CS 412 INTRODUCTION TO MACHINE LEARNING

Detection of Alzheimer’s using Deep Learning

Project Report

TEAM:

Kalyan Kumar Paladugula
(kpalad4@uic.edu)

Sudha Anusha Sagi
(ssagi2@uic.edu)

Vignesh Narayanaswamy
(vnaray21@uic.edu)

Table of Contents

1. ABSTRACT

Alzheimer's disease, the most common form of dementia, is an irreversible, progressive brain disorder associated with permanent loss of memory and cognitive functioning. Mild Cognitive Impairment (MCI), another form of dementia, can be an early sign of Alzheimer's. Diagnosing MCI using neuroimages is especially hard due to its similarity to AD. In this project, we explore different Deep Learning models that can clearly distinguish between patients who may be normal or have AD or MCI.

2. INTRODUCTION

2.1 About Alzheimer's:

Alzheimer's is the sixth-leading cause of death in the United States[1]. It is the only top-10 cause of death in the United States with no known cure. It is estimated that Alzheimer's and other forms of dementia will cost $305 billion in 2020 and the expenses are projected to reach $1.1 trillion by 2050. Currently, more than 5 million people in the US have Alzheimer's and this number would reach 14 million by 2050[1]. Someone in the US develops Alzheimer's every 65 seconds[2]. By 2050 this is projected to go down to 33 seconds[2]. Alzheimer's is not just a disease of old age: 200,000 people under age 65 have early-onset Alzheimer's disease. It has a very high mortality rate: 1 in 3 seniors dies due to Alzheimer's or another form of dementia. Although the mortality rates of the other major diseases decreased significantly, the deaths from Alzheimer's disease have increased significantly. Between 2000 and 2018, the number of recorded deaths from Alzheimer's increased by 246%, while the number of deaths from the number one cause of death (heart disease) decreased by 7.8%[1].

AD is characterized by memory impairment, language dysfunction, and impairment of recognition, leading to AD patients having difficulty managing themselves with neurofibrillary tangles[4]. Though some treatments may temporarily suppress the symptoms of AD, there is still no powerful evidence that can tell the reason or stop its progression[5]. As their condition gets worse, patients become more and more dependent on others. This also puts a heavy psychological and economic burden on caregivers.

Since MCI could be an early sign of AD, extensive research is being carried out to develop techniques for early detection at pre-symptomatic stages to slow or prevent the progression to AD. Advanced neuroimaging techniques, such as magnetic resonance imaging (MRI), positron emission tomography (PET), Diffusion Tensor Imaging (DTI), Cerebrospinal Fluid Flow (CSF) test, have been developed to identify structural and molecular biomarkers[6]. But their high dimensionality and multi-modality make these techniques difficult to analyze for humans[6]. Hence, machine learning approaches have attracted significant attention in the field of high dimensional image analysis[6]. Pattern analysis methods, such as linear discriminant analysis (LDA), Principal Component Analysis (PCA), logistic regression (LR), support vector machine (SVM), and support vector machine recursive feature elimination (SVM-RFE), have been used for early detection of AD and the prediction of AD progression[6].

However, Machine Learning algorithms such as ones mentioned above require data preprocessing and feature extraction prior to application. Since those processes can be highly time-consuming, another family of machine learning called Deep Learning has become prominent due to their automated nature. These deep learning techniques extract the biomarkers (most important features) on their own for classification.

3. LITERATURE REVIEW

3.1 Related Work:

We divided the methods used by the researchers into two types: Pure Deep Learning, and Hybrid Deep Learning. In Pure Deep Learning, deep learning was used for both feature extraction and classification whereas in the Hybrid one, traditional machine learning models, like SVM, Logistic Regression, and Multi-Kernel SVM, were used for classification. The former approach gave better results when used on large datasets and while the latter performed better for small datasets.

Some researchers converted the multi-label classification problem into a binary classification problem and used transfer learning later[7]. They first built the models on only two classes: AD and NC. Then they added a new class MCI and fine-tuned the weights to accommodate this. This is possible because they used cross-entropy loss which could be extended to multi-class cases[7]. They have shown that in case of limited training data, the models with the transfer learning strategy yielded better results than the ones that classify the input into three classes from scratch. As MCI is the intermediate stage between the cognitive decline of normal aging and the more pronounced decline of AD, first learning to separate the AD and the NC then adding the third class and fine-tuning the network transfers the learned knowledge to classify the middle condition, not jeopardizing the performance of AD Diagnosis[7].

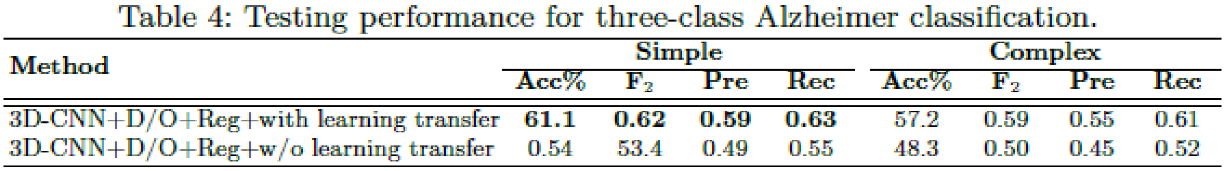
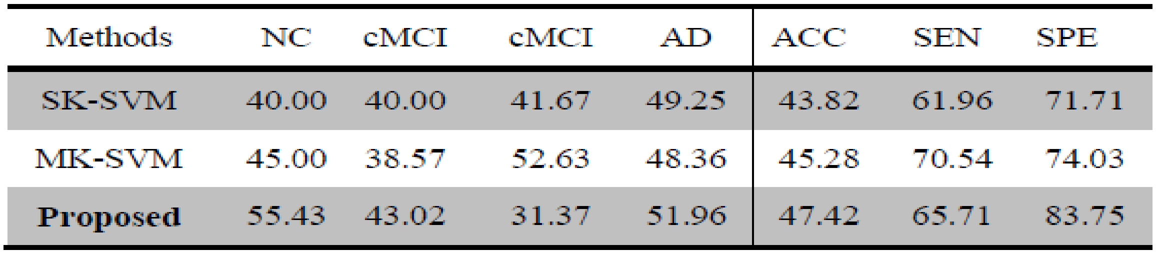


Table 4: Testing performance for three-class Alzheimer classification.

