Dog Breed Identification

Keshav Bansal 101703285 COE13

In [1]:

```
!pip install -q kaggle
```

In [2]:

```
from google.colab import files
files.upload()
```

Choose File No file selected

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
```

Out[2]:

```
{'kaggle.json': b'{"username":"kbansal17","key":"4146685e3f897e16f74f9668abd27eb3"}'}
```

In [3]:

```
# make a new folder in root directory
!mkdir ~/.kaggle
```

In [4]:

```
# copy kaggle.json file in the folder
!cp kaggle.json ~/.kaggle/
```

In [5]:

```
# change the permissions of file
!chmod 600 ~/.kaggle/kaggle.json
```

In [6]:

```
!kaggle datasets list
```

```
Warning: Looks like you're using an outdated API Version, please consider updating (server 1.5.6 /
client 1.5.4)
ref                                                    title
size  lastUpdated          downloadCount
-----------------------------------------------------  --------------------------------------------
-----  -------------------  -------------
shivan118/healthcare-analytics                         Healthcare Analytics
2MB  2020-09-13 17:40:05           1337
datatattle/covid-19-nlp-text-classification            Coronavirus tweets NLP - Text Classificatior
4MB  2020-09-08 11:40:11            784
anmolkumar/health-insurance-cross-sell-prediction      Health Insurance Cross Sell Prediction □ □
6MB  2020-09-11 18:39:31           1916
Cornell-University/arxiv                               arXiv Dataset
888MB  2020-09-22 15:33:49           3007
nipunarora8/age-gender-and-ethnicity-face-data-csv     AGE, GENDER AND ETHNICITY (FACE DATA) CSV
63MB  2020-09-02 13:46:38            664
yoannboyere/co2-ghg-emissionsdata                      CO2_GHG_emissions-data
147KB  2020-09-14 09:59:25            614
ramjidoolla/ipl-data-set                               IPL _Data_Set
1MB  2020-09-14 10:57:42           1180
anikannal/solar-power-generation-data                  Solar Power Generation Data
2MB  2020-08-18 15:52:03           4143
tunguz/us-elections-dataset                            US Elections Dataset
```

```
tunguz/us-elections-dataset                              US Elections Dataset
8MB  2020-09-17 17:02:40           2555
nehaprabhavalkar/av-healthcare-analytics-ii              AV : Healthcare Analytics II
7MB  2020-08-29 03:40:10           2001
imoore/60k-stack-overflow-questions-with-quality-rate    60k Stack Overflow Questions with Quality R
ating   21MB  2020-09-19 20:53:26           1214
jmmvutu/summer-products-and-sales-in-ecommerce-wish      Sales of summer clothes in E-commerce Wish
376KB  2020-08-23 15:16:46           6394
ruchi798/bookcrossing-dataset                            Book-Crossing: User review ratings
25MB  2020-08-11 10:40:25           1139
google/tinyquickdraw                                     QuickDraw Sketches
11GB  2018-04-18 19:38:04           2316
agirlcoding/all-space-missions-from-1957                 All Space Missions from 1957
101KB  2020-08-13 16:18:58           3382
ihelon/lego-minifigures-classification                   LEGO Minifigures
19MB  2020-09-22 07:33:22           1167
datasnaek/youtube-new                                    Trending YouTube Video Statistics
201MB  2019-06-03 00:56:47         105482
zynicide/wine-reviews                                    Wine Reviews
51MB  2017-11-27 17:08:04         112912
residentmario/ramen-ratings                              Ramen Ratings
40KB  2018-01-11 16:04:39          13740
datasnaek/chess                                          Chess Game Dataset (Lichess)
3MB  2017-09-04 03:09:09           9029
```

In [7]:

```
!pip install --upgrade --force-reinstall --no-deps kaggle
```

```
Collecting kaggle
  Downloading
https://files.pythonhosted.org/packages/fc/14/9db40d8d6230655e76fa12166006f952da4697c003610022683c5
15f/kaggle-1.5.8.tar.gz (59kB)
     |████████████████████████████████| 61kB 1.9MB/s
Building wheels for collected packages: kaggle
  Building wheel for kaggle (setup.py) ... done
  Created wheel for kaggle: filename=kaggle-1.5.8-cp36-none-any.whl size=73275
sha256=438ccca3854508e71f9c26befdf99ce29828256dcf9536cf5957008c421bb4cc
  Stored in directory:
/root/.cache/pip/wheels/94/a7/09/68dc83c7c14fdbdf5d3f2b2da5b87e587bfc1e85df69b1130c
Successfully built kaggle
Installing collected packages: kaggle
  Found existing installation: kaggle 1.5.8
    Uninstalling kaggle-1.5.8:
      Successfully uninstalled kaggle-1.5.8
Successfully installed kaggle-1.5.8
```

In [8]:

```
!kaggle --version
```

```
Kaggle API 1.5.8
```

In [9]:

```
!kaggle competitions download -c 'dog-breed-identification'
```

```
Downloading dog-breed-identification.zip to /content
 99% 685M/691M [00:03<00:00, 169MB/s]
100% 691M/691M [00:03<00:00, 182MB/s]
```

In [10]:

```
!unzip dog-breed-identification.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: train/83bcff6b55ee179a7c123fa6103c377a.jpg
  inflating: train/83be6d622ab74a5e7e08b53eb8fd566a.jpg
  inflating: train/83c2d7419b0429b9fe953bc1b6cddbec.jpg
  inflating: train/83cf7d7cd2a759a93e2ffd95bea9c6fb.jpg
  inflating: train/83d405858f0931722ef21e8ac0adee4d.jpg
  inflating: train/fe13d46f5920f0944e6c30e54ac0e2a5.jpg
  inflating: train/fe3d08ee9e1aba1785391b42345c3fc0.jpg
  inflating: train/fe3e760d763e186541e18f303cd7caca.jpg
  inflating: train/fe426e0af99930c0ec3c9ab58b02f8dc.jpg
  inflating: train/fe49341352549164ad921a67647507f1.jpg
  inflating: train/fe4d298d682a42714f33085c9d241cc0.jpg
  inflating: train/fe50bac6c389d137ea01c9cfc7346ca8.jpg
  inflating: train/fe54e87e65fe0c68670c0dd1a923f1f0.jpg
  inflating: train/fe5e4ee18529af1af1861efd550561a3.jpg
  inflating: train/fe624532170510bd80627c0500bafc97.jpg
  inflating: train/fe7171353417898022361453894adf94.jpg
  inflating: train/fe76cbb5f172387f6a5b72739852d608.jpg
  inflating: train/fe78fc42e32174c7178b572bdcf5a129.jpg
  inflating: train/fe7ea4eb63ab5fddea120555790f9187.jpg
  inflating: train/fe8d52ab96ff238ea7d234b508010ece.jpg
  inflating: train/fe9e09be6594f626f0d711bfba10cfe0.jpg
  inflating: train/fea60fdd28de5834520134d6dc77a9a2.jpg
  inflating: train/feafd0730eae85e63a41bbc030755c59.jpg
  inflating: train/feb16cf86c9dac6d476e3c372ba5c279.jpg
  inflating: train/feb9d0ae525ca28aabff74b455e34c16.jpg
  inflating: train/febcab8eb2da444bf83336cffec7eb92.jpg
  inflating: train/fede60fb2acc02a2da0d0a05f760b7d5.jpg
  inflating: train/fee1696ae6725863f84b0da2c05ad892.jpg
  inflating: train/fee672d906b502642597ccbc6acff0bb.jpg
  inflating: train/fee98c990f4d69c6a8467dd0f0668440.jpg
  inflating: train/fef4a58219c8971820a85868a7b073f5.jpg
  inflating: train/fef5d4cdaf50cf159102e803c7d6aa9c.jpg
  inflating: train/fef9c3ab585ad3f778c549fda42c1856.jpg
  inflating: train/fefb453e43ec5e840c323538261493bd.jpg
  inflating: train/ff04baf19edbe449b39619d88da3633c.jpg
  inflating: train/ff05f3976c17fef275cc0306965b3fe4.jpg
  inflating: train/ff0931b1c82289dc2cf02f0b4a165139.jpg
  inflating: train/ff0c4e0e856f1eddcc61facca64440c9.jpg
  inflating: train/ff0d0773ee3eeb6eb90a172d6afd1ea1.jpg
  inflating: train/ff0def9dafea6e633d0d7249554fcb2c.jpg
  inflating: train/ff12508818823987d04e8fa4f5907efe.jpg
  inflating: train/ff181f0d69202b0650e6e5d76e9c13cc.jpg
  inflating: train/ff2523c07da7a6cbeeb7c8f8dafed24f.jpg
  inflating: train/ff3b935868afb51b2d0b75ddc989d058.jpg
  inflating: train/ff47baef46c5876eaf9a403cd6a54d72.jpg
  inflating: train/ff4afeb51a1473f7ba18669a8ff48bc9.jpg
  inflating: train/ff4bb57ce419cd637dd511a1b5474bff.jpg
  inflating: train/ff52a3909f5801a71161cec95d213107.jpg
  inflating: train/ff54d45962b3123bb67052e8e29a60e7.jpg
  inflating: train/ff63ed894f068da8e2bbdfda50a9a9f8.jpg
  inflating: train/ff63fa05a58473138848f80840064d23.jpg
  inflating: train/ff6f47aa8e181b6efa4d0be7b09b5628.jpg
  inflating: train/ff7334b06cee8667a7f30eb00e0b93cf.jpg
  inflating: train/ff7d9c08091acc3b18b869951feeb013.jpg
  inflating: train/ff84992beff3edd99b72718bec9448d2.jpg
  inflating: train/ff8e3fa7e04faca99af85195507ee54d.jpg
  inflating: train/ff91c3c095a50d3d7f1ab52b60e93638.jpg
  inflating: train/ffa0055ec324829882186bae29491645.jpg
  inflating: train/ffa0ad682c6670db3defce2575a2587f.jpg
  inflating: train/ffa16727a9ee462ee3f386be865b199e.jpg
  inflating: train/ffa4e1bf959425bad9228b04af40ac76.jpg
  inflating: train/ffa6a8d29ce57eb760d0f182abada4bf.jpg
  inflating: train/ffbbf7536ba86dcef3f360bda41181b4.jpg
  inflating: train/ffc1717fc5b5f7a6c76d0e4ea7c8f93a.jpg
  inflating: train/ffc2b6b9133a6413c4a013cff29f9ed2.jpg
  inflating: train/ffc532991d3cd7880d27a449ed1c4770.jpg
  inflating: train/ffca1c97cea5fada05b8646998a5b788.jpg
  inflating: train/ffcb610e811817766085054616551f9c.jpg
  inflating: train/ffcde16e7da0872c357fbc7e2168c05f.jpg
  inflating: train/ffcffab7e4beef9a9b8076ef2ca51909.jpg
  inflating: train/ffd25009d635cfd16e793503ac5edef0.jpg
  inflating: train/ffd3f636f7f379c51ba3648a9ff8254f.jpg
  inflating: train/ffe2ca6c940cddfee68fa3cc6c63213f.jpg
  inflating: train/ffe5f6d8e2bff356e9482a80a6e29aac.jpg
  inflating: train/fff43b07992508bc822f33d8ffd902ae.jpg
```

In [11]:

```
import os
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import Model
from os import getcwd
```

In [12]:

```
# Mount your drive having pretrained inceptionV3 model

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [13]:

```
path_inception = f"{getcwd()}/../content/drive/My Drive/Transfer
Learning/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5"

from tensorflow.keras.applications.inception_v3 import InceptionV3

# Create an instance of the inception model from the local pre-trained weights
local_weights_file = path_inception

pre_trained_model = InceptionV3(input_shape = (240, 240, 3),
                                include_top = False,
                                weights = None)

pre_trained_model.load_weights(local_weights_file)

# Making all the layers in pre_trained model non-trainable
for layer in pre_trained_model.layers:
    layer.trainable = False

pre_trained_model.summary()
```

```
Model: "inception_v3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 240, 240, 3) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 119, 119, 32) 864         input_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 119, 119, 32) 96          conv2d[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 119, 119, 32) 0           batch_normalization[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 117, 117, 32) 9216        activation[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 117, 117, 32) 96          conv2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 117, 117, 32) 0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 117, 117, 64) 18432       activation_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 117, 117, 64) 192         conv2d_2[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 117, 117, 64) 0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 58, 58, 64)   0           activation_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 58, 58, 80)   5120        max_pooling2d[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 58, 58, 80)   240         conv2d_3[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 58, 58, 80)   0           batch_normalization_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 56, 56, 192)  138240      activation_3[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 56, 56, 192)  576         conv2d_4[0][0]
__________________________________________________________________________________________________
activation_4 (Activation)       (None, 56, 56, 192)  0           batch_normalization_4[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 27, 27, 192)  0           activation_4[0][0]
```

```
__________________________________________________________________________________________________
mixed10 (Concatenate)           (None, 6, 6, 2048)   0           activation_85[0][0]
                                                                 mixed9_1[0][0]
                                                                 concatenate_1[0][0]
                                                                 activation_93[0][0]
==================================================================================================
Total params: 21,802,784
Trainable params: 0
Non-trainable params: 21,802,784
__________________________________________________________________________________________________
```

In [14]:

```
last_layer = pre_trained_model.get_layer('mixed7')
print(last_layer.output_shape)
last_output = last_layer.output
```

```
(None, 13, 13, 768)
```

In [15]:

```
from tensorflow.keras.optimizers import RMSprop

# Flatten the output layer to 1 dimension
x = layers.Flatten()(last_output)

# Add a fully connected layer with 1024 hidden units and ReLu activation
x = layers.Dense(1024, activation = 'relu')(x)

# Add a dropout rate of 0.2
x = layers.Dropout(0.2)(x)

# Add final softmax layer
x = layers.Dense(120, activation = 'softmax')(x)

model = Model(pre_trained_model.input, x)

model.compile(optimizer = RMSprop(lr = 0.0001),
              loss = 'categorical_crossentropy',
              metrics = ['acc'])

model.summary();
```

```
Model: "functional_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 240, 240, 3) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 119, 119, 32) 864         input_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 119, 119, 32) 96          conv2d[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 119, 119, 32) 0           batch_normalization[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 117, 117, 32) 9216        activation[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 117, 117, 32) 96          conv2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 117, 117, 32) 0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 117, 117, 64) 18432       activation_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 117, 117, 64) 192         conv2d_2[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 117, 117, 64) 0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 58, 58, 64)   0           activation_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 58, 58, 80)   5120        max_pooling2d[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 58, 58, 80)   240         conv2d_3[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 58, 58, 80)   0           batch_normalization_3[0][0]
```

```
=================================================================================================
Total params: 142,006,296
Trainable params: 133,031,032
Non-trainable params: 8,975,264
_________________________________________________________________________________________________
```

In [16]:

```
# make file ids as valid filenames
import pandas as pd
df = pd.read_csv('labels.csv');
df['id'] = df['id'] + '.jpg'
df.head()
```

Out[16]:

	id	breed
0	000bec180eb18c7604dcecc8fe0dba07.jpg	boston_bull
1	001513dfcb2ffafc82cccf4d8bbaba97.jpg	dingo
2	001cdf01b096e06d78e9e5112d419397.jpg	pekinese
3	00214f311d5d2247d5dfe4fe24b2303d.jpg	bluetick
4	0021f9ceb3235effd7fcde7f7538ed62.jpg	golden_retriever

In [17]:

```
len(df)
```

Out[17]:

```
10222
```

In [18]:

```
# random split dataframe for training and validation
import numpy as np
mask = np.random.rand(len(df)) < 0.8

train_df = df[mask]
validation_df = df[~mask]
```

In [19]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Add out data augmentation parameters to ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1/255,
                                   rotation_range = 40,
                                   width_shift_range = 0.2,
                                   height_shift_range = 0.2,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   fill_mode = 'nearest')

test_datagen = ImageDataGenerator(rescale = 1/255)

# Flow training images in batches of 60 from directory using train_datagen
train_generator = train_datagen.flow_from_dataframe(train_df,
                                                    directory = './train/',
                                                    x_col = 'id',
                                                    y_col = 'breed',
                                                    target_size = (240, 240),
                                                    batch_size = 60,
                                                    class_mode = 'categorical')

# Flow validation images in batches of 60 from directory using test_datagen
validation_generator = test_datagen.flow_from_dataframe(validation_df,
                                                        directory = './train/',
                                                        x_col = 'id',
                                                        y_col = 'breed',
                                                        target_size = (240, 240),
                                                        batch_size = 60,
                                                        class_mode = 'categorical')
```

```
Found 8129 validated image filenames belonging to 120 classes.
Found 2093 validated image filenames belonging to 120 classes.
```

In [23]:

```
from keras.callbacks import EarlyStopping
callback = EarlyStopping(monitor = 'loss', patience = 10)
```

In [20]:

```
history = model.fit_generator(train_generator,
                              validation_data = validation_generator,
                              steps_per_epoch = 10,
                              epochs = 80,
                              validation_steps = 2,
                              verbose = 1
                              callbacks = [callback])
```

```
WARNING:tensorflow:From <ipython-input-20-8b45be7ac396>:6: Model.fit_generator (from
tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/80
10/10 [==============================] - 89s 9s/step - loss: 5.4353 - acc: 0.0100 - val_loss: 4.73
47 - val_acc: 0.0417
Epoch 2/80
10/10 [==============================] - 88s 9s/step - loss: 4.7940 - acc: 0.0383 - val_loss: 4.42
91 - val_acc: 0.0667
Epoch 3/80
10/10 [==============================] - 87s 9s/step - loss: 4.6366 - acc: 0.0500 - val_loss: 4.31
34 - val_acc: 0.0750
Epoch 4/80
10/10 [==============================] - 87s 9s/step - loss: 4.4404 - acc: 0.0667 - val_loss: 4.28
53 - val_acc: 0.0917
Epoch 5/80
10/10 [==============================] - 87s 9s/step - loss: 4.3761 - acc: 0.0767 - val_loss: 4.07
56 - val_acc: 0.1417
Epoch 6/80
10/10 [==============================] - 87s 9s/step - loss: 4.2426 - acc: 0.0833 - val_loss: 3.88
60 - val_acc: 0.1500
Epoch 7/80
10/10 [==============================] - 87s 9s/step - loss: 4.0601 - acc: 0.1183 - val_loss: 3.63
79 - val_acc: 0.2167
Epoch 8/80
10/10 [==============================] - 88s 9s/step - loss: 4.0787 - acc: 0.1083 - val_loss: 3.69
26 - val_acc: 0.1750
Epoch 9/80
10/10 [==============================] - 87s 9s/step - loss: 3.8978 - acc: 0.1217 - val_loss: 3.39
68 - val_acc: 0.2333
Epoch 10/80
10/10 [==============================] - 87s 9s/step - loss: 3.8095 - acc: 0.1500 - val_loss: 3.36
79 - val_acc: 0.2250
Epoch 11/80
10/10 [==============================] - 87s 9s/step - loss: 3.6049 - acc: 0.1917 - val_loss: 3.41
00 - val_acc: 0.2250
Epoch 12/80
10/10 [==============================] - 87s 9s/step - loss: 3.5732 - acc: 0.1883 - val_loss: 3.01
09 - val_acc: 0.3250
Epoch 13/80
10/10 [==============================] - 87s 9s/step - loss: 3.5392 - acc: 0.1883 - val_loss: 3.24
21 - val_acc: 0.2667
Epoch 14/80
10/10 [==============================] - 87s 9s/step - loss: 3.3656 - acc: 0.2167 - val_loss: 2.93
43 - val_acc: 0.2833
Epoch 15/80
10/10 [==============================] - 88s 9s/step - loss: 3.2682 - acc: 0.2350 - val_loss: 2.81
62 - val_acc: 0.3083
Epoch 16/80
10/10 [==============================] - 87s 9s/step - loss: 3.2092 - acc: 0.2600 - val_loss: 2.57
```

