Classifying Dementia Progression Using MRI Imaging

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. 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. More recently however, 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 classify the degree 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 in sample size 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.

We used convolutional neural networks (CNNs) to learn discriminative models for the dementia classes. Unlike other analyses, including the Kaggle notebook we cite at the end of the report, our analysis estimates and compares two different models: one model with a pooled class for the mild and moderate dementia class, and one model that represents all four dementia categories separately. One of the goals of our project was to investigate which class representation led to superior predictive performance for a class that had an extremely low sample size relative to the rest of the data. All models were fit with `keras` using GPU-accelerated tensor operations via CUDA and cuDNN on an Nvidia RTX 3080 12GB.

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"))
}
```

The other classes were read in similarly, then the arrays were combined into one large array we named `dementia_data`.

Modeling

Both models use the same CNN specification. There are two convolutional layers, each with a max pooling layer and a dropout layer, and there is one dense layer with a drop out layer. We used drop out probabilities of 0.25 for each drop out layer. We used 3×3 matrices for the kernels in the convolutional layers and max pooled 2×2 grids in the outputs of the convolutional layers for dimensionality reduction. All non-linear functions were ReLU functions except for the softmax layer that outputs class probabilities.

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)
```

Model 1 Out-of-Sample Accuracy

```
# 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

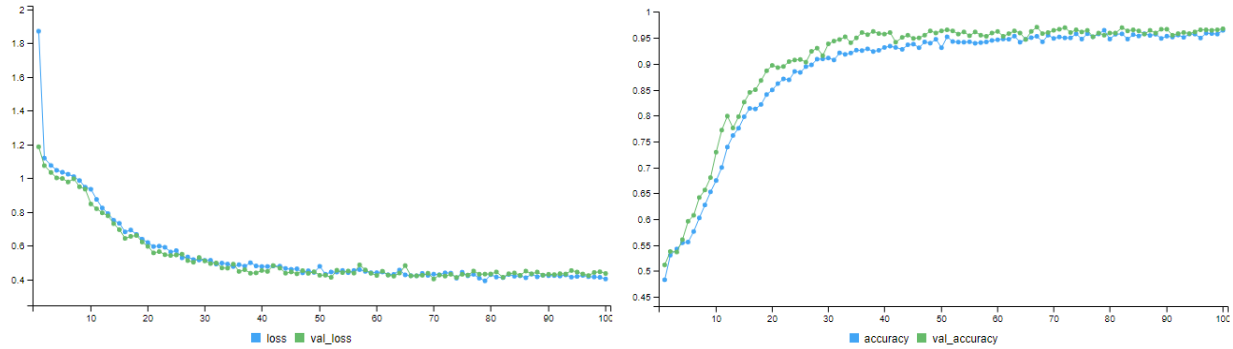
Model 2: 4 classes with one hot encoding

```
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)
```

Model 2 uses the same CNN specification as model 1, except now the class vectors are of length four and not three, so we omit the code for fitting the CNN as it is equivalent to the model 1 code.

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

Model 2 Out-of-Sample Accuracy



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

Conclusion

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. 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.

In all, we found that it was very possible to fit a highly accurate (>90% accuracy for every class) model to predict the extent of Alzheimer's disease from MRI images. Our findings emphasize the utility of MRI imaging as an assistive tool for clinicians in the diagnosis of Alzheimer's disease.

Group Contributions

Sabir set up CUDA with R on his personal machine and used it to train all the CNN models (reducing runtime by up to 2 orders of magnitude), wrote the introduction and data read-in code and their associated report text, and performed the R Markdown formatting of the final report.

Michael wrote the code to stratify the training and testing splits by class and wrote the code containing the CNN architectures + model fitting.

Liyang calculated the predictive performance metrics and compiled the data frames of the results using the estimated CNN models.

Citations

Gaikwad, A. (2022, July 22). Ad Detection With 99.53% accuracy. *Kaggle*. Retrieved December 8, 2022, from https://www.kaggle.com/code/akashgw/ad-detection-with-99-53-accuracy

Kumar, S., & Shastri, S. (2022, March). Alzheimer MRI Preprocessed Dataset. *Kaggle*. Retrieved November 20, 2022, from Kaggle. https://www.kaggle.com/datasets/sachinkumar413/alzheimer-mri-dataset