Main

Group 6

In your final repo, there should be an R markdown file that organizes **all computational steps** for evaluating your proposed Facial Expression Recognition framework.

This file is currently a template for running evaluation experiments. You should update it according to your codes but following precisely the same structure.

```r
if(!require("EBImage")){
  source("https://bioconductor.org/biocLite.R")
  biocLite("EBImage")
}
if(!require("R.matlab")){
  install.packages("R.matlab")
}
```

```
## Warning: package 'R.matlab' was built under R version 3.6.2
```

```r
if(!require("readxl")){
  install.packages("readxl")
}

if(!require("dplyr")){
  install.packages("dplyr")
}
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```r
if(!require("readxl")){
  install.packages("readxl")
}

if(!require("ggplot2")){
  install.packages("ggplot2")
}

if(!require("caret")){
  install.packages("caret")
}
```

```
## Warning: package 'caret' was built under R version 3.6.2
```

```r
if(!require("caTools")){
  install.packages("caTools")
}
```

```
## Warning: package 'caTools' was built under R version 3.6.2
```

```r
if(!require("e1071")){
  install.packages("e1071")
```

```
}
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

```
library(R.matlab)
library(readxl)
library(dplyr)
library(EBImage)
library(ggplot2)
library(caret)
library(caTools)
library(e1071)

set.seed(0)
```

Step 0 set work directories

Provide directories for training images. Training images and Training fiducial points will be in different subfolders.

```
train_dir <- "../data/train_set/"
train_image_dir <- paste(train_dir, "images/", sep="")
train_pt_dir <- paste(train_dir,  "points/", sep="")
train_label_path <- paste(train_dir, "label.csv", sep="")
```

Step 1: set up controls for evaluation experiments.

In this chunk, we have a set of controls for the evaluation experiments.

- (T/F) cross-validation on the training set
- (T/F) process features for training set
- (T/F) run evaluation on an independent test set
- (T/F) process features for test set

```
run.cv=TRUE # run cross-validation on the training set
run.feature.train=TRUE # process features for training set
run.test=TRUE # run evaluation on an independent test set
run.feature.test=TRUE # process features for test set
```

Using cross-validation or independent test set evaluation, we compare the performance of models with different specifications. This code defines the parameters that will be tested for the baseline GBM model.

```
shrinkage <- c(0.001, 0.01, 0.1)
n.minobsinnode <- c(5, 10, 15)
n.trees <- c(200, 300, 400)
param_grid <- expand.grid(shrinkage=shrinkage, n.minobsinnode=n.minobsinnode, n.trees=n.trees)
```

Step 2: import data and train-test split

```
#train-test split
info <- read.csv(train_label_path)
n <- nrow(info)
n_train <- round(n*(4/5), 0)
train_idx <- sample(info$Index, n_train, replace = F)
test_idx <- setdiff(info$Index,train_idx)
```

We did not extract features from the images themselves, so this code chunk only determines the number of images.

```
n_files <- length(list.files(train_image_dir))

#image_list <- list()
#for(i in 1:100){
#   image_list[[i]] <- readImage(paste0(train_image_dir, sprintf("%04d", i), ".jpg"))
#}
```

Fiducial points are stored in matlab format. In this step, we read them and store them in a list.

```
#function to read fiducial points
#input: index
#output: matrix of fiducial points corresponding to the index
readMat.matrix <- function(index){
     return(round(readMat(paste0(train_pt_dir, sprintf("%04d", index), ".mat"))[[1]],0))
}

#load fiducial points
fiducial_pt_list <- lapply(1:n_files, readMat.matrix)
#save(fiducial_pt_list, file="../output/fiducial_pt_list.RData")
```

Step 3: Construct features and responses

For the baseline model, we use the feature extraction from the starter code, which calculates the pairwise distances between the fiducial points.

```
source("../lib/feature.R")
tm_feature_train_base <- NA
if(run.feature.train){
  tm_feature_train_base <- system.time(dat_train <- feature(fiducial_pt_list, train_idx))
}

tm_feature_test_base <- NA
if(run.feature.test){
  tm_feature_test_base <- system.time(dat_test <- feature(fiducial_pt_list, test_idx))
}

#save(dat_train, file="../output/feature_train.RData")
#save(dat_test, file="../output/feature_test.RData")
```

Perform PCA on features

For the advanced model, we run PCA on the features extracted in the previous step.

