Dog Breed Classification

October 16, 2025

1 Import Library

You will need to load all necessary libraries related to TensorFlow and Keras.

Keep in mind that **import keras** is different than **import tensorflow**, so always put **import tensorflow.kears** to make sure you are importing the correct function.

[1]:
```
import os

import tensorflow as tf
#import tensorflow.keras as keras
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.python.keras.engine.training import Model
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.python.keras.layers import Add, Dropout, Flatten, Dense,␣
↪Activation, Conv2D,GlobalAveragePooling2D, MaxPooling2D
from tensorflow.python.keras.layers.normalization import BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# AWS S3 and file manager
import boto3
import s3fs

from sklearn.model_selection import train_test_split

from PIL import Image
import cv2
```

```
WARNING:tensorflow:From
/home/ec2-user/anaconda3/envs/tensorflow_p36/gpu_cuda10.0/lib/python3.6/site-
packages/tensorflow_core/__init__.py:1473: The name tf.estimator.inputs is
deprecated. Please use tf.compat.v1.estimator.inputs instead.
```

2 Reading Images from S3

[2]:
```
s3 =  boto3.resource('s3', region_name='us-east-1')
bucket = s3.Bucket('cap3-hailin-du')
```

[3]:
```
fs = s3fs.S3FileSystem()
```

[ ]:
```
# assign folder path & label
BASEPATH = fs.ls('s3://cap3-hailin-du/Images')
LABELS = set()
paths = []
for d in BASEPATH:
    x = d.split('/')
    LABELS.add(x[2])
    paths.append((d,x[2]))
```

[5]:
```
# checking
s = 'cap3-hailin-du/Images/n02085620-Chihuahua/n02085620_10074.jpg'
x = s.split('/')
print(x[0])
y = x[1] + '/' + x[2] + '/' + x[3]
print(y)
print(x[2])
```

```
cap3-hailin-du
Images/n02085620-Chihuahua/n02085620_10074.jpg
n02085620-Chihuahua
```

3 Resizing Images and Download the Images

Since we cannot read images directly from the S3 bucket, we will resize the images and download them in our notebook.

After that, we will have the images locally or in SageMaker. Keep in mind that if you download the images directly they are in your root folder. Create a folder to store associated images with the correct folder name first.

In order to begin our project, we will need to create a dataframe that includes the path of images and associated labels.

[6]:
```
# Download images
def load_image(path):
    location = path.split('/')
    bucket = s3.Bucket(location[0])
    object = bucket.Object(location[1] + '/' + location[2] + '/' + location[3])
    object.download_file(location[3])
```

```
    image = cv2.imread(location[3])
    image = cv2.resize(image, (224, 224))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return image
```

[7]:
```
# Create dataframe for images path and labels
def generate_df(path):
    location = path.split('/')
    return pd.DataFrame({'image_path':[location[5] + '/' + location[6]],
                         'breed':[location[5]]})
```

[8]:
```
BASEPATH = '/home/ec2-user/SageMaker/Images/'

files = os.listdir(BASEPATH)

LABELS = set()
paths = []
for f in files:
    y = f.split('-')
    LABELS.add(f)
    paths.append((BASEPATH + f, y))
```

[9]:
```
paths[:5]
```

[9]:
```
[('/home/ec2-user/SageMaker/Images/n02108089-boxer', ['n02108089', 'boxer']),
 ('/home/ec2-user/SageMaker/Images/n02113799-standard_poodle',
  ['n02113799', 'standard_poodle']),
 ('/home/ec2-user/SageMaker/Images/n02107683-Bernese_mountain_dog',
  ['n02107683', 'Bernese_mountain_dog']),
 ('/home/ec2-user/SageMaker/Images/n02094433-Yorkshire_terrier',
  ['n02094433', 'Yorkshire_terrier']),
 ('/home/ec2-user/SageMaker/Images/n02096437-Dandie_Dinmont',
  ['n02096437', 'Dandie_Dinmont'])]
```

[10]:
```
# use folder name as breed label
X, y = [], []
i = 0
image = []
for path, label in paths:
    for image_path in os.listdir(path):
        image.append(generate_df(path + '/' + image_path))
```

[11]:
```
image = pd.concat(image)
```

[12]:
```
image.head()
```

[12]:
```
                              image_path            breed
0  n02108089-boxer/n02108089_11616.jpg  n02108089-boxer
0   n02108089-boxer/n02108089_9045.jpg  n02108089-boxer
0   n02108089-boxer/n02108089_9076.jpg  n02108089-boxer
0   n02108089-boxer/n02108089_3028.jpg  n02108089-boxer
0   n02108089-boxer/n02108089_3162.jpg  n02108089-boxer
```

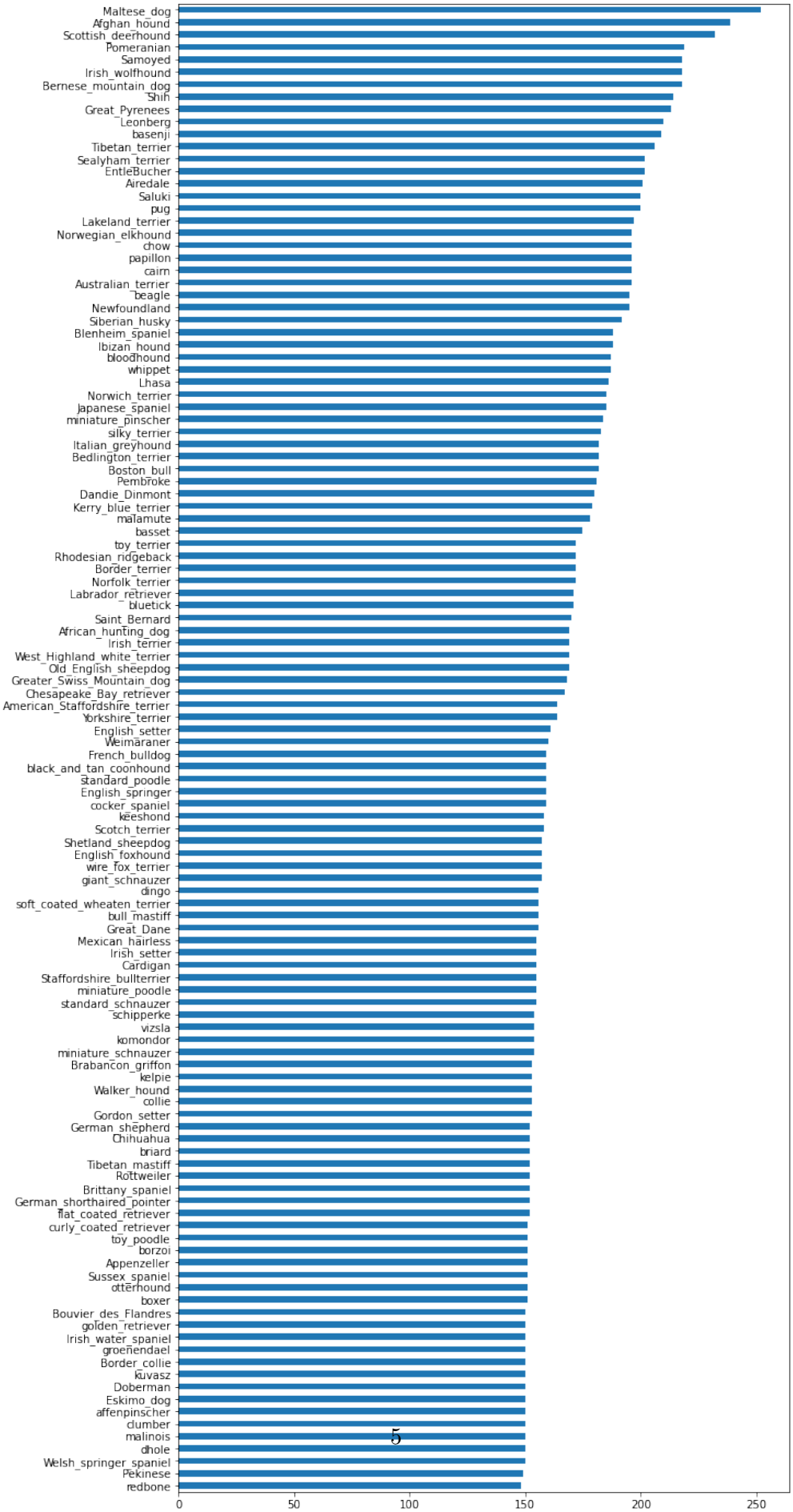
[13]:
```
# clean the label
image['breed'] = image['breed'].apply(lambda x: x.split('-')[1])
```

[14]:
```
image.head()
```

[14]:
```
                              image_path  breed
0  n02108089-boxer/n02108089_11616.jpg  boxer
0   n02108089-boxer/n02108089_9045.jpg  boxer
0   n02108089-boxer/n02108089_9076.jpg  boxer
0   n02108089-boxer/n02108089_3028.jpg  boxer
0   n02108089-boxer/n02108089_3162.jpg  boxer
```

[16]:
```
# to correct some label names
image['breed'] = image['breed'].replace(['black'],'black_and_tan_coonhound')
image['breed'] = image['breed'].replace(['wire'],'wire_fox_terrier')
image['breed'] = image['breed'].replace(['soft'],'soft_coated_wheaten_terrier')
image['breed'] = image['breed'].replace(['flat'],'flat_coated_retriever')
image['breed'] = image['breed'].replace(['curly'],'curly_coated_retriever')
image['breed'] = image['breed'].
 ↪replace(['German_short'],'German_shorthaired_pointer')
```

[17]:
```
image['breed'].value_counts().sort_values(ascending=True).plot.
 ↪barh(figsize=(10,25));
```


Maltese_dog
Afghan_hound
Scottish_deerhound
Pomeranian
Samoyed
Irish_wolfhound
Bernese_mountain_dog
Shih
Great_Pyrenees
Leonberg
basenji
Tibetan_terrier
Sealyham_terrier
EntleBucher
Airedale
Saluki
pug
Lakeland_terrier
Norwegian_elkhound
chow
papillon
cairn
Australian_terrier
beagle
Newfoundland
Siberian_husky
Blenheim_spaniel
Ibizan_hound
bloodhound
whippet
Lhasa
Norwich_terrier
Japanese_spaniel
miniature_pinscher
silky_terrier
Italian_greyhound
Bedlington_terrier
Boston_bull
Pembroke
Dandie_Dinmont
Kerry_blue_terrier
malamute
basset
toy_terrier
Rhodesian_ridgeback
Border_terrier
Norfolk_terrier
Labrador_retriever
bluetick
Saint_Bernard
African_hunting_dog
Irish_terrier
West_Highland_white_terrier
Old_English_sheepdog
Greater_Swiss_Mountain_dog
Chesapeake_Bay_retriever
American_Staffordshire_terrier
Yorkshire_terrier
English_setter
Weimaraner
French_bulldog
black_and_tan_coonhound
Standard_poodle
English_springer
cocker_spaniel
keeshond
Scotch_terrier
Shetland_sheepdog
English_foxhound
wire_fox_terrier
giant_schnauzer
dingo
soft_coated_wheaten_terrier
bull_mastiff
Great_Dane
Mexican_hairless
Irish_setter
Cardigan
Staffordshire_bullterrier
miniature_poodle
standard_schnauzer
schipperke
vizsla
komondor
miniature_schnauzer
Brabancon_griffon
kelpie
Walker_hound
collie
Gordon_setter
German_shepherd
Chihuahua
briard
Tibetan_mastiff
Rottweiler
Brittany_spaniel
German_shorthaired_pointer
flat_coated_retriever
curly_coated_retriever
toy_poodle
borzoi
Appenzeller
Sussex_spaniel
otterhound
boxer
Bouvier_des_Flandres
golden_retriever
Irish_water_spaniel
groenendael
Border_collie
kuvasz
Doberman
Eskimo_dog
affenpinscher
clumber
malinois
dhole
Welsh_springer_spaniel
Pekinese
redbone
0
50
100
150
200
250

[18]:
```
image.breed.unique()
```

[18]:
```
array(['boxer', 'standard_poodle', 'Bernese_mountain_dog',
       'Yorkshire_terrier', 'Dandie_Dinmont', 'Mexican_hairless',
       'Rottweiler', 'bull_mastiff', 'Tibetan_mastiff', 'collie',
       'vizsla', 'clumber', 'curly_coated_retriever', 'Sussex_spaniel',
       'kelpie', 'miniature_schnauzer', 'briard', 'basset',
       'Brabancon_griffon', 'otterhound', 'Brittany_spaniel',
       'Great_Dane', 'bloodhound', 'Irish_setter', 'Japanese_spaniel',
       'German_shepherd', 'Siberian_husky', 'Pekinese',
       'Bedlington_terrier', 'schipperke', 'Greater_Swiss_Mountain_dog',
       'EntleBucher', 'Maltese_dog', 'Appenzeller', 'whippet', 'Lhasa',
       'Saint_Bernard', 'Cardigan', 'Italian_greyhound',
       'Irish_water_spaniel', 'malamute', 'silky_terrier', 'dhole',
       'Bouvier_des_Flandres', 'Shetland_sheepdog', 'Irish_terrier',
       'Irish_wolfhound', 'Pomeranian', 'toy_poodle', 'redbone',
       'Chesapeake_Bay_retriever', 'Saluki', 'Norfolk_terrier',
       'Rhodesian_ridgeback', 'Walker_hound', 'Ibizan_hound',
       'Great_Pyrenees', 'toy_terrier', 'pug', 'Leonberg', 'komondor',
       'Shih', 'bluetick', 'wire_fox_terrier', 'Weimaraner',
       'Labrador_retriever', 'Border_collie', 'Chihuahua',
       'Border_terrier', 'English_setter', 'dingo', 'golden_retriever',
       'West_Highland_white_terrier', 'French_bulldog',
       'German_shorthaired_pointer', 'Norwegian_elkhound',
       'Welsh_springer_spaniel', 'flat_coated_retriever', 'Newfoundland',
       'keeshond', 'giant_schnauzer', 'Scotch_terrier',
       'Scottish_deerhound', 'English_springer', 'papillon',
       'affenpinscher', 'groenendael', 'Samoyed', 'Afghan_hound',
       'beagle', 'Eskimo_dog', 'American_Staffordshire_terrier', 'kuvasz',
       'Lakeland_terrier', 'miniature_pinscher', 'Airedale', 'borzoi',
       'Old_English_sheepdog', 'Staffordshire_bullterrier', 'Doberman',
       'chow', 'Tibetan_terrier', 'cairn', 'cocker_spaniel', 'Pembroke',
       'Kerry_blue_terrier', 'Boston_bull', 'Gordon_setter',
       'Australian_terrier', 'English_foxhound', 'African_hunting_dog',
       'standard_schnauzer', 'soft_coated_wheaten_terrier',
       'black_and_tan_coonhound', 'Norwich_terrier', 'miniature_poodle',
       'Sealyham_terrier', 'basenji', 'malinois', 'Blenheim_spaniel'],
      dtype=object)
```

4 Create Second Labels "Groups" for Images

The dataset only has images with associated breed labels. We will use the American Kennel Club's List of Breeds by Group information to create 9 labels for groups.

[19]:
```
image.loc[(image['breed'] == 'Chihuahua')
          |(image['breed'] == 'Japanese_spaniel')
          |(image['breed'] == 'Maltese_dog')
          |(image['breed'] == 'Pekinese')
          |(image['breed'] == 'Shih')
          |(image['breed'] == 'Blenheim')
          |(image['breed'] == 'papillon')
          |(image['breed'] == 'toy_terrier')
          |(image['breed'] == 'Yorkshire_terrier')
          |(image['breed'] == 'silky_terrier')
          |(image['breed'] == 'soft_coated_wheaten_terrier')
          |(image['breed'] == 'miniature_pinscher')
          |(image['breed'] == 'affenpinscher')
          |(image['breed'] == 'pug')
          |(image['breed'] == 'Pomeranian')
          |(image['breed'] == 'Brabancon_griffon')
          |(image['breed'] == 'toy_poodle')
          |(image['breed'] == 'Blenheim_spaniel'),
          'group'] = 'Toy'

image.loc[(image['breed'] == 'Rhodesian_ridgeback')
          |(image['breed'] == 'Afghan_hound')
          |(image['breed'] == 'basset')
          |(image['breed'] == 'beagle')
          |(image['breed'] == 'bloodhound')
          |(image['breed'] == 'bluetick')
          |(image['breed'] == 'black_and_tan_coonhound')
          |(image['breed'] == 'Walker_hound')
          |(image['breed'] == 'English_foxhound')
          |(image['breed'] == 'redbone')
          |(image['breed'] == 'borzoi')
          |(image['breed'] == 'Irish_wolfhound')
          |(image['breed'] == 'Italian_greyhound')
          |(image['breed'] == 'whippet')
          |(image['breed'] == 'Ibizan_hound')
          |(image['breed'] == 'Norwegian_elkhound')
          |(image['breed'] == 'otterhound')
          |(image['breed'] == 'Saluki')
          |(image['breed'] == 'Scottish_deerhound')
          |(image['breed'] == 'basenji'),
          'group'] = 'Hound'

image.loc[(image['breed'] == 'Weimaraner')
          |(image['breed'] == 'flat_coated_retriever')
          |(image['breed'] == 'curly_coated_retriever')
          |(image['breed'] == 'golden_retriever')
          |(image['breed'] == 'Labrador_retriever')
```

```
          |(image['breed'] == 'Chesapeake_Bay_retriever')
          |(image['breed'] == 'German_shorthaired_pointer')
          |(image['breed'] == 'English_setter')
          |(image['breed'] == 'Irish_setter')
          |(image['breed'] == 'Gordon_setter')
          |(image['breed'] == 'Brittany_spaniel')
          |(image['breed'] == 'clumber')
          |(image['breed'] == 'English_springer')
          |(image['breed'] == 'Welsh_springer_spaniel')
          |(image['breed'] == 'cocker_spaniel')
          |(image['breed'] == 'Sussex_spaniel')
          |(image['breed'] == 'Irish_water_spaniel')
          |(image['breed'] == 'Irish_setter')
          |(image['breed'] == 'Gordon_setter')
          |(image['breed'] == 'Brittany_spaniel')
          |(image['breed'] == 'clumber')
          |(image['breed'] == 'English_springer')
          |(image['breed'] == 'Welsh_springer_spaniel')
          |(image['breed'] == 'cocker_spaniel')
          |(image['breed'] == 'Sussex_spaniel')
          |(image['breed'] == 'Irish_water_spaniel')
          |(image['breed'] == 'vizsla'),
          'group'] = 'Sporting'

image.loc[(image['breed'] == 'Staffordshire_bullterrier')
          |(image['breed'] == 'American_Staffordshire_terrier')
          |(image['breed'] == 'Bedlington_terrier')
          |(image['breed'] == 'Border_terrier')
          |(image['breed'] == 'Norwich_terrier')
          |(image['breed'] == 'wire_fox_terrier')
          |(image['breed'] == 'Lakeland_terrier')
          |(image['breed'] == 'Sealyham_terrier')
          |(image['breed'] == 'Airedale')
          |(image['breed'] == 'cairn')
          |(image['breed'] == 'Australian_terrier')
          |(image['breed'] == 'Dandie_Dinmont')
          |(image['breed'] == 'Boston_bull')
          |(image['breed'] == 'miniature_schnauzer')
          |(image['breed'] == 'Scotch_terrier')
          |(image['breed'] == 'West_Highland_white_terrier')
          |(image['breed'] == 'Kerry_blue_terrier')
          |(image['breed'] == 'Norfolk_terrier')
          |(image['breed'] == 'Irish_terrier'),
          'group'] = 'Terrier'

image.loc[(image['breed'] == 'giant_schnauzer')
          |(image['breed'] == 'standard_schnauzer')
```

```
          |(image['breed'] == 'kuvasz')
          |(image['breed'] == 'komondor')
          |(image['breed'] == 'Rottweiler')
          |(image['breed'] == 'Doberman')
          |(image['breed'] == 'Greater_Swiss_Mountain_dog')
          |(image['breed'] == 'Bernese_mountain_dog')
          |(image['breed'] == 'boxer')
          |(image['breed'] == 'bull_mastiff')
          |(image['breed'] == 'Tibetan_mastiff')
          |(image['breed'] == 'Great_Dane')
          |(image['breed'] == 'Saint_Bernard')
          |(image['breed'] == 'malamute')
          |(image['breed'] == 'Siberian_husky')
          |(image['breed'] == 'Leonberg')
          |(image['breed'] == 'Newfoundland')
          |(image['breed'] == 'Great_Pyrenees')
          |(image['breed'] == 'Samoyed'),
          'group'] = 'Working'

image.loc[(image['breed'] == 'Tibetan_terrier')
          |(image['breed'] == 'Lhasa')
          |(image['breed'] == 'schipperke')
          |(image['breed'] == 'Eskimo_dog')
          |(image['breed'] == 'French_bulldog')
          |(image['breed'] == 'chow')
          |(image['breed'] == 'keeshond')
          |(image['breed'] == 'miniature_poodle')
          |(image['breed'] == 'standard_poodle'),
          'group'] = 'Non-Sporting'

image.loc[(image['breed'] == 'groenendael')
          |(image['breed'] == 'malinois')
          |(image['breed'] == 'briard')
          |(image['breed'] == 'Old_English_sheepdog')
          |(image['breed'] == 'Shetland_sheepdog')
          |(image['breed'] == 'collie')
          |(image['breed'] == 'Border_collie')
          |(image['breed'] == 'Bouvier_des_Flandres')
          |(image['breed'] == 'briard')
          |(image['breed'] == 'German_shepherd')
          |(image['breed'] == 'Shetland_sheepdog')
          |(image['breed'] == 'collie')
          |(image['breed'] == 'Border_collie')
          |(image['breed'] == 'EntleBucher')
          |(image['breed'] == 'Pembroke')
          |(image['breed'] == 'Cardigan'),
          'group'] = 'Herding'
```

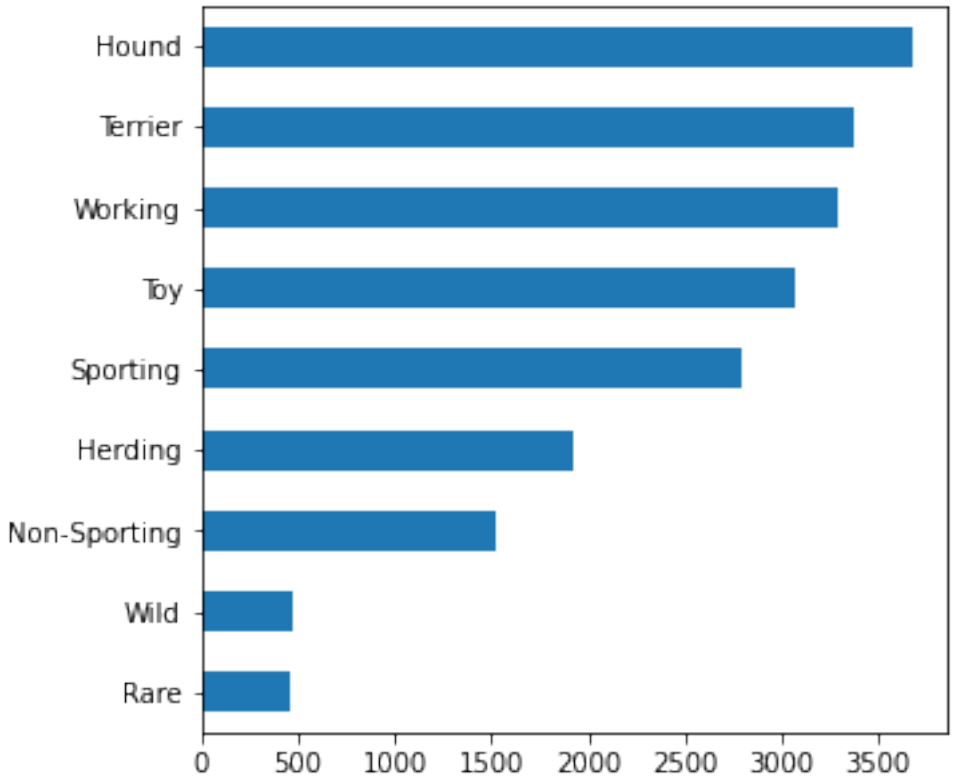
```
image.loc[(image['breed'] == 'kelpie')
          |(image['breed'] == 'Appenzeller')
          |(image['breed'] == 'Mexican_hairless'),
          'group'] = 'Rare'

image.loc[(image['breed'] == 'dingo')
          |(image['breed'] == 'dhole')
          |(image['breed'] == 'African_hunting_dog'),
          'group'] = 'Wild'
```

[20]:
```
image.groupby(['group']).agg(['count'])
```

[20]:

	image_path	breed
	count	count
group		
Herding	1921	1921
Hound	3670	3670
Non-Sporting	1523	1523
Rare	459	459
Sporting	2797	2797
Terrier	3370	3370
Toy	3068	3068
Wild	475	475
Working	3295	3295

[21]:
```
image['group'].value_counts().sort_values(ascending=True).plot.
↪barh(figsize=(5,5));
```



[22]:
```
# check any puppy without group label
image['group'].isnull().values.any()
```

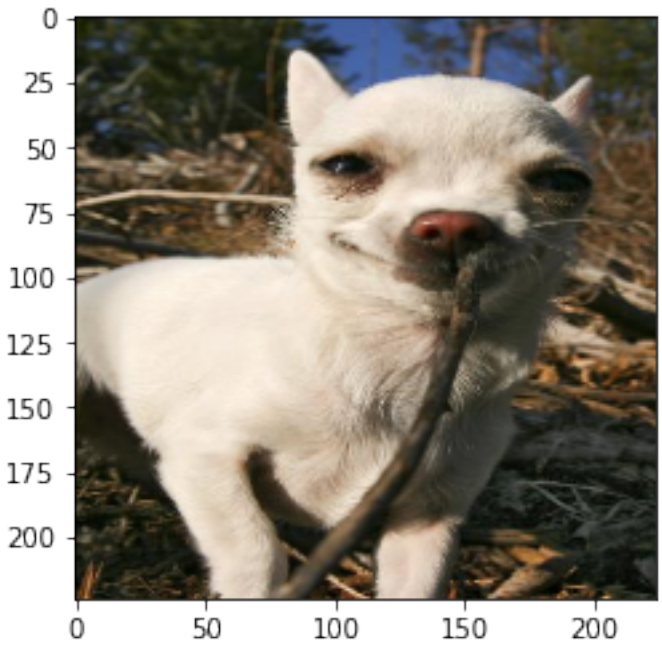
[22]:
```
False
```

[23]:
```
image.group.unique()
```

[23]:
```
array(['Working', 'Non-Sporting', 'Toy', 'Terrier', 'Rare', 'Herding',
       'Sporting', 'Hound', 'Wild'], dtype=object)
```

[24]:
```
# check we can read the image
jup_path = '/home/ec2-user/SageMaker/Images/n02085620-Chihuahua/n02085620_1073.
 ↪jpg'
```

[25]:
```
test = cv2.imread(jup_path)
test = cv2.resize(test, (224, 224))
test = cv2.cvtColor(test, cv2.COLOR_BGR2RGB)
imgplot = plt.imshow(test)
plt.show(imgplot)
```



5 Preprocessing

Before we begin building our CNN model, we will need to convert the dataframe into a form that the model can read the images. using ImageDataGenerator.

