1. Reading Modules

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
from pathlib import Path
import cv2
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
import tensorflow as tf
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
import cv2 as cv
import numpy as np
from scipy import ndimage, misc
import skimage
from keras.applications.inception_v3 import InceptionV3, preprocess_input
from keras.models import Sequential
from keras.layers.pooling import GlobalAveragePooling2D
from tensorflow.keras.layers import Dense, Dropout
```

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

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

```
!unzip /content/drive/MyDrive/Dog_breed.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: train/83bc62b0fffa99a9c94ba0b67a5f7395.jpg
  inflating: train/83bcff6b55ee179a7c123fa6103c377a.jpg
  inflating: train/83be6d622ab74a5e7e08b53eb8fd566a.jpg
  inflating: train/83c2d7419b0429b9fe953bc1b6cddbec.jpg
  inflating: train/83cf7d7cd2a759a93e2ffd95bea9c6fb.jpg
  inflating: train/83d405858f0931722ef21e8ac0adee4d.jpg
  inflating: train/83d4125a4c3c7dc5956563276cb1cd74.jpg
  inflating: train/83f0bb565b2186dbcc6a9d009cb26ff2.jpg
  inflating: train/83fad0718581a696132c96c166472627.jpg
  inflating: train/83fbbcc9a612e3f712b1ba199da61f20.jpg
  inflating: train/8403d8936430c2f05ab7d74d23c2c0cb.jpg
  inflating: train/8406d837b2d7fac1c3cd621abb4c4f9e.jpg
```

```
          inflating: train/840b67d26e5e43f8eb6430f62d4ba1ac.jpg
          inflating: train/840db91ba4600148f3dcb06ec419b421.jpg
          inflating: train/840dbad5a691c22611d85b2488bf4cbb.jpg
          inflating: train/8410ced9ebc1759a7ebce5c42bfb5222.jpg
          inflating: train/841463629c4833816e216cbb041c2778.jpg
          inflating: train/8429dcca4ae91c4e0345e4ba48b0d69f.jpg
          inflating: train/842e3c6e44fda4102fe83d07dac72b3e.jpg
          inflating: train/8431a6ce7c70e5e36698e821eedf24b5.jpg
          inflating: train/8434b6c3cee87e28395197d6fc7d3489.jpg
          inflating: train/8436be99589db6a99cfac1b894421ea6.jpg
          inflating: train/843cbc1fc239d24534859bd272c3bc16.jpg
          inflating: train/843d766d92a7b6d6a85a81e56a99c51f.jpg
          inflating: train/84421c01900b34e3e1ba42f2424fbd33.jpg
          inflating: train/844dde39a9e8987e510e8d46ec4da714.jpg
          inflating: train/8452a26d7243a299ea782a7ba4036f1f.jpg
          inflating: train/8454b5e6546f04871561de8f10d868c7.jpg
          inflating: train/84564a69c0d0fa36e0810188943683a1.jpg
          inflating: train/84605f5fc5ad89a66b9b277e1223e962.jpg
          inflating: train/8463aa43d88bee057082434ccc806bb0.jpg
          inflating: train/8467fbd75a8fe64da70df5410b6c4f09.jpg
          inflating: train/846d6384787fff8dc17d488e6b86c209.jpg
          inflating: train/8470a6fdf4db9b088494aaa9384ba9d0.jpg
          inflating: train/84728e78632c0910a69d33f82e62638c.jpg
          inflating: train/8477ac111ca6a9f11c2edfa43a933cad.jpg
          inflating: train/8480ad94841309fc4ce874c4b1afc90c.jpg
          inflating: train/848133f97b3e97b1b0fab0402e572d98.jpg
          inflating: train/8485bc3f3fd64b90be74d7f020c61f54.jpg
          inflating: train/8486e8159f169e8c3d4697e5c859760f.jpg
          inflating: train/848f7a0b665b118e4a3b85029b1794e0.jpg
          inflating: train/8490222d4744064aa7a8621a1c274965.jpg
          inflating: train/8494afd34e3a2e81bec37e4dfdc67f8d.jpg
          inflating: train/84aaf49fb53d423d4aed05ab79559b0c.jpg
          inflating: train/84ab21940432e5b42cfacc58cd84c861.jpg
          inflating: train/84accc2dc9f5bb3ebee89fe1bf23639c.jpg
          inflating: train/84adb2cc13b65cf25418cde969b9bb0e.jpg
          inflating: train/84b612a8e43c6debbc9951cb24ec9ba0.jpg
          inflating: train/84b62d2def32fc85092cabe2c722c135.jpg
          inflating: train/84bcd47e09b0ef3f0b6e3f47f232a77c.jpg
          inflating: train/84be9b9f59aa586f1b188781b2c47a3e.jpg
          inflating: train/84c6bdd4bb818edd4c088f27312d028f.jpg
          inflating: train/84d2dd9eff021b6095a4b1e2ba3c1c0c.jpg
          inflating: train/84de398dd5408d91b133e2e95628120a.jpg
          inflating: train/84dfe42ce71204b367c2b4000eb6ba5c.jpg
          inflating: train/84e567b15311f0c891858f56f0175867.jpg
          inflating: train/84f5f076b0b951d68f88c8b795b7135e.jpg
```