```
#Perform PCA analysis on training data, and transform features from training data into PCAs
start_time <-  Sys.time()
dat_train_pca <- data.frame(dat_train)
dat_train_pca[,-6007] <- scale(dat_train_pca[,-6007])
pca <- preProcess(x=dat_train_pca[-6007], method="pca", thresh=0.99)
dat_train_pca <- predict(pca, dat_train_pca)
end_time <- Sys.time()
#The total advanced model feature training time is the base feature training time plus PCA time
tm_feature_train_advanced <- difftime(end_time, start_time, units="secs") + tm_feature_train_base[1]
```

```
#Transform features from test data into PCAs
start_time <-  Sys.time()
dat_test_pca <- data.frame(dat_test)
dat_test_pca[,-6007] <- scale(dat_test_pca[,-6007])
dat_test_pca <- predict(pca, dat_test_pca)
end_time <- Sys.time()
#The total advanced model feature training time is the base feature training time plus PCA time
tm_feature_test_advanced <- difftime(end_time, start_time, units="secs") + tm_feature_test_base[1]
```

Step 4: Train a classification model with training features and responses

Baseline model

For the baseline model, we use a GBM model.

```
source("../lib/train_gbm_mp.R")
source("../lib/test_gbm_mp.R")
```

The code below runs cross-validation for the baseline model, in order to choose the best parameters for the GBM model. (Since it takes over 24 hours to run cross-validation on all parameter combinations, we recommend keeping eval=F.)

```
source("../lib/cross_validation_gbm_mp.R")
#load("../output/feature_train.RData")
#load("../output/feature_test.RData")
#load("../output/err_cv_gbm_mp.RData")
if(run.cv){
  model_labels <- rep(NA, nrow(param_grid))
  for(i in 1:nrow(param_grid)){
    model_labels[i] <- paste0("GBM with shrinkage = ",param_grid$shrinkage[i],", n.minobsinnode = ",para
  }
  err_cv <- matrix(0, nrow = nrow(param_grid), ncol = 2)
  for(i in 1:nrow(param_grid)){
    print(model_labels[i])
    err_cv[i,] <- cv.function(dat_train, K, param_grid$shrinkage[i], param_grid$n.minobsinnode[i], para
    #save(err_cv, file="../output/err_cv_gbm_mp.RData")
  }
}
```

Based on the above cross-validation, choose the best parameter values. Our cross-validation found the best values to be 0.1 for shrinkage, 15 for the # of minimum observations in terminal nodes, and 400 for the # of trees.

```
if(run.cv){
  model_best <- which.min(err_cv[,1])
}
par_best <- list(shrinkage = param_grid$shrinkage[model_best], n.minobsinnode = param_grid$n.minobsinno
#save(par_best, file="../output/par_best_gbm_mp.RData")
```

Train the model with the entire training set using the selected GBM parameters.

```
load(file="../output/par_best_gbm_mp.RData")
tm_train_base=NA
tm_train_base <- system.time(fit_train <- train(dat_train, par_best))
```

```
## Warning: package 'gbm' was built under R version 3.6.3
```

```
## Loaded gbm 2.1.5
```

```
#save(fit_train, file="../output/fit_train_gbm_mp.RData")
```

Advanced model

For the advanced model, we use a SVM model.

The code below runs cross-validation for the advanced model, in order to choose the best cost parameter value for the SVM model.

```
c_vals <- 2^(seq(-12, -8, 0.5))
lin_tune <- tune(svm, emotion_idx~., data=dat_train_pca, kernel="linear", ranges=list(cost=c_vals), scal
c <- lin_tune$best.parameters$cost
```

Train the model with the entire training set using the selected SVM parameter.

```
tm_train_advanced=NA
tm_train_advanced <- system.time(pca_svm <- svm(emotion_idx~., data=dat_train_pca, type="C", kernel="lin
#saveRDS(pca_svm, "../output/pca_svm_model.RDS")
```

Step 5: Run test on test images

Baseline model

```
tm_test_base=NA
if(run.test){
  #load(file="../output/fit_train_gbm_mp.RData")
  tm_test_base <- system.time(pred <- test(fit_train, dat_test, par_best))
}
```

Advanced model

```
tm_test_advanced=NA
if(run.test){
  #pca_svm <- readRDS("../output/pca_svm_model.RDS")
  tm_test_advanced <- system.time(pred_advanced <- predict(pca_svm, dat_test_pca[,-1]))
}
```

Evaluation

Baseline model

```
pred <- factor(pred, levels=1:22)
accu <- mean(dat_test$emotion_idx == pred)
cat("The accuracy of the baseline model is", accu*100, "%.\n")
```

