Image Classification of ASL Interactions

Code

B. Kozura

2022-09-20

Hide

Hide

```
# Environment Setup
library(tidyverse)
library(keras)
library(tensorflow)
library(reticulate)
library(hms)
library(imager)library(magick)
```

Hide

Hide

```
wd <- "C:/Users/bkozura/Documents/asl_images/"
setwd(wd)
set.seed(2022)
```

Hide

Hide

```
getS3method("print", "sessionInfo")(sessionInfo()[-8][], locale = FALSE)
```

```
R version 4.2.1 (2022-06-23 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 22000)

Matrix products: default

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base     

other attached packages:
 [1] caret_6.0-93     lattice_0.20-45  magick_2.7.3     imager_0.42.13   magrittr_2.0.3   hms_1.1.2       
 [7] reticulate_1.26  tensorflow_2.9.0 keras_2.9.0      forcats_0.5.2    stringr_1.4.1    dplyr_1.0.10    
[13] purrr_0.3.4      readr_2.1.2      tidyr_1.2.1      tibble_3.1.8     ggplot2_3.3.6    tidyverse_1.3.2 
```

Hide

Hide

```
py_config()
```

```
python:         C:/Users/bkozura/AppData/Local/r-miniconda/envs/r-reticulate/python.exe
libpython:      C:/Users/bkozura/AppData/Local/r-miniconda/envs/r-reticulate/python38.dll
pythonhome:     C:/Users/bkozura/AppData/Local/r-miniconda/envs/r-reticulate
version:        3.8.13 | packaged by conda-forge | (default, Mar 25 2022, 05:59:00) [MSC v.1929 64 bit (AMD64)]
Architecture:   64bit
numpy:          C:/Users/bkozura/AppData/Local/r-miniconda/envs/r-reticulate/Lib/site-packages/numpy
numpy_version:  1.23.3
```

Hide

Hide

```
tf_config()
```

```
TensorFlow v2.9.2 ()
Python v3.8 (C:/Users/bkozura/AppData/Local/r-miniconda/envs/r-reticulate/python.exe)
```

Hide

Hide

```
# -Load Data
path_train <- paste(wd,"/train", sep="")
path_test <- paste(wd,"/test", sep="")
label_list <- dir(path_train)
n_classes <- length(label_list)  # 27 Classes
label_list
```

```
 [1] "A"     "B"     "Blank" "C"     "D"     "E"     "F"     "G"     "H"     "I"     "J"     "K"    
[13] "L"     "M"     "N"     "O"     "P"     "Q"     "R"     "S"     "T"     "U"     "V"     "W"    
[25] "X"     "Y"     "Z"    
```

Hide

Hide

```
img_size <- 227 # height and width
target_size <- c(img_size, img_size)

train_data_gen <- image_data_generator(rescale = 1/255, validation_split = .2)
test_data_gen <- image_data_generator(rescale = 1/255)


train_images <- flow_images_from_directory(path_train,
                                           train_data_gen,
                                           subset = 'training',
                                           target_size = target_size,
                                           class_mode = "categorical",
                                           shuffle=TRUE,
                                           classes = label_list,
                                           seed = 2022)
```

```
Found 19440 images belonging to 27 classes.
```

Hide

Hide

```
validation_images <- flow_images_from_directory(path_train,
                                                train_data_gen,
                                                subset = 'validation',
                                                target_size = target_size,
                                                class_mode = "categorical",
                                                classes = label_list,
                                                seed = 2022)
```

```
Found 4860 images belonging to 27 classes.
```

Hide

Hide

```
test_images <- flow_images_from_directory(path_test,
                                          test_data_gen,
                                          target_size = target_size,
                                          class_mode = "categorical",
                                          classes = label_list,
                                          shuffle = FALSE,
                                          seed = 2022)
```

```
Found 2700 images belonging to 27 classes.
```

Hide

Hide

```
table(train_images$classes)
```

```

  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 
720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 720 
 26 
720 
```

My model is based on AlexNet Architecture, with an additional dropout layer before the first dense layer.