Method	Simple				Complex			
	Acc%	F_2	Pre	Rec	Acc%	F_2	Pre	Rec
3D-CNN+D/O+Reg+with learning transfer	**61.1**	**0.62**	**0.59**	**0.63**	57.2	0.59	0.55	0.61
3D-CNN+D/O+Reg+w/o learning transfer	0.54	53.4	0.49	0.55	48.3	0.50	0.45	0.52

[7]

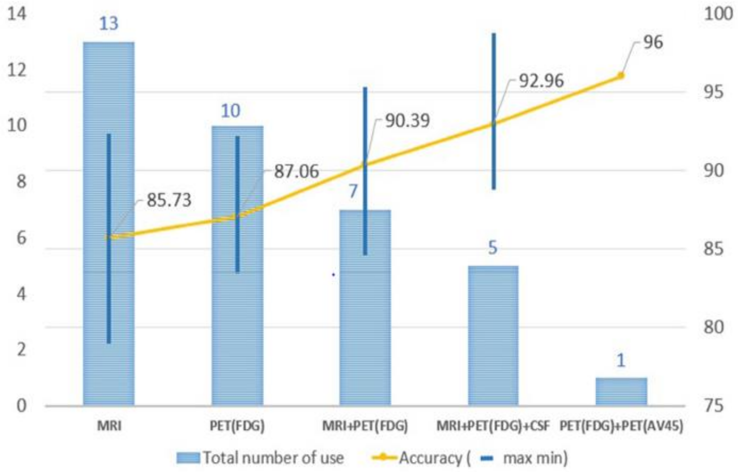
And some further split the MCI class into two: MCI non-converters (ncMCI) or MCI converters (cMCI) based on the risk of progression to AD. In [8], they converted the three-class problem to four-class problem (NC, ncMCI, cMCI, and AD) and applied Pure deep learning with stacked sparse auto-encoders and a softmax regression layer. But they got low accuracy.



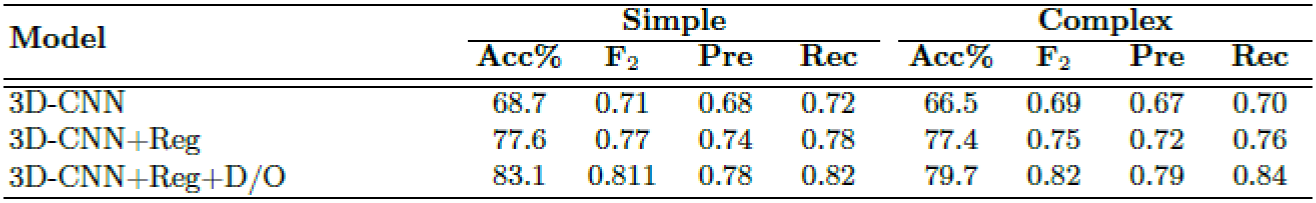
Methods	NC	cMCI	cMCI	AD	ACC	SEN	SPE
SK-SVM	40.00	40.00	41.67	49.25	43.82	61.96	71.71
MK-SVM	45.00	38.57	52.63	48.36	45.28	70.54	74.03
Proposed	55.43	43.02	31.37	51.96	47.42	65.71	83.75

8

Few researchers combined the neuroimaging techniques (MRI+PET, MRI+CSF, MRI+PET+CSF) to improve the accuracy and F1 scores. In [8], they extracted the grey matter volumes from MRI and CMRGlc patterns from PET. And in [10], they used the fusion of MRI, PET, and CSF.
Two or more multimodal neuroimaging data types (MRI+PET, MRI+PET+CSF) resulted in better accuracy[6].



Some researchers used the drop-out technique to avoid the “weight co-adaptation” problem which is a typical cause of overfitting in Deep Learning[9]. It is possible that only a few weights are involved in giving you the desired output. But in this scenario, you will also be looking at only part of the features in the input. This is called "overfitting" and is a result of co-adaptation. Your network has adapted to the dataset. Neural Networks show a gradual decay in performance rather than a complete failure when some of the neurons are disabled (or killed in the case of biological neurons). Basically, a subset of the connections is enough to give the desired output by the network. The dropout technique improved the accuracy of the models[7].



Model	Simple				Complex			
	Acc%	F_2	Pre	Rec	Acc%	F_2	Pre	Rec
3D-CNN	68.7	0.71	0.68	0.72	66.5	0.69	0.67	0.70
3D-CNN+Reg	77.6	0.77	0.74	0.78	77.4	0.75	0.72	0.76
3D-CNN+Reg+D/O	83.1	0.811	0.78	0.82	79.7	0.82	0.79	0.84

7

Some researchers used F1-Score instead of accuracy to improve the recall of AD and MCI classes. As it is difficult to differentiate AD and MCI classes, even if we get high accuracy, we might not get good accuracy on AD and MCI classes (Ex: 84% accuracy - 99% on NC, 70% on MCI, and 80% on AD). If the F1-score is used to pick the best model, we would get better recall scores on all the three classes.

4. DATASET

The dataset is an early version of the Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset floating around the internet. It consists of neuroimages of several subjects taken over a period of time. Preprocessing techniques like skull-stripping and image clipping have been applied which makes feeding them to Deep Learning models easier. Our processed dataset comprises of the following

1. 3D MRI images in the form of npy files (NumPy Arrays)
2. Demographics in the form of a csv file

The whole dataset has a total of 3013 rows (3D scan + demographics information) with some corresponding to subjects scanned multiple times. These 3013 rows are again divided into training, validation, and test datasets. The training dataset has 2109 rows, the validation dataset has 435 rows while the test dataset has 469 rows of data.

Dataset	Number of data points
Training	2109
Validation	435
Test	469

Table: Dataset division for data modeling and testing

5. Exploratory Data Analysis

5.1 Data Analysis – Demographics data:

The demographics dataset taken has five columns: train_valid_test, Age at Scan, data at scan, Sex, and diagnosis. The diagnosis column has three classes: CN (Cognitively Normal), MCI (Mild-Cognitive Impairment), and AD (Alzheimer's disease).

We performed Exploratory Data Analysis (EDA) on our training dataset to get some intuition about how it is structured which gave us some useful insights when designing our models.
Below is a boxplot to visualize how subjects of different age groups are distributed among the three classes. We see the distributions are similar in all classes indicating age might not be a strong feature for our purpose.

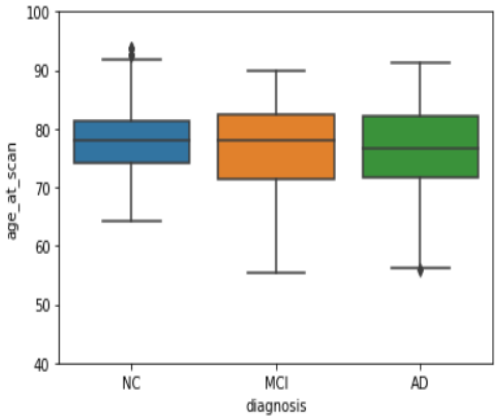


Figure 5.1: Box Plot of distribution of age with respect to the Labels in the Training dataset

We also investigated our training and validation datasets to know the population distribution for each class. Below are the results.