```
## The accuracy of the baseline model is 44 %.
```

```
confusionMatrix(pred, dat_test$emotion_idx)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
##         1  11  0  4  2  0  1  0  0  0  1  0  0  0  0  0  1  0  1  1  0  0  0
##         2   0  9  0  0  0  0  0  2  7  0  0  0  0  0  0  0  1  0  1  2  0  0
##         3   3  0 10  2  0  0  0  0  0  2  0  0  1  0  0  3  1  0  0  0  1  2
##         4   1  0  2  9  1  1  0  0  0  1  3  2  4  0  0  2  0  0  0  0  0  0
##         5   0  0  0  0 14  0  0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0
```

```
##         6    0  1  0  2  0 10  0  0  1  1  1  3  0  0  0  1  0  0  0  0  1  0
##         7    0  0  0  0  2  0 14  2  0  0  0  0  0  0  1  0  1  0  2  0  1  0
##         8    0  3  0  0  1  0  0 19  1  0  0  0  0  0  0  0  1  0  0  1  0  0
##         9    0  3  0  0  0  1  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0  0
##         10   0  0  1  1  0  1  0  0  0  5  3  0  3  0  0  0  0  0  0  0  0  1
##         11   0  0  1  3  0  1  0  0  1  3 13  2  1  0  0  0  0  0  0  0  1  1
##         12   0  0  0  0  0  3  0  0  0  0  3  9  2  0  0  0  0  0  1  0  0  3
##         13   1  0  2  0  0  2  0  0  0  2  1  7  4  0  0  1  0  0  1  0  0  1
##         14   0  0  0  0  2  0  0  0  0  0  0  2  0 19  6  0  1  1  0  0  2  0
##         15   0  0  0  0  1  0  0  0  0  0  0  0  1  2  7  1  0  1  2  0  0  0
##         16   0  0  1  0  0  0  0  0  0  0  0  1  0  0  0 10  0  0  0  0  0  2
##         17   0  0  0  0  4  0  0  2  0  0  0  0  0  0  1  2 10  6  1  3  1  0
##         18   0  0  0  0  2  0  1  1  1  0  0  0  0  0  0  1  6  8  2  2  0  0
##         19   0  0  1  0  0  0  0  0  0  0  0  0  0  0  1  1  0  1  7  4  2  1
##         20   0  0  0  0  0  0  4  1  1  0  1  1  2  1  1  0  1  0  6  4  3  3
##         21   0  0  1  0  1  1  2  0  1  0  2  1  1  0  0  0  0  0  4  3  9  4
##         22   0  1  2  0  1  0  2  0  0  1  1  0  1  0  0  0  0  0  3  3  2  4
##
## Overall Statistics
##
##                Accuracy : 0.44
##                  95% CI : (0.396, 0.4848)
##     No Information Rate : 0.062
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4133
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.6875   0.5294   0.4000   0.4737   0.4828   0.4762
## Specificity            0.9773   0.9731   0.9684   0.9647   0.9936   0.9770
## Pos Pred Value         0.5000   0.4091   0.4000   0.3462   0.8235   0.4762
## Neg Pred Value         0.9895   0.9833   0.9684   0.9789   0.9689   0.9770
## Prevalence             0.0320   0.0340   0.0500   0.0380   0.0580   0.0420
## Detection Rate         0.0220   0.0180   0.0200   0.0180   0.0280   0.0200
## Detection Prevalence   0.0440   0.0440   0.0500   0.0520   0.0340   0.0420
## Balanced Accuracy      0.8324   0.7512   0.6842   0.7192   0.7382   0.7266
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class: 12
## Sensitivity            0.6087   0.7037   0.5357    0.3125    0.4643    0.3214
## Specificity            0.9811   0.9852   0.9915    0.9793    0.9703    0.9746
## Pos Pred Value         0.6087   0.7308   0.7895    0.3333    0.4815    0.4286
## Neg Pred Value         0.9811   0.9831   0.9730    0.9773    0.9683    0.9603
## Prevalence             0.0460   0.0540   0.0560    0.0320    0.0560    0.0560
## Detection Rate         0.0280   0.0380   0.0300    0.0100    0.0260    0.0180
## Detection Prevalence   0.0460   0.0520   0.0380    0.0300    0.0540    0.0420
## Balanced Accuracy      0.7949   0.8445   0.7636    0.6459    0.7173    0.6480
##                      Class: 13 Class: 14 Class: 15 Class: 16 Class: 17
## Sensitivity             0.2000    0.8636    0.4118    0.4167    0.4348
## Specificity             0.9625    0.9707    0.9834    0.9916    0.9581
## Pos Pred Value          0.1818    0.5758    0.4667    0.7143    0.3333
## Neg Pred Value          0.9665    0.9936    0.9794    0.9712    0.9723
```

