KOÇ
ÜNİVERSİTESİ
COLLEGE OF ENGINEERING

COMP 448/548
Medical Image Analysis

Spring 23
Final Project Report

Yiğit Can Ateş 75617 - Melis Oktayoğlu 64388

Dataset and Problem Description

Alzheimer's disease (AD) is a progressive neurodegenerative disorder that primarily affects the brain, leading to a decline in cognitive abilities, memory loss, and changes in behavior and thinking. It is the most common cause of dementia, a syndrome characterized by a severe decline in cognitive function. The exact cause of AD is not yet fully understood, but it is believed to be influenced by a combination of genetic, environmental, and lifestyle factors. The disease is characterized by the accumulation of abnormal protein aggregates in the brain tissue, including beta-amyloid plaques and tau tangles, which disrupt the normal functioning of neural cells. Symptoms of Alzheimer's disease typically develop slowly and worsen over time. Initially, individuals may experience mild memory loss and difficulties with concentration and learning. As the disease progresses, more severe cognitive impairments become evident.

As the prevalence of AD continues to rise, there is an urgent need to develop effective diagnostic techniques and gain insights into the disease's progression. Magnetic Resonance Imaging (MRI) has emerged as a valuable modality for investigating the structural alterations in the brain associated with AD. In this study, we leverage a preprocessed MRI dataset containing 6400 images, each resized into 128 x 128 pixels. This dataset encompasses four classes representing different stages of dementia: Mild Demented (896 images), Moderate Demented (64 images), Non Demented (3200 images), Very Mild Demented (2240 images). Several example images from the dataset are given in figure 1. The dataset is freely available at: https://www.kaggle.com/datasets/sachinkumar413/alzheimer-mri-dataset

Convolutional Neural Networks (CNNs) have revolutionized the field of medical image analysis, offering powerful tools for automated interpretation, diagnosis, and treatment planning. One of the key strengths of CNNs lies in their ability to automatically learn hierarchical features from raw image data, without the need for explicit feature engineering.

The primary objective of this project is to develop a deep CNN model for the diagnosis and classification of AD based on MRI images. For developing our model, we used a CNN architecture designed for a similar medical image analysis problem: pneumonia detection from X-ray images. The code for this pneumonia detection model can be accessed at: https://github.com/abhinavsagar/kaggle-notebooks

With this project, we anticipate contributing to the growing body of knowledge in Alzheimer's disease research, with the ultimate goal of facilitating early detection, accurate diagnosis, and effective management of this debilitating disorder.

```
NonDementia: 2560
MildDementia: 717
ModerateDementia: 52
VeryMildDementia: 1802
```