```
filelist  = []

for dirname, _, filenames in os.walk('/content/train'):
    for filename in filenames:
        filelist.append (os.path.join(dirname, filename))
```

```
filelist[:5]
```

```
['/content/train/b9f96dd0c9f3dc7e755d9b8cbb124f3b.jpg',
 '/content/train/f706682a30021cc74cd9416dac25e943.jpg',
 '/content/train/8f3e10fab6ea57479f91a5c6efc11351.jpg',
 '/content/train/65a3a8d1011f95e937d77e3a79700dad.jpg',
 '/content/train/324759773574e9bd6d6ba9c58e1550f9.jpg']
```

```
labels = pd.read_csv("labels.csv")
```

```
labels
```

	id	breed
0	000bec180eb18c7604dcecc8fe0dba07	boston_bull
1	001513dfcb2ffafc82cccf4d8bbaba97	dingo
2	001cdf01b096e06d78e9e5112d419397	pekinese
3	00214f311d5d2247d5dfe4fe24b2303d	bluetick
4	0021f9ceb3235effd7fcde7f7538ed62	golden_retriever
...	...	...
10217	ffd25009d635cfd16e793503ac5edef0	borzoi
10218	ffd3f636f7f379c51ba3648a9ff8254f	dandie_dinmont
10219	ffe2ca6c940cddfee68fa3cc6c63213f	airedale
10220	ffe5f6d8e2bff356e9482a80a6e29aac	miniature_pinscher
10221	fff43b07992508bc822f33d8ffd902ae	chesapeake_bay_retriever

10222 rows × 2 columns

```
len(labels['breed'].unique())
pd.value_counts(labels.breed)
```

```
scottish_deerhound      126
maltese_dog             117
afghan_hound            116
entlebucher             115
bernese_mountain_dog    114
                       ...
golden_retriever         67
brabancon_griffon        67
komondor                 67
eskimo_dog               66
briard                   66
Name: breed, Length: 120, dtype: int64
```

```
from os.path import join
```

```
image_dir = '/content/train'
```

```
filelist = []
image_path1 = []

for dir_name, _, filenames in os.walk('/content/train'):
  print(filenames)
  for filename in filenames:
    image_path1.append(join(image_dir, filename))
```

```
['b9f96dd0c9f3dc7e755d9b8cbb124f3b.jpg', 'f706682a30021cc74cd9416dac25e943.jpg',
```

```
image_path1[:3]
```

```
['/content/train/b9f96dd0c9f3dc7e755d9b8cbb124f3b.jpg',
 '/content/train/f706682a30021cc74cd9416dac25e943.jpg',
 '/content/train/8f3e10fab6ea57479f91a5c6efc11351.jpg']
```

```
labels_train = []

for i in labels['breed']:
  labels_train.append(i)
```

```
len(image_path1), len(labels_train)
```

```
(10222, 10222)
```

```
type(image_path1), type(labels_train)
```

```
(list, list)
```

```
df_dog = pd.DataFrame(list(zip(image_path1, labels_train)), columns= ['Imagepath', 'La
```