[26]:
```
# split dataset
train_df, test_df = train_test_split(image,test_size=0.25)
```

[27]:
```
# rescale the images
train_datagen = ImageDataGenerator(
        rescale=1 / 255.0,
        rotation_range=20,
        zoom_range=0.05,
        width_shift_range=0.05,
        height_shift_range=0.05,
        shear_range=0.05,
        horizontal_flip=True,
        fill_mode="nearest",
        validation_split=0.20)
test_datagen = ImageDataGenerator(rescale=1 / 255.0)
```

[28]:
```
os.getcwd()
```

[28]:
```
'/home/ec2-user/SageMaker'
```

[29]:
```
image.head()
```

[29]:

	image_path	breed	group
0	n02108089-boxer/n02108089_11616.jpg	boxer	Working
0	n02108089-boxer/n02108089_9045.jpg	boxer	Working
0	n02108089-boxer/n02108089_9076.jpg	boxer	Working
0	n02108089-boxer/n02108089_3028.jpg	boxer	Working
0	n02108089-boxer/n02108089_3162.jpg	boxer	Working

6 Split the data into three groups to fit the model

We will split the data into three groups, which are training, validation, and test dataset. We also set some parameters, such as the **batch_size** is 8.

In each generator, the data is now being split into X column (image_path) and Y column (breed).

[32]:
```
batch_size = 8
jup_path = '/home/ec2-user/SageMaker/Images/'
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=jup_path,
    x_col="image_path",
    y_col="breed",
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode="categorical",
    subset='training',
    shuffle=True,
    seed=42
)
valid_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=jup_path,
    x_col="image_path",
    y_col="breed",
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode="categorical",
    subset='validation',
    shuffle=True,
    seed=42
)
test_generator = test_datagen.flow_from_dataframe(
    dataframe=test_df,
    directory=jup_path,
    x_col="image_path",
    target_size=(224, 224),
```

```
    batch_size=1,
    class_mode=None,
    shuffle=False,
)
```

```
Found 12347 validated image filenames belonging to 120 classes.
Found 3086 validated image filenames belonging to 120 classes.
Found 5145 validated image filenames.
```

7 Convolutional Neural Network (CNN) Model

We will build our CNN model, keep in mnid that the **input_shape** and the **model.add(Dense(120, activation='softmax'))** are something you should pay attention to.

The input_shape is related to your image size, and the last layer is related to how many labels you are going to predict. For example, 120 means a total of 120 breeds.

For other layers, we can always modify it base on personal preference. These layers are related to how to train (how neural network connect) the model.

The last step is to train the model with the number of **EPOCH** you want to run.

[587]:
```
# CNN Model
model = Sequential()

model.add(Conv2D(16, (3, 3), padding='same', use_bias=False, input_shape=(224,␣
↪224, 3)))
model.add(BatchNormalization(axis=3, scale=False))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(4, 4), strides=(4, 4), padding='same'))
model.add(Dropout(0.2))

model.add(Conv2D(32, (3, 3), padding='same', use_bias=False))
model.add(BatchNormalization(axis=3, scale=False))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(4, 4), strides=(4, 4), padding='same'))
model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), padding='same', use_bias=False))
model.add(BatchNormalization(axis=3, scale=False))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(4, 4), strides=(4, 4), padding='same'))
model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3), padding='same', use_bias=False))
model.add(BatchNormalization(axis=3, scale=False))
model.add(Activation("relu"))
model.add(Flatten())
model.add(Dropout(0.2))
```

```
model.add(Dense(512, activation='relu'))
model.add(Dense(120, activation='softmax'))
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_19 (Conv2D)           (None, 224, 224, 16)      432
_________________________________________________________________
batch_normalization_16 (Batc (None, 224, 224, 16)      48
_________________________________________________________________
activation_16 (Activation)   (None, 224, 224, 16)      0
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 56, 56, 16)        0
_________________________________________________________________
dropout_16 (Dropout)         (None, 56, 56, 16)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 56, 56, 32)        4608
_________________________________________________________________
batch_normalization_17 (Batc (None, 56, 56, 32)        96
_________________________________________________________________
activation_17 (Activation)   (None, 56, 56, 32)        0
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
dropout_17 (Dropout)         (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 14, 14, 64)        18432
_________________________________________________________________
batch_normalization_18 (Batc (None, 14, 14, 64)        192
_________________________________________________________________
activation_18 (Activation)   (None, 14, 14, 64)        0
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 4, 4, 64)          0
_________________________________________________________________
dropout_18 (Dropout)         (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 4, 4, 128)         73728
_________________________________________________________________
batch_normalization_19 (Batc (None, 4, 4, 128)         384
_________________________________________________________________
activation_19 (Activation)   (None, 4, 4, 128)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0
_________________________________________________________________
dropout_19 (Dropout)         (None, 2048)              0
```

```
_________________________________________________________________
dense_2 (Dense)              (None, 512)               1049088
_________________________________________________________________
dense_3 (Dense)              (None, 120)               61560
=================================================================
Total params: 1,208,568
Trainable params: 1,208,088
Non-trainable params: 480
_________________________________________________________________
```

[588]:
```
EPOCHS = 10

checkpointer = ModelCheckpoint(filepath='/home/ec2-user/SageMaker/weights.best.
 ↪from_base1.hdf5',
                               verbose=1, save_best_only=True)

model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

base_model = model.fit(train_generator,
               validation_data=(train_generator),
               steps_per_epoch = train_generator.n // train_generator.batch_size,
               validation_steps = valid_generator.n // valid_generator.
 ↪batch_size,
               epochs=EPOCHS, callbacks=[checkpointer], verbose=1)
```

```
Epoch 1/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7449 - acc:
0.0169Epoch 1/10
 385/1543 [======>…] - ETA: 1:59 - loss: 4.6129 - acc:
0.0266
Epoch 00001: val_loss improved from inf to 4.61287, saving model to
/home/ec2-user/SageMaker/weights.best.from_base1.hdf5
1543/1543 [==============================] - 241s 156ms/step - loss: 4.7448 -
acc: 0.0169 - val_loss: 4.6129 - val_acc: 0.0266
Epoch 2/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.4946 - acc:
0.0298Epoch 1/10
 385/1543 [======>…] - ETA: 1:59 - loss: 4.5147 - acc:
0.0367
Epoch 00002: val_loss improved from 4.61287 to 4.51472, saving model to
/home/ec2-user/SageMaker/weights.best.from_base1.hdf5
1543/1543 [==============================] - 203s 131ms/step - loss: 4.4949 -
acc: 0.0298 - val_loss: 4.5147 - val_acc: 0.0367
Epoch 3/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.4519 - acc:
0.0337Epoch 1/10
```

```
 385/1543 [======>…] - ETA: 2:02 - loss: 4.5182 - acc:
0.0260
Epoch 00003: val_loss did not improve from 4.51472
1543/1543 [==============================] - 204s 132ms/step - loss: 4.4519 -
acc: 0.0337 - val_loss: 4.5182 - val_acc: 0.0260
Epoch 4/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.4384 - acc:
0.0357Epoch 1/10
 385/1543 [======>…] - ETA: 1:58 - loss: 4.3238 - acc:
0.0416
Epoch 00004: val_loss improved from 4.51472 to 4.32378, saving model to
/home/ec2-user/SageMaker/weights.best.from_base1.hdf5
1543/1543 [==============================] - 202s 131ms/step - loss: 4.4385 -
acc: 0.0357 - val_loss: 4.3238 - val_acc: 0.0416
Epoch 5/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.4337 - acc:
0.0375Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 4.3208 - acc:
0.0412
Epoch 00005: val_loss improved from 4.32378 to 4.32079, saving model to
/home/ec2-user/SageMaker/weights.best.from_base1.hdf5
1543/1543 [==============================] - 203s 131ms/step - loss: 4.4339 -
acc: 0.0374 - val_loss: 4.3208 - val_acc: 0.0412
Epoch 6/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.4283 - acc:
0.0364Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 4.4012 - acc:
0.0464
Epoch 00006: val_loss did not improve from 4.32079
1543/1543 [==============================] - 202s 131ms/step - loss: 4.4290 -
acc: 0.0364 - val_loss: 4.4012 - val_acc: 0.0464
Epoch 7/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.4086 - acc:
0.0384Epoch 1/10
 385/1543 [======>…] - ETA: 1:59 - loss: 4.3066 - acc:
0.0396
Epoch 00007: val_loss improved from 4.32079 to 4.30656, saving model to
/home/ec2-user/SageMaker/weights.best.from_base1.hdf5
1543/1543 [==============================] - 203s 132ms/step - loss: 4.4093 -
acc: 0.0384 - val_loss: 4.3066 - val_acc: 0.0396
Epoch 8/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.3757 - acc:
0.0453Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 4.2692 - acc:
0.0581
Epoch 00008: val_loss improved from 4.30656 to 4.26924, saving model to
/home/ec2-user/SageMaker/weights.best.from_base1.hdf5
1543/1543 [==============================] - 204s 132ms/step - loss: 4.3759 -
```

```
acc: 0.0453 - val_loss: 4.2692 - val_acc: 0.0581
Epoch 9/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.3648 - acc:
0.0457Epoch 1/10
 385/1543 [======>…] - ETA: 1:59 - loss: 4.3918 - acc:
0.0526
Epoch 00009: val_loss did not improve from 4.26924
1543/1543 [==============================] - 205s 133ms/step - loss: 4.3647 -
acc: 0.0457 - val_loss: 4.3918 - val_acc: 0.0526
Epoch 10/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.3431 - acc:
0.0483Epoch 1/10
 385/1543 [======>…] - ETA: 1:59 - loss: 4.3811 - acc:
0.0351
Epoch 00010: val_loss did not improve from 4.26924
1543/1543 [==============================] - 204s 132ms/step - loss: 4.3429 -
acc: 0.0482 - val_loss: 4.3811 - val_acc: 0.0351
```

8 Checking the Performance

As we can see below, the result is pretty bad with around 3.5% accuracy, but it still outperforms random guess.

Also, the model shows a sign that we can continue to train the model to improve the accuracy by increasing the number of EPOCH. Also, we should always pay attention to the **model.compile** where you are going to select the **optimizer** and the **metrics**.

We also want to see how the confusion matrix performs.

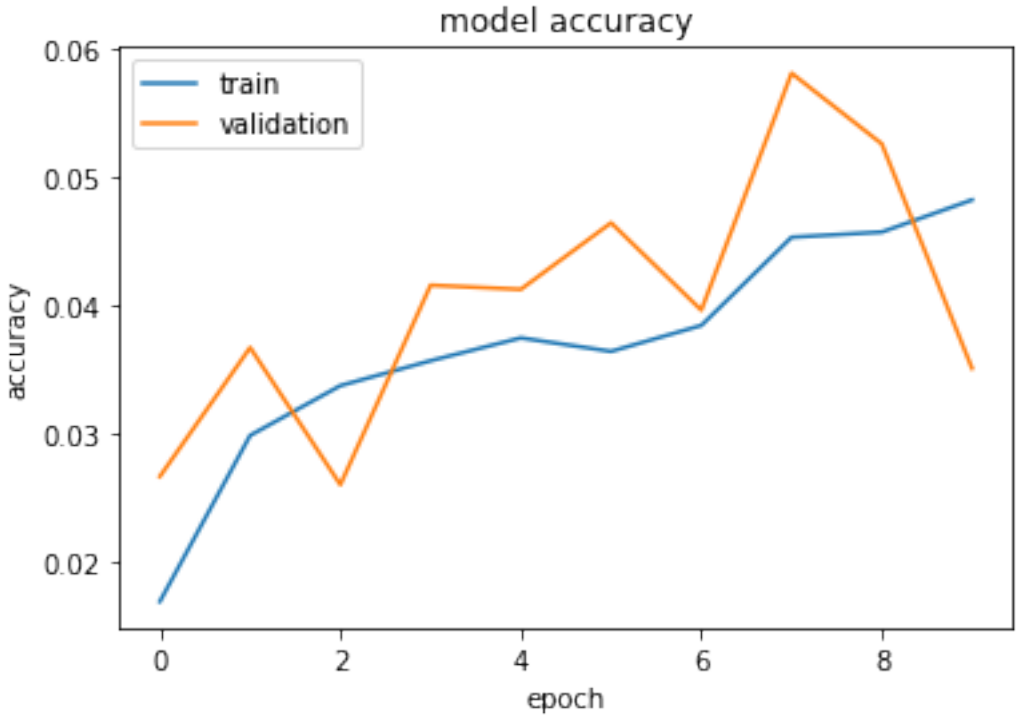
We can always build other models and compare them with the baseline model.

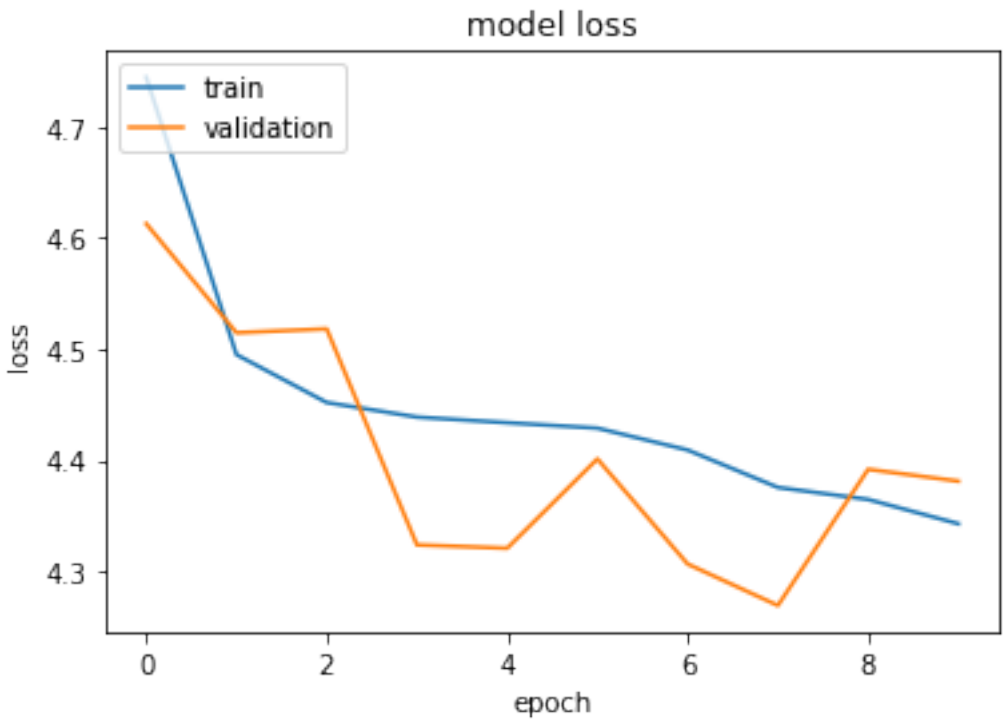
[46]:
```
def report_accuracy(base_model, generator):
    score = base_model.model.evaluate_generator(generator)
    print('Loss:', score[0])
    print('Accuracy:', score[1])
```

[591]:
```
print('For training set:')
report_accuracy(base_model, train_generator)
print('------------------')
print('For validation set:')
report_accuracy(base_model,valid_generator)
```

```
For training set:
Loss: 4.386176852364614
Accuracy: 0.03879485
------------------
For validation set:
Loss: 4.472330960585046
Accuracy: 0.034024626
```

[592]:
```
plt.plot(base_model.history['acc'])
plt.plot(base_model.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(base_model.history['loss'])
plt.plot(base_model.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```





[51]:
```
!pip install mlxtend
```

```
Collecting mlxtend
  Downloading mlxtend-0.18.0-py2.py3-none-any.whl (1.3 MB)
     |                  | 1.3 MB 19.1 MB/s eta 0:00:01
Requirement already satisfied: scipy>=1.2.1 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
mlxtend) (1.5.3)
Requirement already satisfied: numpy>=1.16.2 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
mlxtend) (1.18.5)
Requirement already satisfied: matplotlib>=3.0.0 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
mlxtend) (3.2.1)
Requirement already satisfied: pandas>=0.24.2 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
mlxtend) (1.1.5)
Requirement already satisfied: joblib>=0.13.2 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
mlxtend) (1.0.1)
Requirement already satisfied: scikit-learn>=0.20.3 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
mlxtend) (0.24.1)
Requirement already satisfied: setuptools in
```

```
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
mlxtend) (49.6.0.post20210108)
Requirement already satisfied: python-dateutil>=2.1 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
matplotlib>=3.0.0->mlxtend) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
matplotlib>=3.0.0->mlxtend) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
matplotlib>=3.0.0->mlxtend) (2.4.7)
Requirement already satisfied: cycler>=0.10 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
matplotlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: six in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
cycler>=0.10->matplotlib>=3.0.0->mlxtend) (1.15.0)
Requirement already satisfied: pytz>=2017.2 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
pandas>=0.24.2->mlxtend) (2021.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages (from
scikit-learn>=0.20.3->mlxtend) (2.1.0)
Installing collected packages: mlxtend
Successfully installed mlxtend-0.18.0
```

[52]:
```
import mlxtend
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import classification_report, confusion_matrix
```

[612]:
```
target = list(image.breed.unique())
```

[620]:
```
Y_pred = model.predict_generator(valid_generator)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
cf = confusion_matrix(valid_generator.classes, y_pred)
print(cf)
print('Classification Report')
target_names = target
print(classification_report(valid_generator.classes, y_pred,␣
 ↪target_names=target_names))
```

```
Confusion Matrix
[[2 0 3 … 0 0 0]
 [0 1 2 … 0 0 0]
 [1 0 2 … 0 0 0]
 …
 [0 0 4 … 0 0 0]
```

```
 [0 0 6 … 0 0 0]
 [0 0 4 … 0 0 0]]
Classification Report
```

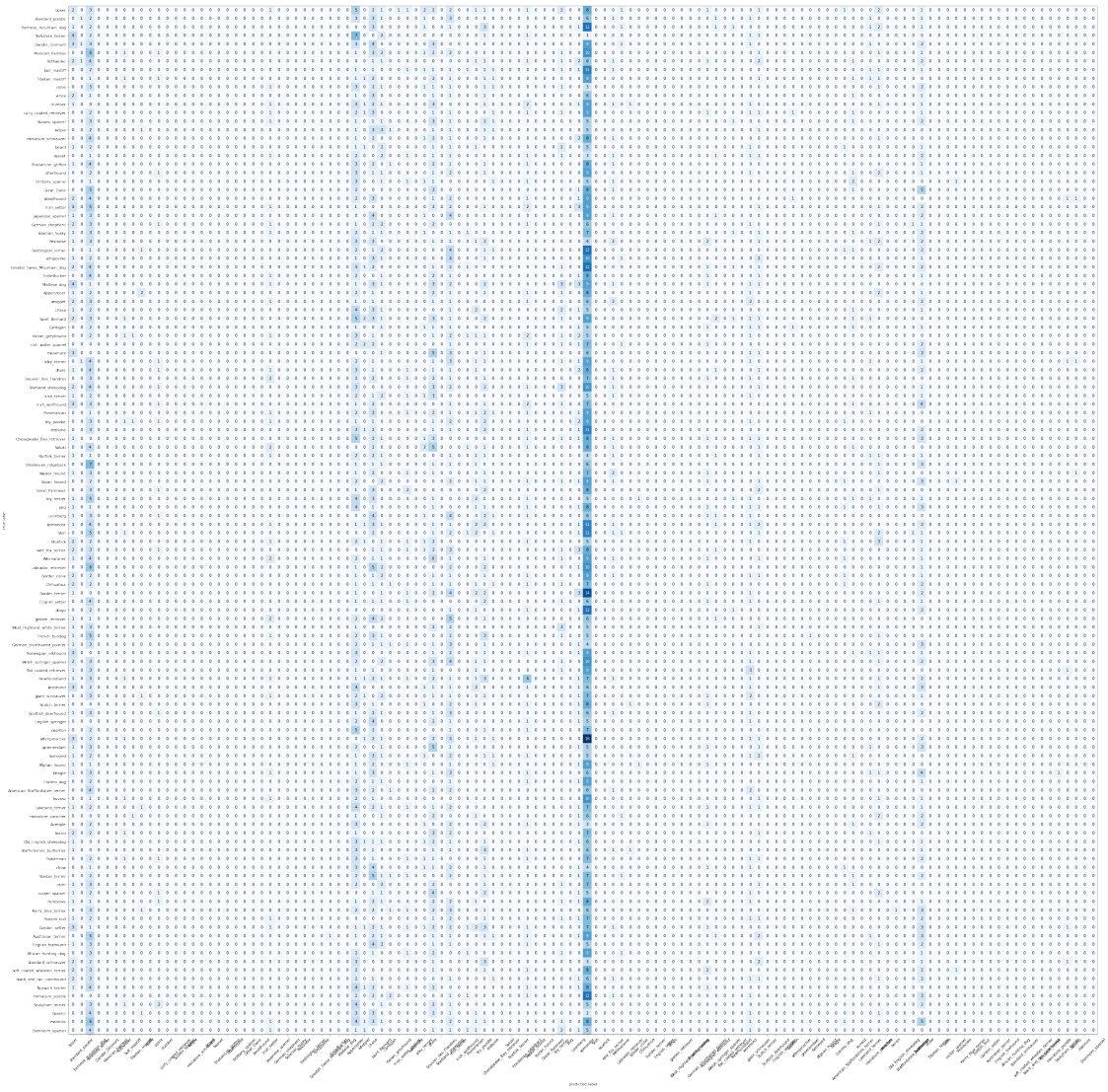
	precision	recall	f1-score	support
boxer	0.02	0.05	0.03	42
standard_poodle	0.25	0.04	0.07	25
Bernese_mountain_dog	0.01	0.06	0.01	31
Yorkshire_terrier	0.00	0.00	0.00	20
Dandie_Dinmont	0.00	0.00	0.00	31
Mexican_hairless	0.00	0.00	0.00	33
Rottweiler	0.00	0.00	0.00	28
bull_mastiff	0.00	0.00	0.00	28
Tibetan_mastiff	0.00	0.00	0.00	25
collie	0.00	0.00	0.00	22
vizsla	0.00	0.00	0.00	18
clumber	0.00	0.00	0.00	30
curly_coated_retriever	0.00	0.00	0.00	29
Sussex_spaniel	0.00	0.00	0.00	23
kelpie	0.00	0.00	0.00	21
miniature_schnauzer	0.00	0.00	0.00	24
briard	0.00	0.00	0.00	20
basset	0.00	0.00	0.00	20
Brabancon_griffon	0.00	0.00	0.00	26
otterhound	0.00	0.00	0.00	26
Brittany_spaniel	0.00	0.00	0.00	19
Great_Dane	0.00	0.00	0.00	26
bloodhound	0.00	0.00	0.00	28
Irish_setter	0.03	0.03	0.03	33
Japanese_spaniel	0.00	0.00	0.00	26
German_shepherd	0.00	0.00	0.00	25
Siberian_husky	0.00	0.00	0.00	24
Pekinese	0.00	0.00	0.00	29
Bedlington_terrier	0.00	0.00	0.00	29
schipperke	0.00	0.00	0.00	27
Greater_Swiss_Mountain_dog	0.00	0.00	0.00	36
EntleBucher	0.00	0.00	0.00	23
Maltese_dog	0.00	0.00	0.00	37
Appenzeller	0.01	0.08	0.02	26
whippet	0.00	0.00	0.00	23
Lhasa	0.02	0.12	0.03	24
Saint_Bernard	0.01	0.03	0.02	40
Cardigan	0.00	0.00	0.00	19
Italian_greyhound	0.00	0.00	0.00	26
Irish_water_spaniel	0.00	0.00	0.00	21
malamute	0.00	0.00	0.00	24
silky_terrier	0.00	0.00	0.00	31
dhole	0.01	0.03	0.01	30

Bouvier_des_Flandres	0.00	0.00	0.00	25
Shetland_sheepdog	0.01	0.06	0.02	34
Irish_terrier	0.00	0.00	0.00	21
Irish_wolfhound	0.00	0.00	0.00	27
Pomeranian	0.03	0.04	0.03	27
toy_poodle	0.02	0.07	0.03	27
redbone	0.00	0.00	0.00	28
Chesapeake_Bay_retriever	0.00	0.00	0.00	31
Saluki	0.00	0.00	0.00	29
Norfolk_terrier	0.00	0.00	0.00	21
Rhodesian_ridgeback	0.00	0.00	0.00	23
Walker_hound	0.00	0.00	0.00	22
Ibizan_hound	0.00	0.00	0.00	30
Great_Pyrenees	0.00	0.00	0.00	23
toy_terrier	0.00	0.00	0.00	28
pug	0.00	0.00	0.00	28
Leonberg	0.00	0.00	0.00	27
komondor	0.01	0.30	0.02	37
Shih	0.00	0.00	0.00	34
bluetick	0.00	0.00	0.00	25
wire_fox_terrier	0.00	0.00	0.00	29
Weimaraner	0.00	0.00	0.00	32
Labrador_retriever	0.00	0.00	0.00	30
Border_collie	0.00	0.00	0.00	26
Chihuahua	0.00	0.00	0.00	24
Border_terrier	0.00	0.00	0.00	34
English_setter	0.00	0.00	0.00	21
dingo	0.00	0.00	0.00	21
golden_retriever	0.00	0.00	0.00	25
West_Highland_white_terrier	0.00	0.00	0.00	18
French_bulldog	0.00	0.00	0.00	26
German_shorthaired_pointer	0.00	0.00	0.00	23
Norwegian_elkhound	0.00	0.00	0.00	26
Welsh_springer_spaniel	0.00	0.00	0.00	33
flat_coated_retriever	0.00	0.00	0.00	22
Newfoundland	0.00	0.00	0.00	38
keeshond	0.02	0.04	0.02	27
giant_schnauzer	0.00	0.00	0.00	25
Scotch_terrier	0.00	0.00	0.00	22
Scottish_deerhound	0.00	0.00	0.00	24
English_springer	0.00	0.00	0.00	17
papillon	0.00	0.00	0.00	19
affenpinscher	0.00	0.00	0.00	37
groenendael	0.00	0.00	0.00	25
Samoyed	0.00	0.00	0.00	23
Afghan_hound	0.00	0.00	0.00	20
beagle	0.00	0.00	0.00	27
Eskimo_dog	0.00	0.00	0.00	19

```
American_Staffordshire_terrier       0.00      0.00      0.00        23
                        kuvasz       0.00      0.00      0.00        19
              Lakeland_terrier       0.00      0.00      0.00        27
            miniature_pinscher       0.04      0.11      0.06        18
                      Airedale       0.00      0.00      0.00        19
                        borzoi       0.00      0.00      0.00        20
          Old_English_sheepdog       0.00      0.00      0.00        21
     Staffordshire_bullterrier       0.00      0.00      0.00        20
                      Doberman       0.01      0.08      0.02        24
                         chow        0.00      0.00      0.00        21
               Tibetan_terrier       0.00      0.00      0.00        27
                         cairn       0.00      0.00      0.00        21
                cocker_spaniel       0.00      0.00      0.00        21
                      Pembroke       0.00      0.00      0.00        26
            Kerry_blue_terrier       0.00      0.00      0.00        27
                   Boston_bull       0.00      0.00      0.00        21
                Gordon_setter        0.00      0.00      0.00        36
            Australian_terrier       0.00      0.00      0.00        32
              English_foxhound       0.00      0.00      0.00        21
          African_hunting_dog        0.00      0.00      0.00        22
           standard_schnauzer        0.00      0.00      0.00        18
   soft_coated_wheaten_terrier       0.00      0.00      0.00        23
       black_and_tan_coonhound       0.00      0.00      0.00        23
               Norwich_terrier       0.00      0.00      0.00        26
            miniature_poodle         0.00      0.00      0.00        26
            Sealyham_terrier         0.00      0.00      0.00        21
                     basenji         0.00      0.00      0.00        19
                    malinois         0.00      0.00      0.00        36
            Blenheim_spaniel         0.00      0.00      0.00        20

                    accuracy                             0.01      3086
                   macro avg       0.00      0.01      0.00      3086
                weighted avg       0.00      0.01      0.00      3086