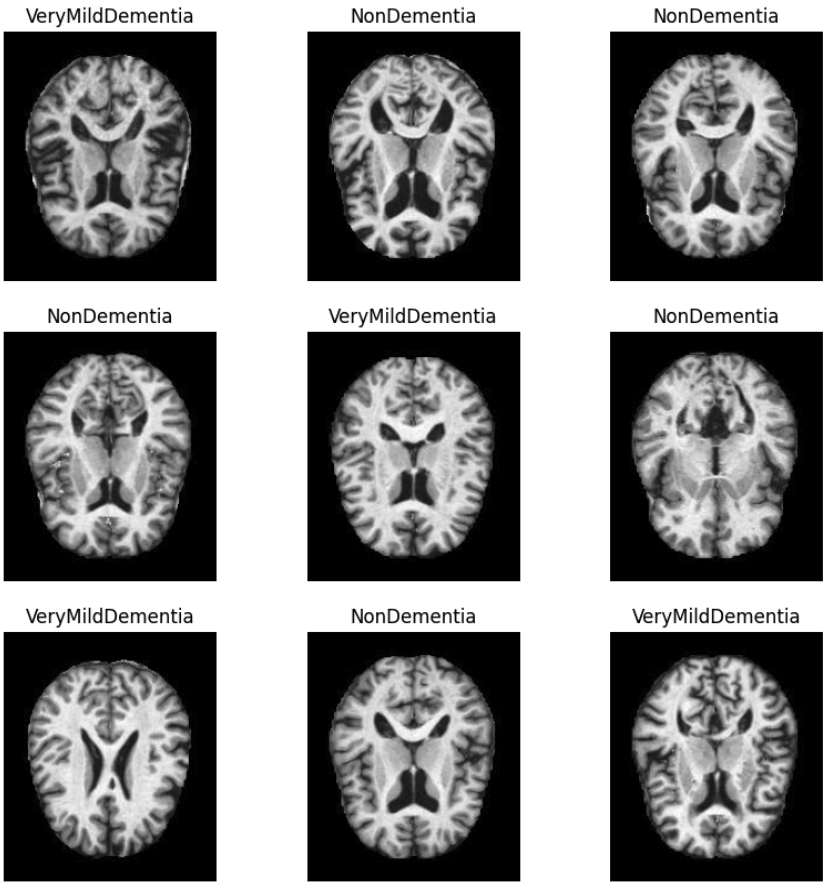

Figure 1. Training dataset class counts and example images from the dataset.

Methods

Our model is a deep convolutional neural network (CNN) architecture for Alzheimer's disease stage classification on MRI images. The reason behind this choice was that during our survey, we were intrigued by the CNN models' performances for the multiclass classification in this problem. We used PyTorch for the implementation of the architecture, as we used it in our homework 3, Tensorboard for our performance graphs, and GoogleColab as our platform and GPU source.

There are two blocks repeatedly used in the model:

1. **ConvBlock**: This block represents a convolutional block. It consists of two convolutional layers with ReLU activation, followed by batch normalization and max pooling. The purpose of ConvBlock is to extract features from the input images and downsample the spatial dimensions.
2. **DenseBlock**: This block represents a dense (fully connected) block. It consists of a linear (fully connected) layer with ReLU activation, followed by batch normalization, and dropout. DenseBlock is responsible for processing the flattened feature vectors obtained from the convolutional layers.

For the overall model, there are first 2 2D convolutions followed by a max pooling layer, then 3 of the convolution blocks followed by a dropout, and another conv block followed by another dropout. Then 3 dense layers for the classification prediction.
The model architecture leverages the convolutional layers to extract hierarchical features from the input MRI images. The subsequent dense blocks process these features to predict the Alzheimer's disease stage. The model uses **ReLU** activation, **batch normalization**, and **dropout** to introduce non-linearity, improve training stability, and reduce overfitting. Also, since this is a multiclass problem, we converted our labels to **one-hot encodings**. Also, we picked a batch size of 32, and 150 epochs. Forthe optimizer we picked ADAM with a learning rate 0.0013, and weight decay, and cross-entropy for our loss function. The reason behind these choices were their widespread use in the field.
By training this model on a labeled dataset of MRI images, it can learn to classify the images into one of four classes representing different stages of Alzheimer's disease. The sigmoid activation function in the last layer allows the model to provide class probabilities.

```python
class ConvBlock(nn.Module):
    def __init__(self, in_dim, out_dim):
        super(ConvBlock, self).__init__()
        self.conv1 = nn.Conv2d(in_dim, in_dim, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(in_dim, out_dim, kernel_size=3, padding=1)
        self.bn = nn.BatchNorm2d(out_dim)
        self.pool = nn.MaxPool2d(kernel_size=2)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = self.bn(x)
        x = self.pool(x)
        return x


class DenseBlock(nn.Module):
    def __init__(self, in_dim, out_dim, dropout_rate):
        super(DenseBlock, self).__init__()
        self.fc = nn.Linear(in_dim, out_dim)
        self.bn = nn.BatchNorm1d(out_dim)
        self.dropout = nn.Dropout(dropout_rate)

    def forward(self, x):
        x = F.relu(self.fc(x))
        x = self.bn(x)
        x = self.dropout(x)
        return x
```

```python
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2)

        self.conv_block1 = ConvBlock(32, 64)
        self.conv_block2 = ConvBlock(64,128)
        self.conv_block3 = ConvBlock(128,256)
        self.dropout1 = nn.Dropout(0.2)
        self.conv_block4 = ConvBlock(256,512)
        self.dropout2 = nn.Dropout(0.2)

        self.flatten = nn.Flatten()

        self.dense_block1 = DenseBlock(15360, 512,0.7)
        self.dense_block2 = DenseBlock(512, 128, 0.5)
        self.dense_block3 = DenseBlock(128, 64, 0.3)

        self.fc = nn.Linear(64, 4)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = self.pool(x)

        x = self.conv_block1(x)
        x = self.conv_block2(x)
        x = self.conv_block3(x)
        x = self.dropout1(x)

        x = self.conv_block4(x)
        x = self.dropout2(x)

        x = self.flatten(x)

        x = self.dense_block1(x)
        x = self.dense_block2(x)
        x = self.dense_block3(x)

        x = self.fc(x)
        x = torch.sigmoid(x)
        return x
```

Figure 2. Code for the model architecture.

