Data loaders are defined using the tensorflow library. Only the training data is augmented for improving accuracy, the data is rotated at random angles and a channel is added. For validation preprocessing a channel is added. Data augmentation is done during training.

Model definition:
For 3D CNN model definition we need to define the CNN layers, the convolution layer, the pooling layer and the dense layer. The Tensorflow library keras is used for defining layers. We do batch normalization at the end of each layer. The activation function “relu” is used for the Convolution layer. For dense layer the activation function “sigmoid” is used.

Training the model:
First the model is compiled, the metrics used for the execution of the model are :

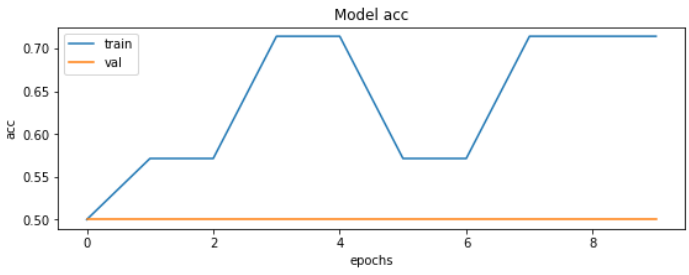
Initial_learning_rate = 0.0001
Decay_steps = 100000
Decay_rate = 0.96
Epochs = 10

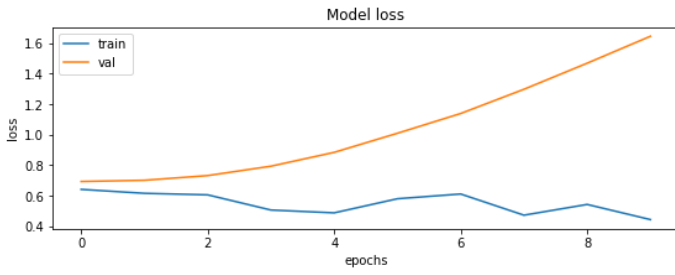
We use the callback for checkpoint at the end of each epoch and save the file as 3d_image_classification.h5

Plotting the training and validation accuracy over epochs:
The data set is plotted using matplotlib for accuracy and loss for the execution.

Making the prediction:
The dataset from the validation is predicted using the built model and the model confidence is calculated.

3.2 Results
For the execution of the CNN model for 10 epochs it took about 30-38 minutes. At the end of each execution the model is predicted and the accuracy and loss plot for the model is give in the figure below for the first execution.





While we see the model accuracy improves over time for training data but not for the validation. The model loss decreases over time for training but increases linearly for validation.

The above execution was for the prediction of alzheimer's patients and the model confidence statistics are as follows:

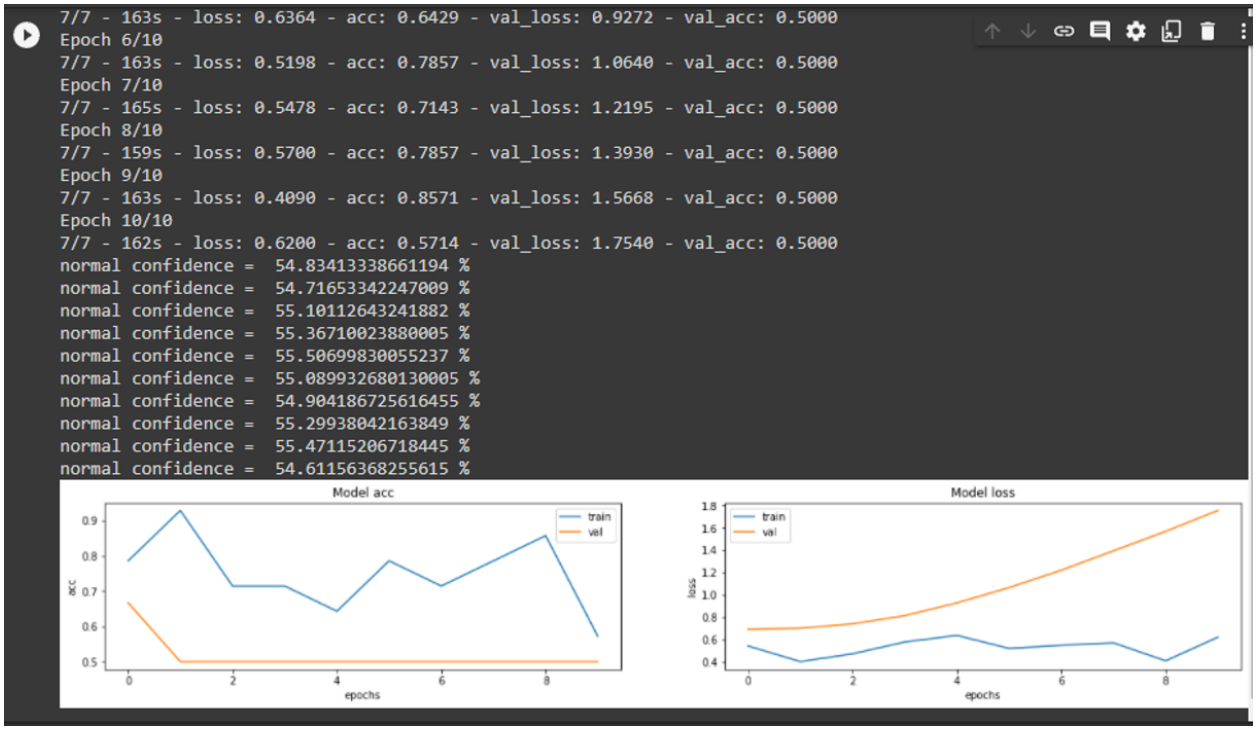
mri scans with normal brain: 10
mri scans with abnormal brain: 10
Number of samples in train and validation are 14 and 6.
Model: "3dcnn"