/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
```

```
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[633]:
```
plot_confusion_matrix(conf_mat=cf,
                      class_names=target,
                      figsize=(50,50));
```



[634]:
```
# second model
model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape=(224,224,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(120))
model.add(Activation('softmax'))

model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_23 (Conv2D)           (None, 222, 222, 32)      896
_________________________________________________________________
activation_20 (Activation)   (None, 222, 222, 32)      0
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 109, 109, 32)      9248
_________________________________________________________________
activation_21 (Activation)   (None, 109, 109, 32)      0
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 54, 54, 32)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 52, 52, 64)        18496
_________________________________________________________________
activation_22 (Activation)   (None, 52, 52, 64)        0
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 26, 26, 64)        0
_________________________________________________________________
flatten_5 (Flatten)          (None, 43264)             0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                2768960
_________________________________________________________________
activation_23 (Activation)   (None, 64)                0
_________________________________________________________________
dropout_20 (Dropout)         (None, 64)                0
```

```
_________________________________________________________________
dense_5 (Dense)              (None, 120)               7800
_________________________________________________________________
activation_24 (Activation)   (None, 120)               0
=================================================================
Total params: 2,805,400
Trainable params: 2,805,400
Non-trainable params: 0
_________________________________________________________________
```

[635]:
```
EPOCHS = 10

checkpointer = ModelCheckpoint(filepath='/home/ec2-user/SageMaker/weights.best.
 ↪from_base2.hdf5',
                               verbose=1, save_best_only=True)
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

base_model2 = model.fit(train_generator,
               validation_data=(train_generator),
               steps_per_epoch = train_generator.n // train_generator.batch_size,
               validation_steps = valid_generator.n // valid_generator.
 ↪batch_size,
               epochs=EPOCHS, callbacks=[checkpointer], verbose=1)
```

```
Epoch 1/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7894 - acc:
0.0109Epoch 1/10
 385/1543 [======>…] - ETA: 2:02 - loss: 4.7793 - acc:
0.0123
Epoch 00001: val_loss improved from inf to 4.77933, saving model to
/home/ec2-user/SageMaker/weights.best.from_base2.hdf5
1543/1543 [==============================] - 206s 134ms/step - loss: 4.7895 -
acc: 0.0109 - val_loss: 4.7793 - val_acc: 0.0123
Epoch 2/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7819 - acc:
0.0121Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 4.7791 - acc:
0.0133
Epoch 00002: val_loss improved from 4.77933 to 4.77907, saving model to
/home/ec2-user/SageMaker/weights.best.from_base2.hdf5
1543/1543 [==============================] - 205s 133ms/step - loss: 4.7818 -
acc: 0.0121 - val_loss: 4.7791 - val_acc: 0.0133
Epoch 3/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7788 - acc:
0.0121Epoch 1/10
 385/1543 [======>…] - ETA: 2:01 - loss: 4.7561 - acc:
```

```
0.0110
Epoch 00003: val_loss improved from 4.77907 to 4.75614, saving model to
/home/ec2-user/SageMaker/weights.best.from_base2.hdf5
1543/1543 [==============================] - 205s 133ms/step - loss: 4.7788 -
acc: 0.0122 - val_loss: 4.7561 - val_acc: 0.0110
Epoch 4/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7516 - acc:
0.0138Epoch 1/10
 385/1543 [======>…] - ETA: 1:58 - loss: 4.6982 - acc:
0.0214
Epoch 00004: val_loss improved from 4.75614 to 4.69819, saving model to
/home/ec2-user/SageMaker/weights.best.from_base2.hdf5
1543/1543 [==============================] - 205s 133ms/step - loss: 4.7516 -
acc: 0.0138 - val_loss: 4.6982 - val_acc: 0.0214
Epoch 5/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7143 - acc:
0.0185Epoch 1/10
 385/1543 [======>…] - ETA: 1:59 - loss: 4.6596 - acc:
0.0208
Epoch 00005: val_loss improved from 4.69819 to 4.65955, saving model to
/home/ec2-user/SageMaker/weights.best.from_base2.hdf5
1543/1543 [==============================] - 204s 132ms/step - loss: 4.7142 -
acc: 0.0185 - val_loss: 4.6596 - val_acc: 0.0208
Epoch 6/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.6795 - acc:
0.0184Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 4.6019 - acc:
0.0231
Epoch 00006: val_loss improved from 4.65955 to 4.60186, saving model to
/home/ec2-user/SageMaker/weights.best.from_base2.hdf5
1543/1543 [==============================] - 205s 133ms/step - loss: 4.6795 -
acc: 0.0185 - val_loss: 4.6019 - val_acc: 0.0231
Epoch 7/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.6666 - acc:
0.0185Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 4.6378 - acc:
0.0211
Epoch 00007: val_loss did not improve from 4.60186
1543/1543 [==============================] - 204s 132ms/step - loss: 4.6666 -
acc: 0.0185 - val_loss: 4.6378 - val_acc: 0.0211
Epoch 8/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.6430 - acc:
0.0212Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 4.5749 - acc:
0.0221
Epoch 00008: val_loss improved from 4.60186 to 4.57489, saving model to
/home/ec2-user/SageMaker/weights.best.from_base2.hdf5
1543/1543 [==============================] - 205s 133ms/step - loss: 4.6431 -
```

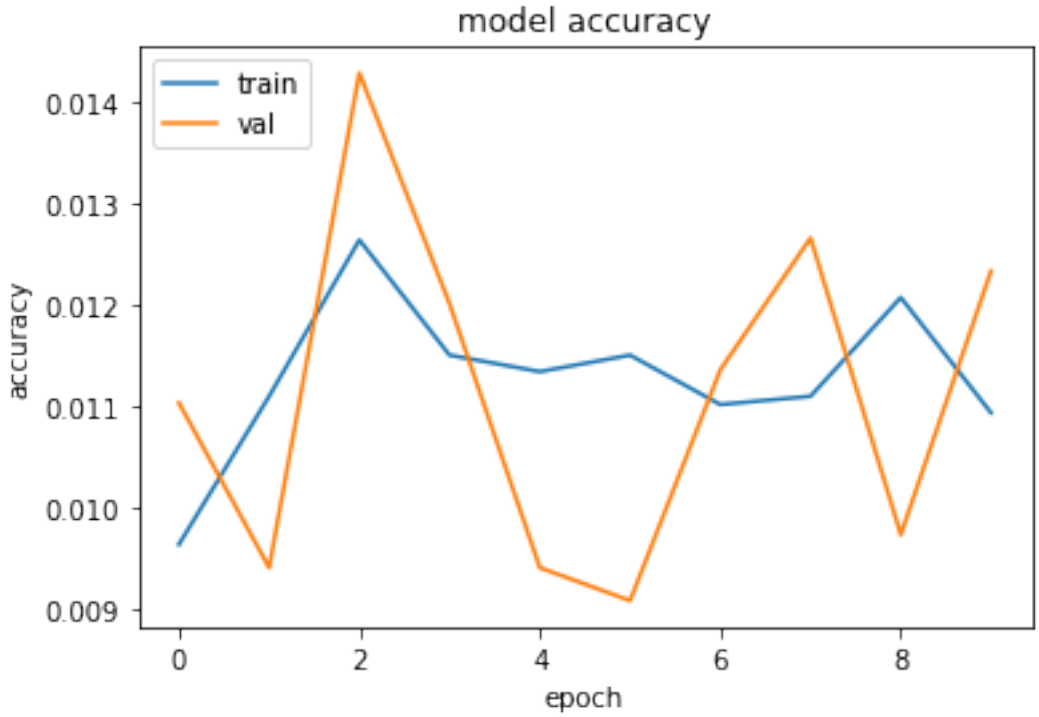
```
acc: 0.0212 - val_loss: 4.5749 - val_acc: 0.0221
Epoch 9/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.6144 - acc:
0.0242Epoch 1/10
 385/1543 [======>…] - ETA: 1:59 - loss: 4.5241 - acc:
0.0308
Epoch 00009: val_loss improved from 4.57489 to 4.52405, saving model to
/home/ec2-user/SageMaker/weights.best.from_base2.hdf5
1543/1543 [==============================] - 204s 132ms/step - loss: 4.6147 -
acc: 0.0242 - val_loss: 4.5241 - val_acc: 0.0308
Epoch 10/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.5827 - acc:
0.0264Epoch 1/10
 385/1543 [======>…] - ETA: 2:01 - loss: 4.4543 - acc:
0.0344
Epoch 00010: val_loss improved from 4.52405 to 4.45431, saving model to
/home/ec2-user/SageMaker/weights.best.from_base2.hdf5
1543/1543 [==============================] - 205s 133ms/step - loss: 4.5827 -
acc: 0.0263 - val_loss: 4.4543 - val_acc: 0.0344
```

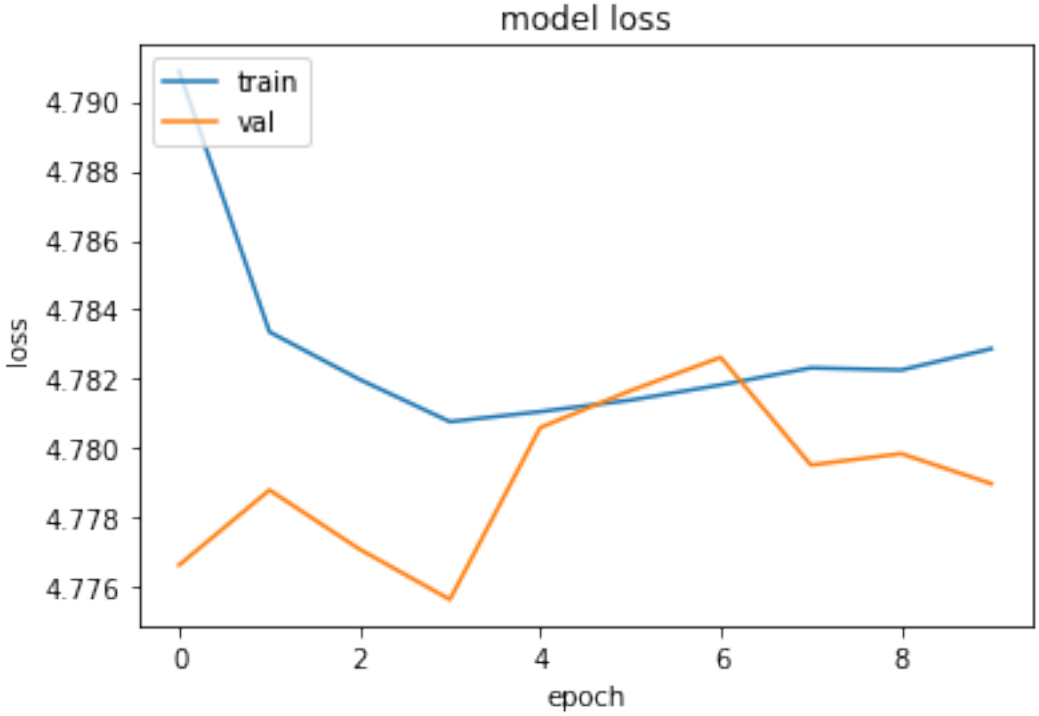
[636]:
```
print('For training set:')
report_accuracy(base_model2, train_generator)
print('------------------')
print('For validation set:')
report_accuracy(base_model2, valid_generator)
```

```
For training set:
Loss: 4.453738087199513
Accuracy: 0.036122136
------------------
For validation set:
Loss: 4.485526751977792
Accuracy: 0.030784186
```

[165]:
```
plt.plot(base_model2.history['acc'])
plt.plot(base_model2.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.plot(base_model2.history['loss'])
plt.plot(base_model2.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
```

```
plt.show()
```


model accuracy
train
val
0.014
0.013
0.012
0.011
0.010
0.009
accuracy
0
2
4
6
8
epoch


model loss
train
val
4.790
4.788
4.786
4.784
4.782
4.780
4.778
4.776
loss
0
2
4
6
8
epoch

9 CNN Model for the Group Label

Since the breed labels are doing poorly, we want to check how well the model will work on classifying groups.

As you may notice that we now changed the Y Column from **breed** to **group**.

We are using the same layer we built for our breed.

[637]:
```
batch_size = 8
jup_path = '/home/ec2-user/SageMaker/Images'
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=jup_path,
    x_col="image_path",
    y_col="group",
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode="categorical",
    subset='training',
    shuffle=True,
    seed=42
)
valid_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=jup_path,
    x_col="image_path",
    y_col="group",
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode="categorical",
    subset='validation',
    shuffle=True,
    seed=42
)
test_generator = test_datagen.flow_from_dataframe(
    dataframe=test_df,
    directory=jup_path,
    x_col="image_path",
    target_size=(224, 224),
    batch_size=1,
    class_mode=None,
    shuffle=False,
)
```

```
Found 12347 validated image filenames belonging to 9 classes.
Found 3086 validated image filenames belonging to 9 classes.
```

```
Found 5145 validated image filenames.
```

[638]:
```
model = Sequential()

model.add(Conv2D(16, (3, 3), padding='same', use_bias=False, input_shape=(224,␣
↪224, 3)))
model.add(BatchNormalization(axis=3, scale=False))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(4, 4), strides=(4, 4), padding='same'))
model.add(Dropout(0.2))

model.add(Conv2D(32, (3, 3), padding='same', use_bias=False))
model.add(BatchNormalization(axis=3, scale=False))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(4, 4), strides=(4, 4), padding='same'))
model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), padding='same', use_bias=False))
model.add(BatchNormalization(axis=3, scale=False))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(4, 4), strides=(4, 4), padding='same'))
model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3), padding='same', use_bias=False))
model.add(BatchNormalization(axis=3, scale=False))
model.add(Activation("relu"))
model.add(Flatten())
model.add(Dropout(0.2))

model.add(Dense(512, activation='relu'))
model.add(Dense(9, activation='softmax'))
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_26 (Conv2D)           (None, 224, 224, 16)      432
_________________________________________________________________
batch_normalization_20 (Batc (None, 224, 224, 16)      48
_________________________________________________________________
activation_25 (Activation)   (None, 224, 224, 16)      0
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 56, 56, 16)        0
_________________________________________________________________
dropout_21 (Dropout)         (None, 56, 56, 16)        0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 56, 56, 32)        4608
```

```
_________________________________________________________________
batch_normalization_21 (Batc (None, 56, 56, 32)        96
_________________________________________________________________
activation_26 (Activation)   (None, 56, 56, 32)        0
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
dropout_22 (Dropout)         (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 14, 14, 64)        18432
_________________________________________________________________
batch_normalization_22 (Batc (None, 14, 14, 64)        192
_________________________________________________________________
activation_27 (Activation)   (None, 14, 14, 64)        0
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 4, 4, 64)          0
_________________________________________________________________
dropout_23 (Dropout)         (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 4, 4, 128)         73728
_________________________________________________________________
batch_normalization_23 (Batc (None, 4, 4, 128)         384
_________________________________________________________________
activation_28 (Activation)   (None, 4, 4, 128)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 2048)              0
_________________________________________________________________
dropout_24 (Dropout)         (None, 2048)              0
_________________________________________________________________
dense_6 (Dense)              (None, 512)               1049088
_________________________________________________________________
dense_7 (Dense)              (None, 9)                 4617
=================================================================
Total params: 1,151,625
Trainable params: 1,151,145
Non-trainable params: 480
_________________________________________________________________
```

[ ]:
```
# EPOCHS = 10

checkpointer = ModelCheckpoint(filepath='/home/ec2-user/SageMaker/weights.best.
 ↪from_class_base1.hdf5',
                               verbose=1, save_best_only=True)

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
base_group_model = model.fit(train_generator,
              validation_data=(train_generator),
              steps_per_epoch = train_generator.n // train_generator.batch_size,
              validation_steps = valid_generator.n // valid_generator.
 ↪batch_size,
              epochs=EPOCHS, callbacks=[checkpointer], verbose=1)
```

10 Checking Performance for Groups

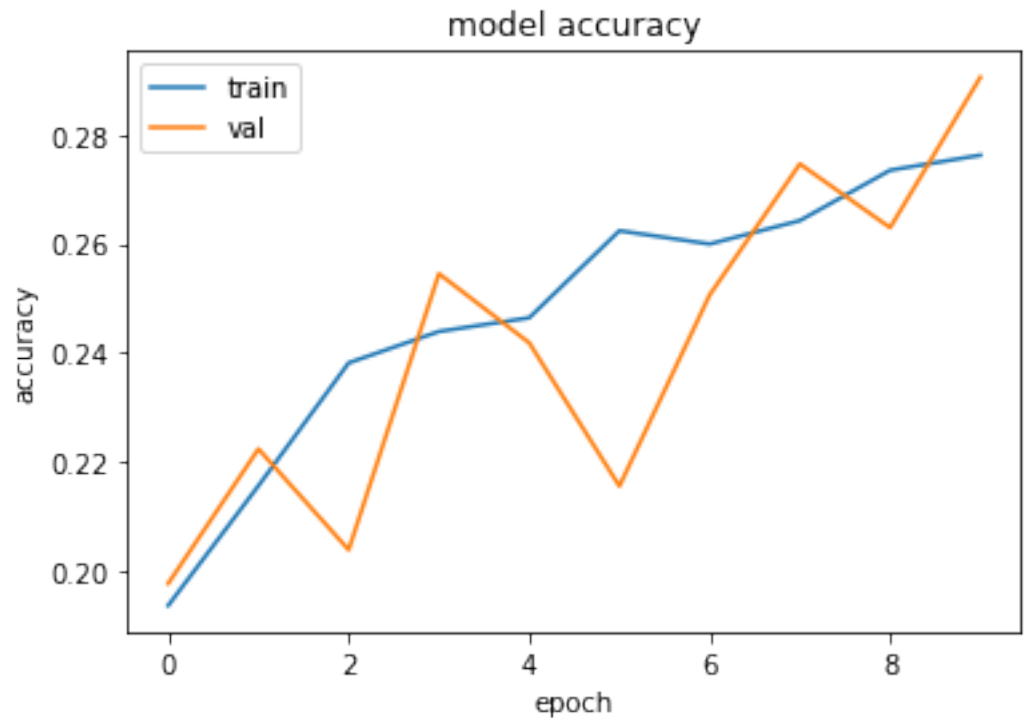
The accuracy is not bad, we have around 28%, which definitely better than a random guess.

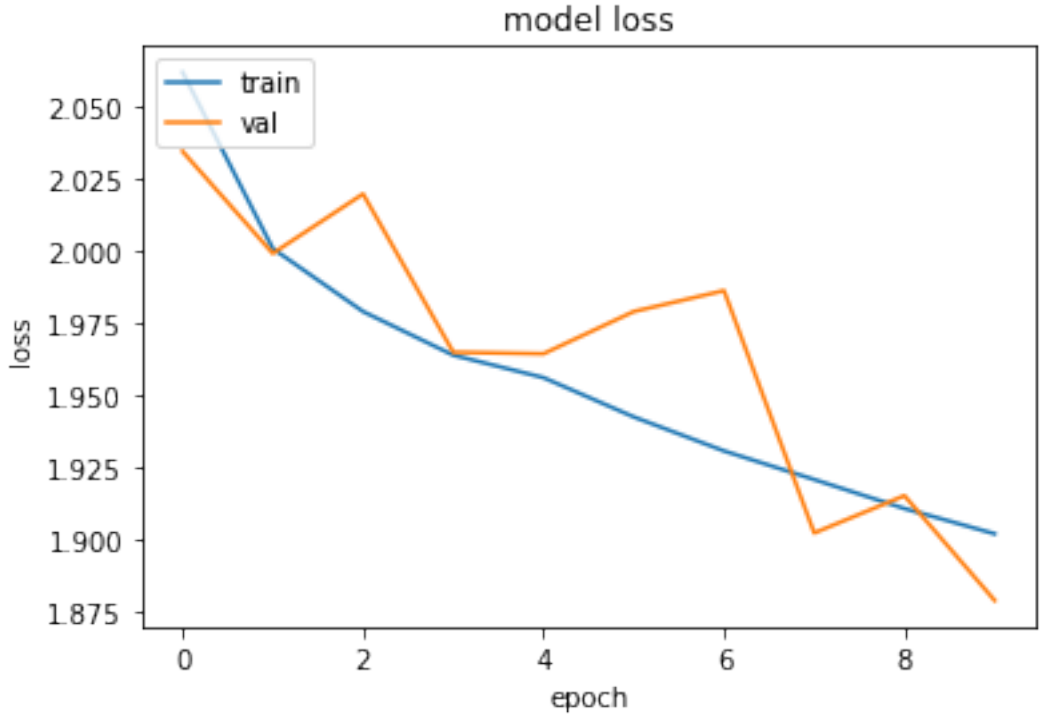
We tried to use different optimizers but the difference is not much.

[640]:
```
print('For training set:')
report_accuracy(base_group_model, train_generator)
print('------------------')
print('For validation set:')
report_accuracy(base_group_model, valid_generator)
```

```
For training set:
Loss: 1.8766826538687542
Accuracy: 0.28184983
------------------
For validation set:
Loss: 1.910685713118222
Accuracy: 0.25631887
```

[641]:
```
plt.plot(base_group_model.history['acc'])
plt.plot(base_group_model.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.plot(base_group_model.history['loss'])
plt.plot(base_group_model.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```


model accuracy
train
val
0.28
0.26
0.24
0.22
0.20
accuracy
0
2
4
6
8
epoch


model loss
train
val
2.050
2.025
2.000
1.975
1.950
1.925
1.900
1.875
loss
0
2
4
6
8
epoch

[642]:
```
EPOCHS = 10

checkpointer = ModelCheckpoint(filepath='/home/ec2-user/SageMaker/weights.best.
 ↪from_class_base1.hdf5',
                               verbose=1, save_best_only=True)

model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

base_group_model2 = model.fit(train_generator,
               validation_data=(train_generator),
               steps_per_epoch = train_generator.n // train_generator.batch_size,
               validation_steps = valid_generator.n // valid_generator.
 ↪batch_size,
               epochs=EPOCHS, callbacks=[checkpointer], verbose=1)
```

```
Epoch 1/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.8939 - acc:
0.2855Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 1.8968 - acc:
0.2737
Epoch 00001: val_loss improved from inf to 1.89683, saving model to
/home/ec2-user/SageMaker/weights.best.from_class_base1.hdf5
1543/1543 [==============================] - 206s 134ms/step - loss: 1.8940 -
acc: 0.2854 - val_loss: 1.8968 - val_acc: 0.2737
Epoch 2/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.9004 - acc:
0.2786Epoch 1/10
 385/1543 [======>…] - ETA: 1:59 - loss: 1.8776 - acc:
0.3032
Epoch 00002: val_loss improved from 1.89683 to 1.87762, saving model to
/home/ec2-user/SageMaker/weights.best.from_class_base1.hdf5
1543/1543 [==============================] - 203s 132ms/step - loss: 1.9004 -
acc: 0.2785 - val_loss: 1.8776 - val_acc: 0.3032
Epoch 3/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.9023 - acc:
0.2843Epoch 1/10
 385/1543 [======>…] - ETA: 1:59 - loss: 1.9172 - acc:
0.2578
Epoch 00003: val_loss did not improve from 1.87762
1543/1543 [==============================] - 203s 132ms/step - loss: 1.9022 -
acc: 0.2845 - val_loss: 1.9172 - val_acc: 0.2578
Epoch 4/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.9107 - acc:
0.2810Epoch 1/10
 385/1543 [======>…] - ETA: 1:59 - loss: 1.8609 - acc:
0.3003
```

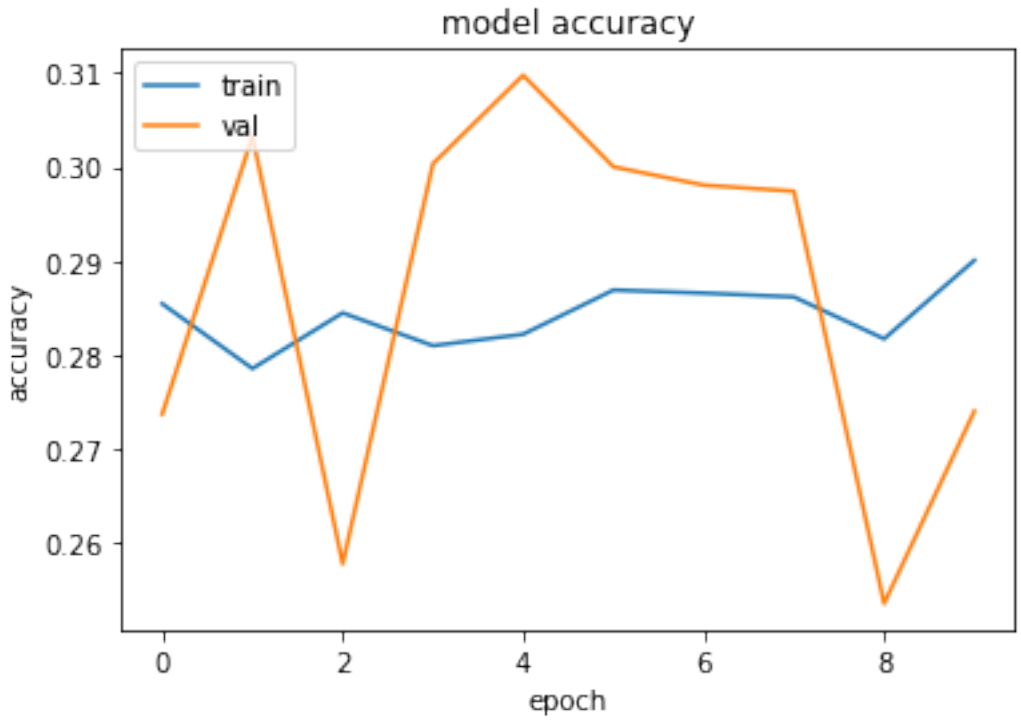
```
Epoch 00004: val_loss improved from 1.87762 to 1.86094, saving model to
/home/ec2-user/SageMaker/weights.best.from_class_base1.hdf5
1543/1543 [==============================] - 206s 133ms/step - loss: 1.9106 -
acc: 0.2810 - val_loss: 1.8609 - val_acc: 0.3003
Epoch 5/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.8998 - acc:
0.2821Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 1.8758 - acc:
0.3097
Epoch 00005: val_loss did not improve from 1.86094
1543/1543 [==============================] - 204s 132ms/step - loss: 1.8997 -
acc: 0.2822 - val_loss: 1.8758 - val_acc: 0.3097
Epoch 6/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.9007 - acc:
0.2868Epoch 1/10
 385/1543 [======>…] - ETA: 2:01 - loss: 1.8650 - acc:
0.3000
Epoch 00006: val_loss did not improve from 1.86094
1543/1543 [==============================] - 204s 133ms/step - loss: 1.9007 -
acc: 0.2869 - val_loss: 1.8650 - val_acc: 0.3000
Epoch 7/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.9068 - acc:
0.2866Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 1.8569 - acc:
0.2981
Epoch 00007: val_loss improved from 1.86094 to 1.85685, saving model to
/home/ec2-user/SageMaker/weights.best.from_class_base1.hdf5
1543/1543 [==============================] - 205s 133ms/step - loss: 1.9069 -
acc: 0.2866 - val_loss: 1.8569 - val_acc: 0.2981
Epoch 8/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.9052 - acc:
0.2864Epoch 1/10
 385/1543 [======>…] - ETA: 2:01 - loss: 1.8827 - acc:
0.2974
Epoch 00008: val_loss did not improve from 1.85685
1543/1543 [==============================] - 204s 132ms/step - loss: 1.9053 -
acc: 0.2862 - val_loss: 1.8827 - val_acc: 0.2974
Epoch 9/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.9139 - acc:
0.2817Epoch 1/10
 385/1543 [======>…] - ETA: 2:00 - loss: 1.9284 - acc:
0.2536
Epoch 00009: val_loss did not improve from 1.85685
1543/1543 [==============================] - 203s 131ms/step - loss: 1.9136 -
acc: 0.2817 - val_loss: 1.9284 - val_acc: 0.2536
Epoch 10/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.9093 - acc:
0.2902Epoch 1/10
```

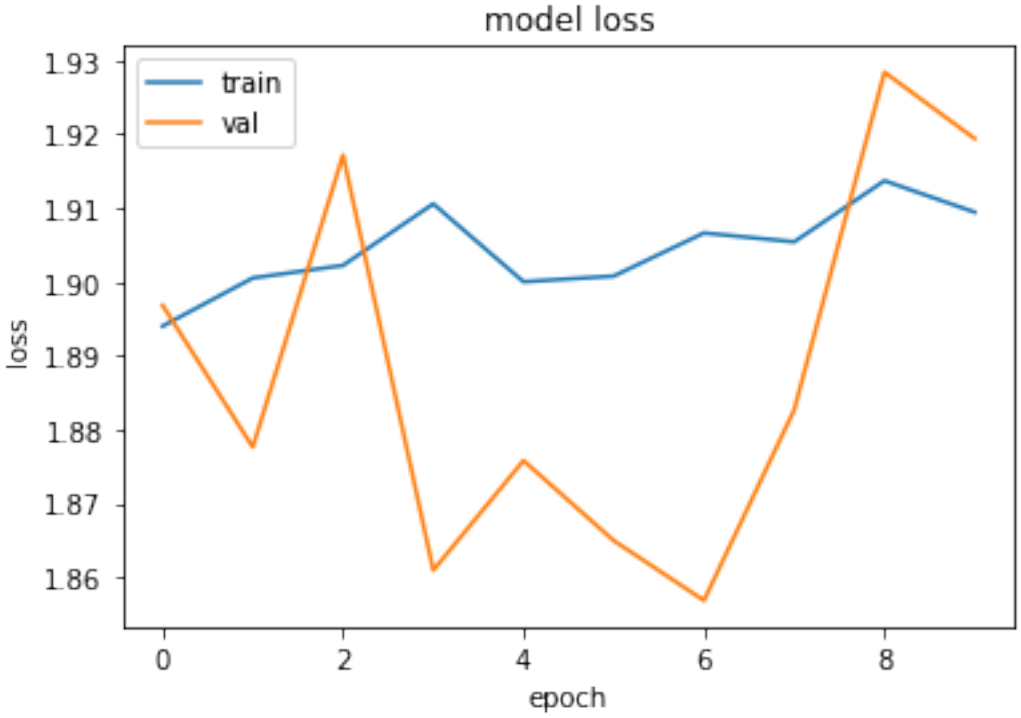
```
 385/1543 [======>…] - ETA: 2:00 - loss: 1.9194 - acc:
0.2740
Epoch 00010: val_loss did not improve from 1.85685
1543/1543 [==============================] - 203s 131ms/step - loss: 1.9094 -
acc: 0.2901 - val_loss: 1.9194 - val_acc: 0.2740
```

[643]:
```
print('For training set:')
report_accuracy(base_group_model2, train_generator)
print('------------------')
print('For validation set:')
report_accuracy(base_group_model2, valid_generator)
```

