KHK model report

Data acquisition

1. Custom data  
    Ken, ali and kaho have created about 3000 fist and palm gesture images
2. Online data

* figshare dataset includes 4 categories of hand gestures: open hand, fist, right hand, left hand + one negative class  
  <https://figshare.com/articles/Hand_gestures_raw_images/11394585/1>
* Hand Images Databases have 3000 palm images <https://xwww.mutah.edu.jo/biometrix/hand-images-databases.html>
* rock-paper-scissors datasets

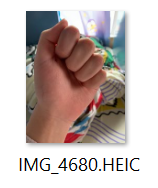
<https://github.com/alessandro-giusti/rock-paper-scissors/tree/master/datasets/final/testing/c0>

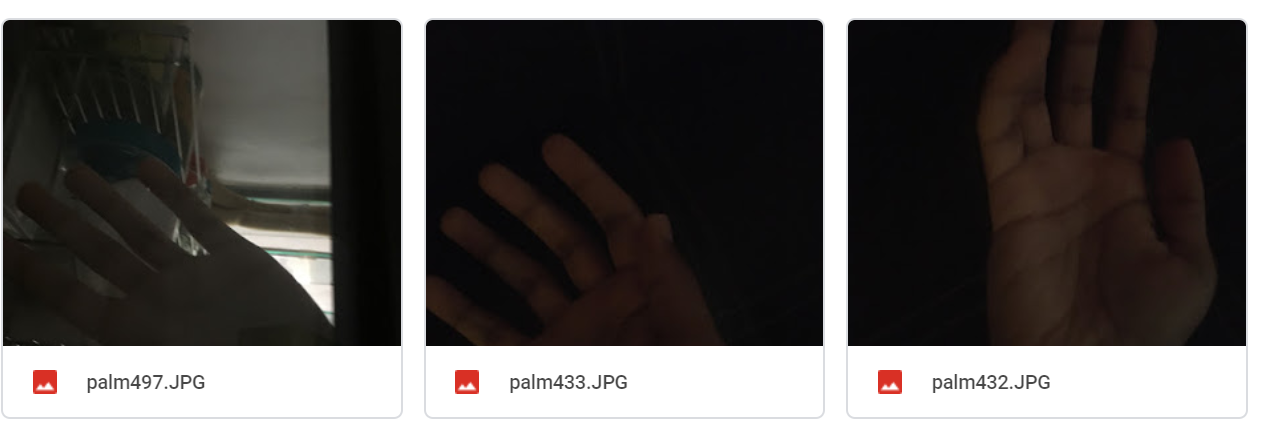
* manually download some image data form image provider such as google image, unsplash, pixabay.   
  https://unsplash.com

1. possible dataset (**did not use in this model**)

* <https://www.kaggle.com/gti-upm/leapgestrecog/version/1>
* <https://www.gti.ssr.upm.es/data/HandGesture_database.html>
* <https://www.kaggle.com/datamunge/sign-language-mnist>
* <https://www.kaggle.com/ardamavi/sign-language-digits-dataset>
* <https://www-prima.inrialpes.fr/FGnet/data/10-Gesture/gestures/main.html>
* <https://sites.google.com/view/11khands>

Data cleaning



1. Remove .HEIC image file that iphone uses because opencv has trouble processing it. There was a small amount of .HEIC file like about 15 images in   
   custom made image dataset.
2. Remove dark image which also contain about 10 images in custom made image dataset.  
   

