Final Report

Sabir Meah, Michael Miller, Liyang Yuan

2022-12-16

All the files for this project, including the full code (non-important code is hidden in this report due to space constraints) and data can be found in its GitHub repo: https://github.com/smeah/biostat625group1project

Introduction

Alzheimer's disease is the most common cause of dementia and the 6th leading cause of death in American adults (Alzheimer's Association; CDC). Despite its large impact on the American populace, the science of its etiology is rapidly evolving and many things about it are still not understood. Alzheimer's disease is typically diagnosed through clinical evaluation, as dementia is a disease defined by its symptoms, not underlying biological phenomena, but more recently, radiology, such as CT scans, PET scans, and MRIs can inform and support a clinical Alzheimer's disease diagnosis.

On that line of thinking, we set off to create a machine learning model that can accurately predict the presence of Alzheimer's disease from MRI imaging. We used an Alzheimer's MRI imaging dataset from Kaggle. The data includes a total of 6400 MRI images from individuals classified into four stages: no dementia, very mild dementia, mild dementia, and moderate dementia, with each successive class being smaller than the last. Each MRI result is a 128x128 .jpg image. The images have been preprocessed so that they are of a standard form and orientation, but we still will need to do the step of processing the data and gray scale standardization from .jpg images into a tabular format suitable for standard R functions.

Data Read-In

We read in the 128x128 images from .jpg files to three dimensional arrays, with the first dimension corresponding to the index of each image, then the latter two to greyscale pixel values, which we extracted using the `readJPEG()` function from the `jpeg` package.

```
no_dementia <- array(NA, dim = c(3200, 128, 128))
for (i in 1:3200){
  no_dementia[i,,] <- readJPEG(paste0(getwd(),
                                      "/Data/Non_Demented/non_", i, ".jpg"))
}
```

```
verymild_dementia <- array(NA, dim = c(2240, 128, 128))
for (i in 1:2240){
  verymild_dementia[i,,] <- readJPEG(paste0(getwd(),
                                            "/Data/Very_Mild_Demented/verymild_", i, ".jpg"))
}
```

```
mild_dementia <- array(NA, dim = c(896, 128, 128))
for (i in 1:896){
  mild_dementia[i,,] <- readJPEG(paste0(getwd(),
                                        "/Data/Mild_Demented/mild_", i, ".jpg"))
}
```

```
moderate_dementia <- array(NA, dim = c(64, 128, 128))
for (i in 1:64){
  moderate_dementia[i,,] <- readJPEG(paste0(getwd(),
                                            "/Data/Moderate_Demented/moderate_", i, ".jpg"))
}
```

```
lenNoDem <- 3200
lenVeryMildDem <- 2240
lenMildDem <- 896
lenModDem <- 64
totalLen <- lenNoDem + lenVeryMildDem + lenMildDem + lenModDem
```

```
dementia_data <- array(NA, dim = c(6400, 128, 128))
dementia_data[1:lenNoDem,,] <- no_dementia
dementia_data[3201:(lenNoDem + lenVeryMildDem),,] <- verymild_dementia
dementia_data[5441:(lenNoDem + lenVeryMildDem + lenMildDem),,] <- mild_dementia
dementia_data[6337:totalLen,,] <- moderate_dementia
```