```
df_dog
```

	Imagepath	Labels
0	/content/train/b9f96dd0c9f3dc7e755d9b8cbb124f3...	boston_bull
1	/content/train/f706682a30021cc74cd9416dac25e94...	dingo
2	/content/train/8f3e10fab6ea57479f91a5c6efc1135...	pekinese
3	/content/train/65a3a8d1011f95e937d77e3a79700da...	bluetick
4	/content/train/324759773574e9bd6d6ba9c58e1550f...	golden_retriever
...	...	...

```
a=df_dog.Labels.value_counts()[:10]
```

```
a
```

```
scottish_deerhound      126
maltese_dog             117
afghan_hound            116
entlebucher             115
bernese_mountain_dog    114
shih-tzu                112
great_pyrenees          111
pomeranian              111
basenji                 110
samoyed                 109
Name: Labels, dtype: int64
```

```
labels_needed = ['scottish_deerhound', 'maltese_dog', 'afghan_hound', 'entlebucher',
'bernese_mountain_dog', 'shih-tzu', 'great_pyrenees', 'pomeranian', 'basenji',
                 'samoyed']
```

```
all_lables = set(df_dog.Labels.unique())
len(all_lables)
```

```
120
```

```
s = all_lables.difference(set(labels_needed))
print(len(s))
```

```
110
```

```
not_needed_lables = list(s)
not_needed_lables
```

```
['west_highland_white_terrier',
 'vizsla',
 'yorkshire_terrier',
 'lhasa',
 'australian_terrier',
```

```
        'irish_wolfhound',
        'collie',
        'welsh_springer_spaniel',
        'rhodesian_ridgeback',
        'doberman',
        'toy_poodle',
        'cairn',
        'toy_terrier',
        'border_terrier',
        'affenpinscher',
        'keeshond',
        'wire-haired_fox_terrier',
        'dandie_dinmont',
        'appenzeller',
        'pekinese',
        'brabancon_griffon',
        'standard_poodle',
        'standard_schnauzer',
        'boxer',
        'weimaraner',
        'miniature_schnauzer',
        'english_springer',
        'clumber',
        'kerry_blue_terrier',
        'chihuahua',
        'dingo',
        'english_setter',
        'walker_hound',
        'malamute',
        'japanese_spaniel',
        'greater_swiss_mountain_dog',
        'bedlington_terrier',
        'bloodhound',
        'border_collie',
        'old_english_sheepdog',
        'miniature_pinscher',
        'leonberg',
        'kuvasz',
        'silky_terrier',
        'irish_setter',
        'eskimo_dog',
        'norwich_terrier',
        'ibizan_hound',
        'dhole',
        'english_foxhound',
        'malinois',
        'american_staffordshire_terrier',
        'sealyham_terrier',
        'african_hunting_dog',
        'german_short-haired_pointer',
        'scotch_terrier',
        'german_shepherd',
        'soft-coated_wheaten_terrier',

#df_temp = df_dog.loc[df_dog["Labels"] in labels_needed]
```

```
'''# Set the index of the DataFrame to the country name
df_temp = df_dog.set_index("Labels")
df_temp.head()'''
```

```
'# Set the index of the DataFrame to the country name\ndf_temp = df_dog.set_inde
x("Labels")\ndf_temp.head()'
```

```
#df_temp.shape
```

```
#df_temp = df_temp.drop(not_needed_lables)
#df_temp
```

Taking only those 10 labels here.

[ ] ↳ *7 cells hidden*

Creating a dataframe with file paths and the labels for them