Data preprocessing

*rootPath = '/content/gdrive/My Drive/DATASETS/fileshareHand/raw\_open\_hand/'*

*indices = os.listdir(rootPath)*

*np.random.seed(502)*

*np.random.shuffle(indices)*

*# http://cs230.stanford.edu/blog/split/*

*print("total len(indices): {}".format(len(indices)))*

*split\_1 = int(0.8 \* len(indices))*

*train\_filenames = indices[:split\_1]*

*print("total len(train): {}".format(len(train\_filenames)))*

*test\_filenames = indices[split\_1:]*

*print("total len(test): {}".format(len(test\_filenames)))*

*# move file to file , use os.rename() function*

*outputTrainFile = "/content/gdrive/My Drive/DATASETS/train/palm\_train/"*

*outputTestFile = "/content/gdrive/My Drive/DATASETS/test/palm\_test/"*

*for i in range(0,len(train\_filenames)):*

*oldPath= rootPath+train\_filenames[i]*

*newPath= outputTrainFile+train\_filenames[i]*

*os.rename(oldPath,newPath)*

*# if i==0:*

*# print(oldPath,newPath)*

*# else:*

*# break*

*for j in range(0,len(test\_filenames)):*

*oldPath= rootPath+test\_filenames[j]*

*newPath= outputTestFile+test\_filenames[j]*

*os.rename(oldPath,newPath)*

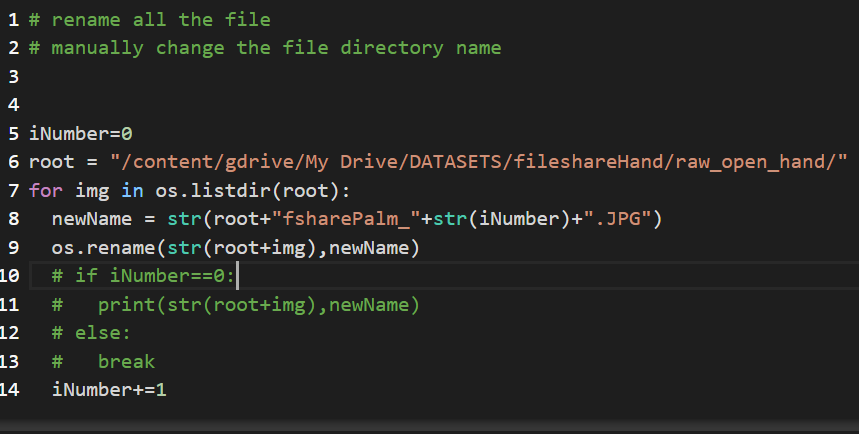
*# if j==0:*

*# print(oldPath,newPath)*

*# else:*

*# break*

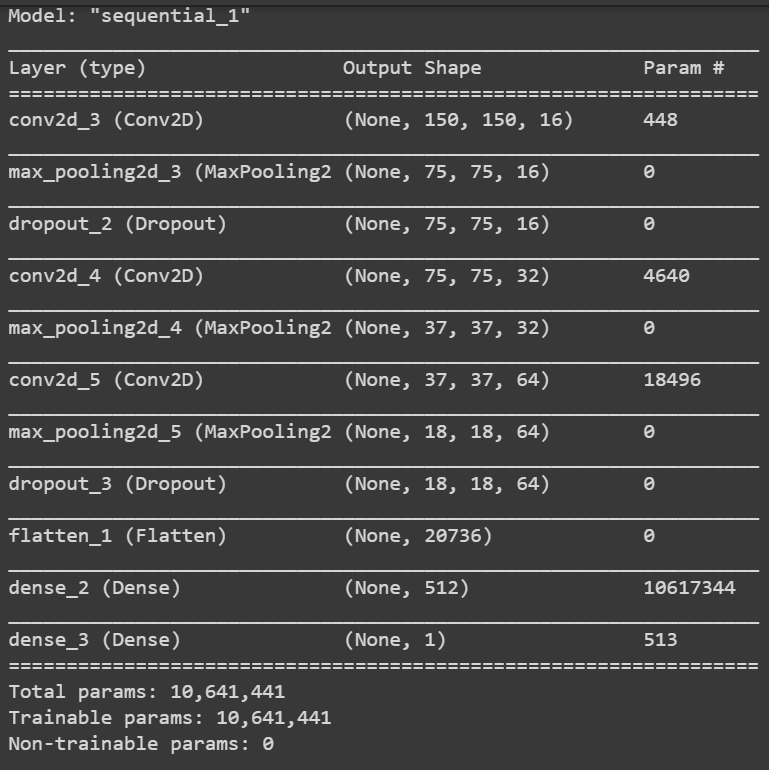
1. 8:2 train & test split   
     
   first shuffle the data,  
   then split data set to  
   training set by total# \*0.8   
     
   use os.rename()  
   function to put training set to training file directory.
2. Rename some of the image for clarification.   
     