Modeling

write here

Model 1: 3 classes with one hot encoding

For the first model we combined the mild dementia and moderate dementia cases, resulting in three distinct classes.

```
lenNoDem <- 3200
lenVeryMildDem <- 2240
lenMildDem <- 896
lenModDem <- 64

img_rows <- 128
img_cols <- 128

# Set up class indicators. One hot encoding for the three classes
totalLen <- lenNoDem+lenVeryMildDem+lenMildDem+lenModDem
noDemClass <- cbind(rep(1,lenNoDem), rep(0,lenNoDem),
                    rep(0,lenNoDem))
veryMildDemClass <- cbind(rep(0,lenVeryMildDem), rep(1,lenVeryMildDem),
                          rep(0,lenVeryMildDem))
mildModDemClass <- cbind(rep(0,lenMildDem+lenModDem), rep(0,lenMildDem+lenModDem),
                         rep(1,lenMildDem+lenModDem))
classMat <- rbind(noDemClass,
                  veryMildDemClass,
                  mildModDemClass)

# Stratify sampling by class to ensure that classes are proportionally represented
# in the training and testing data sets
noDemTestIdx <- sample(lenNoDem, round(lenNoDem/4.0), replace = FALSE)
noDemTrainIdx <- setdiff(1:lenNoDem, noDemTestIdx)

veryMildDemTestIdx <- sample((lenNoDem+1):(lenNoDem+lenVeryMildDem),
                             round(lenVeryMildDem/4.0), replace = FALSE)
```

```
veryMildDemTrainIdx <- setdiff((lenNoDem+1):(lenNoDem+lenVeryMildDem),
                               veryMildDemTestIdx)

mildDemTestIdx <- sample((lenNoDem+lenVeryMildDem+1):
                           (lenNoDem+lenVeryMildDem+lenMildDem),
                           round((lenMildDem)/4.0), replace = FALSE)
mildDemTrainIdx <- setdiff((lenNoDem+lenVeryMildDem+1):
                           (lenNoDem+lenVeryMildDem+lenMildDem),
                           mildDemTestIdx)

modDemTestIdx <- sample((lenNoDem+lenVeryMildDem+lenMildDem+1):
                         (lenNoDem+lenVeryMildDem+lenMildDem+lenModDem),
                          round((lenModDem)/4.0), replace = FALSE)
modDemTrainIdx <- setdiff((lenNoDem+lenVeryMildDem+lenMildDem+1):
                           (lenNoDem+lenVeryMildDem+lenMildDem+lenModDem),
                            modDemTestIdx)

mildModDemTrainIdx <- c(mildDemTrainIdx,modDemTrainIdx)
mildModDemTestIdx <- c(mildDemTrainIdx,modDemTrainIdx)

x_train <- dementia_data[c(noDemTrainIdx,veryMildDemTrainIdx,
                           mildModDemTrainIdx),,]
y_train <- classMat[c(noDemTrainIdx,veryMildDemTrainIdx,
                      mildModDemTrainIdx),]
x_test <- dementia_data[c(noDemTestIdx,veryMildDemTestIdx,
                          mildModDemTestIdx),,]
y_test <- classMat[c(noDemTestIdx,veryMildDemTestIdx,
                     mildModDemTestIdx),]

# Shuffle training data because keras does not shuffle before taking validation set
shuffleIdx <- sample(1:round(3.0*totalLen/4.0))
x_train <- x_train[shuffleIdx,,]
y_train <- y_train[shuffleIdx,]

# Reshape into tensor form in order to be compatible with keras
x_train <- array_reshape(x_train, c(nrow(x_train), img_rows, img_cols, 1))
x_test <- array_reshape(x_test, c(nrow(x_test), img_rows, img_cols, 1))
input_shape <- c(img_rows, img_cols, 1)

batch_size <- 32
num_classes <- 3
epochs <- 100
```

```
mod1 <- keras_model_sequential() %>%
  layer_conv_2d(filters = 64, kernel_size = c(3,3),
                activation = 'relu', input_shape = input_shape) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3,3),
                activation = 'relu', input_shape = input_shape) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_flatten() %>%
  layer_dense(units = 128, activation = 'relu',
```

```
              kernel_regularizer=regularizer_l1_l2(l1=1e-4,l2=1e-5)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = num_classes,
              activation = 'softmax')

mod1 %>% compile(
  loss = loss_categorical_crossentropy,
  optimizer = optimizer_adam(),
  metrics = c('accuracy')
)

cnn_history <- mod1 %>% fit(
  x_train, y_train,
  batch_size = batch_size,
  epochs = epochs,
  validation_split = 0.2
)
```

```
# Predict on the test data
predictProbs = predict(mod1, x_test)
```

Process Model 1 Results

```
# Calculate class predictions using the softmax probabilities
predictClass <- matrix(NA, nrow = nrow(x_test), ncol = num_classes)
for (i in 1:nrow(x_test)) {
  classVec <- rep(0, num_classes)
  classVec[which(predictProbs[i,] == max(predictProbs[i,]))] <- 1
  predictClass[i,] <- classVec
}
```