```
df = pd.DataFrame(list(zip(image_path1, labels_train)), columns= ['Filepath', 'Labels'
df
```

	Filepath	Labels
0	/content/train/b9f96dd0c9f3dc7e755d9b8cbb124f3...	boston_bull
1	/content/train/f706682a30021cc74cd9416dac25e94...	dingo
2	/content/train/8f3e10fab6ea57479f91a5c6efc1135...	pekinese
3	/content/train/65a3a8d1011f95e937d77e3a79700da...	bluetick
4	/content/train/324759773574e9bd6d6ba9c58e1550f...	golden_retriever
...	...	...
10217	/content/train/5c13e38df48763724a42552504b8dde...	borzoi
10218	/content/train/c9bbc4ce586c0d73e14bee1b1e674ba...	dandie_dinmont
10219	/content/train/41cdc849e6032e410cf32c6a274fe2a...	airedale
10220	/content/train/6fa11f3d4cd5e972b5be8a871674017...	miniature_pinscher
10221	/content/train/a69dbc3bb27b3b0dd9b74b7f2da1311...	chesapeake_bay_retriever

10222 rows × 2 columns

```
from sklearn.utils import shuffle
df = (df.sample(frac = 1).reset_index()).drop(columns = 'index')
```

```
df
```

	Filepath	Labels
0	/content/train/2a8ac4ec28af4aa4bbb7e35dda82c6e...	west_highland_white_terrier
1	/content/train/1c5575083fe9e346d66eac01d2cc548...	brabancon_griffon
2	/content/train/ec483170d4a9c12f9f7bd0d691de7c6...	basset
3	/content/train/42fc4f86c553289b9f3a89171e840c7...	english_springer
4	/content/train/18b79147982f9a14c768a256d3696a1...	standard_schnauzer
...	...	...
10217	/content/train/c25b1b2e1919a58239ac16ee53bdd44...	whippet
10218	/content/train/7f1746ba7ed8254df3be3c2a1ab97e5...	irish_water_spaniel
10219	/content/train/6ca9149d85c705eab3f870619ee87e5...	kuvasz
10220	/content/train/c043f726c52f93c83fa0100a8a77648...	beagle
10221	/content/train/fdfcc3d2e40970fbfb8521bd29e9fb4...	otterhound

10222 rows × 2 columns

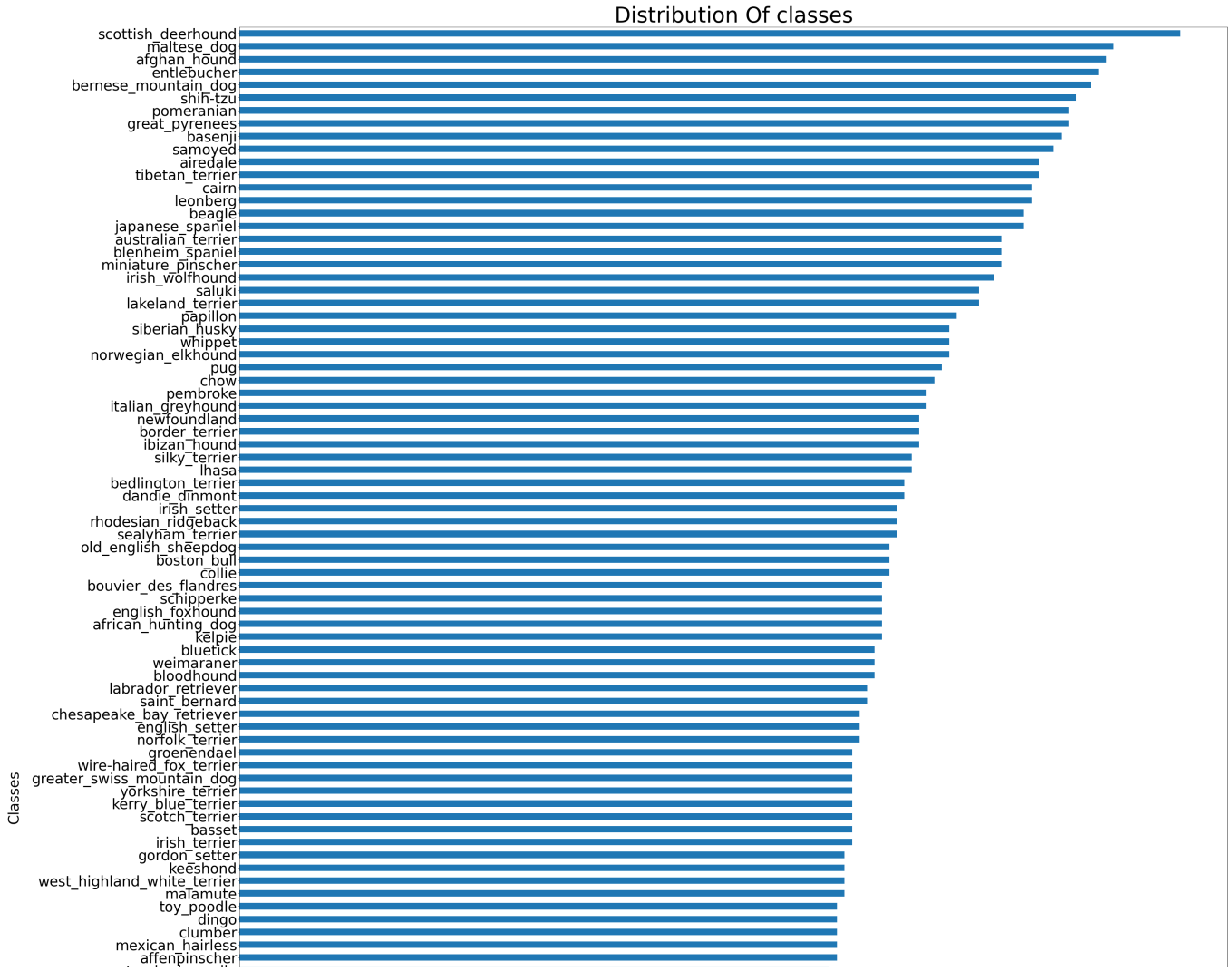
Displaying first 12 pictures