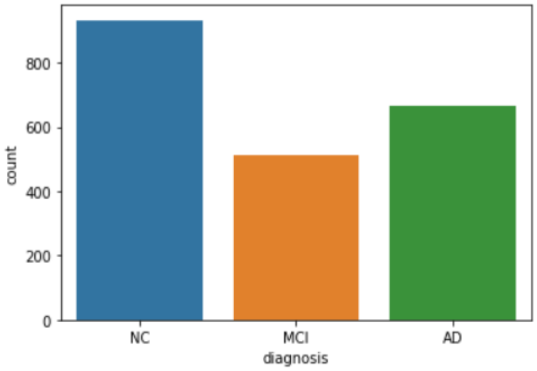


Figure 5.2 Distribution of the 3 class labels among the Training and Validation sets.

The classes do not seem to be balanced; there are more observations that correspond to the class Normal Control(NC). So, we will choose the best model that has a higher F1 Score.

We also performed some preliminary analysis to find out whether gender has an impact on diagnosis, we found the following observations:

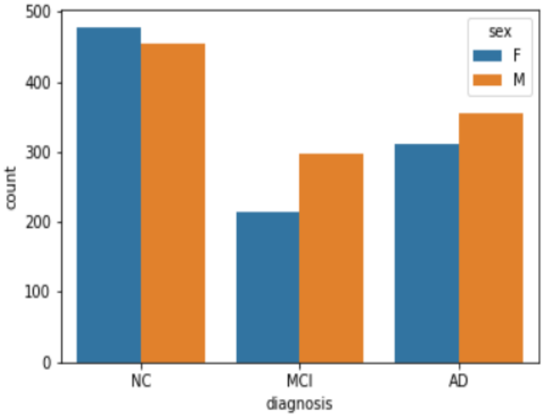


Figure 5.3: Distribution of 3 class labels among the Training dataset with respect to gender.

In case of observations with NC class, there is not much difference in the male and female ratio but for MCI and AD classes there is a noticeable difference.

Our data analysis on demographics has shown that gender is an important feature for classification and age is not that important.

5.2 Data Analysis – MRI images: The dataset has a total of 3013 data points which includes all the subjects: CN(Clinically Normal), MCI(Mild Cognitive impairment), and AD(Alzheimer's disease). The MRI images are 3D images and these MRI images are represented in the form of 2D slices. So, each subject has 62 2D brain slices corresponding to one MRI image. Consecutive 62 images are considered as one 3D image.

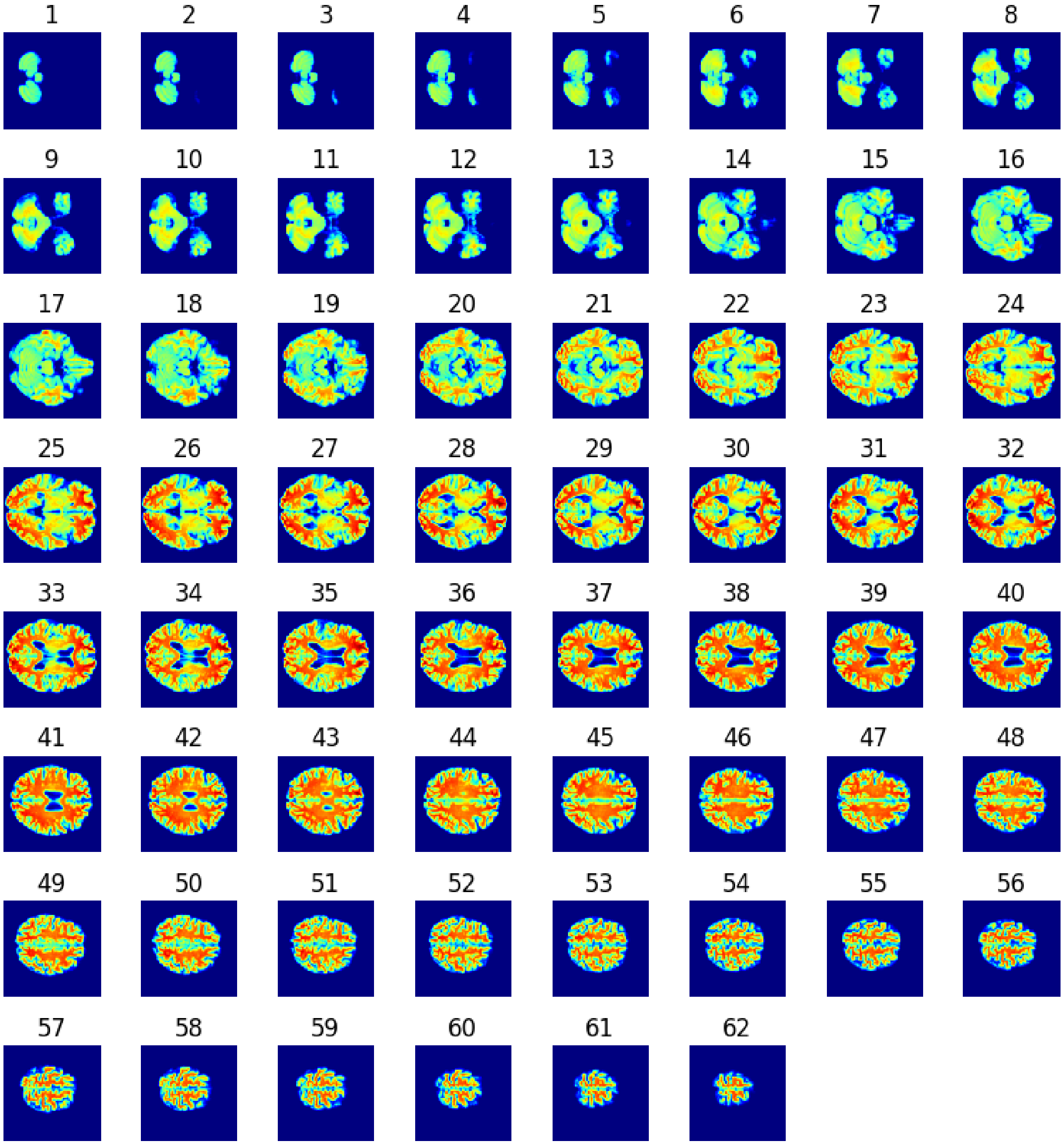


Figure 5.4: 2D slices of a 3D MRI images. Each slice represents a section of the brain image cut horizontally[11].