```
## Prevalence               0.0400    0.0440    0.0340    0.0480    0.0460
## Detection Rate           0.0080    0.0380    0.0140    0.0200    0.0200
## Detection Prevalence     0.0440    0.0660    0.0300    0.0280    0.0600
## Balanced Accuracy        0.5813    0.9172    0.6976    0.7041    0.6964
##                      Class: 18 Class: 19 Class: 20 Class: 21 Class: 22
## Sensitivity              0.4211    0.2258    0.1818    0.3913    0.1818
## Specificity              0.9667    0.9765    0.9477    0.9560    0.9644
## Pos Pred Value           0.3333    0.3889    0.1379    0.3000    0.1905
## Neg Pred Value           0.9769    0.9502    0.9618    0.9702    0.9624
## Prevalence               0.0380    0.0620    0.0440    0.0460    0.0440
## Detection Rate           0.0160    0.0140    0.0080    0.0180    0.0080
## Detection Prevalence     0.0480    0.0360    0.0580    0.0600    0.0420
## Balanced Accuracy        0.6939    0.6012    0.5648    0.6736    0.5731
```

Advanced model

```r
pred_advanced <- factor(pred_advanced, levels=1:22)
accu_advanced <- mean(dat_test$emotion_idx == pred_advanced)
cat("The accuracy of the advanced model is", accu_advanced*100, "%.\n")
```

```
## The accuracy of the advanced model is 52.2 %.
```

```r
confusionMatrix(pred_advanced, dat_test$emotion_idx)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
##         1  13  0  5  1  0  0  2  0  0  0  0  0  0  0  0  1  0  1  1  0  0  0
##         2   0 12  0  0  0  0  0  2  3  0  0  0  0  0  0  0  0  0  0  0  0  0
##         3   2  0 11  0  0  0  0  0  0  1  0  0  0  0  0  1  1  0  3  0  0  4
##         4   0  0  0 11  1  3  1  0  0  4  6  2  4  0  0  1  0  0  0  0  0  1
##         5   0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
##         6   1  1  0  3  0  9  0  0  0  1  1  3  2  0  0  0  0  0  0  0  1  2
##         7   0  0  0  0  1  0 14  1  0  0  0  0  0  0  0  0  1  0  0  0  2  0
##         8   0  2  0  0  0  0  0 21  0  0  0  0  0  0  0  0  2  0  0  0  0  0
##         9   0  2  0  0  0  1  0  0 21  0  0  0  0  0  0  0  0  0  1  2  0  0
##         10  0  0  2  2  0  0  0  0  0  5  3  1  1  0  0  1  0  0  0  0  1  4
##         11  0  0  1  0  0  3  0  0  0  1 12  3  1  0  0  0  0  0  0  0  0  0
##         12  0  0  0  0  0  2  0  0  1  0  3 12  2  0  0  0  0  0  0  0  0  0
##         13  0  0  1  2  0  1  0  0  0  3  2  5  7  0  0  0  0  0  2  0  0  2
##         14  0  0  0  0  1  1  0  0  2  0  0  1  0 18  4  0  1  0  1  1  0  0
##         15  0  0  0  0  0  0  1  0  0  0  0  0  0  2 10  0  0  1  3  0  1  0
##         16  0  0  2  0  0  0  0  0  0  0  0  0  2  0  0 17  0  0  1  1  2  0
##         17  0  0  0  0  1  0  0  2  0  0  0  0  0  0  1  0  8  3  2  2  0  0
##         18  0  0  0  0  5  0  1  0  0  0  0  0  0  1  0  3  8 11  1  1  0  0
##         19  0  0  2  0  0  0  0  0  0  0  0  0  0  0  1  0  0  2  9  1  2  0
##         20  0  0  0  0  0  0  2  1  0  0  0  0  0  1  0  0  1  0  2  6  2  1
##         21  0  0  0  0  0  1  2  0  1  0  0  0  1  0  1  0  1  0  2  6 11  5
##         22  0  0  1  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  3  2  1  3
##
## Overall Statistics
##
##                Accuracy : 0.522
##                  95% CI : (0.4772, 0.5665)
```