   For example, In second round of creating custom data I rename the data that I created as palmv2\_01jpg, fistv2\_01.jpg.

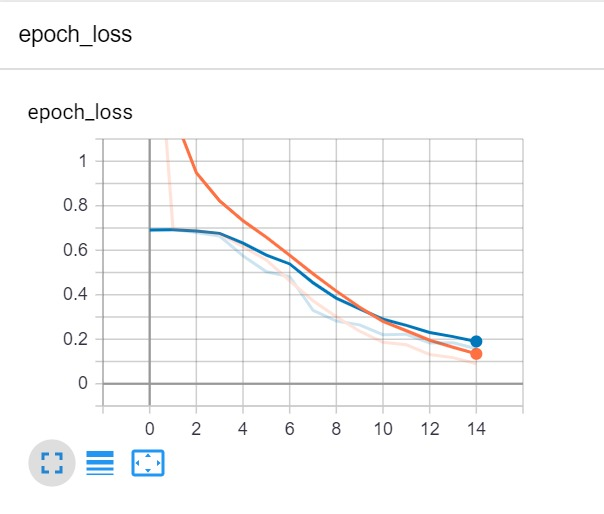
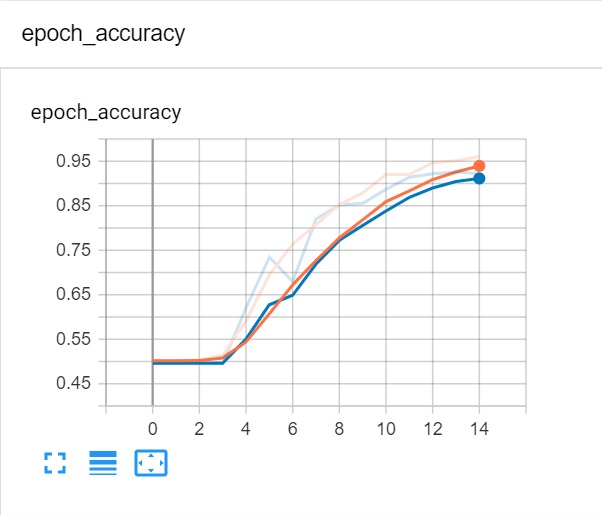
kaho’s second round image is named as KHpalmv2\_01.jpg, KHfistv2\_01.jpg.  
  
data from the fileshare is named fshareFist\_01.jpg, fsharePalm\_01.jpg  


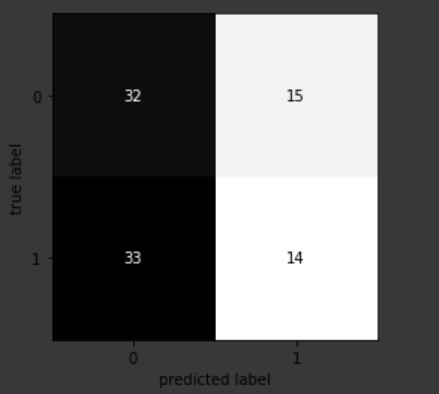
Modeling

Stage ONE:

First model

  
initial custom made image is used only. (kaho ken ali) total size = 1740

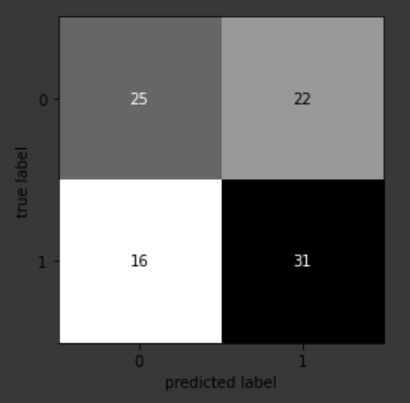




Training and validation data are similar, so the accuracy is high. However result is not good. Test accuracy is around .50

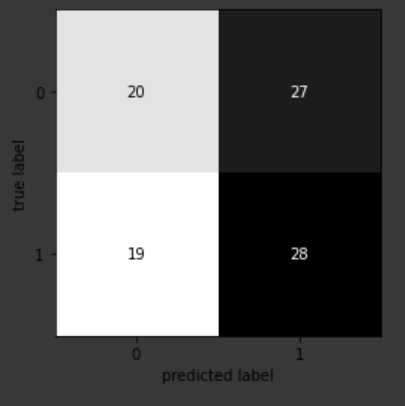
Data with only 3 of our hands, it is homogeneous, we must increase the variety and quantity.

*Adding data augmentation*

**

Test acc is around .55, slightly better than previous model, but still sucks.

*No drop out layers*

**

Test acc is .61,

Stage TWO:

increased image size, including the custom image with diverse background and online datasets.

**Model name : training\_7\_**

Used equal size training ,validation and test dataset

**Architecture** :

model= Sequential([

Conv2D(64, 3, padding='same', activation='relu', input\_shape=(IMG\_HEIGHT, IMG\_WIDTH ,3)),

#BatchNormalization(),

MaxPooling2D(pool\_size=(2, 2)),

Conv2D(64, 3, padding='same', activation='relu'),

#BatchNormalization(),

MaxPooling2D(pool\_size=(2, 2)),

#Dropout(rate=0.2),

Conv2D(128, 3, padding='same', activation='relu'),

#BatchNormalization(),

MaxPooling2D(pool\_size=(2, 2)),

#Dropout(rate=0.2),

Flatten(),

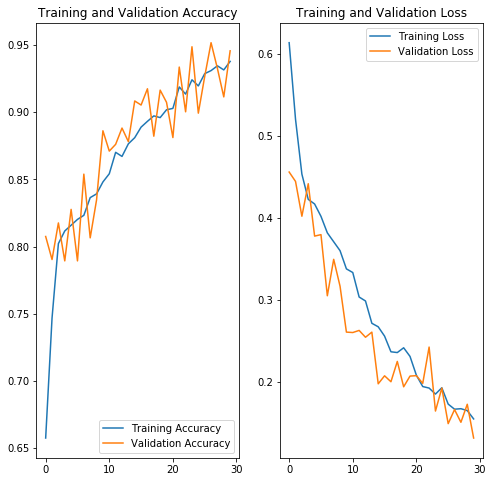
Dropout(rate=0.2),

Dense(512, activation='relu'),

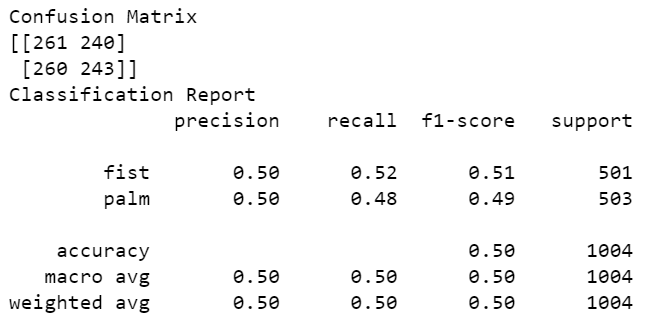
Dense(1)