```
For training set:
Loss: 1.9128635660045505
Accuracy: 0.2823358
------------------
For validation set:
Loss: 1.9388576115968932
Accuracy: 0.2624757
```

[646]:
```
plt.plot(base_group_model2.history['acc'])
plt.plot(base_group_model2.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.plot(base_group_model2.history['loss'])
plt.plot(base_group_model2.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```


model accuracy
train
val
accuracy
epoch
0.31
0.30
0.29
0.28
0.27
0.26
0
2
4
6
8


model loss
train
val
loss
epoch
1.93
1.92
1.91
1.90
1.89
1.88
1.87
1.86
0
2
4
6
8

[648]:
```
target = image.group.unique()
target
```

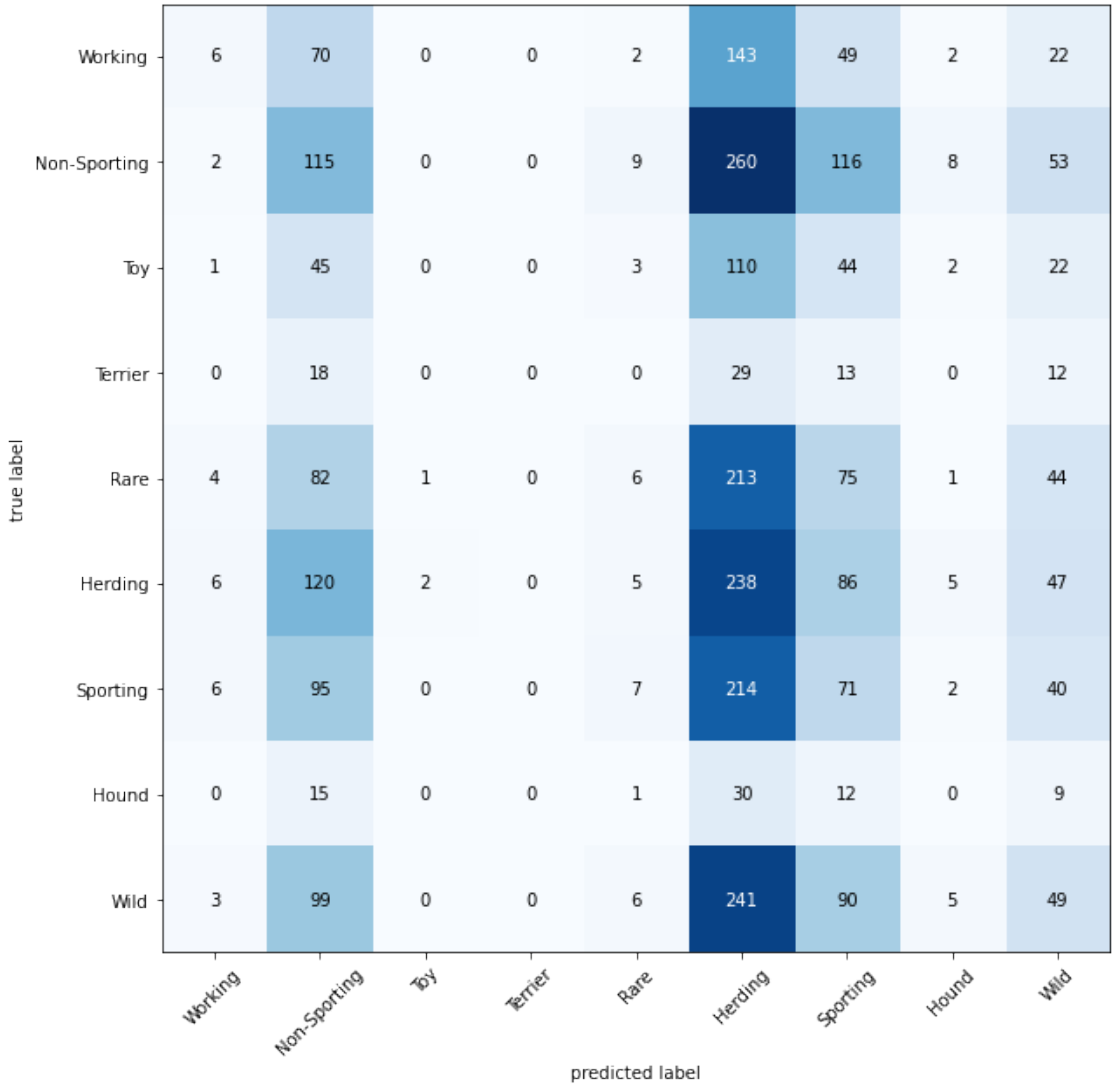
[648]:
```
array(['Working', 'Non-Sporting', 'Toy', 'Terrier', 'Rare', 'Herding',
       'Sporting', 'Hound', 'Wild'], dtype=object)
```