```
f,a = plt.subplots(nrows=4, ncols=3,figsize=(13, 7),
                        subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(a.flat):
    ax.imshow(plt.imread(df.Filepath[i]))
    ax.set_title(df.Labels[i])

plt.tight_layout()
plt.show()
```


```
ax=pd.value_counts(df['Labels'],ascending=True).plot(kind='barh',
                                                     fontsize="40",
                                                     title="Distribution Of classes'
                                                     figsize=(50,80))
ax.set(xlabel="Images per class", ylabel="Classes")
ax.xaxis.label.set_size(40)
ax.yaxis.label.set_size(40)
ax.title.set_size(60)
plt.show()
```



Double-click (or enter) to edit

Checking for class imbalance


```
df.Labels.value_counts()
```

```
scottish_deerhound      126
maltese_dog             117
afghan_hound            116
entlebucher             115
bernese_mountain_dog    114
                       ...
komondor                 67
golden_retriever         67
brabancon_griffon        67
eskimo_dog               66
briard                   66
Name: Labels, Length: 120, dtype: int64
```

- Class imbalance present.

Splitting the data And Creating data generator

```
train_ratio = .75
validation_ratio = 0.10
test_ratio = 0.25

train, test = train_test_split(df, test_size = test_ratio )
val, test = train_test_split(test, test_size=test_ratio/(test_ratio + validation_ratio
```

```
img_datagen = ImageDataGenerator(rescale=1./255,
                                 rotation_range=30,
                                 width_shift_range=0.2,
                                 height_shift_range=0.2,
                                 horizontal_flip = 'true')
```

```
x_train =  img_datagen.flow_from_dataframe(dataframe = train,  x_col='Filepath', y_col
x_val = img_datagen.flow_from_dataframe(dataframe = val,  x_col='Filepath', y_col='Lab
x_test = img_datagen.flow_from_dataframe(dataframe = test,  x_col='Filepath', y_col='I
```

```
Found 7666 validated image filenames belonging to 120 classes.
Found 730 validated image filenames belonging to 120 classes.
Found 1826 validated image filenames belonging to 120 classes.
```

```
x_train
```

```
<keras.preprocessing.image.DataFrameIterator at 0x7fac43b12b90>
```

Modelling

```
i_model = InceptionV3(weights= 'imagenet', include_top=False, input_shape=(299, 299, 3
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicatio
87916544/87910968 [==============================] - 0s 0us/step
87924736/87910968 [==============================] - 0s 0us/step
```