So, similarly, every set of 62 elements of npy files are mapped to the respective subject in the same order. From figure 5.4 we can identify that each 2D slice is a part of 3D MRI image which is sectioned horizontally. This way it provides a clear view of every part in the image which makes it easier to feed the data to a neural network. The 3D projection of same MRI 2D slices for the same subject[0] can be viewed as below.

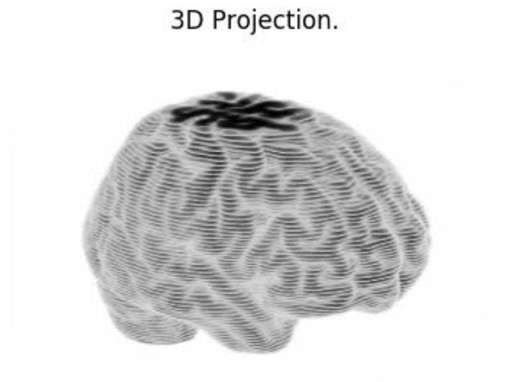


Figure 5.5: 3D projection of 2D slices

6. Data Manipulation

During the initial phases of training, we noticed, reading data from .npy files and storing them in separate variables took a very long time and often led to OutOfMemory exceptions. To avoid this issue, we combined each 3D image and corresponding demographics information to create a table like file objects (Tensorflow Records) with ".tfrecords" extension. Compressing these files further led to faster and a more streamlined training process with significant runtime reduction.

7. DEEP LEARNING MODELS

There are several pre-trained models including InceptionNets, ResNets, and VGG models that perform very well for specific datasets. However, two major limitations of such models did not allow us to utilize them for our own project.

1. They are usually pre-trained on some common image datasets like ImageNet and not on medical neuroimages.
2. They are constrained to 2D images rather than 3D

We also tested several recent 3D model architectures such as VoxNets and PointCNNs. These are state-of-the-art models for object recognition from 3D point clouds. But they were less efficient in performing our specific classification task which is more complicated and nuanced than simple object recognition.

This pushed us to design our own novel architectures that performed surprisingly well despite their simplicity relative to the models discussed above. In the forthcoming subsections, we discuss our models in greater detail.

Before we delve deep into the architecture of our models, we give brief information about some core concepts and building blocks of our models.

7.1 About 3D Convolutional Neural Network:

A 3D CNN[10] remains regardless of what we say a CNN that is very much similar to 2D CNN. In a 3D CNN, the kernels move through three dimensions of data (height, length, and depth) and produce 3D activation maps.

The 3D convolutions are the generalization of the 2D convolution. Here in 3D convolution, the filter depth is smaller than the input layer depth (kernel size < channel size). As a result, the 3D filter can move in all 3-direction (height, width, breadth of the image). At each position, the element-wise multiplication and addition provide one number. Since the filter slides through a 3D space, the output numbers are arranged in a 3D space as well. The output is then 3D data.

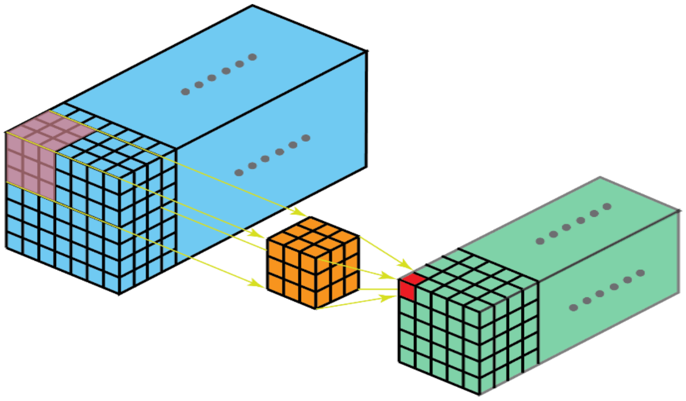


Figure 7.1: In 3D convolution, a 3D filter can move in all 3-direction (height, width, channel of the image). At each position, the element-wise multiplication and addition provide one number. Since the filter slides through a 3D space, the output numbers are arranged i*n a 3D space as well. The output is then a 3D data.*

Similar to 2D convolutions that encode spatial relationships of objects in a 2D domain, 3D convolutions can describe the spatial relationships of objects in the 3D space. Such a 3D relationship is important for some applications, such as in 3D segmentations/reconstructions of biomedical imagining, e.g. CT and MRI where objects meander around in the 3D space.

3D CNN varies from 2D CNN as follows:1. 3D Convolutional Layers:

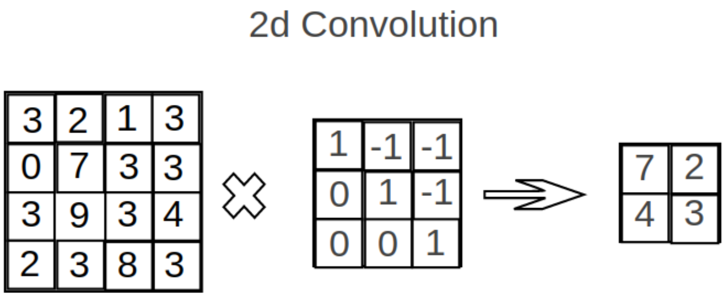


Figure 7.2 2D Convolution

Originally a 2d Convolution Layer[10] is an inner product (entry per entry multiplication) of the input and the filters, where filters and inputs are 2d matrices. In a 3d Convolution Layer, the same operations are used. We do these operations on multiple pairs of 2d matrices. Padding options and slides step options work the same way.