```
10/10 [==============================] - 87s 9s/step - loss: 3.2092 - acc: 0.2600 - val_loss: 2.07
64 - val_acc: 0.4167
Epoch 17/80
10/10 [==============================] - 87s 9s/step - loss: 3.1292 - acc: 0.2633 - val_loss: 2.79
53 - val_acc: 0.3333
Epoch 18/80
10/10 [==============================] - 87s 9s/step - loss: 3.0854 - acc: 0.2600 - val_loss: 2.46
70 - val_acc: 0.3917
Epoch 19/80
10/10 [==============================] - 83s 8s/step - loss: 2.9465 - acc: 0.2707 - val_loss: 2.49
71 - val_acc: 0.3917
Epoch 20/80
10/10 [==============================] - 87s 9s/step - loss: 2.8534 - acc: 0.3250 - val_loss: 2.40
74 - val_acc: 0.4333
Epoch 21/80
10/10 [==============================] - 88s 9s/step - loss: 2.7556 - acc: 0.3417 - val_loss: 2.57
61 - val_acc: 0.3000
Epoch 22/80
10/10 [==============================] - 83s 8s/step - loss: 2.8787 - acc: 0.3040 - val_loss: 2.54
50 - val_acc: 0.3667
Epoch 23/80
10/10 [==============================] - 87s 9s/step - loss: 2.8301 - acc: 0.3050 - val_loss: 2.33
53 - val_acc: 0.3583
Epoch 24/80
10/10 [==============================] - 87s 9s/step - loss: 2.7176 - acc: 0.3450 - val_loss: 2.20
64 - val_acc: 0.3917
Epoch 25/80
10/10 [==============================] - 87s 9s/step - loss: 2.5592 - acc: 0.3667 - val_loss: 2.36
75 - val_acc: 0.3750
Epoch 26/80
10/10 [==============================] - 87s 9s/step - loss: 2.6440 - acc: 0.3383 - val_loss: 2.33
49 - val_acc: 0.4167
Epoch 27/80
10/10 [==============================] - 88s 9s/step - loss: 2.4218 - acc: 0.4100 - val_loss: 2.25
13 - val_acc: 0.4000
Epoch 28/80
10/10 [==============================] - 87s 9s/step - loss: 2.5870 - acc: 0.3433 - val_loss: 2.21
85 - val_acc: 0.3667
Epoch 29/80
10/10 [==============================] - 87s 9s/step - loss: 2.5525 - acc: 0.3267 - val_loss: 2.25
56 - val_acc: 0.4083
Epoch 30/80
10/10 [==============================] - 87s 9s/step - loss: 2.3851 - acc: 0.3983 - val_loss: 1.98
98 - val_acc: 0.4750
Epoch 31/80
10/10 [==============================] - 87s 9s/step - loss: 2.3590 - acc: 0.3633 - val_loss: 2.00
68 - val_acc: 0.4667
Epoch 32/80
10/10 [==============================] - 87s 9s/step - loss: 2.4541 - acc: 0.4000 - val_loss: 1.89
03 - val_acc: 0.5250
Epoch 33/80
10/10 [==============================] - 87s 9s/step - loss: 2.3360 - acc: 0.4083 - val_loss: 1.91
01 - val_acc: 0.5167
Epoch 34/80
10/10 [==============================] - 84s 8s/step - loss: 2.3878 - acc: 0.3691 - val_loss: 1.93
29 - val_acc: 0.5083
Epoch 35/80
10/10 [==============================] - 87s 9s/step - loss: 2.3020 - acc: 0.4167 - val_loss: 1.74
44 - val_acc: 0.5000
Epoch 36/80
10/10 [==============================] - 87s 9s/step - loss: 2.2049 - acc: 0.4450 - val_loss: 2.00
28 - val_acc: 0.4583
Epoch 37/80
10/10 [==============================] - 87s 9s/step - loss: 2.3544 - acc: 0.3950 - val_loss: 1.70
35 - val_acc: 0.5667
Epoch 38/80
10/10 [==============================] - 87s 9s/step - loss: 2.1669 - acc: 0.4450 - val_loss: 1.80
94 - val_acc: 0.5500
Epoch 39/80
10/10 [==============================] - 87s 9s/step - loss: 2.1112 - acc: 0.4533 - val_loss: 1.78
41 - val_acc: 0.4833
Epoch 40/80
10/10 [==============================] - 88s 9s/step - loss: 2.2920 - acc: 0.4033 - val_loss: 1.57
99 - val_acc: 0.5500
Epoch 41/80
10/10 [==============================] - 83s 8s/step - loss: 2.0795 - acc: 0.4499 - val_loss: 1.47
50 - val_acc: 0.5833
Epoch 42/80
Epoch 42/80
10/10 [==============================] - 86s 9s/step - loss: 2.1490 - acc: 0.4150 - val_loss: 1.90
32 - val_acc: 0.4583
Epoch 43/80
10/10 [==============================] - 86s 9s/step - loss: 2.0662 - acc: 0.4633 - val_loss: 1.87
99 - val_acc: 0.4750
Epoch 44/80
10/10 [==============================] - 87s 9s/step - loss: 1.9190 - acc: 0.4733 - val_loss: 1.64
53 - val_acc: 0.5750
Epoch 45/80
10/10 [==============================] - 87s 9s/step - loss: 2.0427 - acc: 0.4650 - val_loss: 1.59
95 - val_acc: 0.5500
Epoch 46/80
10/10 [==============================] - 88s 9s/step - loss: 1.9688 - acc: 0.4800 - val_loss: 1.40
50 - val_acc: 0.6250
Epoch 47/80
10/10 [==============================] - 87s 9s/step - loss: 1.9776 - acc: 0.4933 - val_loss: 1.73
28 - val_acc: 0.5667
Epoch 48/80
10/10 [==============================] - 87s 9s/step - loss: 1.9540 - acc: 0.4850 - val_loss: 1.66
56 - val_acc: 0.4833
Epoch 49/80
10/10 [==============================] - 87s 9s/step - loss: 1.8268 - acc: 0.5150 - val_loss: 1.40
38 - val_acc: 0.5583
Epoch 50/80
10/10 [==============================] - 87s 9s/step - loss: 1.9403 - acc: 0.4933 - val_loss: 1.74
75 - val_acc: 0.5583
Epoch 51/80
10/10 [==============================] - 87s 9s/step - loss: 1.8565 - acc: 0.4850 - val_loss: 1.70
72 - val_acc: 0.5250
Epoch 52/80
10/10 [==============================] - 86s 9s/step - loss: 1.9499 - acc: 0.4783 - val_loss: 1.58
72 - val_acc: 0.5500
Epoch 53/80
10/10 [==============================] - 88s 9s/step - loss: 1.9701 - acc: 0.4833 - val_loss: 1.38
32 - val_acc: 0.6167
Epoch 54/80
10/10 [==============================] - 83s 8s/step - loss: 1.7823 - acc: 0.5167 - val_loss: 1.62
99 - val_acc: 0.5333
Epoch 55/80
10/10 [==============================] - 83s 8s/step - loss: 1.8994 - acc: 0.4851 - val_loss: 1.41
65 - val_acc: 0.5250
Epoch 56/80
10/10 [==============================] - 86s 9s/step - loss: 1.7594 - acc: 0.5283 - val_loss: 1.50
44 - val_acc: 0.6083
Epoch 57/80
10/10 [==============================] - 87s 9s/step - loss: 1.8698 - acc: 0.4983 - val_loss: 1.25
97 - val_acc: 0.6667
Epoch 58/80
10/10 [==============================] - 87s 9s/step - loss: 1.7545 - acc: 0.5067 - val_loss: 1.48
25 - val_acc: 0.5250
Epoch 59/80
10/10 [==============================] - 84s 8s/step - loss: 1.6991 - acc: 0.5466 - val_loss: 1.67
25 - val_acc: 0.5667
Epoch 60/80
10/10 [==============================] - 87s 9s/step - loss: 1.7755 - acc: 0.5267 - val_loss: 1.62
59 - val_acc: 0.5667
Epoch 61/80
10/10 [==============================] - 87s 9s/step - loss: 1.9139 - acc: 0.4683 - val_loss: 1.33
00 - val_acc: 0.6000
Epoch 62/80
10/10 [==============================] - 87s 9s/step - loss: 1.6993 - acc: 0.5283 - val_loss: 1.58
00 - val_acc: 0.5417
Epoch 63/80
10/10 [==============================] - 83s 8s/step - loss: 1.7105 - acc: 0.5378 - val_loss: 1.40
68 - val_acc: 0.5417
Epoch 64/80
10/10 [==============================] - 87s 9s/step - loss: 1.6766 - acc: 0.5333 - val_loss: 1.34
59 - val_acc: 0.6333
Epoch 65/80
10/10 [==============================] - 87s 9s/step - loss: 1.7764 - acc: 0.5150 - val_loss: 1.41
65 - val_acc: 0.6167
Epoch 66/80
10/10 [==============================] - 88s 9s/step - loss: 1.6697 - acc: 0.5317 - val_loss: 1.53
77 - val_acc: 0.5250
Epoch 67/80
10/10 [==============================] - 87s 9s/step - loss: 1.6308 - acc: 0.5383 - val_loss: 1.65
76 - val_acc: 0.5750
```