Hide

Hide

```
# Custom Layer functions

con_layer <- function(x,filters, ksize, bias, strides=1, padding='SAME'){
  layer_conv_2d(x, filters = filters,
                kernel_size = c(ksize, ksize),
                padding = padding,
                activation = "relu",
                strides=c(strides, strides),
                use_bias = TRUE,
                kernel_initializer='random_uniform',
                bias_initializer='random_uniform')}

pool_layer <- function(x){
  layer_max_pooling_2d(x,
                       pool_size = c(3, 3),
                       strides=2,
                       padding= 'VALID') }

fully_connected_layer <- function(units, activation = "relu", name){
  layer_dense(units = units,
              activation = activation,
              kernel_initializer='random_uniform',
              bias_initializer='random_uniform') }
```

Hide

Hide

```
# Model Function

model_a <- function(learning_rate=0.0001, dropout_rate=0.00000001){
  k_clear_session()

  model <- keras_model_sequential(input_shape = c(img_size,img_size,3)) %>%

    con_layer(filters=96, ksize=11, strides = 4, padding = 'VALID') %>%
    pool_layer() %>%

    con_layer(filters=256, ksize=5) %>%
    pool_layer() %>%

    con_layer(filters=384, ksize=3) %>%
    con_layer(filters=384, ksize=3) %>%
    con_layer(filters=256, ksize=3) %>%
    pool_layer() %>%
    layer_flatten() %>%
    layer_dropout(dropout_rate)%>%

    layer_dense(units = 4096,
                activation = "relu",
                kernel_initializer='random_uniform',
                bias_initializer='random_uniform')  %>%

    layer_dense(units = 4096,
                activation = "relu",
                kernel_initializer='random_uniform',
                bias_initializer='random_uniform')  %>%

    layer_dense(name="Output", units = n_classes, activation = "softmax")

  model %>% compile(
    loss = "categorical_crossentropy",
    optimizer = optimizer_adam(learning_rate= learning_rate),
    metrics = "accuracy"
  )


  return(model)
}
```

Hide

Hide

```
# Create Model with chosen parameters
my_model <- model_a(learning_rate=0.00001)
summary(my_model)
```

```
Model: "sequential"
____________________________________________________________________________________________________
 Layer (type)                                Output Shape                               Param #
====================================================================================================
 conv2d_4 (Conv2D)                           (None, 55, 55, 96)                         34944
 max_pooling2d_2 (MaxPooling2D)              (None, 27, 27, 96)                         0
 conv2d_3 (Conv2D)                           (None, 27, 27, 256)                        614656
 max_pooling2d_1 (MaxPooling2D)              (None, 13, 13, 256)                        0
 conv2d_2 (Conv2D)                           (None, 13, 13, 384)                        885120
 conv2d_1 (Conv2D)                           (None, 13, 13, 384)                        1327488
 conv2d (Conv2D)                             (None, 13, 13, 256)                        884992
 max_pooling2d (MaxPooling2D)                (None, 6, 6, 256)                          0
 flatten (Flatten)                           (None, 9216)                               0
 dropout (Dropout)                           (None, 9216)                               0
 dense_1 (Dense)                             (None, 4096)                               37752832
 dense (Dense)                               (None, 4096)                               16781312
 Output (Dense)                              (None, 27)                                 110619
====================================================================================================
Total params: 58,391,963
Trainable params: 58,391,963
Non-trainable params: 0
____________________________________________________________________________________________________
```

Hide

Hide

```
# Save Pre-Training Model
my_model %>% save_model_tf("/asl_alexnet_model_pretraining")
```