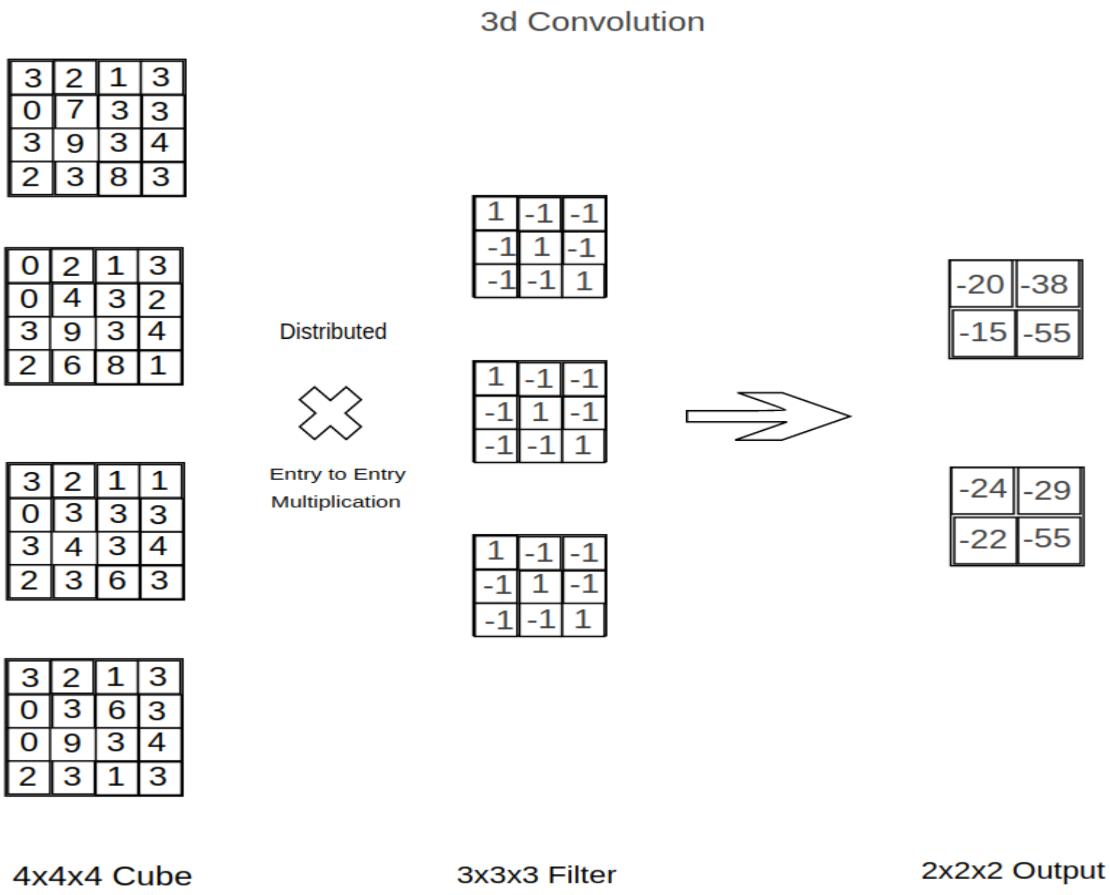


Figure 7.3 3D Convolution

7.2 Building blocks of our Architecture:

Input block:

Accepts input of dimension 62x96x96x1 and feeds it to convolutional blocks.

3D Convolutional blocks:

Contain ‘n’ filters of size 3x3 that perform three-dimensional convolutions with a stride of 2 on the input. The value for ‘n’ varies with each model. Having convolutions with a stride of 2 allowed us to not include pooling blocks like MaxPool and AveragePool that are not capable of preserving details in the data.

BatchNormalization:

Used in conjunction with convolutional blocks to regularize outputs and reduce the need for explicit Dropouts.

Activations:

For all our convolutional blocks as well as Dense layers, LeakyReLU activations are used. For deep networks, it is paramount that gradients flow to each and every layer so the model trains properly. LeakyReLU activations have this property and can significantly reduce the risk of vanishing and exploding gradients. For our output, we have used Softmax activation as it more closely resembles a probability distribution.

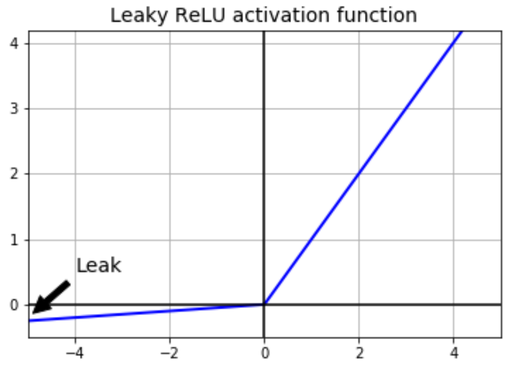


Figure 7.4 Visualization of LeakyReLU activation

Flatten Blocks:

Converts the output of convolutional blocks to a vector so it can be fed to a fully connected layer

Dense blocks:

Capture details from the output of convolutional layers and introduce further non-linearity.

Dropout layers:

Used with Dense layers to “dropout” a fraction of the outputs to reduce the chances of overfitting.

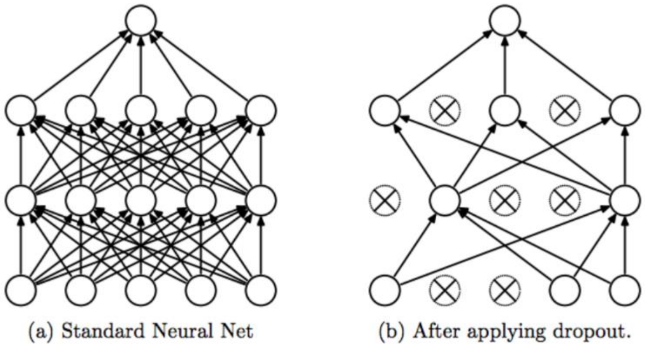


Figure 7.5 Visualization of Dropout

Output block:

Outputs a 1x3 vector specifying the probability that given input belongs to each of the three classes.

7.3 3D Deep Convolutional Neural Network (Baseline):

As a baseline to aid performance comparisons we created a straight-forward deep convolutional network capable of operating on three-dimensional images with multiple image channels. Visualization of the model architecture is given in figure 7.6.

We can observe that the architecture starts with 62 x 96 x 96 x 1 input layer nodes and reduces to 30 x 30 x 47 x 8 nodes after its first 3D convolutional layer and normalization. Similarly, it reduces to 6 x 11 x 11 x 8 after 3rd batch normalization. Finally, after some more intermediate layers, the final output we get is in the form of 3 nodes corresponding to CN, MCI, and AD.

For more details about the architecture, please go through Figure 6.5 which shows a detailed design of the layers and nodes.

