

# Project 3 - Main File

*Group 3: Chaoyue Tan, Han Lin, Hongyang Yang, Peilin Qiu, Wyatt Thompson*

Relevant packages needed for this file

```
list.of.packages <- c("e1071", "ggplot2", "gbm", "caret", "randomForest", "EBImage", "xgboost", "OpenImageR", "stringr")

packages.needed=setdiff(list.of.packages,
                        intersect(installed.packages()[,1],
                                list.of.packages))

if(length(packages.needed)>0){
  install.packages(packages.needed, dependencies = TRUE)
}

library("stringr")
library("gbm")
library("ggplot2")
library("caret")
library("randomForest")
library("EBImage")
library("xgboost")
library("OpenImageR")
```

**Step 0: specify directories.**

```
#setwd('.')
#image_train.dir # raw image folder
label_train.dir <- '../data/label_train.csv'
feature_sift_train.dir <- '../data/sift_train.csv'
# feature_hog_train.dir
# feature_lbp_train.dir
#
#
image_test.dir <- '../data/images'
#
feature_sift_test.dir <- '../data/sift_test.csv'
feature_hog_test.dir <- '../output/feature_hog_test.RData'
#feature_sift_pca_test.dir <- '../output/feature_sift_pca_test.RData'
feature_lbp_test.dir <- '../output/feature_lbp_test.RData'
feature_gray_test.dir <- '../output/feature_gray_test.RData'
feature_test.dir <- '../output/feature_test.RData'
#
# predict_label_base.dir
# predict_label_adv1.dir
# predict_label_adv2.dir
```

**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 for GBM
- (number) K, the number of CV folds
- (T/F) Out of Bag Estimate (similar to cross-validation) on training set for Random Forest
- (T/F) process features for training set
- (T/F) run evaluation on an independent test set

```

run.cv=FALSE # run cross-validation on the training set
K_folds <- 3 # number of CV folds
run.feature.train=FALSE # process features for all pictures

run.train = FALSE # if true, train model on training data, else use saved model
run.test=TRUE # run evaluation on an independent test set
run.feature.test=FALSE # process features for test set

```

## Step 2: Import training label and sift feature

```

label_train <- as.vector(read.csv(label_train.dir,as.is = T)[,2])
sift_train_feature <- read.csv(feature_sift_train.dir,as.is=T)[,-1]

```

## Step 3: construct visual features

```

source("../lib/feature.R")
#extract_feature

#if(run.feature.train){
  #tm_extract_HOG <- system.time(HOG_features <- HOG_extract(image_train.dir))

  #print(tm_extract_HOG)

  #tm_feature_train <-
  # system.time(feature_train <- extract_feature(sift_train_feature,image_train.dir))[3]
  #save(feature_train,file=feature_output.dir)
#}else{

feature_train <- get(load('../output/feature_train.RData'))

#}

if(run.feature.test){

  sift_test_feature <- read.csv(feature_sift_test.dir,as.is=T)[,-1]
  #tm_feature_pca <- system.time(feature_sift_pca_test <- feature.pca(sift_test_feature))
  tm_feature_hog <- system.time(feature_hog_test <- HOG_extract(image_test.dir))[3]
  tm_feature_gray <- system.time(feature_gray_test <- gray_extractFeature(image_test.dir))[3]

  tm_feature_test <- tm_feature_gray + tm_feature_hog

  #save(feature_sift_pca_test,file = feature_sift_pca_test.dir)
  save(feature_hog_test,file = feature_hog_test.dir)
  save(feature_gray_test,file = feature_gray_test.dir)

  #load(feature_sift_pca_test.dir)
  load(feature_hog_test.dir)
  load(feature_gray_test.dir)

  #feature_lbp_test <- read.csv(feature_lbp_test.dir,as.is=T)[1:100,]
  feature_lbp_test <- get(load(feature_lbp_test.dir))

```

```

#feature_gray_test <- read.csv(feature_gray_test.dir,as.is=T)
#feature_lbp_test <- get(load(feature_lbp_test.dir))

feature_test <- cbind(sift_test_feature,feature_hog_test,feature_lbp_test,feature_gray_test)
save(feature_test,file = feature_test.dir)

}else{
  feature_test <- get(load(feature_test.dir)) # feature_test
}

# if(run.evaluation){
#   load('../output/train_index.RData')
#   #train_index <- sort(sample(1:length(label_train),0.7*length(label_train)))
#   sift_train_feature <- sift_train_feature[train_index,]
#   feature_train <- feature_train[train_index,]
#   feature_test <- feature_train[-train_index,]
#   label_train <- label_train[train_index]
#   label_test <- label_train[-train_index]
# }

```

#### Step 4: Train baseline model (GBM with SIFT)

```

source('../lib/train.R')
source('../lib/cross_validation.R')

if(run.train){
  tm_train_base <- system.time(base_fit <- gbm_train(sift_train_feature,label_train,run.cv = run.cv))[3]
  save(base_fit,file='../output/base_fit.RData')
}else{
  load('../output/base_fit.RData')
  tm_train_base <- 152.279
}

```

#### Step 5: Train advanced models

##### XGBoost with SIFT + HOG + LBP + Gray

```

if(run.train){
  tm_train_xgb <- system.time(xgb_fit <- xgb_train(feature_train,label_train,run.cv=run.cv))[3]
  save(xgb_fit,file='../output/xgb_fit.RData')
}else{
  load('../output/xgb_fit.RData')
  tm_train_xgb <- 193.722
}

```

#### Step 6: Make prediction

```

source('../lib/test.R')
if(run.test){
  sift_test_feature <- read.csv(feature_sift_test.dir,as.is = T)[,-1]

  tm_predict_base <- system.time(pred_label_base <- gbm_test(base_fit,sift_test_feature))[3]
  #save(pred_label_base,file=predict_label_base.dir)
}

```

```

tm_predict_xgb <- system.time(pred_label_xgb <- xgb_test(xgb_fit,feature_test))[3]
#save(pred_label_gbm,file=predict_label_gbm.dir)

pred_label <- cbind(pred_label_base,pred_label_xgb)
write.csv(pred_label,'../output/labels.csv')
}

```

## Step 7: Summarize running time

```

#cat("Time for constructing test features=", tm_feature_train[1], "s \n")
cat("Time for training model=", tm_train_base, "s \n")

```

```
## Time for training model= 152.279 s
```

```
cat("Time for making prediction=", tm_train_xgb, "s \n")
```

```
## Time for making prediction= 193.722 s
```