Results

Since accuracy is not a sufficient metric for imbalanced datasets, we used AUC (Area Under the Curve). AUC (Area Under the Curve) is a widely used evaluation metric for binary classification problems. It measures the overall performance of a classifier by calculating the area under the receiver operating characteristic (ROC) curve. The ROC curve is created by plotting the true positive rate against the false positive rate at various classification thresholds.

Since we are dealing with a multi-class classification problem, we applied AUC in a one-vs-all fashion to extend its use. For each class in a multi-class problem we treated the problem as a binary classification task, where the positive class was the corresponding class and the negative class was everything else. The final AUC score is computed as the average of the individual class AUC scores. The validation loss, validation accuracy and validation AUC values for the 150th epoch is given in table 1.

Table 1. The validation loss, validation accuracy and validation AUC values for the 150th epoch.

Epoch	Validation Loss	Validation Accuracy	Validation AUC
150/150	0.9021	0.8345	0.9253

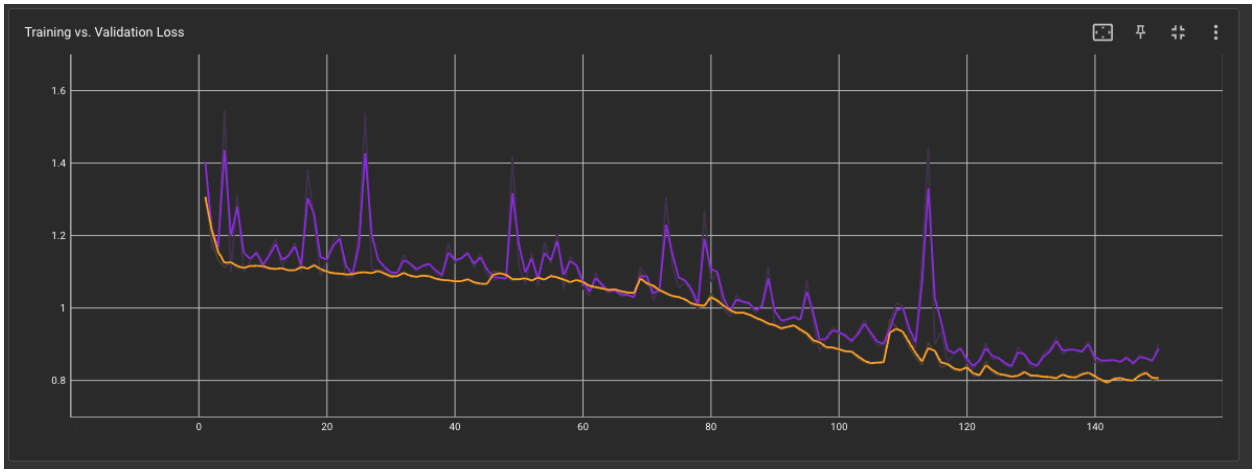

Figure 3. Train Loss and Validation loss vs epoch number

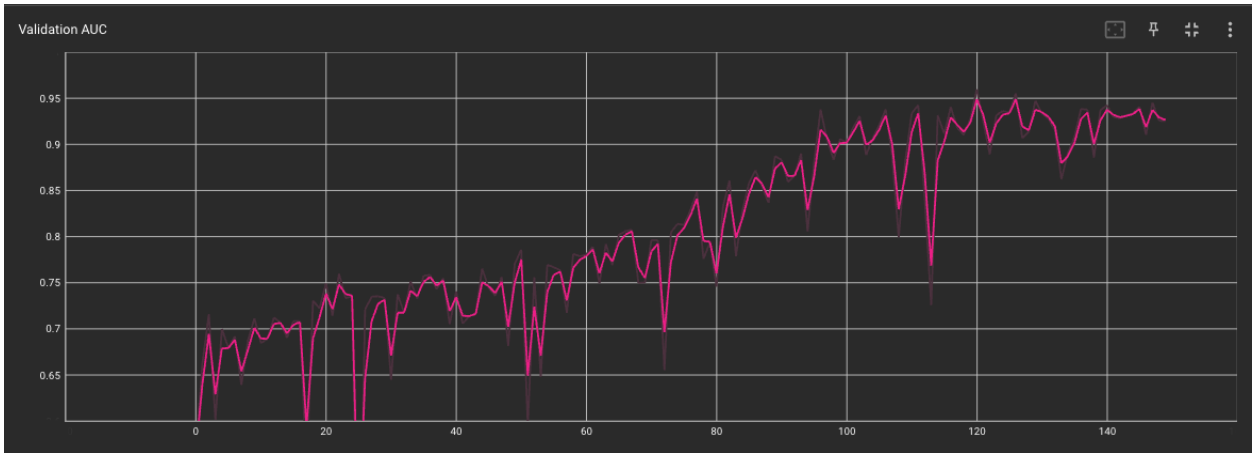

Figure 4. Running validation AUC vs epoch number.

Table 2. The test accuracy and test AUC values.

Test Accuracy	Test AUC
0.6419	0.7336