```
for layer in i_model.layers:
    layer.trainable = False
```

```
i_model.summary()
```

```
Model: "inception_v3"
__________________________________________________________________________________
 Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
```

```
input_1 (InputLayer)           [(None, 299, 299, 3  0           []
                               )]

conv2d (Conv2D)                (None, 149, 149, 32  864         ['input_1[0][0]
                               )

batch_normalization (BatchNorm  (None, 149, 149, 32  96         ['conv2d[0][0]'
alization)                     )

activation (Activation)        (None, 149, 149, 32  0           ['batch_normali
                               )

conv2d_1 (Conv2D)              (None, 147, 147, 32  9216        ['activation[0]
                               )

batch_normalization_1 (BatchNo  (None, 147, 147, 32  96         ['conv2d_1[0][0
rmalization)                   )

activation_1 (Activation)      (None, 147, 147, 32  0           ['batch_normali
                               )

conv2d_2 (Conv2D)              (None, 147, 147, 64  18432       ['activation_1[
                               )

batch_normalization_2 (BatchNo  (None, 147, 147, 64  192        ['conv2d_2[0][0
rmalization)                   )

activation_2 (Activation)      (None, 147, 147, 64  0           ['batch_normali
                               )

max_pooling2d (MaxPooling2D)   (None, 73, 73, 64)   0           ['activation_2[

conv2d_3 (Conv2D)              (None, 73, 73, 80)   5120        ['max_pooling2d

batch_normalization_3 (BatchNo  (None, 73, 73, 80)  240         ['conv2d_3[0][0
rmalization)

activation_3 (Activation)      (None, 73, 73, 80)   0           ['batch_normali

conv2d_4 (Conv2D)              (None, 71, 71, 192)  138240      ['activation_3[

batch_normalization_4 (BatchNo  (None, 71, 71, 192)  576        ['conv2d_4[0][0
rmalization)

activation_4 (Activation)      (None, 71, 71, 192)  0           ['batch_normali

max_pooling2d_1 (MaxPooling2D)  (None, 35, 35, 192)  0         ['activation_4[

conv2d_8 (Conv2D)              (None, 35, 35, 64)   12288       ['max_pooling2d_

batch_normalization_8 (BatchNo  (None, 35, 35, 64)  192         ['conv2d_8[0][0
rmalization)

activation_8 (Activation)      (None, 35, 35, 64)   0           ['batch_normali
```

```
model = Sequential()
model.add(i_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(128))
model.add(Dropout(0.2))
model.add(Dense(120, activation = 'softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 8, 8, 2048)        21802784

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 128)               262272

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 120)               15480

=================================================================
Total params: 22,080,536
Trainable params: 277,752
Non-trainable params: 21,802,784
_________________________________________________________________
```

Double-click (or enter) to edit

```
model.compile(optimizer = SGD(),
              loss="categorical_crossentropy",
              metrics=["accuracy"])
```

```
history = model.fit(x_train, validation_data = x_val,steps_per_epoch = 175,validation_
                epochs = 15, verbose = 2)
```

```
Epoch 1/15
175/175 - 532s - loss: 4.9531 - accuracy: 0.0109 - val_loss: 4.8723 - val_accura
Epoch 2/15
175/175 - 494s - loss: 4.8924 - accuracy: 0.0103 - val_loss: 4.8979 - val_accura
Epoch 3/15
175/175 - 494s - loss: 4.8695 - accuracy: 0.0091 - val_loss: 4.8745 - val_accura
Epoch 4/15
175/175 - 501s - loss: 4.8188 - accuracy: 0.0092 - val_loss: 4.8624 - val_accura
Epoch 5/15
175/175 - 493s - loss: 4.8236 - accuracy: 0.0177 - val_loss: 4.8699 - val_accura
Epoch 6/15
175/175 - 493s - loss: 4.7902 - accuracy: 0.0154 - val_loss: 4.9134 - val_accura
Epoch 7/15
175/175 - 490s - loss: 4.7553 - accuracy: 0.0263 - val_loss: 4.9106 - val_accura
Epoch 8/15
```