Hide

Hide

```
# Train Model

start_time <- Sys.time()
batch_size <- 128
epochs <- 25
hist <- my_model %>% fit(
  train_images,
  epochs = epochs,
  validation_data = validation_images,
  verbose = 2,
  callbacks = list(callback_early_stopping(monitor = "val_accuracy", min_delta=0.001, patience = 2, restore_best_weights = T
RUE))
)
```

```
Epoch 1/25
608/608 - 443s - loss: 2.8366 - accuracy: 0.1977 - val_loss: 2.1524 - val_accuracy: 0.3747 - 443s/epoch - 728ms/step
Epoch 2/25
608/608 - 456s - loss: 1.5258 - accuracy: 0.5501 - val_loss: 1.1808 - val_accuracy: 0.6634 - 456s/epoch - 749ms/step
Epoch 3/25
608/608 - 487s - loss: 0.8508 - accuracy: 0.7462 - val_loss: 0.7921 - val_accuracy: 0.7650 - 487s/epoch - 800ms/step
Epoch 4/25
608/608 - 475s - loss: 0.5512 - accuracy: 0.8415 - val_loss: 0.6692 - val_accuracy: 0.7942 - 475s/epoch - 781ms/step
Epoch 5/25
608/608 - 475s - loss: 0.3845 - accuracy: 0.8890 - val_loss: 0.5377 - val_accuracy: 0.8298 - 475s/epoch - 782ms/step
Epoch 6/25
608/608 - 471s - loss: 0.2749 - accuracy: 0.9253 - val_loss: 0.4441 - val_accuracy: 0.8687 - 471s/epoch - 775ms/step
Epoch 7/25
608/608 - 474s - loss: 0.2011 - accuracy: 0.9435 - val_loss: 0.4011 - val_accuracy: 0.8788 - 474s/epoch - 779ms/step
Epoch 8/25
608/608 - 480s - loss: 0.1386 - accuracy: 0.9634 - val_loss: 0.4122 - val_accuracy: 0.8782 - 480s/epoch - 789ms/step
Epoch 9/25
608/608 - 487s - loss: 0.1008 - accuracy: 0.9747 - val_loss: 0.3648 - val_accuracy: 0.8955 - 487s/epoch - 800ms/step
Epoch 10/25
608/608 - 468s - loss: 0.0793 - accuracy: 0.9808 - val_loss: 0.3577 - val_accuracy: 0.8986 - 468s/epoch - 771ms/step
Epoch 11/25
608/608 - 468s - loss: 0.0537 - accuracy: 0.9880 - val_loss: 0.3442 - val_accuracy: 0.9072 - 468s/epoch - 770ms/step
Epoch 12/25
608/608 - 461s - loss: 0.0461 - accuracy: 0.9902 - val_loss: 0.3221 - val_accuracy: 0.9121 - 461s/epoch - 759ms/step
Epoch 13/25
608/608 - 453s - loss: 0.0453 - accuracy: 0.9899 - val_loss: 0.2918 - val_accuracy: 0.9218 - 453s/epoch - 745ms/step
Epoch 14/25
608/608 - 470s - loss: 0.0419 - accuracy: 0.9903 - val_loss: 0.4364 - val_accuracy: 0.8899 - 470s/epoch - 773ms/step
Epoch 15/25
608/608 - 468s - loss: 0.0307 - accuracy: 0.9939 - val_loss: 0.2779 - val_accuracy: 0.9270 - 468s/epoch - 769ms/step
Epoch 16/25
608/608 - 485s - loss: 0.0169 - accuracy: 0.9963 - val_loss: 0.3107 - val_accuracy: 0.9204 - 485s/epoch - 798ms/step
Epoch 17/25
608/608 - 488s - loss: 0.0339 - accuracy: 0.9915 - val_loss: 0.3459 - val_accuracy: 0.9060 - 488s/epoch - 802ms/step
```

Hide

Hide

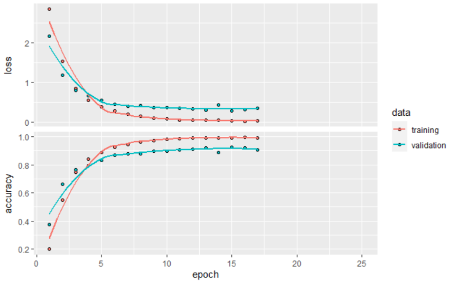
```
end_time  <- Sys.time()
end_time - start_time
```

```
Time difference of 2.224575 hours
```

Hide

Hide

```
plot(hist)
```



Hide

Hide

```r
# Predict on Example Image
test_image <- image_load("example.png",target_size = target_size)
x <- image_to_array(test_image)
x <- array_reshape(x, c(1, dim(x)))
x <- x/255
pred <- my_model %>% predict(x)
```

```
1/1 [==============================] - 0s 99ms/step
⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠
1/1 [==============================] - 0s 99ms/step
```

Hide

Hide

```r
pred <- data.frame("Class" = label_list, "Probability" = t(pred))
pred <- pred[order(pred$Probability, decreasing=T),][1:5,]
pred$Probability <- paste(format(100*pred$Probability,2),"%")
pred
```

	Class <chr>	Probability <chr>
5	D	9.991615e+01 %
22	U	8.381176e-02 %
27	Z	3.128087e-05 %
25	X	1.599010e-07 %
15	N	5.326599e-09 %

5 rows

Hide

Hide

```r
# Check accuracy
my_model %>% evaluate(train_images)
```

```
       loss    accuracy 
0.009826553 0.999382734 
```

Hide

Hide

```r
my_model %>% evaluate(test_images)
```

```
     loss  accuracy 
 0.2397213 0.9377778 
```

Hide

Hide

```r
#Predict on test data
predictions <- my_model %>%  predict(test_images, generator = test_data_gen) %>% as.data.frame
```