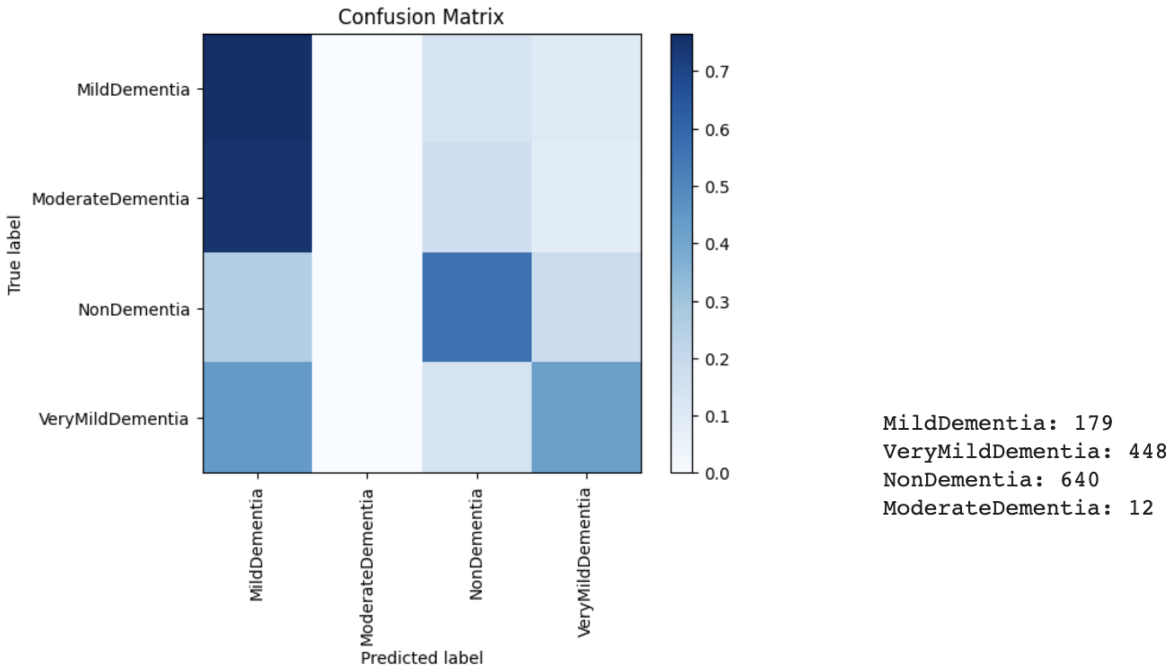

Figure 5. Normalized Confusion Matrix, and class counts of the test dataset on the right.

Discussion

From Figure 3, the training and validation loss is around the same, we see no divergence in them. Thus, we have no problems with overfitting, where the training loss is small and the validation loss is higher. Also, both seem to have reached a plateau meaning that we have reached our model's performance capacity. In fact, the validation loss is slightly higher in the latest epoch, thus in fact we could've done an early stopping to prevent it. However, our test AUC is significantly lower than the validation AUC by 0.73, this suggests an overfitting to the data. To counter this, we could've increased our dropout rates, used a higher batch size, and could have increased our learning rate. Also, our validation AUC (Figure 4.) also seems to consistently improve and has reached a plateau near 0.95.

In our test normalized confusion matrix, our model is highly accurate in classifying mild dementia cases, however, due to the class imbalance we have low performance in moderate dementia cases (in total 12 images in the test set), which are classified as mild dementia in the majority of the cases. We also have significant performance in Non Dementia classification, which is the highest count class in the test dataset, yet have again low performance in very mild dementia cases with some of them classified as mild dementia.