```
175/175 - 491s - loss: 4.7584 - accuracy: 0.0114 - val_loss: 4.9324 - val_accura
Epoch 9/15
175/175 - 494s - loss: 4.7069 - accuracy: 0.0297 - val_loss: 4.8762 - val_accura
Epoch 10/15
175/175 - 496s - loss: 4.7038 - accuracy: 0.0251 - val_loss: 4.9336 - val_accura
Epoch 11/15
175/175 - 494s - loss: 4.7062 - accuracy: 0.0246 - val_loss: 4.9181 - val_accura
Epoch 12/15
175/175 - 495s - loss: 4.6826 - accuracy: 0.0263 - val_loss: 4.9275 - val_accura
Epoch 13/15
175/175 - 495s - loss: 4.6296 - accuracy: 0.0371 - val_loss: 4.9421 - val_accura
Epoch 14/15
175/175 - 497s - loss: 4.6405 - accuracy: 0.0360 - val_loss: 4.9258 - val_accura
Epoch 15/15
175/175 - 495s - loss: 4.6319 - accuracy: 0.0320 - val_loss: 4.9452 - val_accura
```

SAVING THE MODEL & LOADING THE MODEL

```
from keras.models import model_from_json
model_dog1_json = model.to_json()
with open("/content/drive/MyDrive/model_dog1.json", "w") as json_file:
    json_file.write(model_dog1_json)
# serialize weights to HDF5
model.save_weights("/content/drive/MyDrive/model_dog1.h5")
print("Saved model to disk")

'''
LOADING THE WEIGHTS OF THE DEEP LEARNING NETWORK
# load json and create model
json_file = open('model_cat.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("model.h5")
print("Loaded model from disk")
'''
```

```
Saved model to disk
'\nLOADING THE WEIGHTS OF THE DEEP LEARNING NETWORK\n# load json and create mode
l\njson_file = open(\'model_cat.json\', \'r\')\nloaded_model_json = json_file.re
ad()\njson_file.close()\nloaded_model = model_from_json(loaded_model_json)\n# lo
```

Predicting on test data

```
predictions = model.predict(x_test)
predictions = np.argmax(predictions, axis=1)
predictions
```

```
array([42, 65, 85, ..., 35, 95, 95])
```

```
labels = x_train.class_indices
labels
```

```
{'affenpinscher': 0,
 'afghan_hound': 1,
 'african_hunting_dog': 2,
 'airedale': 3,
 'american_staffordshire_terrier': 4,
 'appenzeller': 5,
 'australian_terrier': 6,
 'basenji': 7,
 'basset': 8,
 'beagle': 9,
 'bedlington_terrier': 10,
 'bernese_mountain_dog': 11,
 'black-and-tan_coonhound': 12,
 'blenheim_spaniel': 13,
 'bloodhound': 14,
 'bluetick': 15,
 'border_collie': 16,
 'border_terrier': 17,
 'borzoi': 18,
 'boston_bull': 19,
 'bouvier_des_flandres': 20,
 'boxer': 21,
 'brabancon_griffon': 22,
 'briard': 23,
 'brittany_spaniel': 24,
 'bull_mastiff': 25,
 'cairn': 26,
 'cardigan': 27,
 'chesapeake_bay_retriever': 28,
 'chihuahua': 29,
 'chow': 30,
 'clumber': 31,
 'cocker_spaniel': 32,
 'collie': 33,
 'curly-coated_retriever': 34,
 'dandie_dinmont': 35,
 'dhole': 36,
 'dingo': 37,
 'doberman': 38,
 'english_foxhound': 39,
 'english_setter': 40,
 'english_springer': 41,
 'entlebucher': 42,
 'eskimo_dog': 43,
 'flat-coated_retriever': 44,
 'french_bulldog': 45,
```

```
        'german_shepherd': 46,
        'german_short-haired_pointer': 47,
        'giant_schnauzer': 48,
        'golden_retriever': 49,
        'gordon_setter': 50,
        'great_dane': 51,
        'great_pyrenees': 52,
        'greater_swiss_mountain_dog': 53,
        'groenendael': 54,
        'ibizan_hound': 55,
        'irish_setter': 56,
        'irish_terrier': 57
```