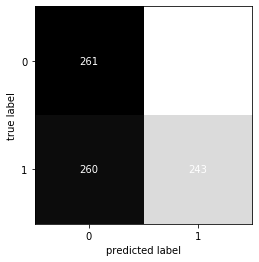
])

**Remarks**: didn’t add dropout before flatten() layer, changed the conv2d filter param to higher

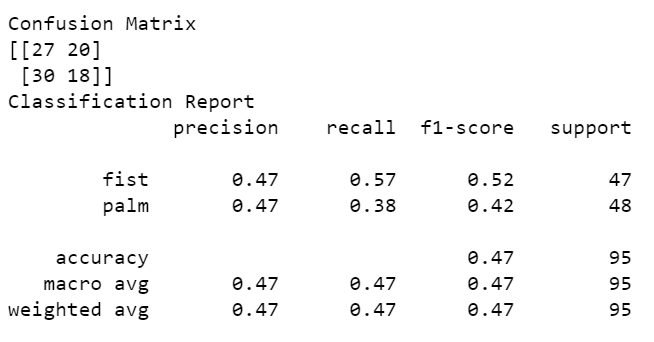


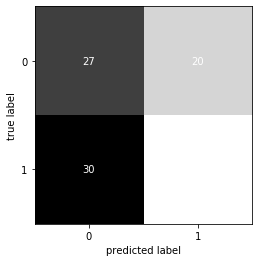
Result in val set





Result in test set





Problem here : in second attempt to predict test and val dataset the model predicted all input data as 0

**Possible reason of problem:**

* [Dying ReLU](https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks) https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks
* Too small / too big learning rate
* Vastly unbalanced data
* Initialization / Optimization  
    
  <https://stackoverflow.com/questions/41488279/neural-network-always-predicts-the-same-class>

**Possible action to tackle problem:**

* decrease learning rate from 0.001 to 0.0001 🗸
* switch activate function from relu to LeakyRelu 🗸

note(LeakyReLU): https://adventuresinmachinelearning.com/vanishing-gradient-problem-tensorflow/

**Model name: training\_8\_LeakyRelu**

Used equal size training, validation and test dataset

**Architecture** :

model= Sequential([

Conv2D(32, 3, padding='same', activation=LeakyReLU(alpha=0.01), input\_shape=(IMG\_HEIGHT, IMG\_WIDTH ,3)),

#BatchNormalization(),

MaxPooling2D(pool\_size=(2, 2)),

Conv2D(64, 3, padding='same', activation=LeakyReLU(alpha=0.01)),

#BatchNormalization(),

MaxPooling2D(pool\_size=(2, 2)),

#Dropout(rate=0.2),

Conv2D(128, 3, padding='same', activation=LeakyReLU(alpha=0.01)),

#BatchNormalization(),

MaxPooling2D(pool\_size=(2, 2)),

#Dropout(rate=0.2),

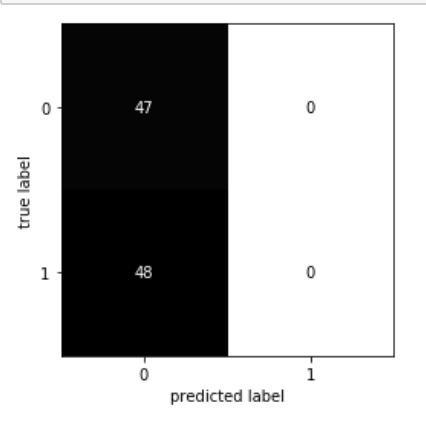
Flatten(),

Dropout(rate=0.2),

Dense(512, activation=LeakyReLU(alpha=0.01)),

Dense(1)

])



Investigation on weird confusion matrix above

Bebugging:

1. Not the data
2. Not the training
3. correct data input format
4. correct model checkpoint saving
5. incorrect in model loading

**Problem in model loading :**

**I loaded the pre-trained model weights before the creation of model, it is a logic error.**

**Solution : First create the model, then load the pre-trained weights into the created model.**