No.Dementia	Very.Mild.Dementia	Mild.Moderate.Dementia
0.965	0.9303571	0.9861111

Model 2: 4 classes with one hot encoding

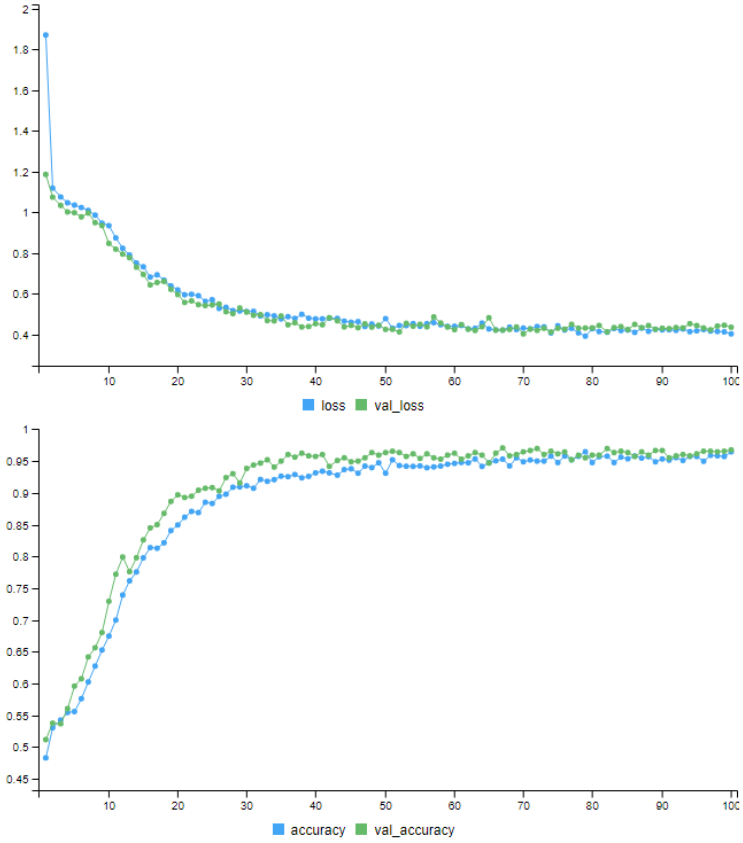
For the second model we represented mild dementia and moderate dementia with separate classes, resulting in four distinct classes.

```
lenNoDem <- 3200
lenVeryMildDem <- 2240
lenMildDem <- 896
lenModDem <- 64

img_rows <- 128
img_cols <- 128

# Set up class indicators. One hot encoding for the four classes.
totalLen <- lenNoDem+lenVeryMildDem+lenMildDem+lenModDem

noDemClass <- cbind(rep(1,lenNoDem), rep(0,lenNoDem),
                    rep(0,lenNoDem), rep(0,lenNoDem))
veryMildDemClass <- cbind(rep(0,lenVeryMildDem), rep(1,lenVeryMildDem),
```

```
                          rep(0,lenVeryMildDem), rep(0,lenVeryMildDem))
mildDemClass <- cbind(rep(0,lenMildDem) ,rep(0,lenMildDem),
                      rep(1,lenMildDem), rep(0,lenMildDem))
modDemClass <- cbind(rep(0,lenModDem), rep(0,lenModDem),
                     rep(0,lenModDem), rep(1,lenModDem))
classMat <- rbind(noDemClass, veryMildDemClass,
                  mildDemClass, modDemClass)
```

```
# Predict on the test data
predictProbs = predict(mod2, x_test)
```

Process Model 2 Results



No.Dementia	Very.Mild.Dementia	Mild.Dementia	Moderate.Dementia
0.97875	0.9464286	0.9642857	1

Conclusion

A key difference in our project compared to other analyses of this data set is that we compared model performance between a three class representation of dementia progression and a four class representation of dementia progression. As shown in the results, the out of sample prediction for the no dementia and very mild dementia cases improved in model 2, but the mild + moderate dementia out of sample prediction in model 1 performed better than the mild dementia out of sample prediction in model 2. However, model 2 achieved perfection out of sample prediction for the moderate dementia cases, but this is not a very reliable result given that there were only 16 data points for the moderate dementia.

In summary, we conclude that model 2 is the better model because 1) it actually does predict the moderate dementia case well, contrary to our first intuition that it would perform poorly due to the low sample size, and 2) it performs better on the no dementia and very mild dementia cases.