```
76 - val_acc: 0.5750
Epoch 68/80
10/10 [==============================] - 87s 9s/step - loss: 1.5997 - acc: 0.5567 - val_loss: 1.33
09 - val_acc: 0.6250
Epoch 69/80
10/10 [==============================] - 87s 9s/step - loss: 1.6147 - acc: 0.5750 - val_loss: 1.63
06 - val_acc: 0.5583
Epoch 70/80
10/10 [==============================] - 87s 9s/step - loss: 1.5708 - acc: 0.5767 - val_loss: 1.16
32 - val_acc: 0.6583
Epoch 71/80
10/10 [==============================] - 87s 9s/step - loss: 1.5419 - acc: 0.5850 - val_loss: 1.49
25 - val_acc: 0.5667
Epoch 72/80
10/10 [==============================] - 87s 9s/step - loss: 1.6408 - acc: 0.5417 - val_loss: 1.40
54 - val_acc: 0.6167
Epoch 73/80
10/10 [==============================] - 88s 9s/step - loss: 1.6503 - acc: 0.5183 - val_loss: 1.47
29 - val_acc: 0.6250
Epoch 74/80
10/10 [==============================] - 87s 9s/step - loss: 1.7218 - acc: 0.5250 - val_loss: 1.33
08 - val_acc: 0.6250
Epoch 75/80
10/10 [==============================] - 87s 9s/step - loss: 1.5897 - acc: 0.5550 - val_loss: 1.50
70 - val_acc: 0.5917
Epoch 76/80
10/10 [==============================] - 87s 9s/step - loss: 1.5784 - acc: 0.5667 - val_loss: 1.45
79 - val_acc: 0.5500
Epoch 77/80
10/10 [==============================] - 87s 9s/step - loss: 1.5488 - acc: 0.5633 - val_loss: 1.43
41 - val_acc: 0.5917
Epoch 78/80
10/10 [==============================] - 88s 9s/step - loss: 1.4794 - acc: 0.5833 - val_loss: 1.21
90 - val_acc: 0.6750
Epoch 79/80
10/10 [==============================] - 87s 9s/step - loss: 1.6184 - acc: 0.5517 - val_loss: 1.37
41 - val_acc: 0.6083
Epoch 80/80
10/10 [==============================] - 87s 9s/step - loss: 1.5304 - acc: 0.5417 - val_loss: 1.34
97 - val_acc: 0.5917
```

In [21]:

```
import cv2

# list to store images of test set
test_set = []

# list to store all file ids of images in test directory
test_set_ids = []

for image in os.listdir('test'):
    test_set_ids.append(os.path.splitext(image)[0])
    image = cv2.imread('test/'+ image)
    test_set.append(cv2.resize(image, (240, 240)))     # resizing test images to target size
```

In [25]:

```
test_set_ids[0]
```

Out[25]:

```
'8d3a3f80d624dae142d10827ef3c4bfd'
```

In [22]:

```
# create a numpy array of images and rescale them

test_set = np.array(test_set, np.float32)/255.0
print(test_set[0])
```

```
[[[0.10980392 0.15294118 0.18431373]
  [0.11764706 0.16078432 0.19215687]
  [0.11372549 0.15686275 0.1882353 ]
  [0.11372549 0.15686275 0.1882353 ]
  ...
  [0.17254902 0.21568628 0.24705882]
  [0.18431373 0.22745098 0.25882354]
  [0.17254902 0.21568628 0.24705882]]

 [[0.10588235 0.14901961 0.18039216]
  [0.11372549 0.15686275 0.1882353 ]
  [0.11372549 0.15686275 0.1882353 ]
  ...
  [0.1882353  0.23137255 0.2627451 ]
  [0.1882353  0.23137255 0.2627451 ]
  [0.1882353  0.23137255 0.2627451 ]]

 [[0.09803922 0.14901961 0.18039216]
  [0.10196079 0.15294118 0.18431373]
  [0.10196079 0.15294118 0.18431373]
  ...
  [0.18431373 0.22745098 0.25882354]
  [0.1882353  0.23137255 0.2627451 ]
  [0.21176471 0.25490198 0.28627452]]

 ...