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 128, 128, 64, 1)] 0
_________________________________________________________________
conv3d (Conv3D)              (None, 126, 126, 62, 64)  1792
_________________________________________________________________
max_pooling3d (MaxPooling3D) (None, 63, 63, 31, 64)    0
_________________________________________________________________
batch_normalization (BatchNo (None, 63, 63, 31, 64)    256
_________________________________________________________________
conv3d_1 (Conv3D)            (None, 61, 61, 29, 64)    110656
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 30, 30, 14, 64)    0
_________________________________________________________________
batch_normalization_1 (Batch (None, 30, 30, 14, 64)    256
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 28, 28, 12, 128)   221312
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 14, 14, 6, 128)    0
_________________________________________________________________
batch_normalization_2 (Batch (None, 14, 14, 6, 128)    512
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 12, 12, 4, 128)    442496
_________________________________________________________________
max_pooling3d_3 (MaxPooling3 (None, 6, 6, 2, 128)      0
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 2, 128)      512
_________________________________________________________________
global_average_pooling3d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 512)               66048
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 844,353
```

Trainable params: 843,585
Non-trainable params: 768

____
Epoch 1/10
7/7 - 168s - loss: 0.6412 - acc: 0.5000 - val_loss: 0.6930 - val_acc: 0.5000
Epoch 2/10
7/7 - 165s - loss: 0.6154 - acc: 0.5714 - val_loss: 0.7010 - val_acc: 0.5000
Epoch 3/10
7/7 - 164s - loss: 0.6054 - acc: 0.5714 - val_loss: 0.7317 - val_acc: 0.5000
Epoch 4/10
7/7 - 166s - loss: 0.5058 - acc: 0.7143 - val_loss: 0.7933 - val_acc: 0.5000
Epoch 5/10
7/7 - 166s - loss: 0.4873 - acc: 0.7143 - val_loss: 0.8841 - val_acc: 0.5000
Epoch 6/10
7/7 - 165s - loss: 0.5799 - acc: 0.5714 - val_loss: 1.0093 - val_acc: 0.5000
Epoch 7/10
7/7 - 166s - loss: 0.6109 - acc: 0.5714 - val_loss: 1.1389 - val_acc: 0.5000
Epoch 8/10
7/7 - 166s - loss: 0.4720 - acc: 0.7143 - val_loss: 1.2978 - val_acc: 0.5000
Epoch 9/10
7/7 - 166s - loss: 0.5425 - acc: 0.7143 - val_loss: 1.4689 - val_acc: 0.5000
Epoch 10/10
7/7 - 166s - loss: 0.4431 - acc: 0.7143 - val_loss: 1.6456 - val_acc: 0.5000
abnormal confidence = 51.859575510025024 %
abnormal confidence = 51.85898542404175 %
abnormal confidence = 51.65600776672363 %
abnormal confidence = 51.548534631729126 %
abnormal confidence = 51.49175524711609 %
abnormal confidence = 51.651179790496826 %
abnormal confidence = 51.819366216659546 %
abnormal confidence = 51.490265130996704 %
abnormal confidence = 51.52765512466431 %
abnormal confidence = 51.88692808151245 %

For another batch of execution the output is as follows:



Analysis of results:

The model built is only able to classify the subject by 1 to 5 % i.e., variability is very less. It is so because there is not enough data to train the model. With only 10 normal and 10 Alzheimer dataset the classifier is only able to do the task by a minute percentage. While the training accuracy increased but the validation accuracy did not.

3.3 Discussion

As stated above and according to the prediction we can say the data is needed to be much more to be able to clearly classify the normal images from the Alzheimer ones. The model has been trained on downsampled data due to limitation of CPU/GPU, the accuracy of the model can be improved by training on full resolution and a larger dataset.

4. Related Work

3D image classification is a very useful field and can also be used for action recognition, action detection and video segmentation.

5. Future Work

Future work would include gathering more datasets and making the training accuracy better. The classification has been used for grayscale images but it can also be used for RGB images, in real world the data from all fields such as neuroscience, art, music, graphics can be used to make inferences on the top of the 3D CNN classifier.

6. Conclusion

Even with augmenting the data and preprocessing the variability of a maximum of 5% is achieved. The classifier works as expected but is not properly trained with enough data. This can be mitigated by feeding more data to the model.

https://colab.research.google.com/drive/1WWj81yFMUYBYjFdQWfUj9qJhkQVMza2R?usp=sharing