[649]:
```
Y_pred = model.predict_generator(valid_generator)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
cf = confusion_matrix(valid_generator.classes, y_pred)
print(cf)
print('Classification Report')
target_names = target
print(classification_report(valid_generator.classes, y_pred,␣
 ↪target_names=target_names))
```

```
Confusion Matrix
[[  6  70   0   0   2 143  49   2  22]
 [  2 115   0   0   9 260 116   8  53]
 [  1  45   0   0   3 110  44   2  22]
 [  0  18   0   0   0  29  13   0  12]
 [  4  82   1   0   6 213  75   1  44]
 [  6 120   2   0   5 238  86   5  47]
 [  6  95   0   0   7 214  71   2  40]
 [  0  15   0   0   1  30  12   0   9]
 [  3  99   0   0   6 241  90   5  49]]
Classification Report
              precision    recall  f1-score   support

     Working       0.21      0.02      0.04       294
Non-Sporting       0.17      0.20      0.19       563
         Toy       0.00      0.00      0.00       227
     Terrier       0.00      0.00      0.00        72
        Rare       0.15      0.01      0.03       426
     Herding       0.16      0.47      0.24       509
    Sporting       0.13      0.16      0.14       435
       Hound       0.00      0.00      0.00        67
        Wild       0.16      0.10      0.12       493

    accuracy                           0.16      3086
   macro avg       0.11      0.11      0.08      3086
weighted avg       0.14      0.16      0.12      3086


/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
```

```
  _warn_prf(average, modifier, msg_start, len(result))
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[650]:
```
plot_confusion_matrix(conf_mat=cf,
                      class_names=target,
                      figsize=(10,10));
```



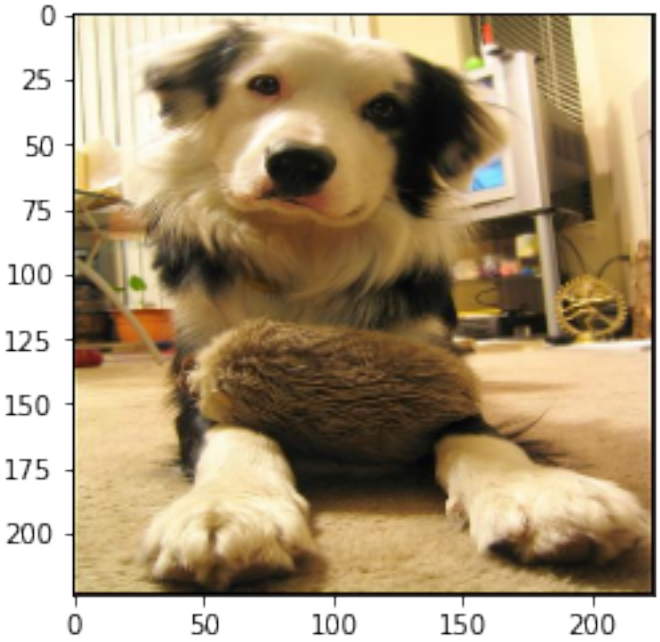
11 Investigate how the model misclassified labels

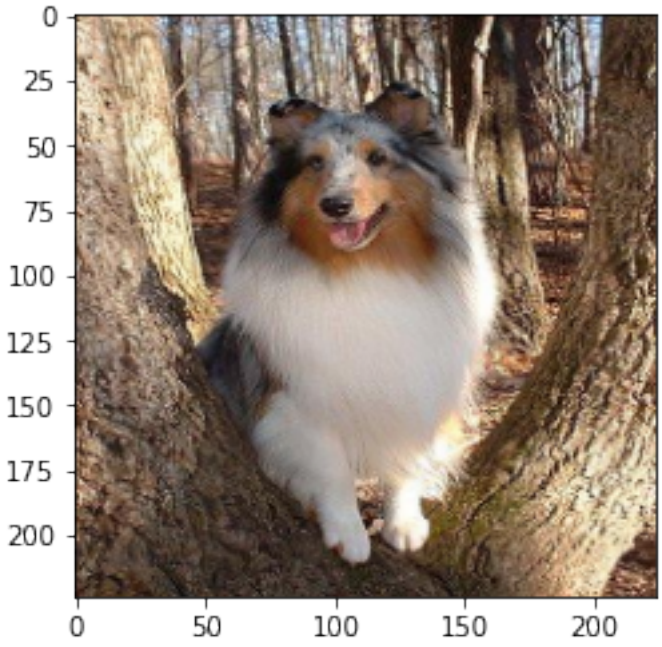
The Herding group has been classified the most, while the Toy group has the lowest.

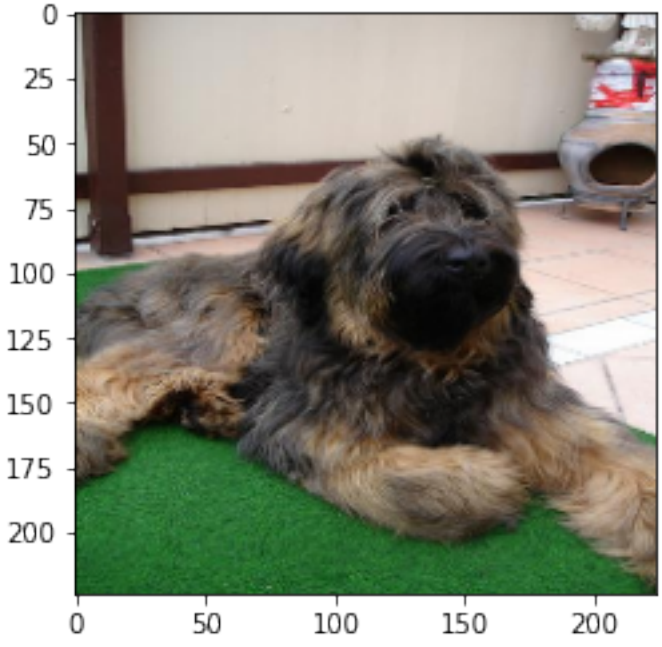
[651]:
```
img_path = '/home/ec2-user/SageMaker/Images/'
```

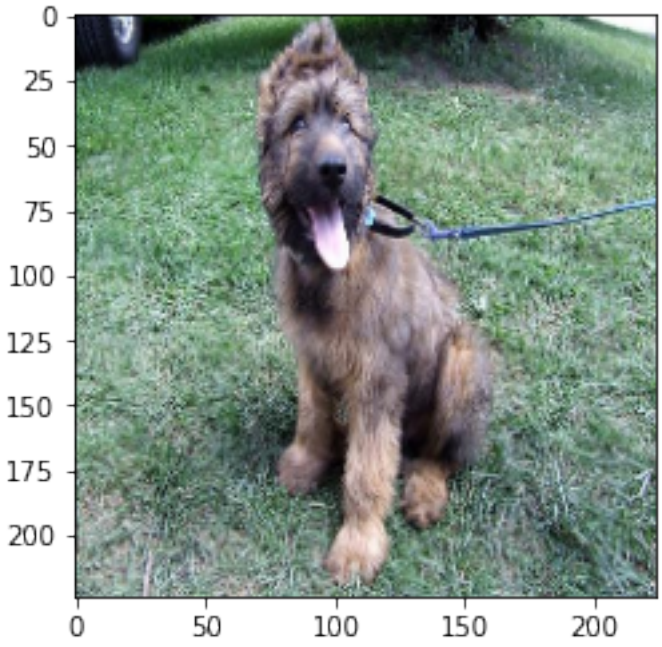
[696]:
```
herd = image.loc[(image['group'] == 'Herding')]
non_sport = image.loc[(image['group'] == 'Non-Sporting')]
toy = image.loc[(image['group'] == 'Toy')]
```

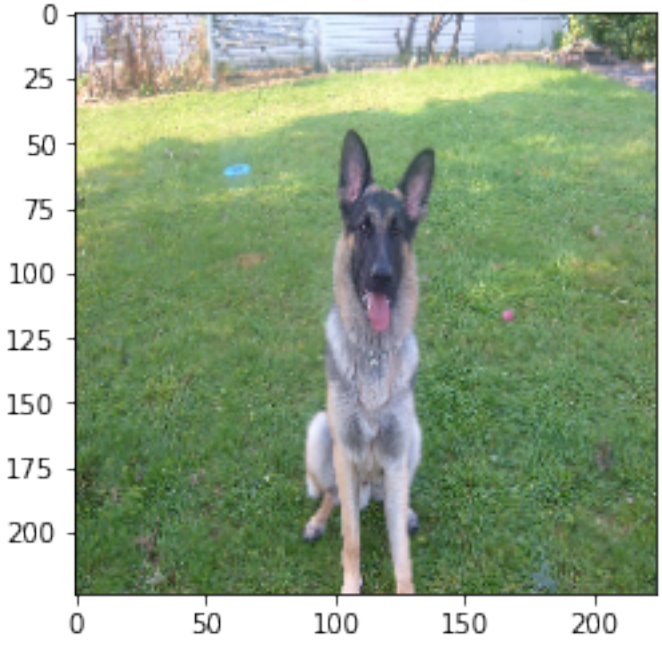
[693]:
```
for i in herd['image_path'][::100]:
    test = cv2.imread(img_path + i)
    test = cv2.resize(test, (224, 224))
    test = cv2.cvtColor(test, cv2.COLOR_BGR2RGB)
    imgplot = plt.imshow(test)
    plt.show(imgplot)
```

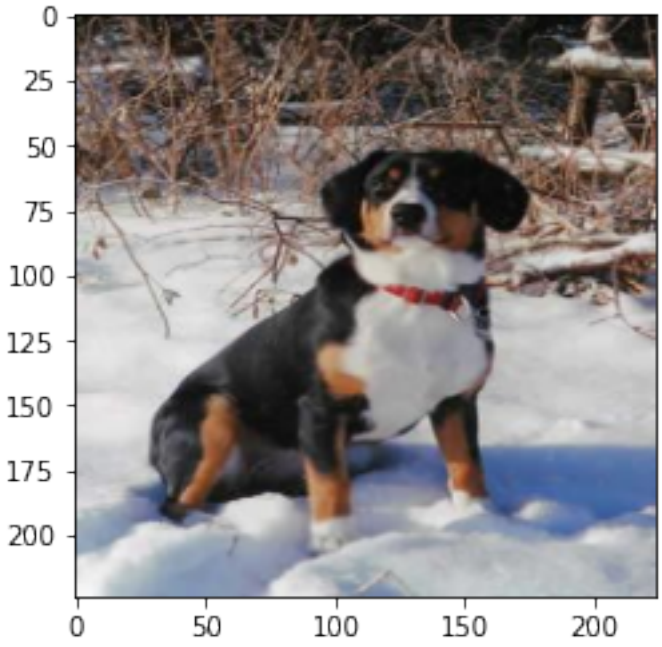


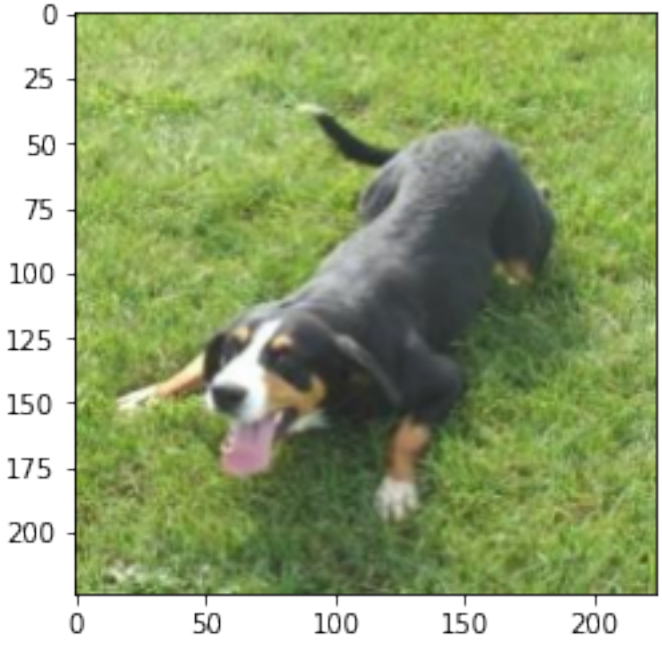

0
25
50
75
100
125
150
175
200
0
50
100
150
200

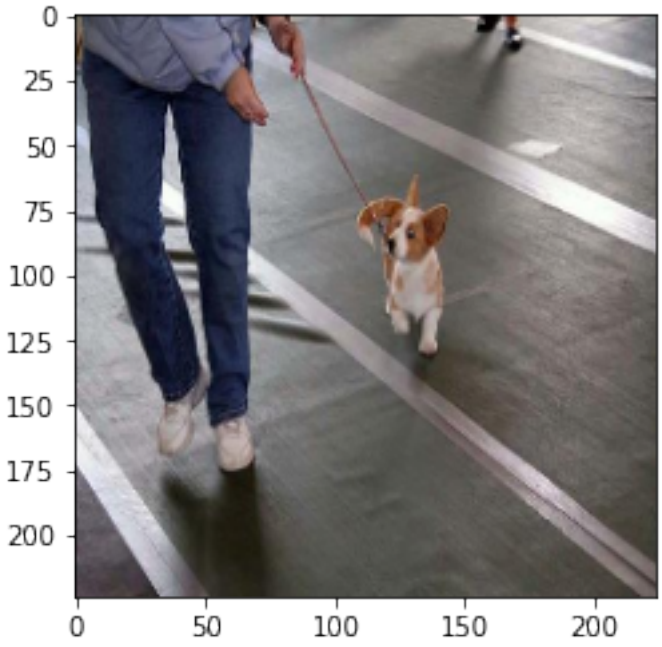

0
25
50
75
100
125
150
175
200
0
50
100
150
200

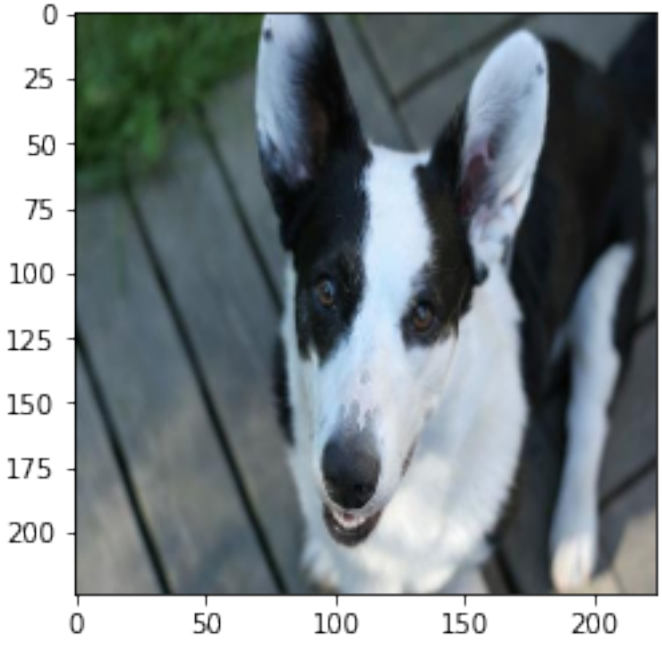

0
25
50
75
100
125
150
175
200
0
50
100
150
200

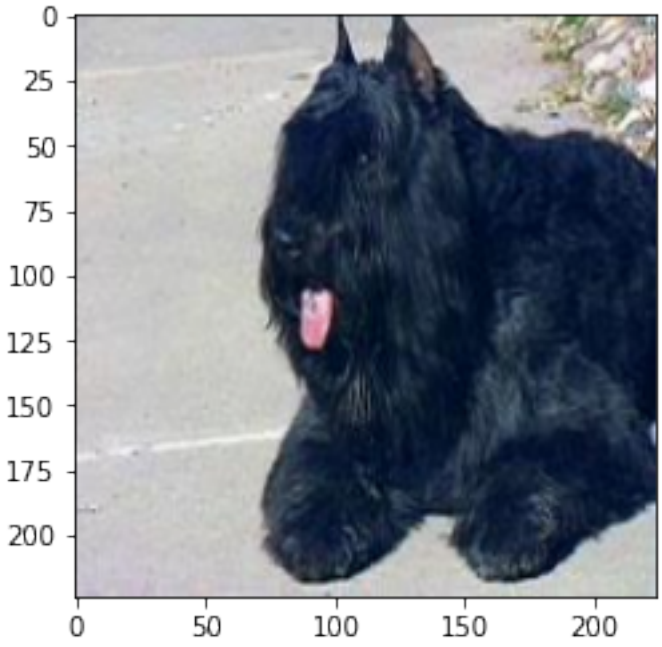

0
25
50
75
100
125
150
175
200
0
50
100
150
200

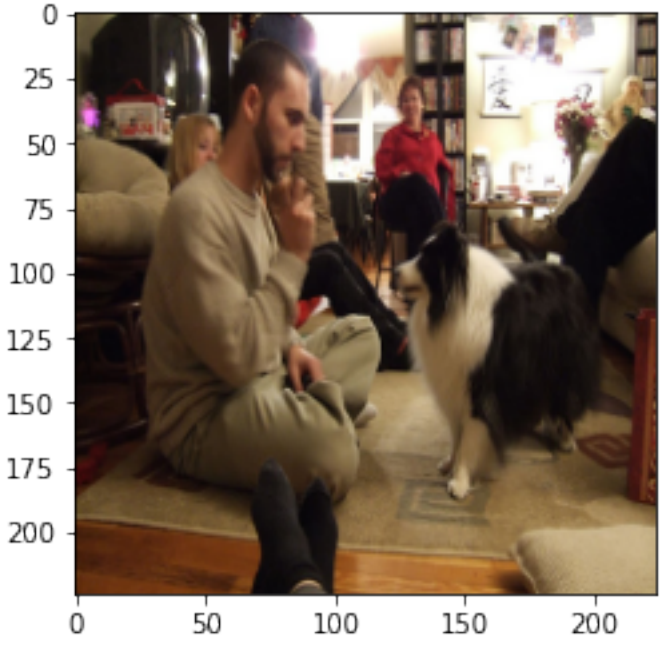

0
25
50
75
100
125
150
175
200
0
50
100
150
200

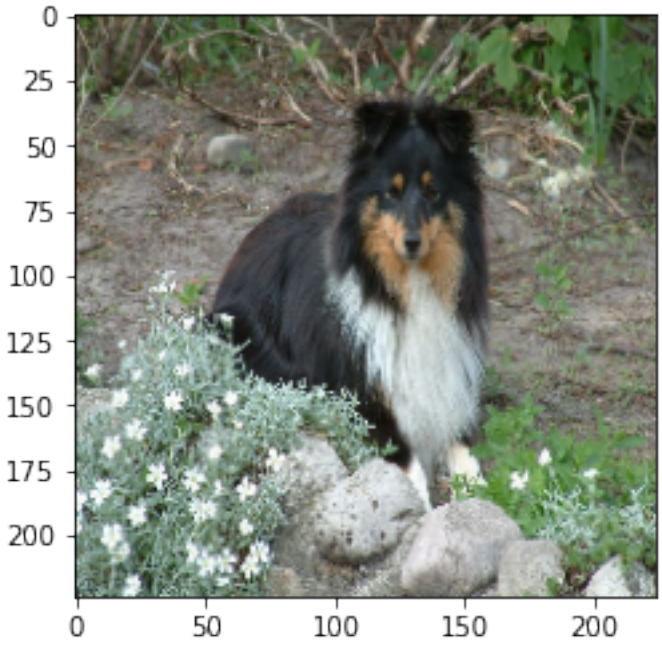

0
25
50
75
100
125
150
175
200
0
50
100
150
200

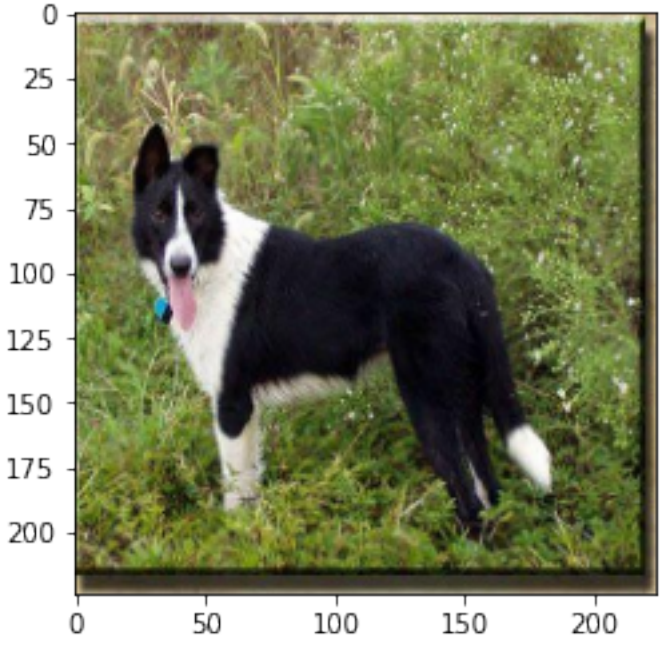

0
25
50
75
100
125
150
175
200
0
50
100
150
200

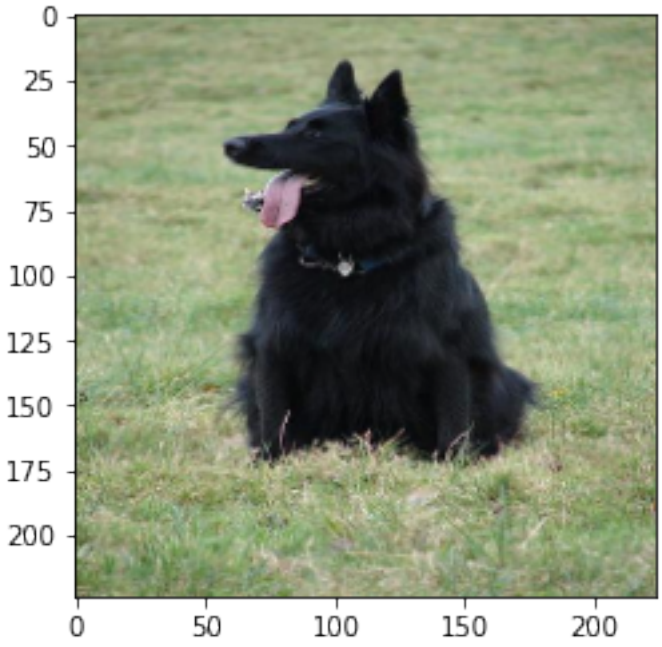

0
25
50
75
100
125
150
175
200
0
50
100
150
200