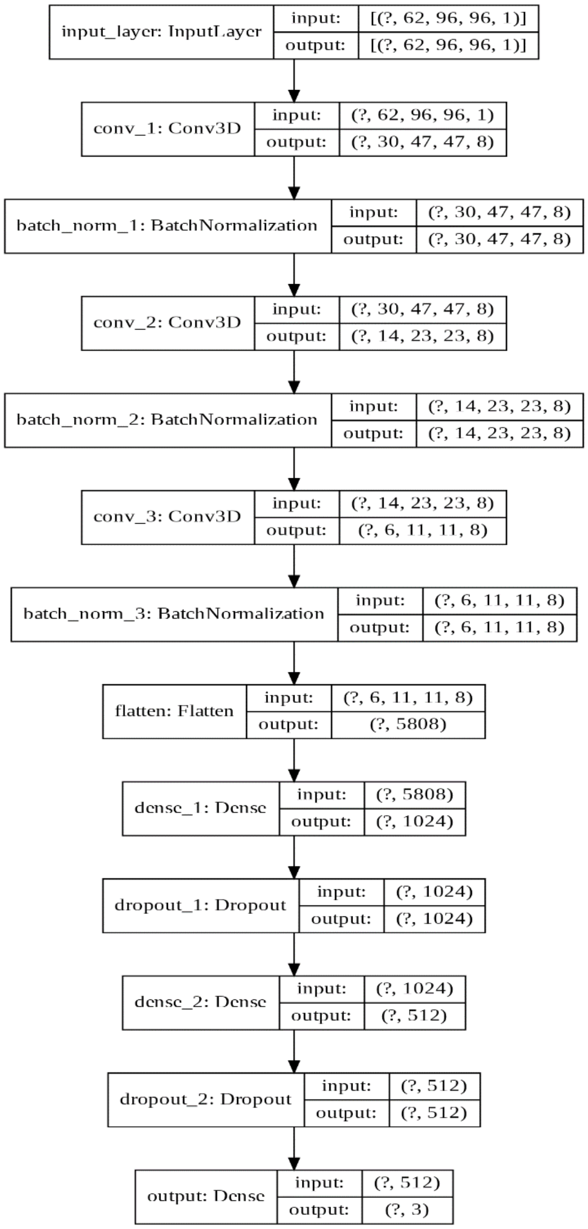

input_layer: InputLayer
input: [(?, 62, 96, 96, 1)]
output: [(?, 62, 96, 96, 1)]
conv_1: Conv3D
input: (?, 62, 96, 96, 1)
output: (?, 30, 47, 47, 8)
batch_norm_1: BatchNormalization
input: (?, 30, 47, 47, 8)
output: (?, 30, 47, 47, 8)
conv_2: Conv3D
input: (?, 30, 47, 47, 8)
output: (?, 14, 23, 23, 8)
batch_norm_2: BatchNormalization
input: (?, 14, 23, 23, 8)
output: (?, 14, 23, 23, 8)
conv_3: Conv3D
input: (?, 14, 23, 23, 8)
output: (?, 6, 11, 11, 8)
batch_norm_3: BatchNormalization
input: (?, 6, 11, 11, 8)
output: (?, 6, 11, 11, 8)
flatten: Flatten
input: (?, 6, 11, 11, 8)
output: (?, 5808)
dense_1: Dense
input: (?, 5808)
output: (?, 1024)
dropout_1: Dropout
input: (?, 1024)
output: (?, 1024)
dense_2: Dense
input: (?, 1024)
output: (?, 512)
dropout_2: Dropout
input: (?, 512)
output: (?, 512)
output: Dense
input: (?, 512)
output: (?, 3)

Figure 7.6 3D Deep Convolutional Neural Network – Baseline model

The scores for the Baseline 3D deep convolutional neural network are as follows,

We can see that the model was able to detect NC with close to 87% accuracy. The model finds difficulty in separating MCI and AD patients and this is because there are a lot of similarities in the critical regions of MRI images of patients with MCI and AD.

	Normal Control	Mild Cognitive Impairment	Alzheimer's	Average
Accuracy	87.4%	60%	77.77%	74.66%
F1 Scores	83.23%	64.56%	74.6%	74.1%
Recall	87.42%	58.59%	77.77%	74.6%

Performance of the Baseline 3D convolutional model.

7.4 Parallel Deep Convolutional Neural Network:

This architecture uses parallel Convolutional neural networks. From figure 7.7. we can observe that the architecture separates out into two branches that are working parallelly. The architecture starts with 62 x 96 x 96 x 1 input layer nodes and reduces to 30 x 30 x 47 x 8 nodes after its first 3D convolutional layer and its normalization in first branch and 30 x 30 x 47 x 32. This is because the number of filters in each block in the branches are different. Similarly, it reduces to 14 x 23 x 23 x 8 and 14 x 23 x 23 x 16 in branches one and two respectively after second batch normalization. Then we concatenate the two branches and merge the layers together. Finally, after some more fully connected layers, the final output we get is in the form of 3 nodes corresponding to CN, MCI, and AD. For more details about the architecture, please go through Figure 6.6 which shows a detailed design of the layers and nodes.

The scores for the parallel model are as follows,

We can see that the total accuracy of the model has increased by 1.4% in this parallel model. We can also see that the model was able to detect NC with almost close to 90% accuracy. This model detects NC patients better than our previous baseline model.

	Normal Control	Mild Cognitive Impairment	Alzheimer's	Average
Accuracy	92.5%	61.2%	74.2%	76%
F1 Scores	84.24%	67.14%	72.5%	74.6%
Recall	92.45%	60%	73.8%	74.6%

Performance of the Parallel Convolutional model.