```
import plotly.graph_objects as go
from IPython.display import display, Image
```

ACCURACY VISUALIZATION

```
plt.clf()
fig = go.Figure()
fig.add_trace(go.Scatter(
                    y=history.history['accuracy'],
                    name='Train'))

fig.add_trace(go.Scatter(
                    y=history.history['val_accuracy'],
                    name='Valid'))


fig.update_layout(height=450,
                  width=600,
                  title='Accuracy for Dog breed',
                  xaxis_title='Epoch',
                  yaxis_title='Accuracy')

fig.write_html('acc_race.html', include_plotlyjs='cdn')

fig.show()
```

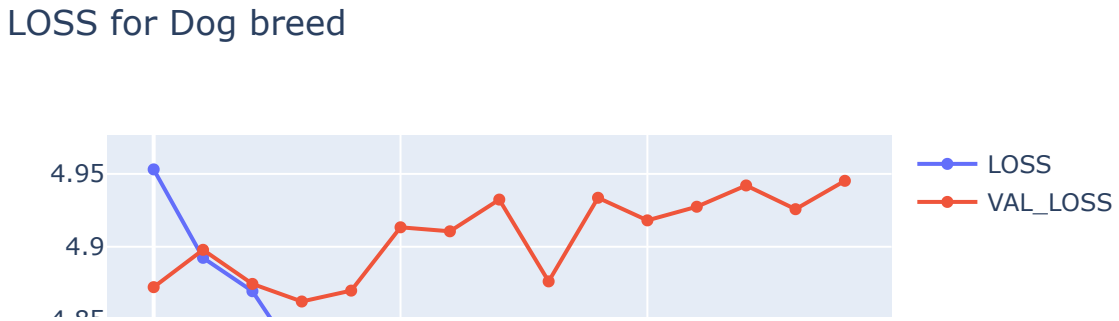

```
plt.clf()
fig = go.Figure()
fig.add_trace(go.Scatter(
                    y=history.history['loss'],
                    name='LOSS'))

fig.add_trace(go.Scatter(
                    y=history.history['val_loss'],
                    name='VAL_LOSS'))


fig.update_layout(height=450,
                  width=600,
                  title='LOSS for Dog breed',
                  xaxis_title='Epoch',
                  yaxis_title='LOSS')

#fig.write_html('acc_race.html', include_plotlyjs='cdn')

fig.show()
```



LOADING THE HISTORY AND SAVING THE HISTORY

```
#SAVING THE HISTORY

np.save('/content/drive/MyDrive/my_history_dog.npy',history.history)

#Loading the history

#history=np.load('/content/drive/MyDrive/my_history_dog.npy',allow_pickle='TRUE').iter


test["Labels"].replace(x_train.class_indices, inplace = True)
```

Evaluating the test data

Test Accuracy

```
test_accuracy = model.evaluate(x_test)[1] * 100
print('Test accuracy is : ',test_accuracy, '%' )
```

```
183/183 [==============================] - 424s 2s/step - loss: 4.9982 - accuracy
Test accuracy is :  0.5476451478898525 %
```

Confusion Matrix

```
confusion_matrix(test.Labels , predictions)
```

```
array([[0, 0, 0, ..., 0, 0, 0],
```

```
       [0, 0, 0, ..., 0, 0, 0],
       [0, 2, 0, ..., 0, 0, 0],
       ...,
       [0, 1, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 3, 0, ..., 0, 0, 0]])
```

F1 Score

```
from sklearn.metrics import accuracy_score, f1_score
print('F1 score is',f1_score(test.Labels, predictions, average = 'weighted'))
```

```
F1 score is 0.002283508439841087
```

ROC - AUC Score

```
predicted_probab =model.predict(x_test)
predicted_probab
```

```
print("ROC- AUC score is", roc_auc_score( test.Labels, predicted_probab, multi_class='
```

```
ROC- AUC score is 0.47385584434466915
```