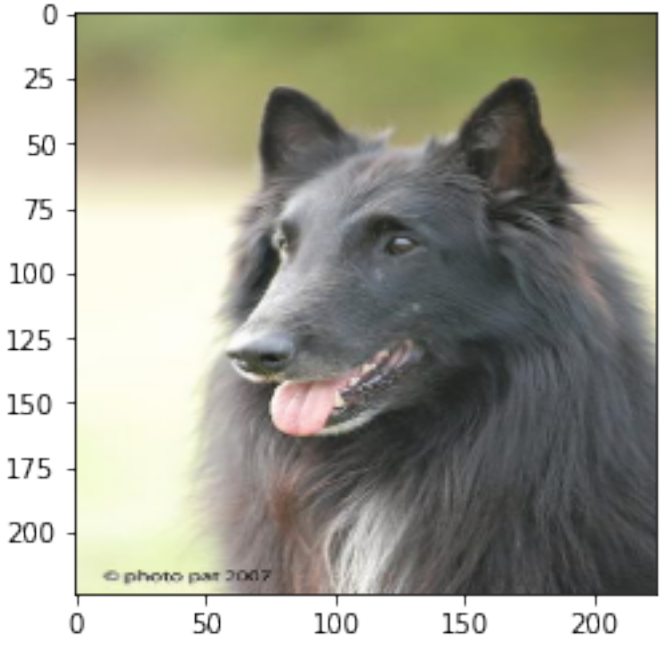

0
25
50
75
100
125
150
175
200
0
50
100
150
200

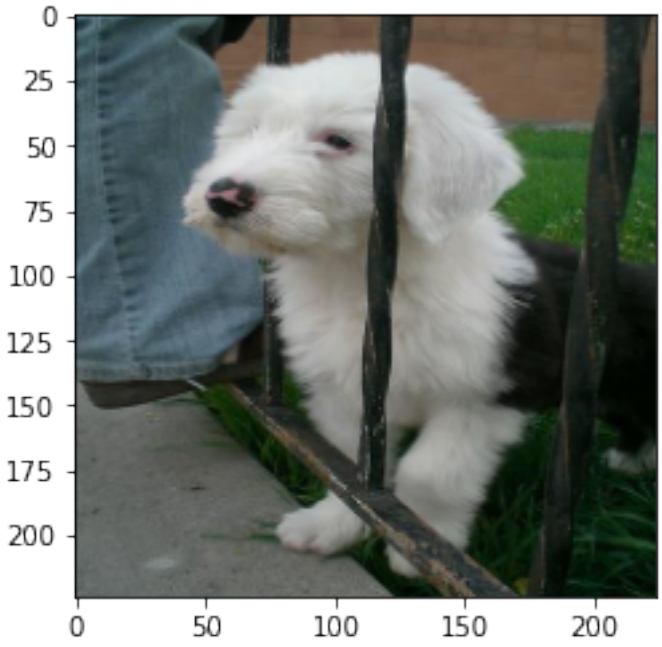

0
25
50
75
100
125
150
175
200
0
50
100
150
200

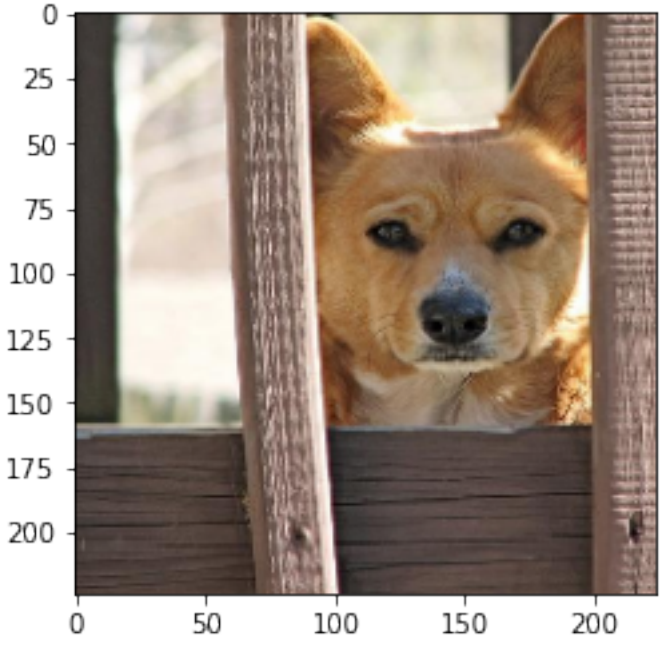

0
25
50
75
100
125
150
175
200
0
50
100
150
200

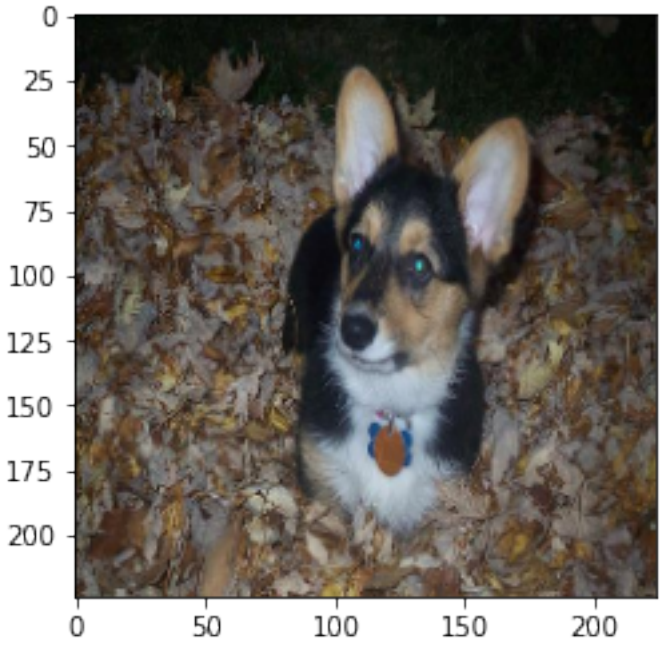

0
25
50
75
100
125
150
175
200
0
50
100
150
200

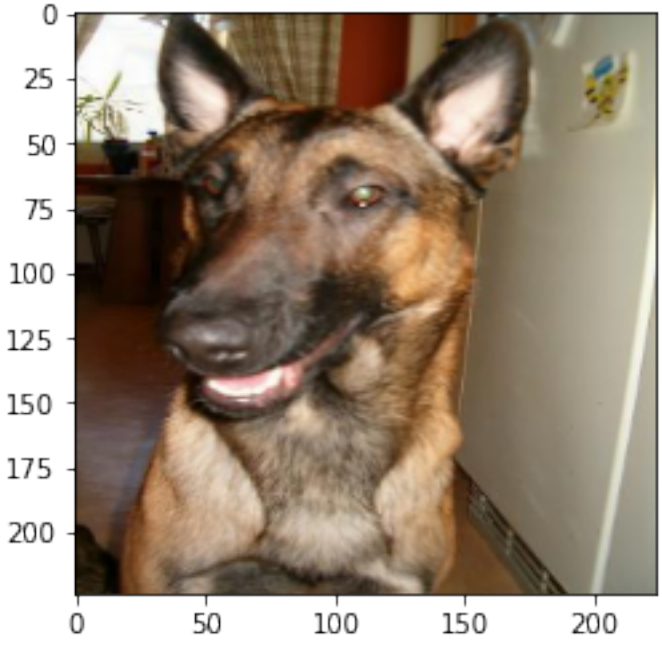

0
25
50
75
100
125
150
175
200
0
50
100
150
200

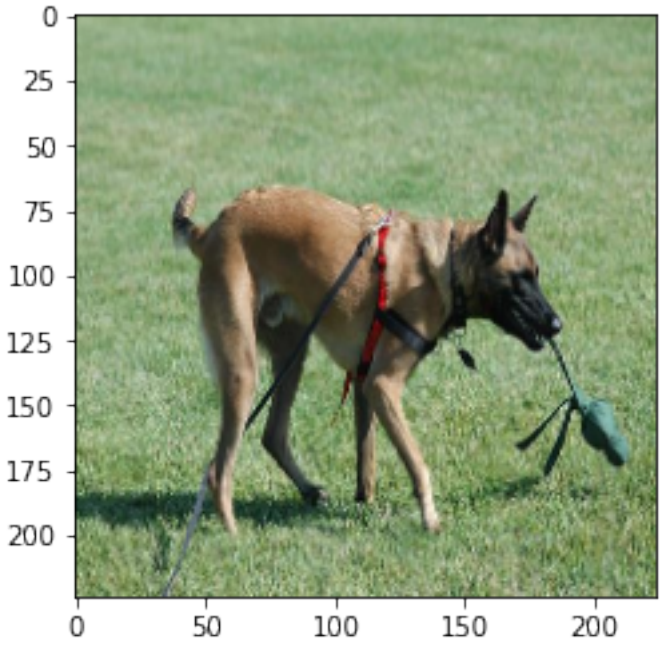

0
25
50
75
100
125
150
175
200
© photo par 2007
0
50
100
150
200


0
25
50
75
100
125
150
175
200
0
50
100
150
200

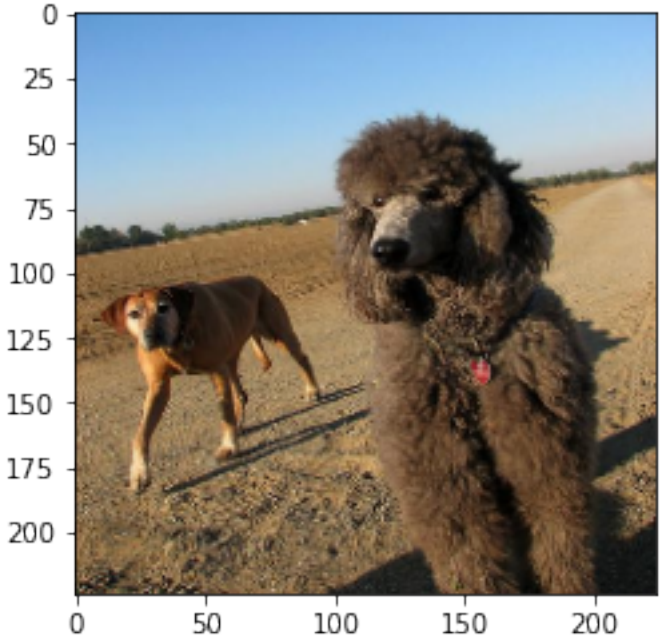

0
25
50
75
100
125
150
175
200
0
50
100
150
200

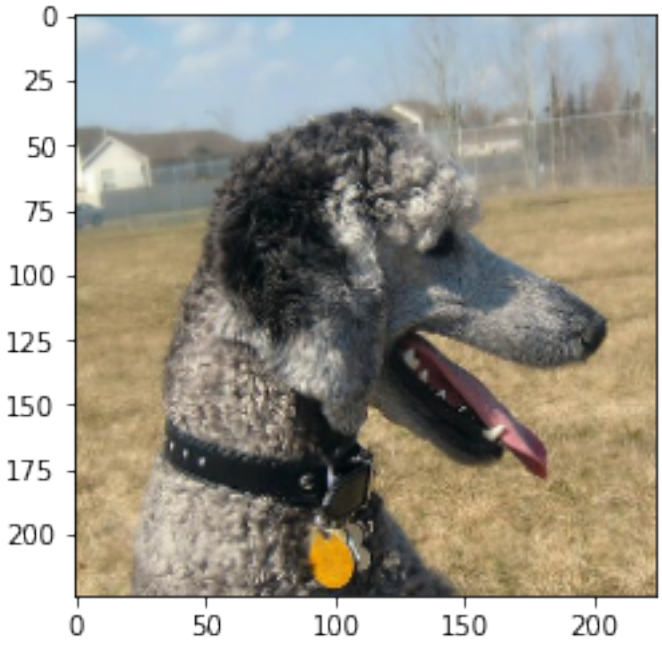

0
25
50
75
100
125
150
175
200
0
50
100
150
200

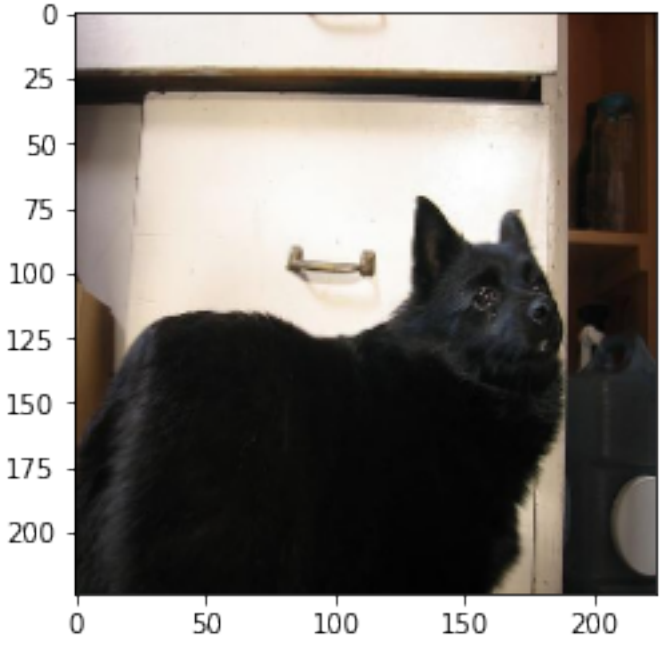

0
25
50
75
100
125
150
175
200
0
50
100
150
200

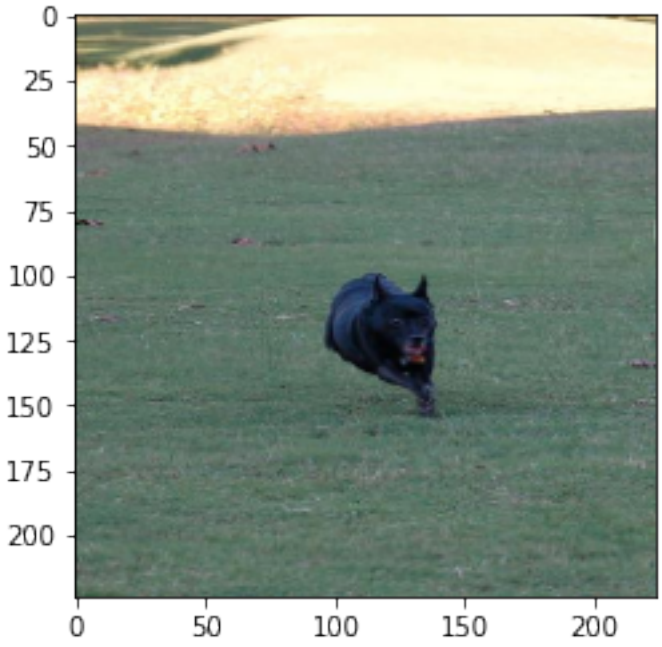


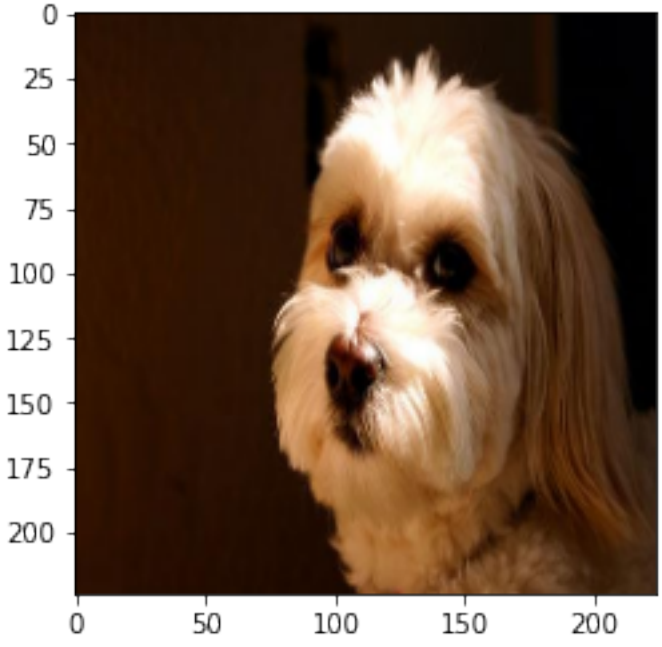
[695]:
```
for i in non_sport['image_path'][::100]:
    test = cv2.imread(img_path + i)
    test = cv2.resize(test, (224, 224))
    test = cv2.cvtColor(test, cv2.COLOR_BGR2RGB)
    imgplot = plt.imshow(test)
    plt.show(imgplot)
```

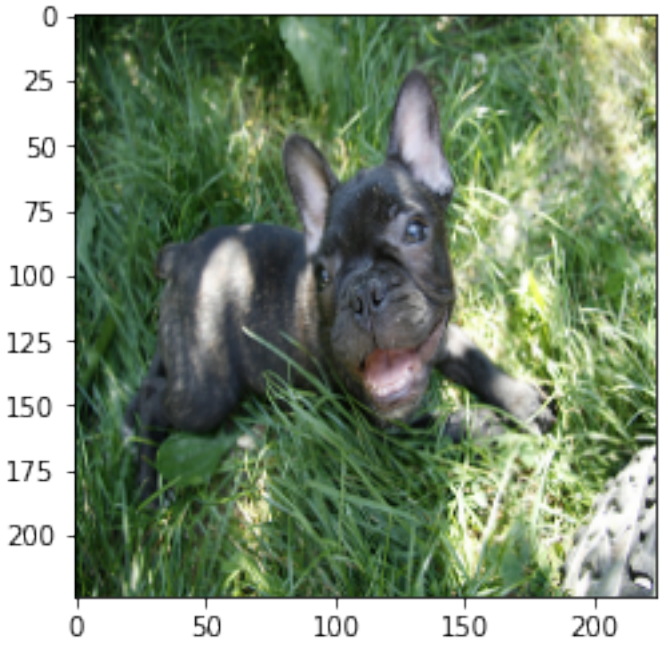


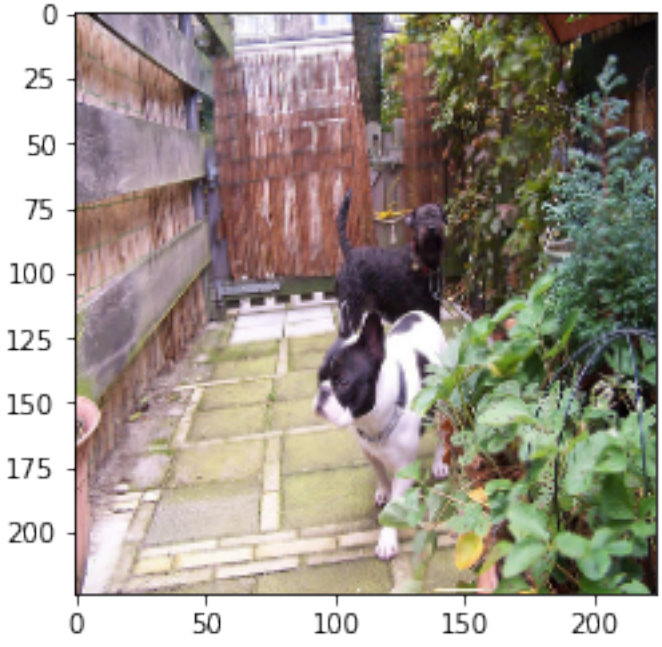

0
25
50
75
100
125
150
175
200
0
50
100
150
200

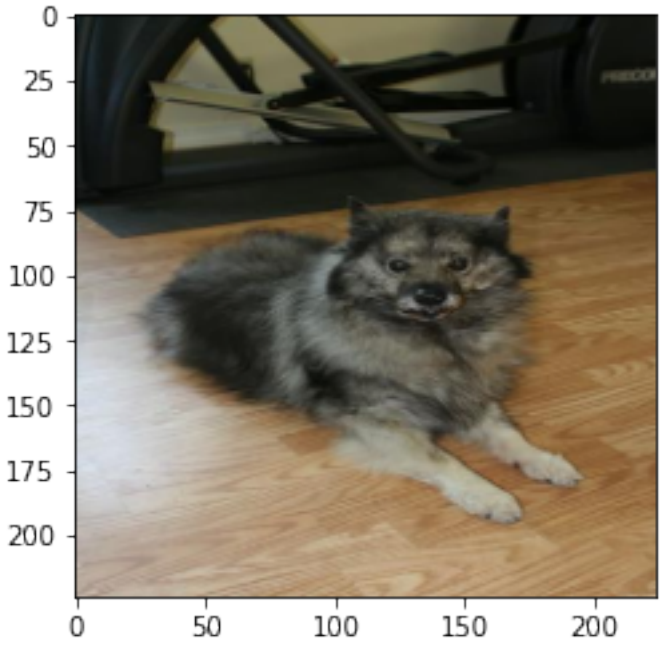

0
25
50
75
100
125
150
175
200
0
50
100
150
200

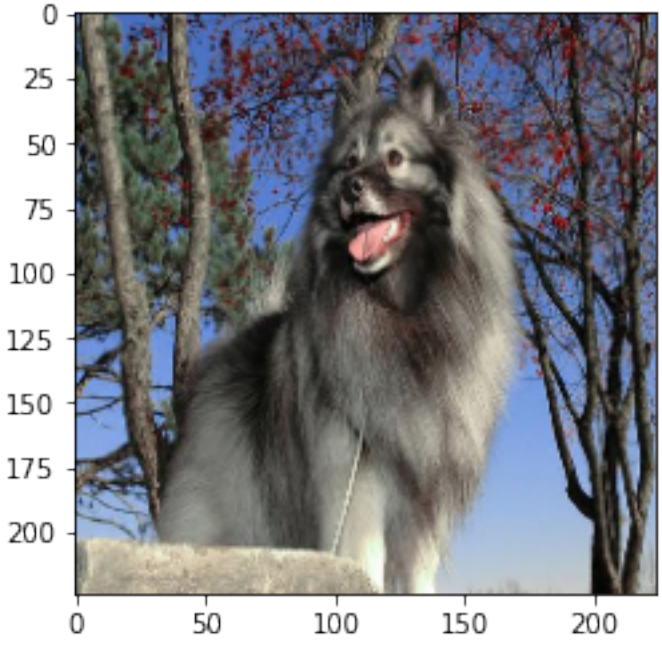

0
25
50
75
100
125
150
175
200
0
50
100
150
200

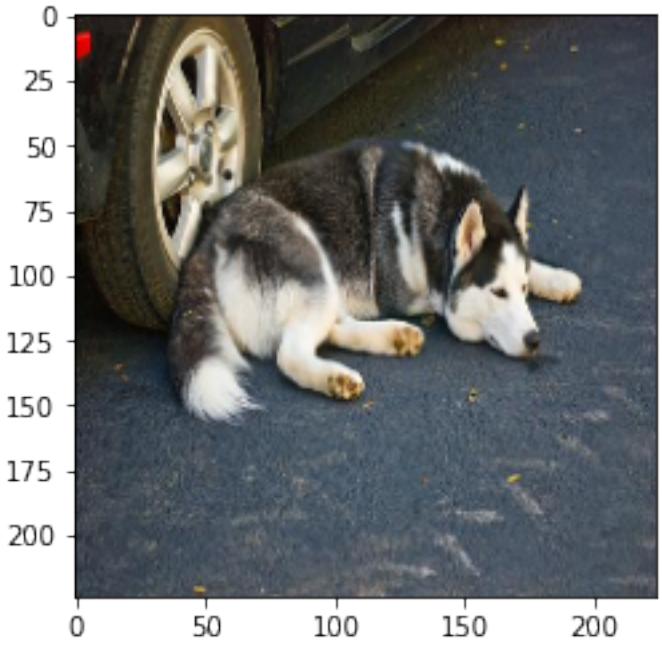

0
25
50
75
100
125
150
175
200
0
50
100
150
200

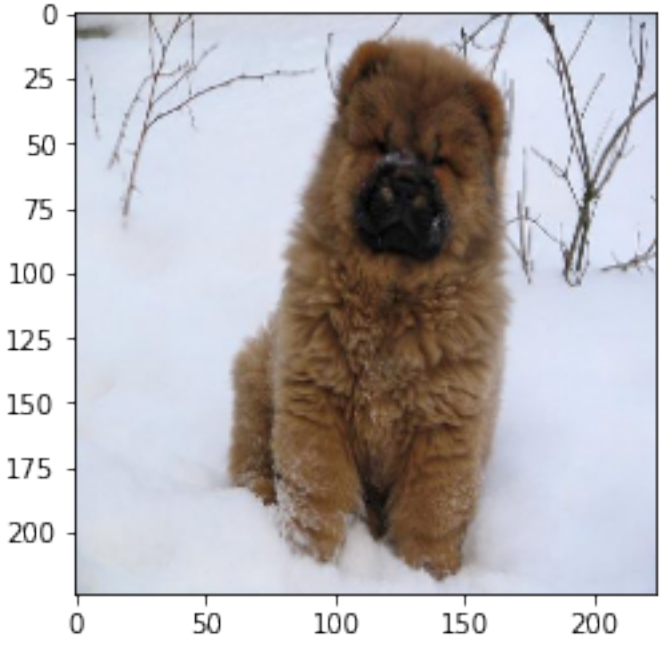

0
25
50
75
100
125
150
175
200
0
50
100
150
200

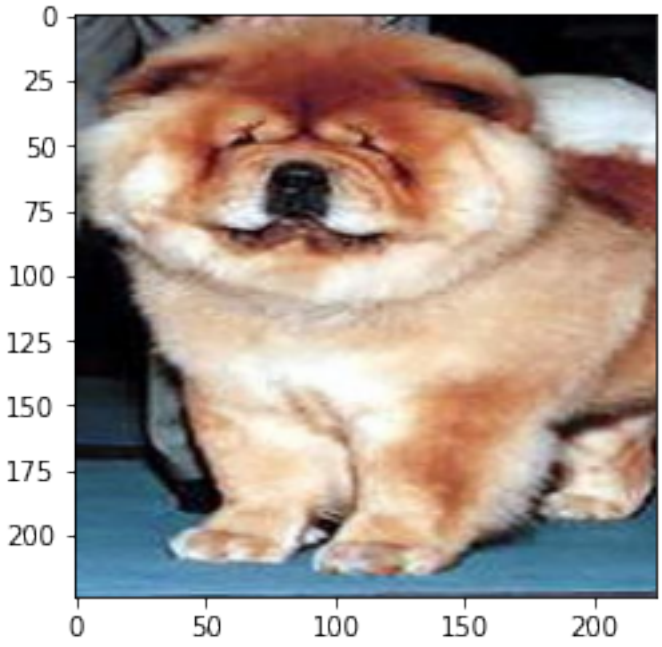

0
25
50
75
100
125
150
175
200
0
50
100
150
200

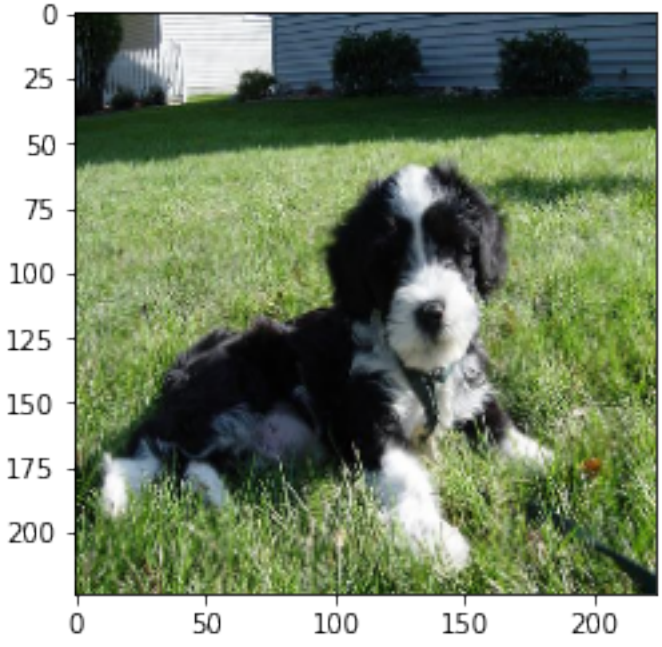

0
25
50
75
100
125
150
175
200
0
50
100
150
200

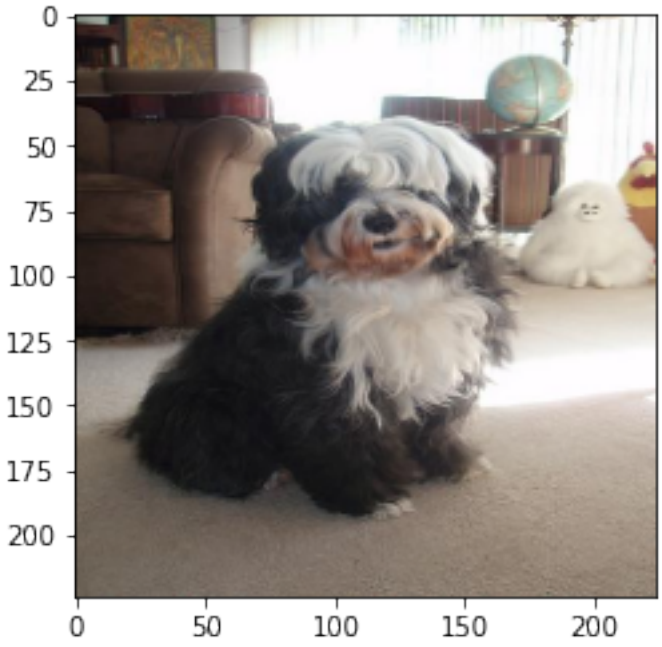

0
25
50
75
100
125
150
175
200
0
50
100
150
200

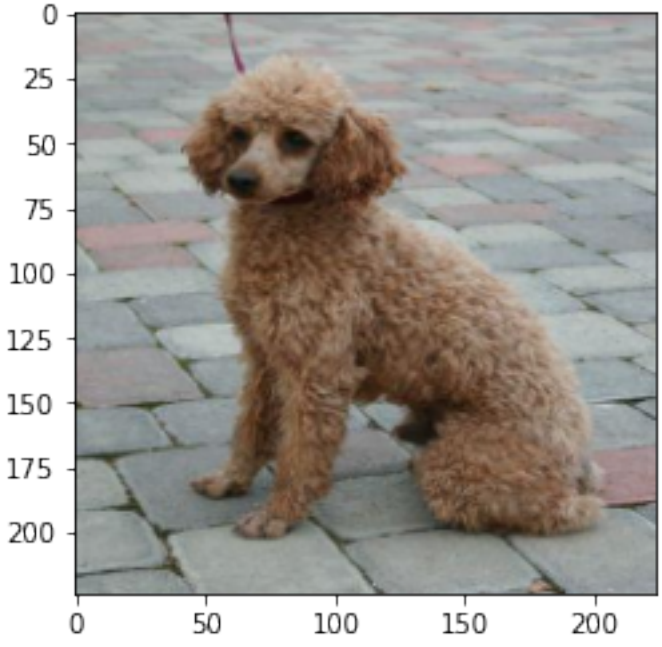

0
25
50
75
100
125
150
175
200
0
50
100
150
200

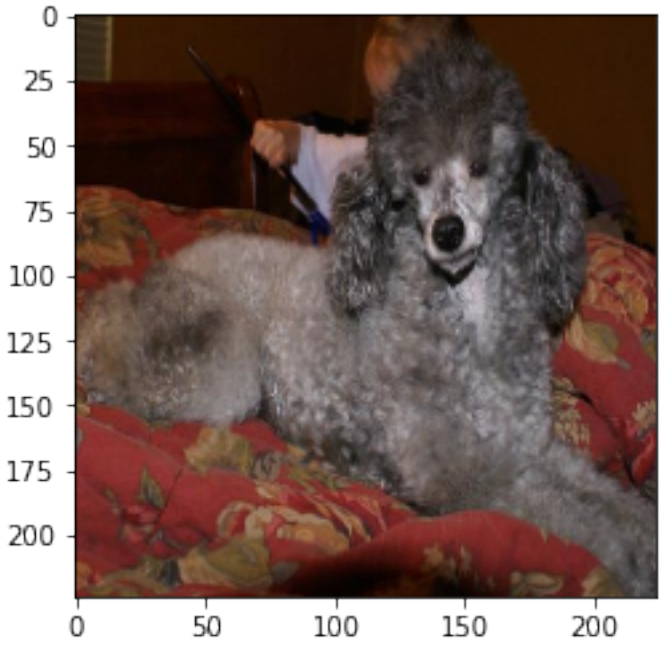

0
25
50
75
100
125
150
175
200
0
50
100
150
200


0
25
50
75
100
125
150
175
200
0
50
100
150
200

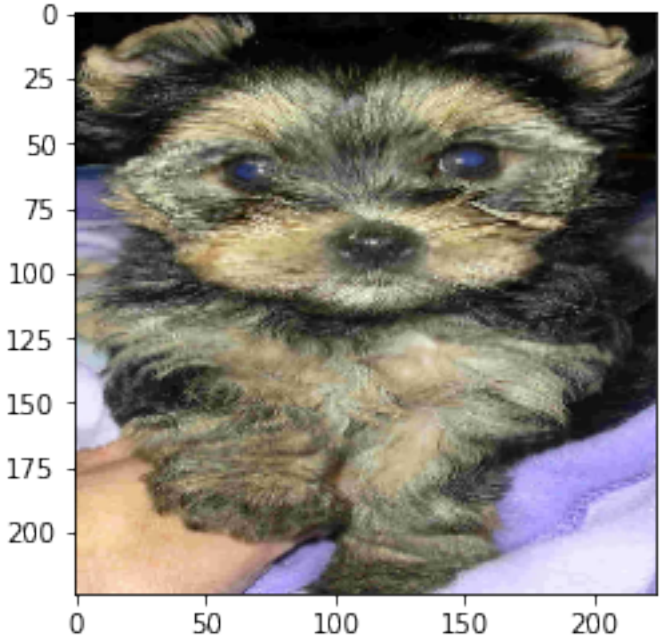

0
25
50
75
100
125
150
175
200
0
50
100
150
200

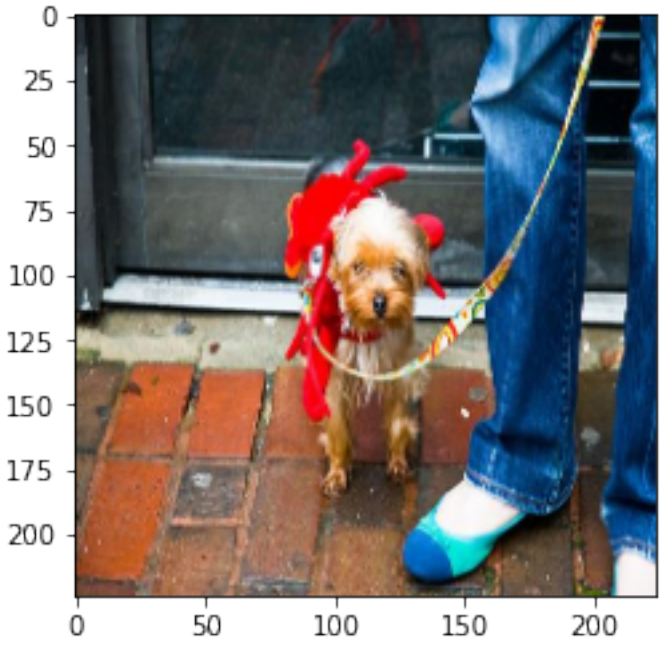

0
25
50
75
100
125
150
175
200
0
50
100
150
200

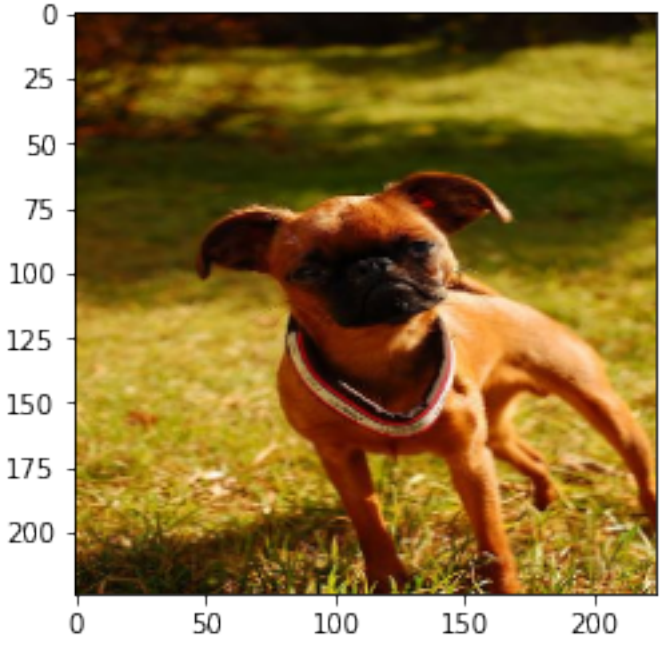

0
25
50
75
100
125
150
175
200
0
50
100
150
200

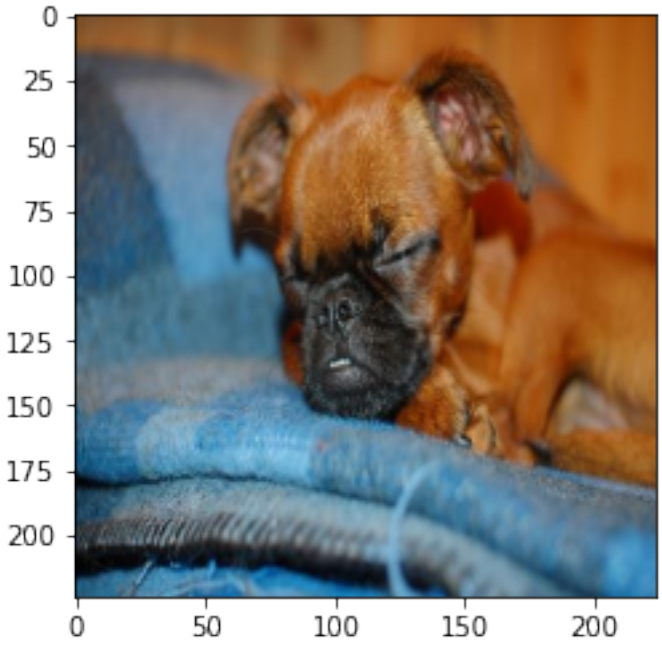


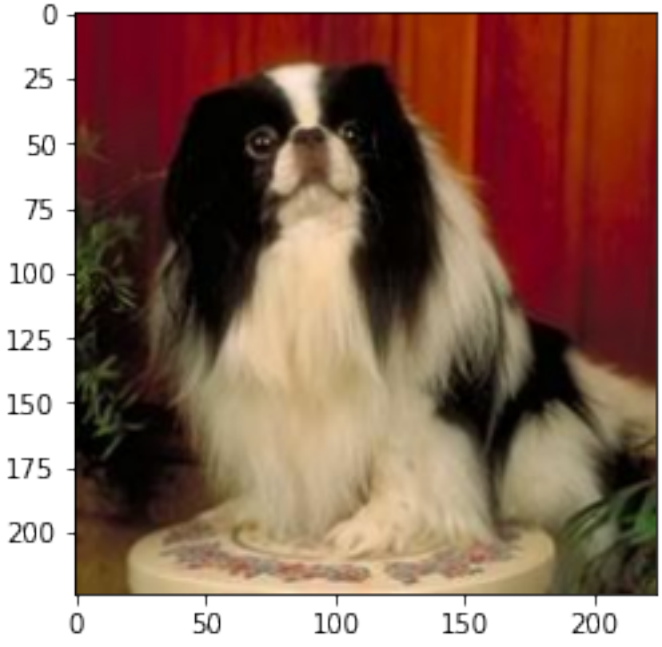
[697]:
```
for i in toy['image_path'][::100]:
    test = cv2.imread(img_path + i)
    test = cv2.resize(test, (224, 224))
    test = cv2.cvtColor(test, cv2.COLOR_BGR2RGB)
    imgplot = plt.imshow(test)
    plt.show(imgplot)
```

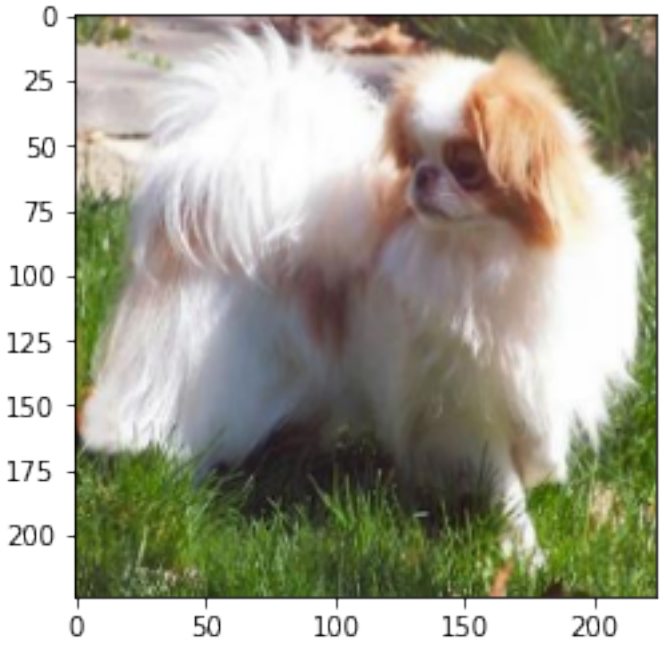


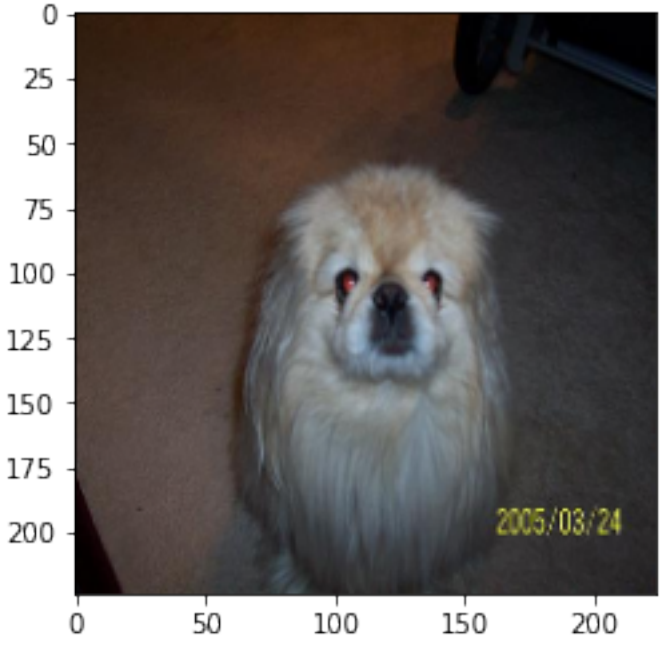

0
25
50
75
100
125
150
175
200
0
50
100
150
200

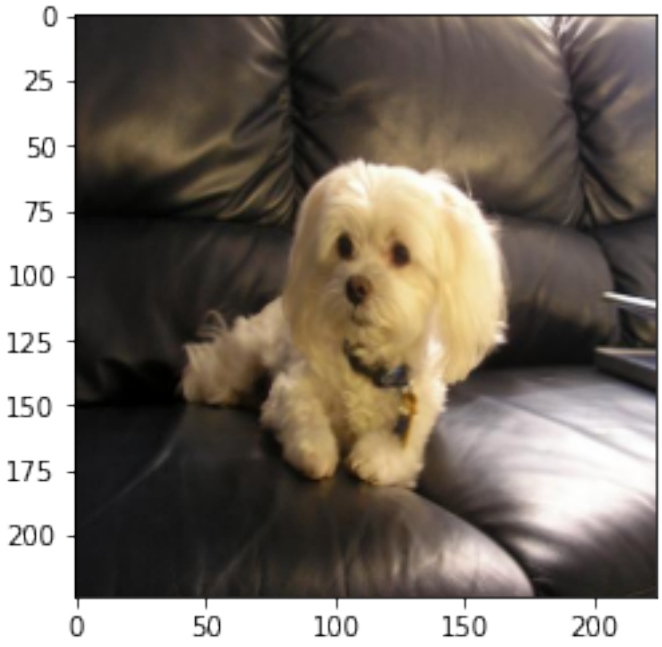

0
25
50
75
100
125
150
175
200
0
50
100
150
200

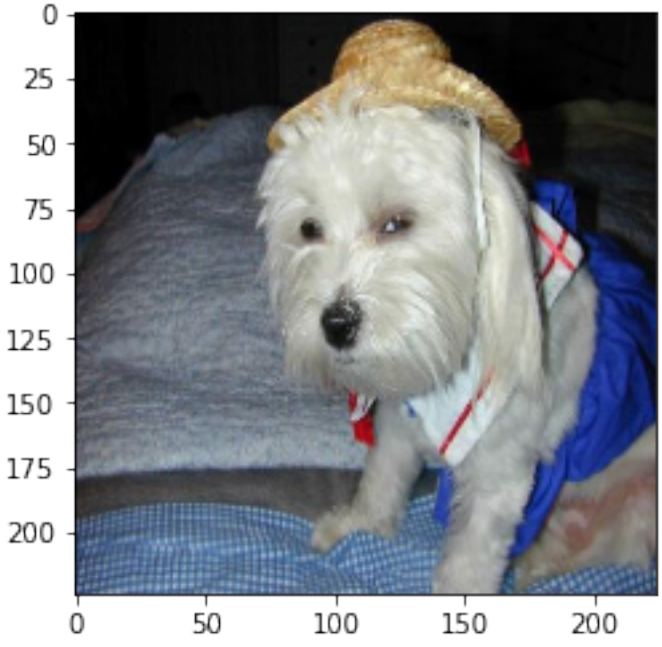

0
25
50
75
100
125
150
175
200
0
50
100
150
200

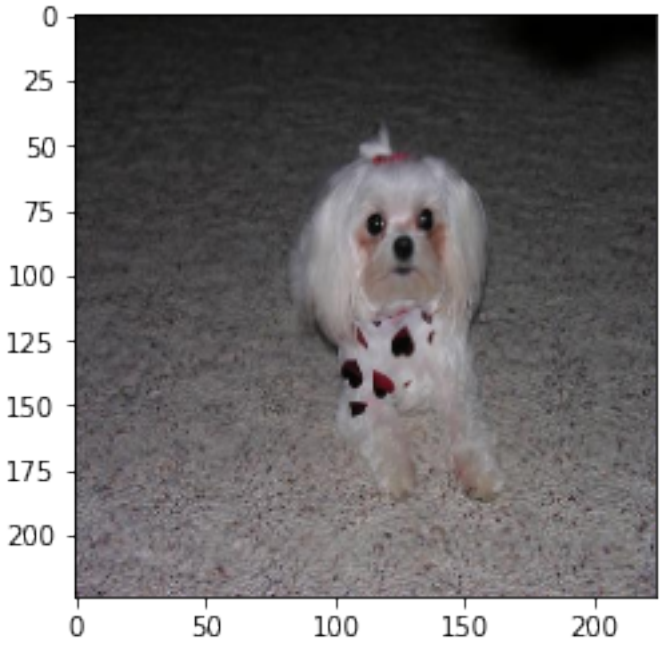

0
25
50
75
100
125
150
175
200
0
50
100
150
200

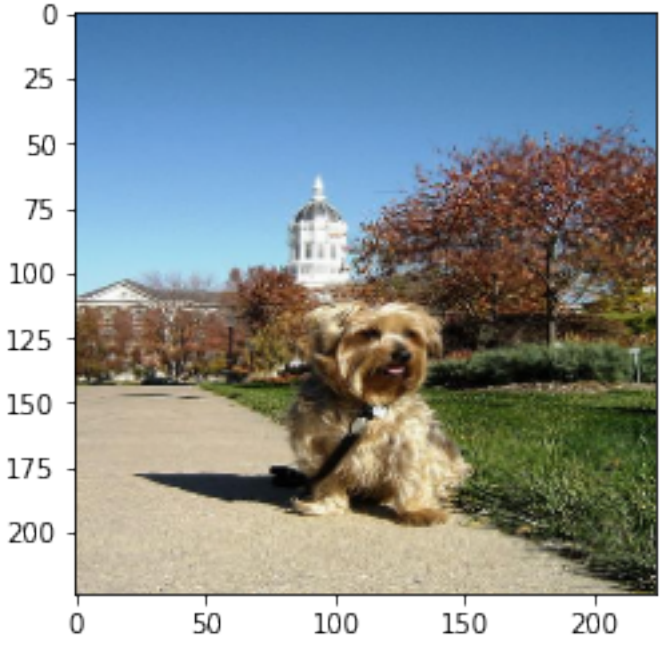

0
25
50
75
100
125
150
175
200
0
50
100
150
200