 [[0.1254902  0.2        0.25882354]
  [0.12941177 0.20392157 0.2627451 ]
  [0.12941177 0.20392157 0.2627451 ]
  ...
  [0.2        0.23921569 0.26666668]
  [0.21568628 0.25490198 0.28235295]
  [0.22352941 0.2627451  0.2901961 ]]

 [[0.10980392 0.18431373 0.24313726]
  [0.10980392 0.18431373 0.24313726]
  [0.11372549 0.1882353  0.24705882]
  ...
  [0.20392157 0.24705882 0.2627451 ]
  [0.21176471 0.25490198 0.27058825]
  [0.21176471 0.25490198 0.27058825]]

 [[0.12941177 0.20392157 0.2627451 ]
  [0.12941177 0.20392157 0.2627451 ]
  [0.13333334 0.20784314 0.26666668]
  ...
  [0.20784314 0.2509804  0.26666668]
  [0.20784314 0.2509804  0.26666668]
  [0.2        0.24313726 0.25882354]]]
```

In [23]:

```
predictions = model.predict(test_set)
predictions.shape[0]
```

Out[23]:

```
10357
```

In [24]:

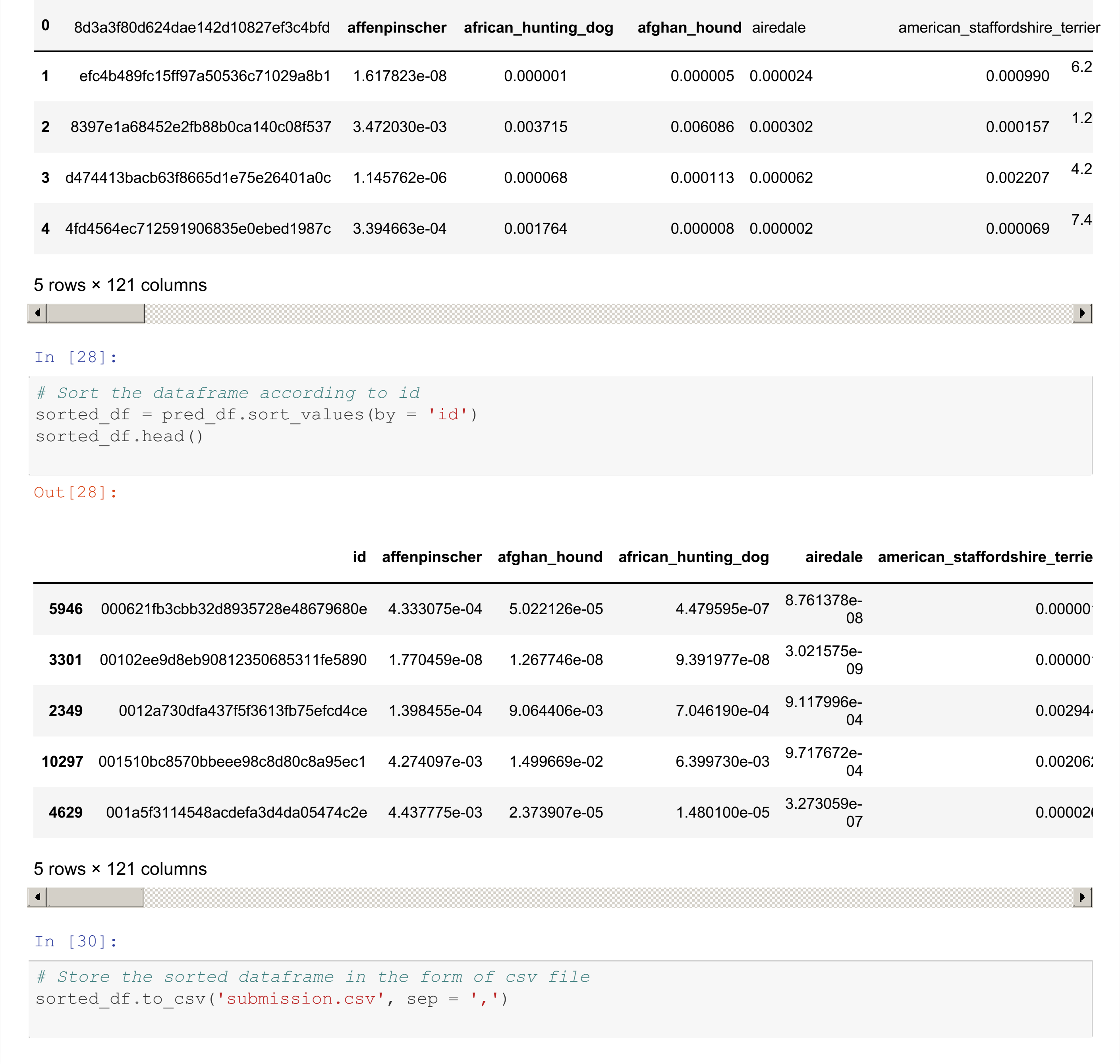
```
# Convert into dataframe
pred_df = pd.DataFrame(predictions)

# Define headers of dataframe
pred_df.columns = list(train_generator.class_indices.keys())

# Add a column containing file ids of images
pred_df.insert(0, 'id', test_set_ids)

pred_df.head()
```

Out[24]:



0	8d3a3f80d624dae142d10827ef3c4bfd	**affenpinscher**	**african_hunting_dog**	**afghan_hound**	airedale	american_staffordshire_terrier	
1	efc4b489fc15ff97a50536c71029a8b1	1.617823e-08	0.000001	0.000005	0.000024	0.000990	6.2
2	8397e1a68452e2fb88b0ca140c08f537	3.472030e-03	0.003715	0.006086	0.000302	0.000157	1.2
3	d474413bacb63f8665d1e75e26401a0c	1.145762e-06	0.000068	0.000113	0.000062	0.002207	4.2
4	4fd4564ec712591906835e0ebed1987c	3.394663e-04	0.001764	0.000008	0.000002	0.000069	7.4

5 rows × 121 columns

In [28]:

```
# Sort the dataframe according to id
sorted_df = pred_df.sort_values(by = 'id')
sorted_df.head()
```

Out[28]:

	id	affenpinscher	afghan_hound	african_hunting_dog	airedale	american_staffordshire_terrie
5946	000621fb3cbb32d8935728e48679680e	4.333075e-04	5.022126e-05	4.479595e-07	8.761378e-08	0.00000
3301	00102ee9d8eb90812350685311fe5890	1.770459e-08	1.267746e-08	9.391977e-08	3.021575e-09	0.00000
2349	0012a730dfa437f5f3613fb75efcd4ce	1.398455e-04	9.064406e-03	7.046190e-04	9.117996e-04	0.00294
10297	001510bc8570bbeee98c8d80c8a95ec1	4.274097e-03	1.499669e-02	6.399730e-03	9.717672e-04	0.00206
4629	001a5f3114548acdefa3d4da05474c2e	4.437775e-03	2.373907e-05	1.480100e-05	3.273059e-07	0.00002

5 rows × 121 columns

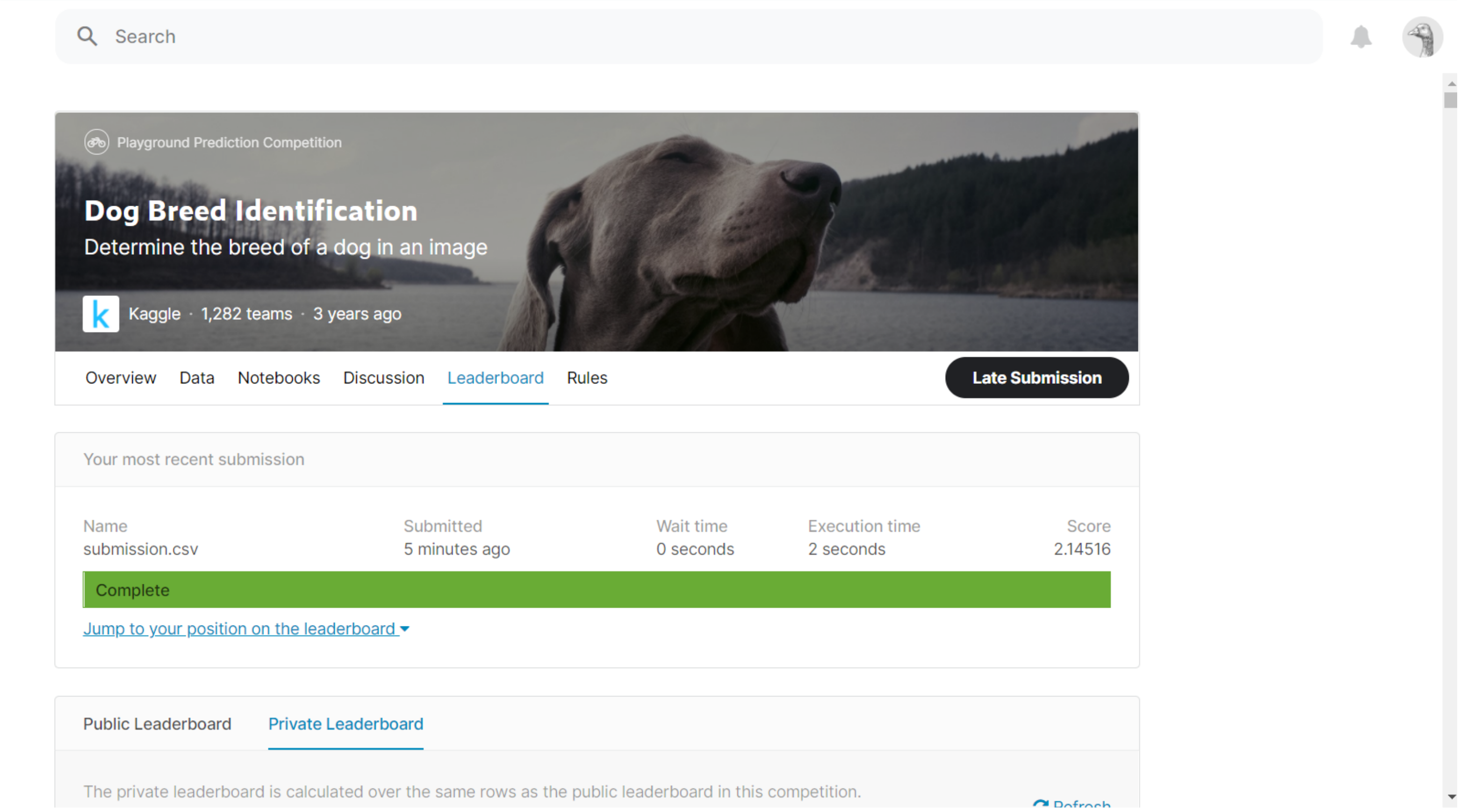
In [30]:

```
# Store the sorted dataframe in the form of csv file
sorted_df.to_csv('submission.csv', sep = ',')
```

Google Colab Notebook Link:

https://colab.research.google.com/drive/1ekQSEwhoqnvBOYFSXkeYPuGwJ7F7qCQ8?usp=sharing

Kaggle Score



Dog Breed Identification

Determine the breed of a dog in an image

Kaggle · 1,282 teams · 3 years ago

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission.csv	5 minutes ago	0 seconds	2 seconds	2.14516

Complete

Jump to your position on the leaderboard

The private leaderboard is calculated over the same rows as the public leaderboard in this competition.