Hide

Hide

```r
names(predictions) <- label_list
predictions$predicted_class <-  label_list[apply(predictions,1,which.max)]
predictions$true_class <- label_list[test_images$classes+1]
 cmatrix <- as.matrix(table(Actual = predictions$true_class, Predicted=predictions$predicted_class))
 cmatrix
```

```
       Predicted
Actual   A  B Blank  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
  A     91  0     0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  7  0
  B      0 99     0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  Blank  0  0    96  2  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
  C      0  0     0 98  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0
  D      0  0     1  0 94  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  3  0  0
  E      0  0     0  0  0 96  0  0  0  0  0  0  0  2  1  0  0  0  0  0  1  0  0  0  0  0  0
  F      0  0     0  0  0  0 99  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  G      0  0     0  0  1  0  0 98  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
  H      0  0     0  1  0  0  0  0 96  0  0  0  0  0  0  0  0  2  0  0  1  0  0  0  0  0  0
  I      0  0     0  0  0  0  0  0  0 97  0  0  0  0  0  1  0  0  1  0  1  0  0  0  0  0  0
  J      0  0     0  0  0  0  0  0  0  0 91  0  1  0  0  0  1  2  0  0  0  0  0  0  0  5  0
  K      0  1     0  0  0  0  0  0  0  0  0 94  0  1  0  0  0  0  1  0  0  2  0  0  0  1  0
  L      0  0     0  0  0  0  1  0  0  0  1  0 98  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  M      0  0     0  0  0  2  0  0  0  0  0  1  0 91  1  0  0  1  0  0  3  0  1  0  0  0  0
  N      1  0     0  0  2  1  0  0  0  0  0  0  0  4 89  0  0  0  0  0  1  1  0  0  1  0  0
  O      0  0     0  1  0  1  1  0  0  1  0  0  0  0  0 94  0  0  0  1  0  0  0  0  1  0  0
  P      0  0     0  0  0  0  0  0  1  0  1  0  0  0  0  0 94  3  0  0  0  0  0  0  0  1  0
  Q      0  0     0  0  1  0  0  0  0  0  2  0  0  0  0  0  5 91  0  0  0  0  0  0  0  1  0
  R      0  1     0  0  0  0  1  0  0  1  0  0  0  2  0  1  0  0 89  0  0  1  0  0  4  0  0
  S      0  0     0  1  0  0  1  0  0  0  0  0  0  0  0 13  0  0  1 79  4  0  0  0  1  0  0
  T      0  0     0  0  0  1  0  0  1  0  0  0  0  0  0  1  0  0  0  2 95  0  0  0  0  0  0
  U      0  0     0  0  0  0  0  0  0  2  0  0  0  0  0  0  1  0  2  0  0 94  0  0  1  0  0
```

```
  V      0  1     0  1  1  0  1  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0 91  2  0  0  0
  W      0  0     0  0  0  0  1  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  3 94  0  0  0
  X      0  0     0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  2  0 95  0  1
  Y      2  0     0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0 96  0
  Z      0  1     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  2  1  0  0 93
```

Hide

Hide

```
summary(metrics[,c(-1)])
```

```
  precision          recall             f1        
 Min.   :0.8468   Min.   :0.7900   Min.   :0.8681  
 1st Qu.:0.9249   1st Qu.:0.9100   1st Qu.:0.9160  
 Median :0.9495   Median :0.9400   Median :0.9447  
 Mean   :0.9396   Mean   :0.9378   Mean   :0.9378  
 3rd Qu.:0.9677   3rd Qu.:0.9600   3rd Qu.:0.9606  
 Max.   :1.0000   Max.   :0.9900   Max.   :0.9899  
```

Hide

Hide

```
macros <- data.frame("Value"=colMeans(metrics[,c(-1)]))
macros["accuracy",] <- rbind(accuracy)
macros["rand_accuracy",] <- rbind(sum(act_dist*pred_dist))
macros["kappa",] <- rbind((macros["accuracy",] - macros["rand_accuracy",]) / (1 - macros["rand_accuracy",]))
macros$Value <- round(as.numeric(macros$Value),6)
macros
```

	Value <dbl>
precision	0.939569
recall	0.937778
f1	0.937799
accuracy	0.937778
rand_accuracy	0.037037
kappa	0.935385

6 rows

Hide

Hide

```
# Save Trained Model
my_model %>% save_model_tf("/asl_alexnet_model")
```

Hide

Hide

```
# Can load anytime using:
# my_model <- load_model_tf("~/asl_images/asl_alexnet_model")
```