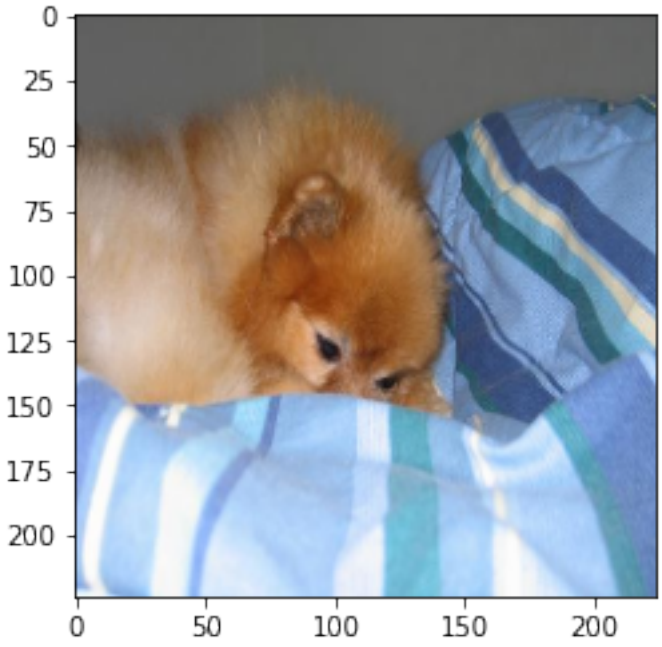

0
25
50
75
100
125
150
175
200
2005/03/24
0
50
100
150
200

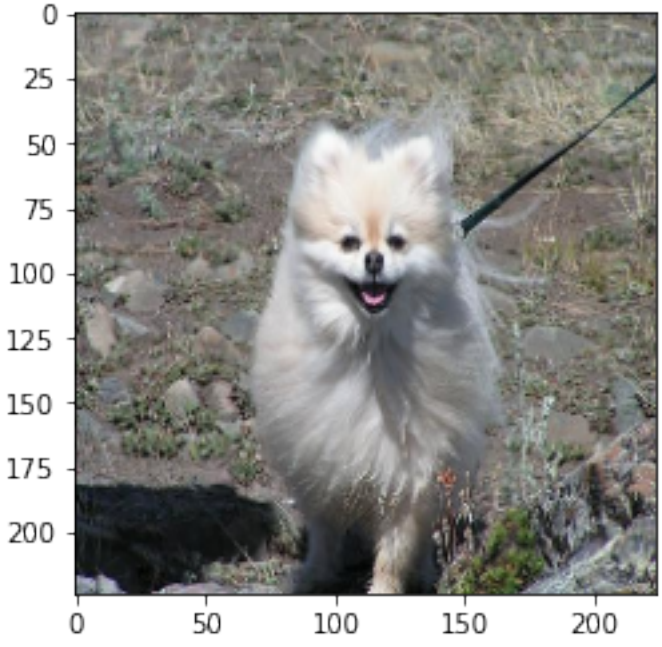

0
25
50
75
100
125
150
175
200
0
50
100
150
200

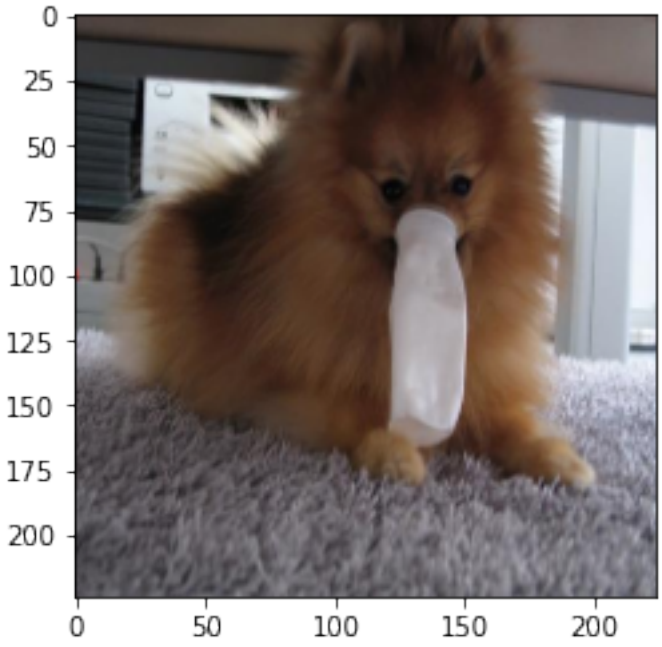

0
25
50
75
100
125
150
175
200
0
50
100
150
200

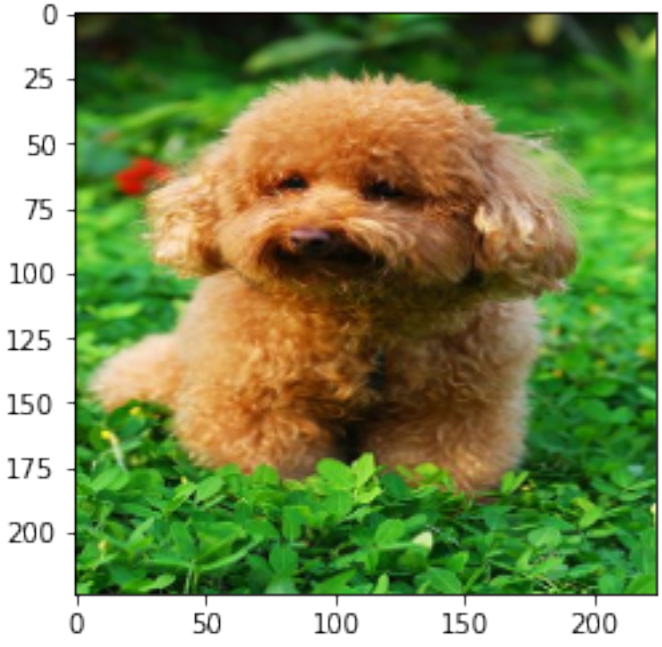

0
25
50
75
100
125
150
175
200
0
50
100
150
200

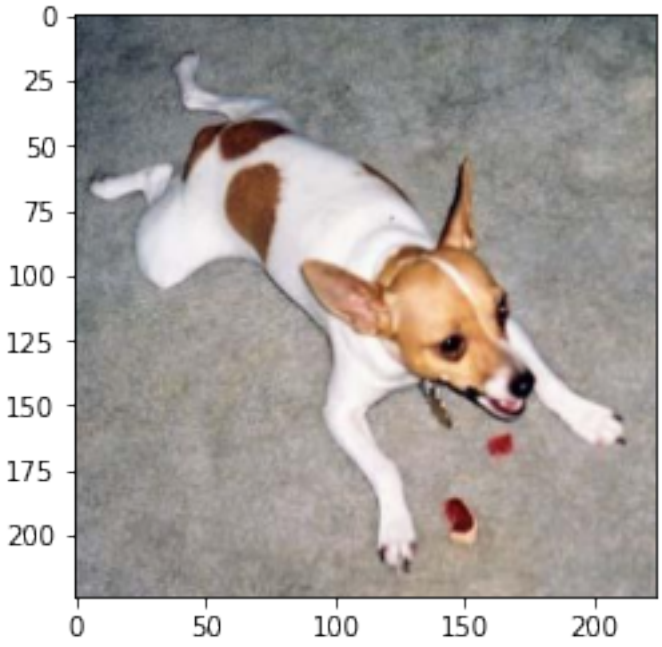

0
25
50
75
100
125
150
175
200
0
50
100
150
200

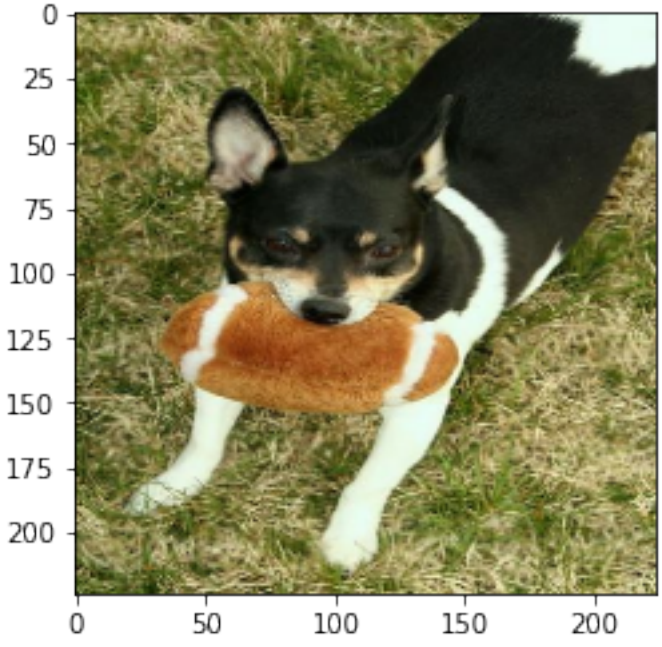

0
25
50
75
100
125
150
175
200
0
50
100
150
200

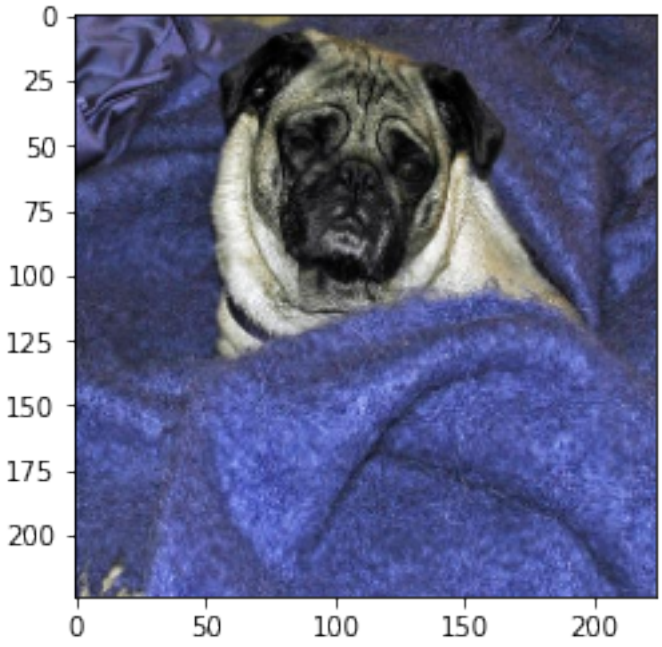

0
25
50
75
100
125
150
175
200
0
50
100
150
200

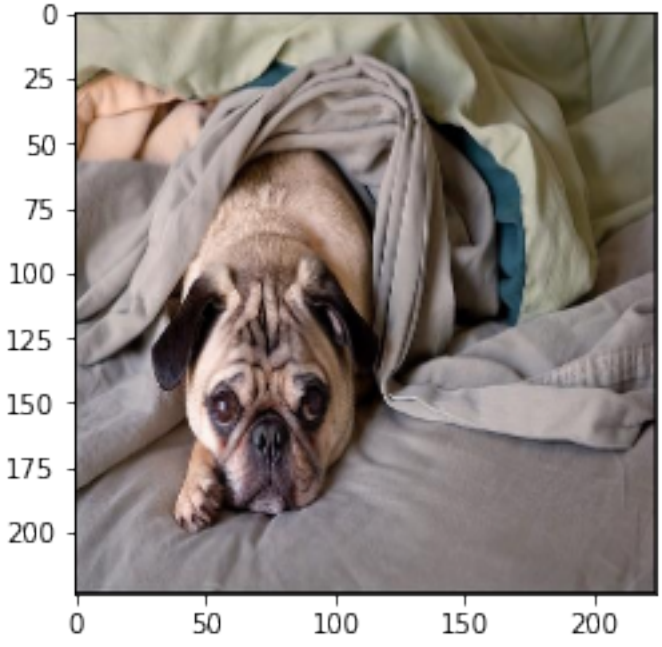

0
25
50
75
100
125
150
175
200
0
50
100
150
200

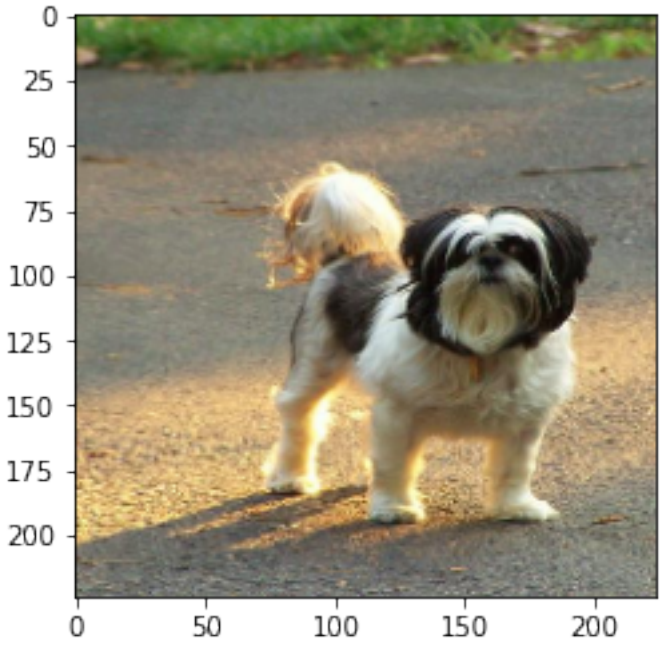

0
25
50
75
100
125
150
175
200
0
50
100
150
200

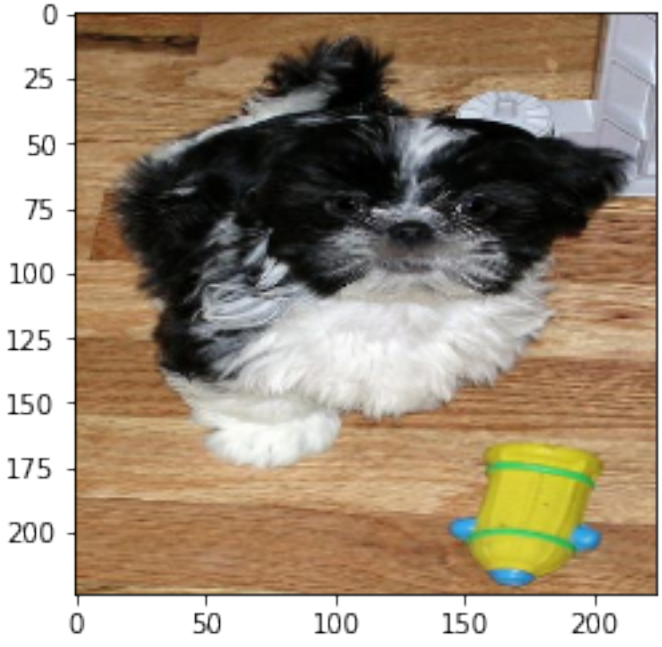

0
25
50
75
100
125
150
175
200
0
50
100
150
200

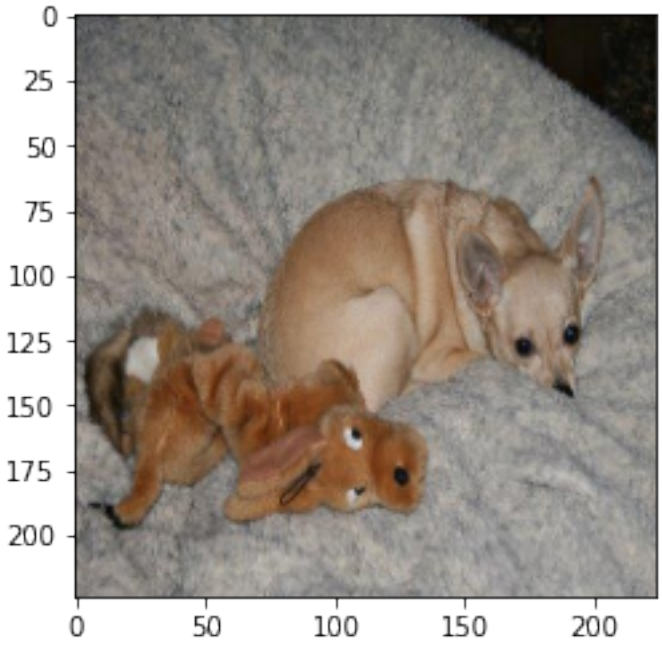

0
25
50
75
100
125
150
175
200
0
50
100
150
200

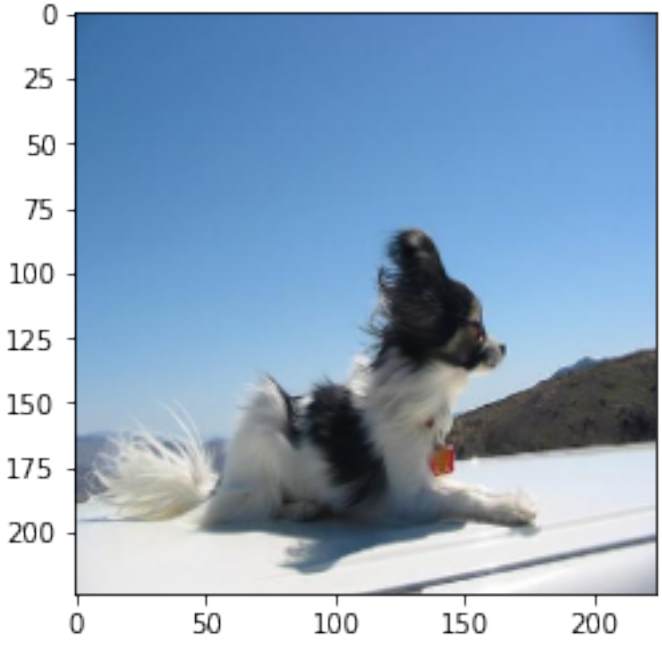

0
25
50
75
100
125
150
175
200
0
50
100
150
200

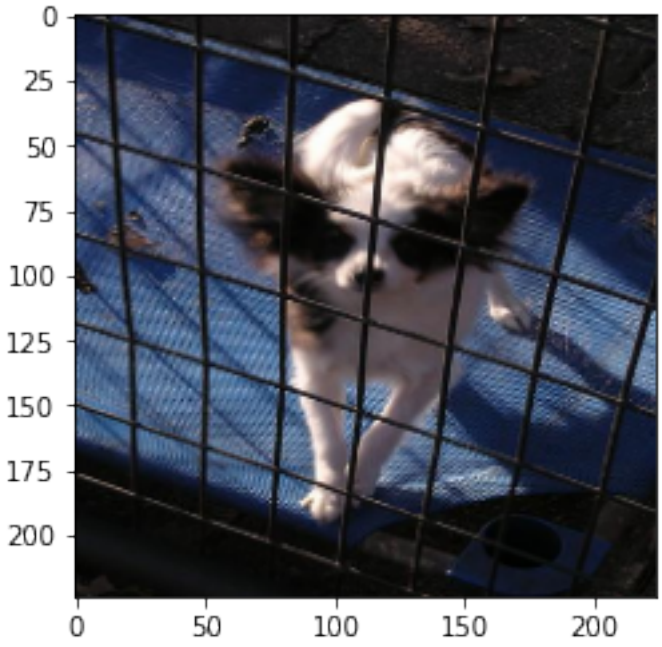

0
25
50
75
100
125
150
175
200
0
50
100
150
200

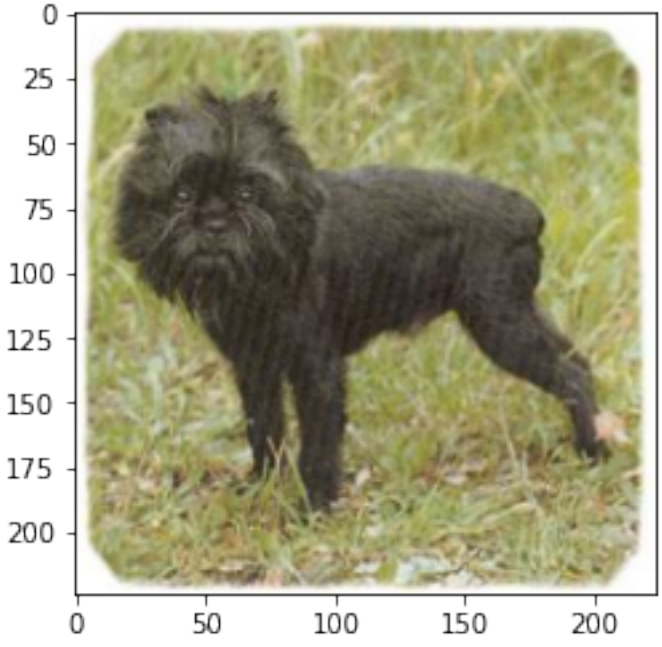

0
25
50
75
100
125
150
175
200
0
50
100
150
200

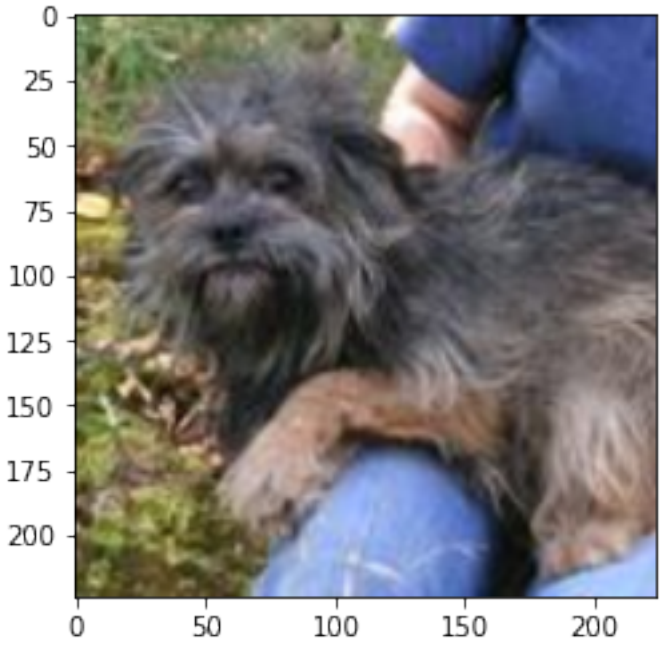

0
25
50
75
100
125
150
175
200
0
50
100
150
200

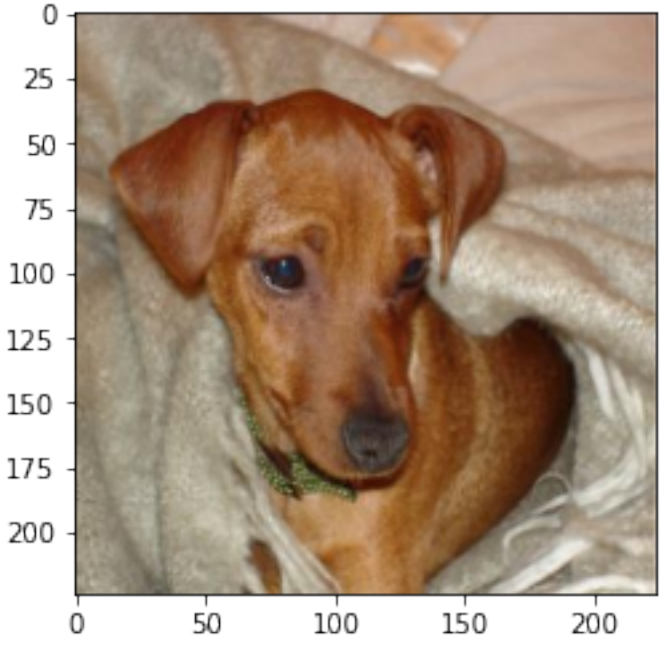

0
25
50
75
100
125
150
175
200
0
50
100
150
200

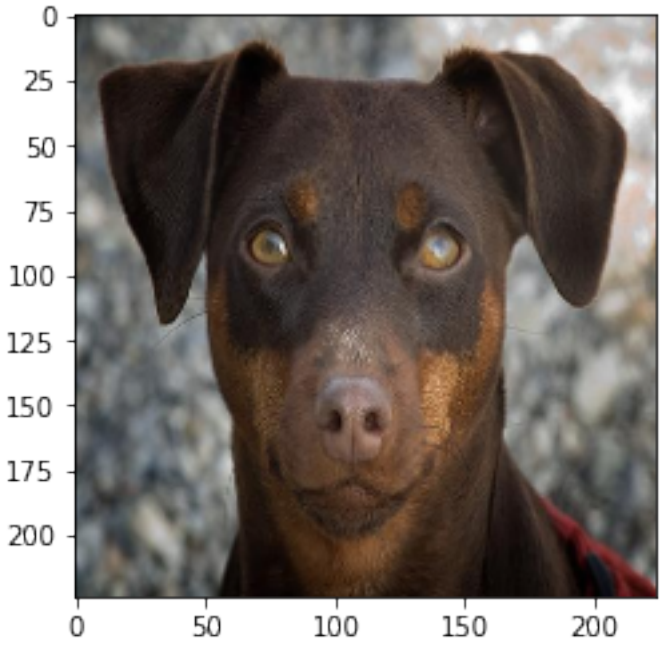

0
25
50
75
100
125
150
175
200
0
50
100
150
200

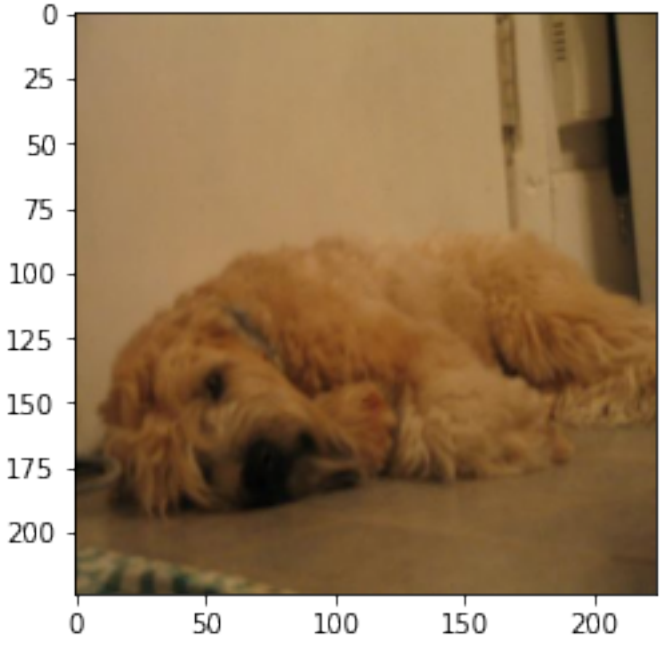

0
25
50
75
100
125
150
175
200
0
50
100
150
200

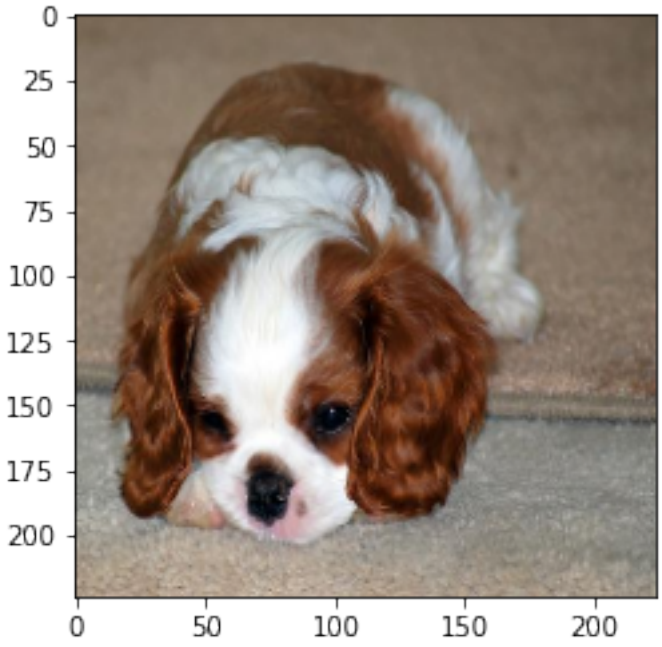

0
25
50
75
100
125
150
175
200
0
50
100
150
200

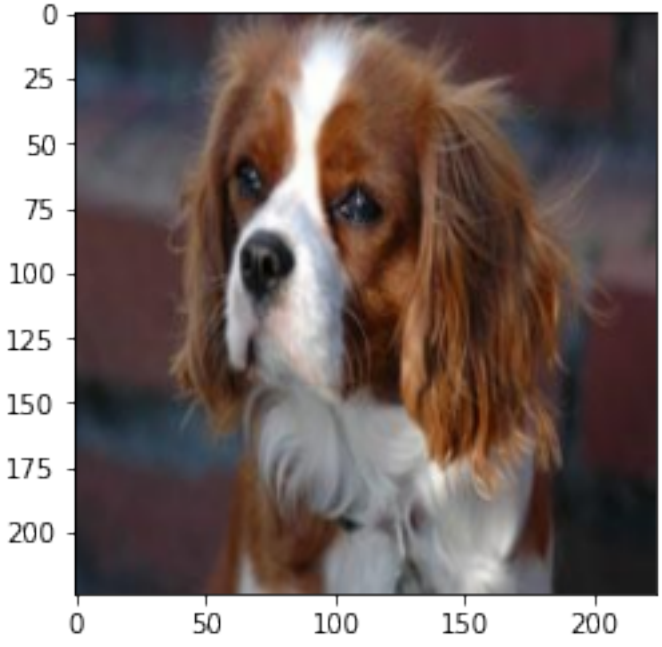

0
25
50
75
100
125
150
175
200
0
50
100
150
200


0
25
50
75
100
125
150
175
200
0
50
100
150
200


0
25
50
75
100
125
150
175
200
0
50
100
150
200


0
25
50
75
100
125
150
175
200
0
50
100
150
200


0
25
50
75
100
125
150
175
200
0
50
100
150
200


0
25
50
75
100
125
150
175
200
0
50
100
150
200

12 Transfer Learning - VGG16 Model

VGG stands for Very Deep Convolutional Networks, consists of 16 layers, developed in 2014. It is one of the famous models back then in the field of deep learning.

We want to see how well it will perform compared with our baseline model.

[714]:
```
#  loading transfer learning model
from tensorflow.keras import applications
```

[704]:
```
# create the model
vgg_model = applications.VGG16(include_top=False, weights='imagenet')
```

[728]:
```
batch_size = 8
jup_path = '/home/ec2-user/SageMaker/Images/'
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=jup_path,
    x_col="image_path",
    y_col="breed",
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode="categorical",
    subset='training',
    shuffle=True,
    seed=42
)
valid_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=jup_path,
    x_col="image_path",
    y_col="breed",
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode="categorical",
    subset='validation',
    shuffle=True,
    seed=42
)
test_generator = test_datagen.flow_from_dataframe(
    dataframe=test_df,
    directory=jup_path,
    x_col="image_path",
    target_size=(224, 224),
    batch_size=1,
    class_mode=None,
    shuffle=False,
)
```