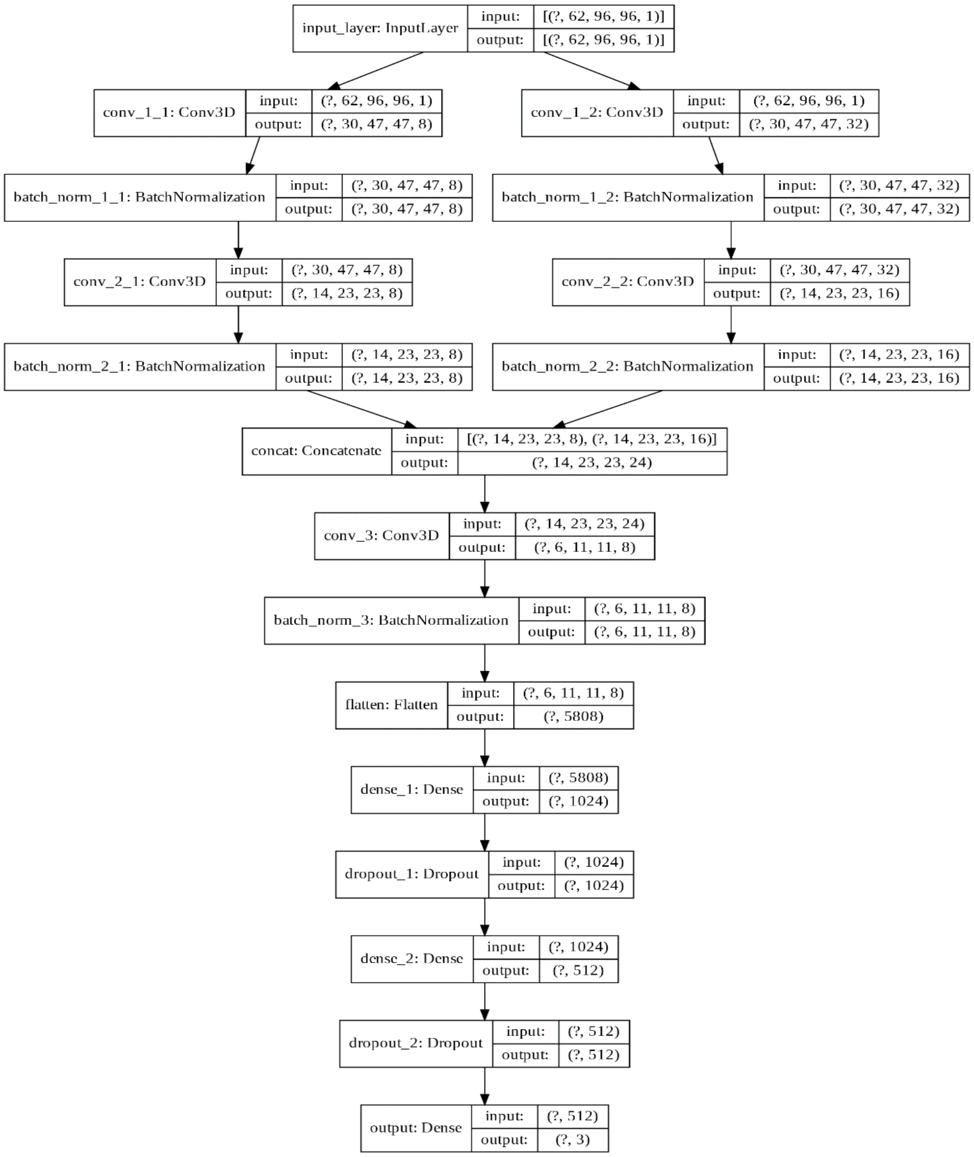

input_layer: InputLayer
input: [(?, 62, 96, 96, 1)]
output: [(?, 62, 96, 96, 1)]
conv_1_1: Conv3D
input: (?, 62, 96, 96, 1)
output: (?, 30, 47, 47, 8)
conv_1_2: Conv3D
input: (?, 62, 96, 96, 1)
output: (?, 30, 47, 47, 32)
batch_norm_1_1: BatchNormalization
input: (?, 30, 47, 47, 8)
output: (?, 30, 47, 47, 8)
batch_norm_1_2: BatchNormalization
input: (?, 30, 47, 47, 32)
output: (?, 30, 47, 47, 32)
conv_2_1: Conv3D
input: (?, 30, 47, 47, 8)
output: (?, 14, 23, 23, 8)
conv_2_2: Conv3D
input: (?, 30, 47, 47, 32)
output: (?, 14, 23, 23, 16)
batch_norm_2_1: BatchNormalization
input: (?, 14, 23, 23, 8)
output: (?, 14, 23, 23, 8)
batch_norm_2_2: BatchNormalization
input: (?, 14, 23, 23, 16)
output: (?, 14, 23, 23, 16)
concat: Concatenate
input: [(?, 14, 23, 23, 8), (?, 14, 23, 23, 16)]
output: (?, 14, 23, 23, 24)
conv_3: Conv3D
input: (?, 14, 23, 23, 24)
output: (?, 6, 11, 11, 8)
batch_norm_3: BatchNormalization
input: (?, 6, 11, 11, 8)
output: (?, 6, 11, 11, 8)
flatten: Flatten
input: (?, 6, 11, 11, 8)
output: (?, 5808)
dense_1: Dense
input: (?, 5808)
output: (?, 1024)
dropout_1: Dropout
input: (?, 1024)
output: (?, 1024)
dense_2: Dense
input: (?, 1024)
output: (?, 512)
dropout_2: Dropout
input: (?, 512)
output: (?, 512)
output: Dense
input: (?, 512)
output: (?, 3)

Figure 7.7 Parallel model

7.5 Mixed Input Parallel Deep Convolutional Neural Network model:

This architecture uses a mixed input architecture that can utilize both image and numerical inputs.

From figure 7.8 we can observe that the architecture starts with 62 x 96 x 96 x 1 input layer nodes and reduces to 14 x 23 x 23 x 8 after second batch normalization. During this stage, a second network is introduced starting with one layer and 2 nodes. After some iterations, the two branches concatenate and merge the layers together. Finally, after some more intermediate layers, the final output we get is in the form of 3 nodes corresponding to CN, MCI, and AD.

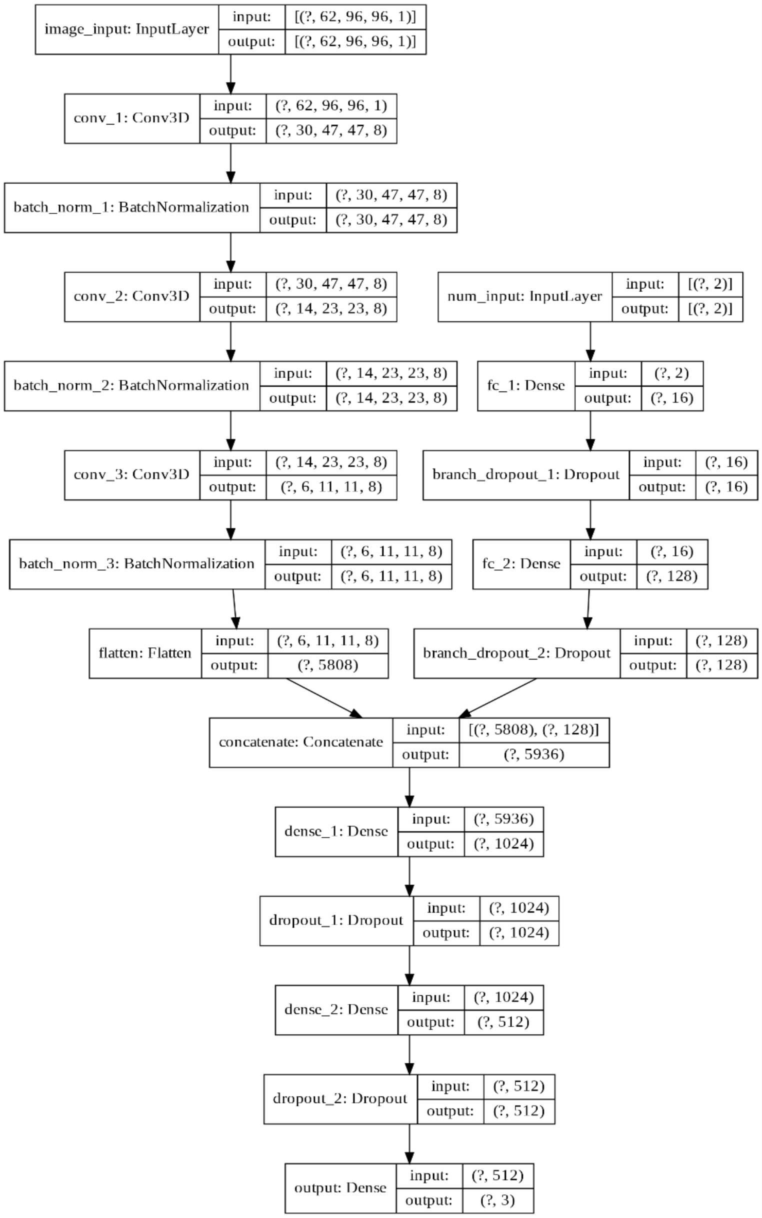
For more details about the architecture, please go through Figure 7.8 which shows a detailed design of the layers and nodes.