```
##     No Information Rate : 0.062
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4993
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.8125   0.7059   0.4400   0.5789   0.6897   0.4286
## Specificity            0.9773   0.9896   0.9747   0.9522   0.9979   0.9687
## Pos Pred Value         0.5417   0.7059   0.4783   0.3235   0.9524   0.3750
## Neg Pred Value         0.9937   0.9896   0.9706   0.9828   0.9812   0.9748
## Prevalence             0.0320   0.0340   0.0500   0.0380   0.0580   0.0420
## Detection Rate         0.0260   0.0240   0.0220   0.0220   0.0400   0.0180
## Detection Prevalence   0.0480   0.0340   0.0460   0.0680   0.0420   0.0480
## Balanced Accuracy      0.8949   0.8478   0.7074   0.7656   0.8438   0.6986
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class: 12
## Sensitivity            0.6087   0.7778   0.7500    0.3125    0.4286    0.4286
## Specificity            0.9895   0.9915   0.9873    0.9690    0.9809    0.9831
## Pos Pred Value         0.7368   0.8400   0.7778    0.2500    0.5714    0.6000
## Neg Pred Value         0.9813   0.9874   0.9852    0.9771    0.9666    0.9667
## Prevalence             0.0460   0.0540   0.0560    0.0320    0.0560    0.0560
## Detection Rate         0.0280   0.0420   0.0420    0.0100    0.0240    0.0240
## Detection Prevalence   0.0380   0.0500   0.0540    0.0400    0.0420    0.0400
## Balanced Accuracy      0.7991   0.8847   0.8686    0.6408    0.7048    0.7058
##                      Class: 13 Class: 14 Class: 15 Class: 16 Class: 17
## Sensitivity             0.3500    0.8182    0.5882    0.7083    0.3478
## Specificity             0.9625    0.9749    0.9834    0.9832    0.9769
## Pos Pred Value          0.2800    0.6000    0.5556    0.6800    0.4211
## Neg Pred Value          0.9726    0.9915    0.9855    0.9853    0.9688
## Prevalence              0.0400    0.0440    0.0340    0.0480    0.0460
## Detection Rate          0.0140    0.0360    0.0200    0.0340    0.0160
## Detection Prevalence    0.0500    0.0600    0.0360    0.0500    0.0380
## Balanced Accuracy       0.6562    0.8965    0.7858    0.8458    0.6624
##                      Class: 18 Class: 19 Class: 20 Class: 21 Class: 22
## Sensitivity             0.5789    0.2903    0.2727    0.4783    0.1364
## Specificity             0.9584    0.9829    0.9791    0.9581    0.9791
## Pos Pred Value          0.3548    0.5294    0.3750    0.3548    0.2308
## Neg Pred Value          0.9829    0.9545    0.9669    0.9744    0.9610
## Prevalence              0.0380    0.0620    0.0440    0.0460    0.0440
## Detection Rate          0.0220    0.0180    0.0120    0.0220    0.0060
## Detection Prevalence    0.0620    0.0340    0.0320    0.0620    0.0260
## Balanced Accuracy       0.7687    0.6366    0.6259    0.7182    0.5577
```

Summarize Running Time

```r
cat("Baseline model: Time for constructing training features=", tm_feature_train_base[1], "s \n")
```

```
## Baseline model: Time for constructing training features= 1.66 s
```

```r
cat("Baseline model: Time for constructing testing features=", tm_feature_test_base[1], "s \n")
```

```
## Baseline model: Time for constructing testing features= 0.14 s
```

```r
cat("Baseline model: Time for training model=", tm_train_base[1], "s \n")
```

```
## Baseline model: Time for training model= 1497.7 s
```

```r
cat("Baseline model: Time for testing model=", tm_test_base[1], "s \n")
```

```
## Baseline model: Time for testing model= 15.64 s
```

```r
cat("Advanced model: Time for constructing training features=", tm_feature_train_advanced, "s \n")
```

```
## Advanced model: Time for constructing training features= 78.54291 s
```

```r
cat("Advanced model: Time for constructing testing features=", tm_feature_test_advanced, "s \n")
```

```
## Advanced model: Time for constructing testing features= 2.89789 s
```

```r
cat("Advanced model: Time for training model=", tm_train_advanced[1], "s \n")
```

```
## Advanced model: Time for training model= 4.01 s
```

```r
cat("Advanced model: Time for testing model=", tm_test_advanced[1], "s \n")
```

```
## Advanced model: Time for testing model= 0.38 s
```

###Reference - Du, S., Tao, Y., & Martinez, A. M. (2014). Compound facial expressions of emotion. Proceedings of the National Academy of Sciences, 111(15), E1454-E1462.