The scores for the mixed input model are as follows,

	Normal Control	Mild Cognitive Impairment	Alzheimer's	Average
Accuracy	86.8%	59.2%	80.4%	75.87%
F1 Scores	82.13%	64.53%	75.92%	74.1%
Recall	86.79%	57.96%	80.3%	75.04%

Performance of the Mixed Input model.

We can see that the total accuracy of the model is almost like the previous model, but this model is able to predict patients with AD better than the previous model. We can also see that the accuracy of this model for the Normal Control class is lower than that of the previous model. Overall, the total accuracy is close to the previous model, but we can find variations within the classes.



image_input: InputLayer | input: | [(?, 62, 96, 96, 1)] | output: | [(?, 62, 96, 96, 1)]
conv_1: Conv3D | input: | (?, 62, 96, 96, 1) | output: | (?, 30, 47, 47, 8)
batch_norm_1: BatchNormalization | input: | (?, 30, 47, 47, 8) | output: | (?, 30, 47, 47, 8)
conv_2: Conv3D | input: | (?, 30, 47, 47, 8) | output: | (?, 14, 23, 23, 8)
num_input: InputLayer | input: | [(?, 2)] | output: | [(?, 2)]
batch_norm_2: BatchNormalization | input: | (?, 14, 23, 23, 8) | output: | (?, 14, 23, 23, 8)
fc_1: Dense | input: | (?, 2) | output: | (?, 16)
conv_3: Conv3D | input: | (?, 14, 23, 23, 8) | output: | (?, 6, 11, 11, 8)
branch_dropout_1: Dropout | input: | (?, 16) | output: | (?, 16)
batch_norm_3: BatchNormalization | input: | (?, 6, 11, 11, 8) | output: | (?, 6, 11, 11, 8)
fc_2: Dense | input: | (?, 16) | output: | (?, 128)
flatten: Flatten | input: | (?, 6, 11, 11, 8) | output: | (?, 5808)
branch_dropout_2: Dropout | input: | (?, 128) | output: | (?, 128)
concatenate: Concatenate | input: | [(?, 5808), (?, 128)] | output: | (?, 5936)
dense_1: Dense | input: | (?, 5936) | output: | (?, 1024)
dropout_1: Dropout | input: | (?, 1024) | output: | (?, 1024)
dense_2: Dense | input: | (?, 1024) | output: | (?, 512)
dropout_2: Dropout | input: | (?, 512) | output: | (?, 512)
output: Dense | input: | (?, 512) | output: | (?, 3)

Figure 7.8 Mixed Input model

7. CONCLUSION

Since the percentage of subjects with no dementia (NC) is considerably higher than that of other classes, any model built on this imbalanced data would be biased towards the NC class. A similar trend is seen in all our models as well. We got more than 85% accuracy for NC class with all the three models and accuracies for the other two classes have not crossed 81%. We selected the best model based on the F1 score, which is robust to the class-imbalance problem. The mean F1 scores for the Baseline, Parallel, and Mixed input models are 74.1, 74.6, and 74.1, respectively. Due to the small difference in the F1 Scores, we considered the average accuracy scores: 74.66, 76, and 75.87. The Parallel model, which has two branches of CNNs, is the best model in terms of both mean F1 score and mean accuracy.

The mediocre results of our models can be attributed to similar patterns seen in both AD and MCI neuroimages that make distinction difficult. Alternatively, the most successful approach has been to use a 3D Auto-Encoder as mentioned in the paper[12]. One can also try using 3D Convolutional Auto-Encoders, in which each convolutional layer is an Auto Encoder.

The working code and other running instructions can be found at https://github.com/Vignesh-Nswamy/AD-Diagnosis.

In the future we want to try the following approaches implemented in the research papers [7], [9], and [6]:

- "Transfer" Learning: Multi-classification problems are converted into a binary classification problem and built models on that and increase the neurons in the output layer later[7].
- "Hybrid" Deep Learning: Deep learning is only used for feature extraction. Traditional machine learning models like SVM, are used for the classification[9].
- Multiple Neuroimaging data types: A fusion of two or more neuroimaging data types (like MRI+PET, and MRI+PET+CSF) instead of a single neuroimaging data[6].

8. List of Packages used

1. OS
2. Numpy – 1.18.1
3. Pandas – 1.0.1
4. Scikit-learn – 0.22.1
5. Matplotlib – 3.1.3
6. Tensorflow – 2.1.0

9. BIBLIOGRAPHY

1. https://www.alz.org/alzheimers-dementia/facts-figures#quickFacts
2. Alzheimer’s Association: 2019 Alzheimer’s Disease Facts and Figures
3. Auto-Detection of Alzheimer's Disease Using Deep Convolutional Neural Networks
4. DaoqiangZhang, DinggangShena, and The Alzheimer's Disease Neuroimaging Initiative, “Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in Alzheimer's disease, ” Neuroimage, Volume 59, Issue 2, Pages 895-907,16 January 2012.
5. Manhua Liu, Daoqiang Zhang, Dinggang Shen, and The Alzheimer's Disease Neuroimaging Initiative, “Ensemble sparse classification of Alzheimer's disease,” Neuroimage, vol. 60, 1106–1116, January 2012.
6. Deep Learning in Alzheimer’s Disease: Diagnostic Classification and Prognostic Prediction Using Neuroimaging Data
7. End-To-End Alzheimer’s Disease Diagnosis and Biomarker Identification - Soheil Esmaeilzadeh, Dimitrios Ioannis Belivanis, Kilian M. Pohl, and Ehsan Adeli
8. Early diagnosis of Alzheimer's disease with deep learning
9. A Robust Deep Model for Improved Classification of AD/MCI Patients
10. 3D Convolutional Neural Networks - Towards Data Science
11. Classifying Alzheimer’s MRI scans using 2D and 3D convolutional neural networks - Bouke Regnerus and Raoul Fasel
12. Predicting Alzheimer’s disease: a neuroimaging study with 3D convolutional neural networks