```
Found 12347 validated image filenames belonging to 120 classes.
Found 3086 validated image filenames belonging to 120 classes.
Found 5145 validated image filenames.
```

[749]:
```
vgg_train = vgg_model.predict(train_generator, verbose=1)
vgg_val = vgg_model.predict(valid_generator, verbose=1)
```

```
1544/1544 [==============================] - 206s 133ms/step
386/386 [==============================] - 50s 131ms/step
```

[708]:
```
# add the last layer to the VGG model
model = Sequential()
model.add(vgg_model)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(120, activation='softmax'))
```

[709]:
```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy', metrics=['accuracy'])
```

[710]:
```
EPOCHS = 10
vgg = model.fit(train_generator,
              validation_data=(train_generator),
              steps_per_epoch = train_generator.n // train_generator.batch_size,
              validation_steps = valid_generator.n // valid_generator.
 ↪batch_size,
              epochs=EPOCHS, verbose=1)
```

```
Epoch 1/10
1542/1543 [============================>.] - ETA: 0s - loss: 7.4353 - acc:
0.0105Epoch 1/10
1543/1543 [==============================] - 209s 136ms/step - loss: 7.4336 -
acc: 0.0105 - val_loss: 4.7784 - val_acc: 0.0123
Epoch 2/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.8480 - acc:
0.0127Epoch 1/10
1543/1543 [==============================] - 208s 135ms/step - loss: 4.8480 -
acc: 0.0127 - val_loss: 4.7773 - val_acc: 0.0146
Epoch 3/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7951 - acc:
0.0110Epoch 1/10
1543/1543 [==============================] - 207s 134ms/step - loss: 4.7951 -
acc: 0.0110 - val_loss: 4.7763 - val_acc: 0.0107
Epoch 4/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.8095 - acc:
0.0145Epoch 1/10
1543/1543 [==============================] - 207s 134ms/step - loss: 4.8095 -
acc: 0.0145 - val_loss: 4.7773 - val_acc: 0.0153
```

```
Epoch 5/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7791 - acc:
0.0151Epoch 1/10
1543/1543 [==============================] - 208s 135ms/step - loss: 4.7791 -
acc: 0.0151 - val_loss: 4.7789 - val_acc: 0.0117
Epoch 6/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7803 - acc:
0.0121Epoch 1/10
1543/1543 [==============================] - 209s 135ms/step - loss: 4.7804 -
acc: 0.0121 - val_loss: 4.7774 - val_acc: 0.0153
Epoch 7/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7874 - acc:
0.0136Epoch 1/10
1543/1543 [==============================] - 208s 135ms/step - loss: 4.7874 -
acc: 0.0136 - val_loss: 4.7778 - val_acc: 0.0120
Epoch 8/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7802 - acc:
0.0125Epoch 1/10
1543/1543 [==============================] - 207s 134ms/step - loss: 4.7801 -
acc: 0.0125 - val_loss: 4.7738 - val_acc: 0.0123
Epoch 9/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7814 - acc:
0.0116Epoch 1/10
1543/1543 [==============================] - 207s 134ms/step - loss: 4.7813 -
acc: 0.0116 - val_loss: 4.7774 - val_acc: 0.0153
Epoch 10/10
1542/1543 [============================>.] - ETA: 0s - loss: 4.7887 - acc:
0.0135Epoch 1/10
1543/1543 [==============================] - 208s 135ms/step - loss: 4.7886 -
acc: 0.0135 - val_loss: 4.7822 - val_acc: 0.0107
```

13 Checking VGG Model Performance

The transfer learning, VGG Model did worst compared to our CNN baseline model.

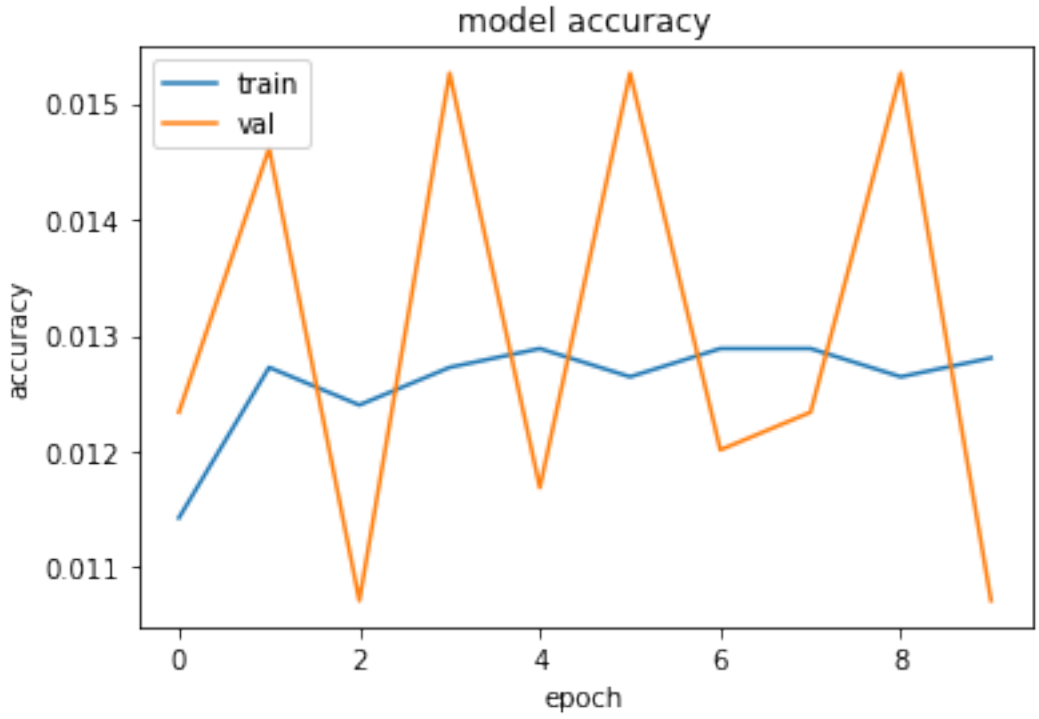
VGG - 1.2% accuracy vs CNN - 3.5% accuracy

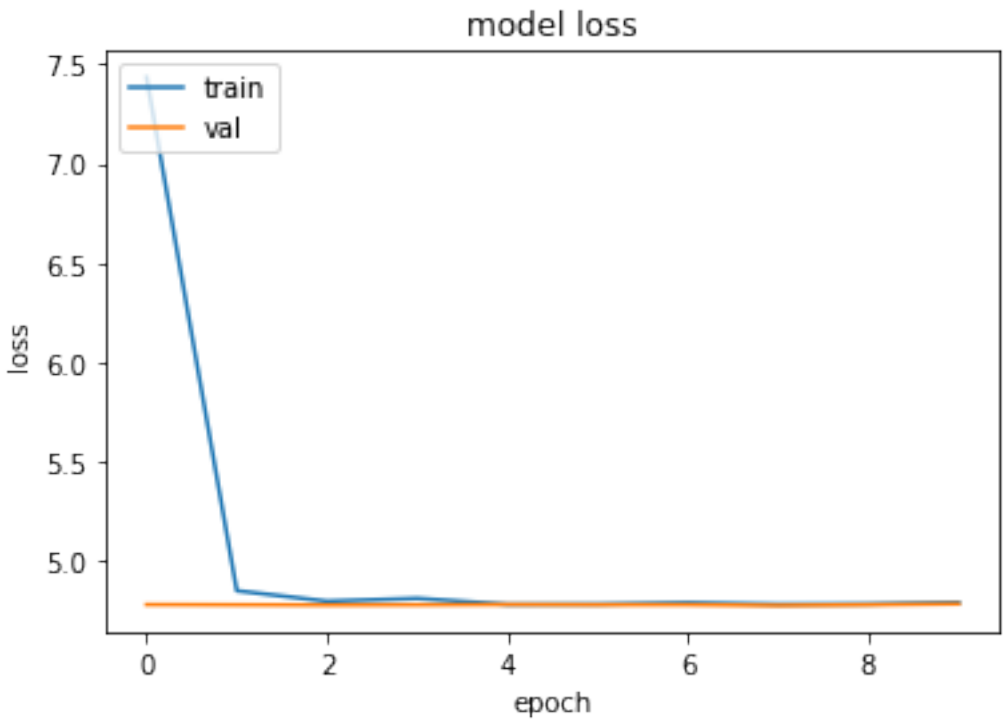
[711]:
```
print('For training set:')
report_accuracy(vgg, train_generator)
print('------------------')
print('For validation set:')
report_accuracy(vgg, valid_generator)
```

```
For training set:
Loss: 4.7782006659038325
Accuracy: 0.012796631
------------------
For validation set:
```

```
Loss: 4.793544256625398
Accuracy: 0.011017499
```

[712]:
```
plt.plot(vgg.history['acc'])
plt.plot(vgg.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.plot(vgg.history['loss'])
plt.plot(vgg.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```





14 Transfer Learning - Inception V3 Model

Inception V3 Model is a pre-trained model on more than a million images, 1,000 classes from the ImageNet database developed by Google. It has stacked layers compare to the traditional CNN model and it is up to 48 layers. V3 here means version 3.

We will see how it will perform.

[33]:
```
from tensorflow.keras.applications import inception_v3
```

[39]:
```
batch_size = 8
jup_path = '/home/ec2-user/SageMaker/Images/'
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=jup_path,
    x_col="image_path",
    y_col="breed",
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode="categorical",
    subset='training',
    shuffle=True,
    seed=42
```

```
)
valid_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=jup_path,
    x_col="image_path",
    y_col="breed",
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode="categorical",
    subset='validation',
    shuffle=True,
    seed=42
)
test_generator = test_datagen.flow_from_dataframe(
    dataframe=test_df,
    directory=jup_path,
    x_col="image_path",
    target_size=(224, 224),
    batch_size=1,
    class_mode=None,
    shuffle=False,
)
```

```
Found 12347 validated image filenames belonging to 120 classes.
Found 3086 validated image filenames belonging to 120 classes.
Found 5145 validated image filenames.
```

[40]:
```
from tensorflow.keras.utils import to_categorical, model_to_dot, plot_model
num_classes = len(train_generator.class_indices)
train_labels = train_generator.classes
train_labels = to_categorical(train_labels, num_classes=num_classes)
valid_labels = valid_generator.classes
valid_labels = to_categorical(valid_labels, num_classes=num_classes)
nb_train_samples = len(train_generator.filenames)
nb_valid_samples = len(valid_generator.filenames)
```

[41]:
```
inception_model = inception_v3.InceptionV3(weights='imagenet',␣
 ↪include_top=False)
```

[42]:
```
model = Sequential()

for layer in inception_model.layers:
    layer.trainable= False
#     print(layer,layer.trainable)

model.add(inception_model)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.2))
```

```
model.add(Dense(120,activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, None, None, 2048)  21802784
_________________________________________________________________
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 120)               245880
=================================================================
Total params: 22,048,664
Trainable params: 245,880
Non-trainable params: 21,802,784
_________________________________________________________________
```

[43]:
```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy', metrics=['accuracy'])
```

[44]:
```
EPOCHS = 10
incept = model.fit(train_generator,
                   steps_per_epoch = nb_train_samples//batch_size,
                   validation_data = valid_generator,
                   validation_steps = nb_valid_samples//batch_size,
                   epochs = EPOCHS,
                   verbose = 1)
```

```
Epoch 1/10
1542/1543 [============================>.] - ETA: 0s - loss: 2.7803 - acc:
0.3554Epoch 1/10
1543/1543 [==============================] - 274s 178ms/step - loss: 2.7802 -
acc: 0.3554 - val_loss: 1.9702 - val_acc: 0.7029
Epoch 2/10
1542/1543 [============================>.] - ETA: 0s - loss: 2.2250 - acc:
0.4839Epoch 1/10
1543/1543 [==============================] - 228s 148ms/step - loss: 2.2255 -
acc: 0.4839 - val_loss: 2.2595 - val_acc: 0.7192
Epoch 3/10
1542/1543 [============================>.] - ETA: 0s - loss: 2.2403 - acc:
0.5111Epoch 1/10
1543/1543 [==============================] - 225s 146ms/step - loss: 2.2398 -
acc: 0.5112 - val_loss: 2.4868 - val_acc: 0.7312
Epoch 4/10
1542/1543 [============================>.] - ETA: 0s - loss: 2.2583 - acc:
```

```
0.5283Epoch 1/10
1543/1543 [==============================] - 230s 149ms/step - loss: 2.2579 -
acc: 0.5282 - val_loss: 2.9167 - val_acc: 0.7198
Epoch 5/10
1542/1543 [============================>.] - ETA: 0s - loss: 2.3424 - acc:
0.5361Epoch 1/10
1543/1543 [==============================] - 224s 145ms/step - loss: 2.3423 -
acc: 0.5361 - val_loss: 2.9434 - val_acc: 0.7328
Epoch 6/10
1542/1543 [============================>.] - ETA: 0s - loss: 2.3555 - acc:
0.5473Epoch 1/10
1543/1543 [==============================] - 223s 144ms/step - loss: 2.3546 -
acc: 0.5475 - val_loss: 3.2083 - val_acc: 0.7325
Epoch 7/10
1542/1543 [============================>.] - ETA: 0s - loss: 2.4043 - acc:
0.5534Epoch 1/10
1543/1543 [==============================] - 221s 143ms/step - loss: 2.4044 -
acc: 0.5534 - val_loss: 3.4329 - val_acc: 0.7273
Epoch 8/10
1542/1543 [============================>.] - ETA: 0s - loss: 2.3864 - acc:
0.5669Epoch 1/10
1543/1543 [==============================] - 224s 145ms/step - loss: 2.3858 -
acc: 0.5670 - val_loss: 3.4564 - val_acc: 0.7351
Epoch 9/10
1542/1543 [============================>.] - ETA: 0s - loss: 2.4275 - acc:
0.5633Epoch 1/10
1543/1543 [==============================] - 224s 145ms/step - loss: 2.4274 -
acc: 0.5633 - val_loss: 3.5580 - val_acc: 0.7305
Epoch 10/10
1542/1543 [============================>.] - ETA: 0s - loss: 2.4884 - acc:
0.5682Epoch 1/10
1543/1543 [==============================] - 223s 144ms/step - loss: 2.4894 -
acc: 0.5681 - val_loss: 3.7386 - val_acc: 0.7263
```

15 Check Performance for Inception V3 Model

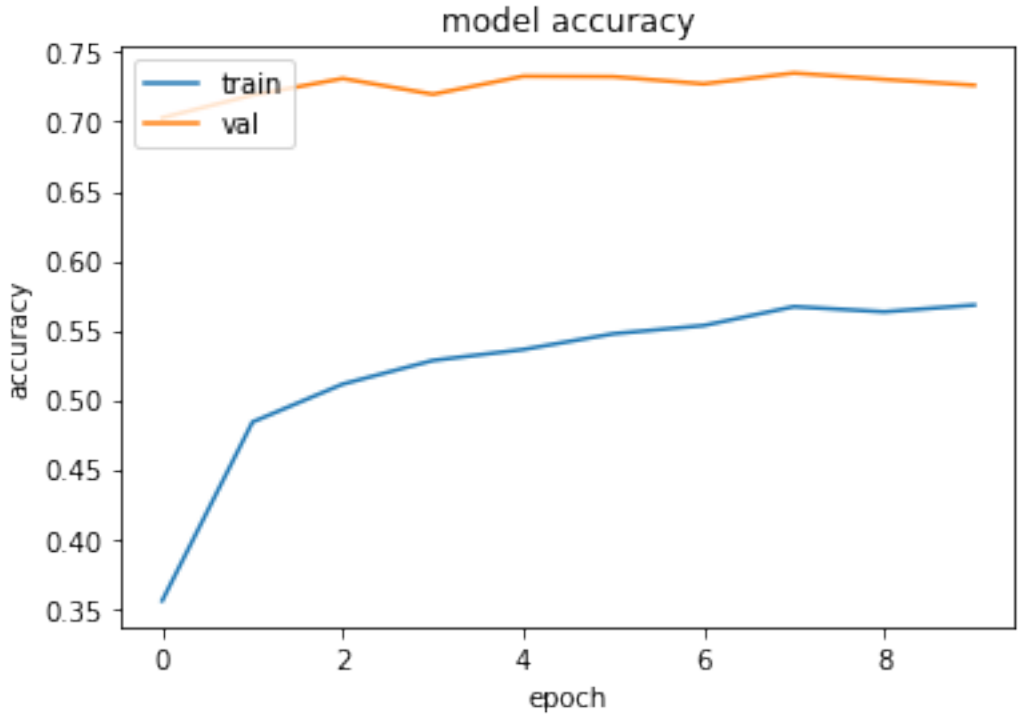
As we can see below that the Inception V3 model reaches its accuracy to 73% to 75% surprisingly. More importantly, the model is not done with the training. We can always increase the number of epoch for improvement.

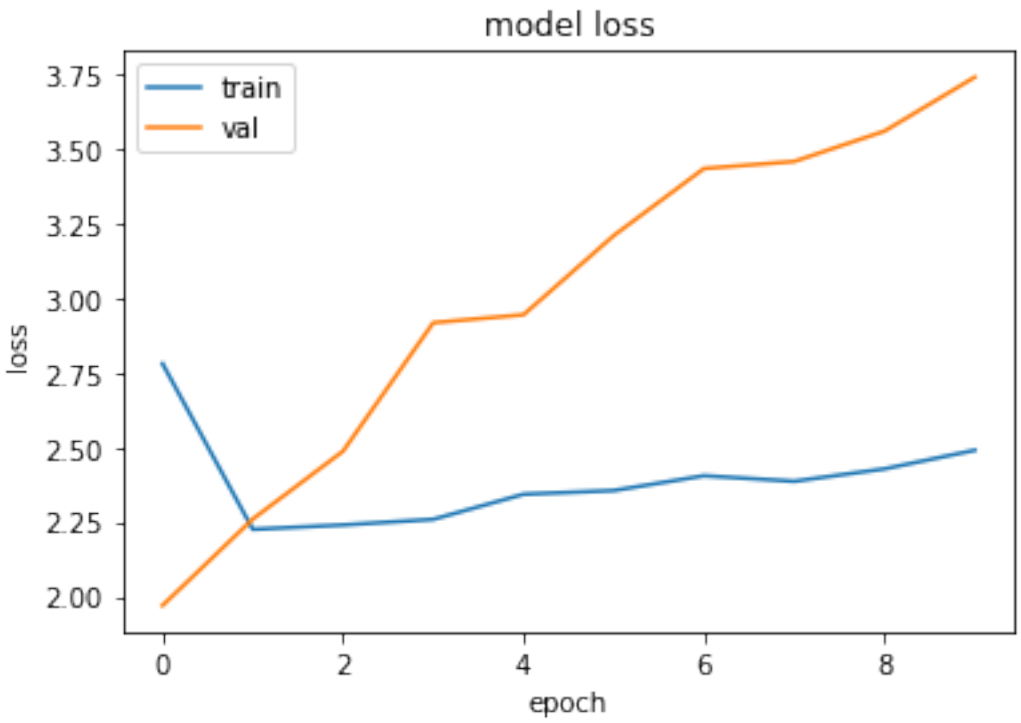
[47]:
```
print('For training set:')
report_accuracy(incept, train_generator)
print('-------------------')
print('For validation set:')
report_accuracy(incept,valid_generator)
```

```
For training set:
Loss: 3.0573402153720206
```

```
Accuracy: 0.75581115
------------------
For validation set:
Loss: 3.71907747693474
Accuracy: 0.7300713
```

[48]:
```
plt.plot(incept.history['acc'])
plt.plot(incept.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.plot(incept.history['loss'])
plt.plot(incept.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```





[49]:
```
target = list(image.breed.unique())
```

[53]:
```
Y_pred = model.predict_generator(valid_generator)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
cf = confusion_matrix(valid_generator.classes, y_pred)
print(cf)
print('Classification Report')
target_names = target
print(classification_report(valid_generator.classes, y_pred,␣
 ↪target_names=target_names))
```

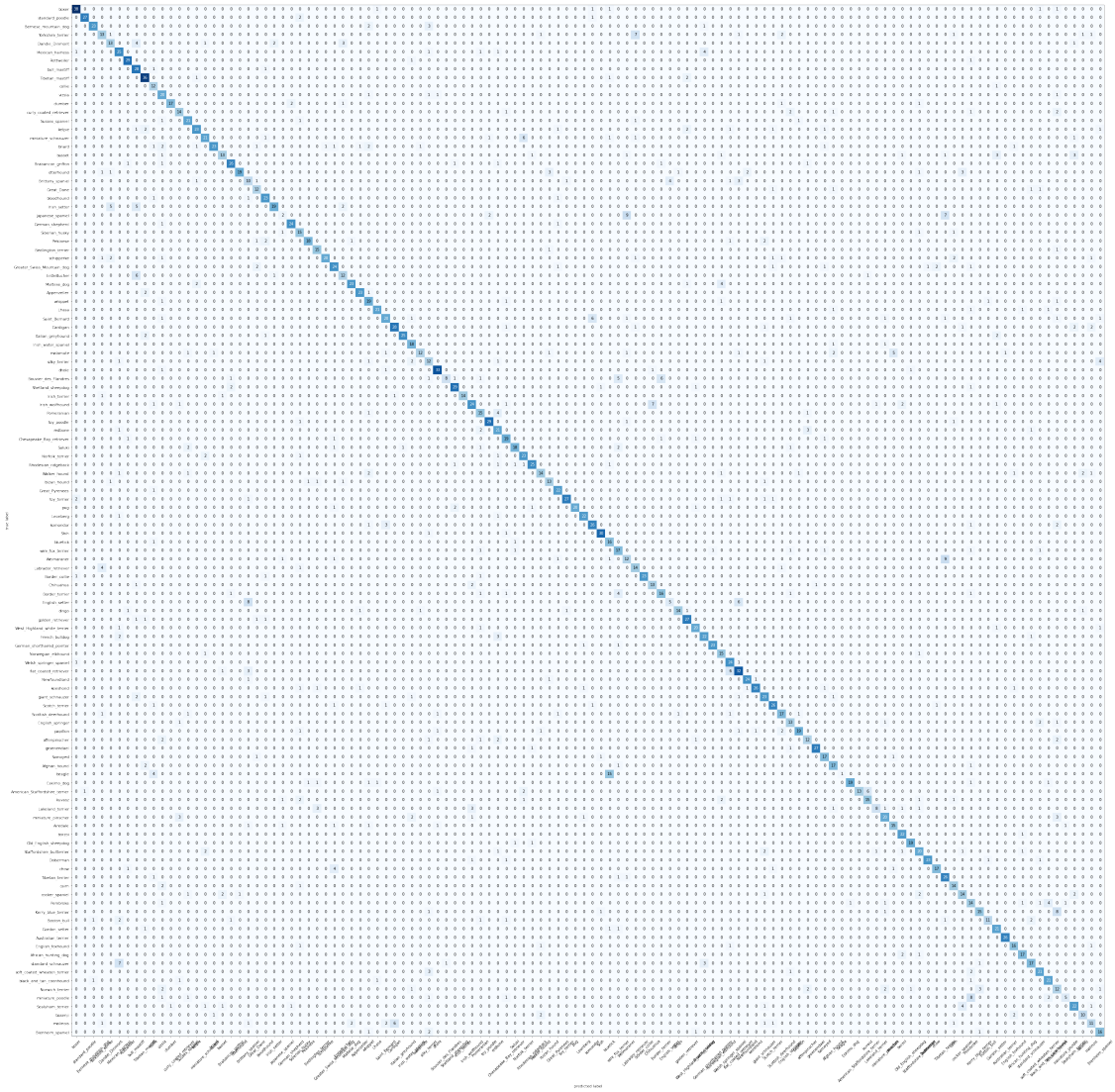
```
Confusion Matrix
[[38  0  0 …  0  0  0]
 [ 0 27  0 …  0  0  0]
 [ 0  0 23 …  0  0  0]
 …
 [ 0  0  0 … 10  0  0]
 [ 0  0  0 …  0 11  0]
 [ 0  0  0 …  0  0 16]]
Classification Report
                              precision    recall  f1-score   support

                       boxer       0.88      0.88      0.88        43
```

standard_poodle	0.96	0.90	0.93	30
Bernese_mountain_dog	0.92	0.79	0.85	29
Yorkshire_terrier	0.59	0.46	0.52	28
Dandie_Dinmont	0.57	0.54	0.55	24
Mexican_hairless	0.61	0.76	0.68	33
Rottweiler	0.88	0.94	0.91	31
bull_mastiff	0.58	0.93	0.72	30
Tibetan_mastiff	0.78	0.90	0.84	40
collie	0.50	0.92	0.65	13
vizsla	0.53	0.83	0.65	24
clumber	0.94	0.77	0.85	22
curly_coated_retriever	0.70	0.67	0.68	21
Sussex_spaniel	0.75	0.84	0.79	25
kelpie	0.82	0.70	0.75	33
miniature_schnauzer	0.75	0.66	0.70	32
briard	0.92	0.64	0.75	36
basset	0.72	0.52	0.60	25
Brabancon_griffon	0.84	0.81	0.83	32
otterhound	0.95	0.61	0.75	31
Brittany_spaniel	0.38	0.48	0.43	21
Great_Dane	0.63	0.75	0.69	16
bloodhound	0.74	0.88	0.81	26
Irish_setter	0.86	0.61	0.72	31
Japanese_spaniel	0.33	0.10	0.15	20
German_shepherd	0.89	0.92	0.91	26
Siberian_husky	0.56	0.83	0.67	18
Pekinese	0.90	0.72	0.80	25
Bedlington_terrier	0.71	0.88	0.79	17
schipperke	0.95	0.67	0.78	30
Greater_Swiss_Mountain_dog	0.69	0.73	0.71	33
EntleBucher	0.63	0.60	0.62	20
Maltese_dog	0.88	0.77	0.82	30
Appenzeller	0.88	0.88	0.88	26
whippet	0.58	0.79	0.67	24
Lhasa	0.88	0.84	0.86	25
Saint_Bernard	0.71	0.59	0.65	34
Cardigan	0.80	0.78	0.79	36
Italian_greyhound	0.96	0.83	0.89	30
Irish_water_spaniel	0.69	0.90	0.78	20
malamute	0.86	0.46	0.60	26
silky_terrier	0.44	0.55	0.49	22
dhole	0.87	0.89	0.88	37
Bouvier_des_Flandres	0.89	0.35	0.50	23
Shetland_sheepdog	0.85	0.85	0.85	34
Irish_terrier	1.00	0.82	0.90	17
Irish_wolfhound	0.75	0.63	0.69	38
Pomeranian	0.58	0.71	0.64	21
toy_poodle	0.91	0.91	0.91	32

redbone	0.57	0.75	0.65	28
Chesapeake_Bay_retriever	0.68	0.79	0.73	24
Saluki	0.82	0.69	0.75	26
Norfolk_terrier	0.70	0.85	0.77	27
Rhodesian_ridgeback	0.86	0.83	0.85	30
Walker_hound	0.74	0.58	0.65	24
Ibizan_hound	0.68	0.72	0.70	18
Great_Pyrenees	0.79	0.96	0.86	23
toy_terrier	0.93	0.79	0.86	34
pug	1.00	0.80	0.89	25
Leonberg	0.88	0.92	0.90	24
komondor	0.74	0.79	0.76	33
Shih	0.91	0.94	0.92	32
bluetick	0.43	0.89	0.58	18
wire_fox_terrier	0.53	0.71	0.61	24
Weimaraner	0.46	0.48	0.47	25
Labrador_retriever	0.56	0.70	0.62	20
Border_collie	1.00	0.88	0.94	26
Chihuahua	0.62	0.76	0.68	17
Border_terrier	0.67	0.64	0.65	25
English_setter	0.56	0.26	0.36	19
dingo	0.88	0.70	0.78	20
golden_retriever	0.82	0.87	0.84	31
West_Highland_white_terrier	0.80	0.83	0.82	24
French_bulldog	0.73	0.79	0.76	28
German_shorthaired_pointer	0.96	0.92	0.94	25
Norwegian_elkhound	0.60	0.88	0.71	17
Welsh_springer_spaniel	0.67	0.92	0.77	26
flat_coated_retriever	0.70	0.76	0.73	42
Newfoundland	0.89	0.89	0.89	27
keeshond	0.93	0.79	0.85	33
giant_schnauzer	0.68	0.79	0.73	29
Scotch_terrier	0.79	0.84	0.81	31
Scottish_deerhound	0.71	0.71	0.71	24
English_springer	0.62	0.72	0.67	18
papillon	0.86	0.73	0.79	26
affenpinscher	0.63	0.57	0.60	21
groenendael	0.96	0.90	0.93	30
Samoyed	0.74	0.85	0.79	20
Afghan_hound	0.68	0.81	0.74	21
beagle	1.00	0.05	0.10	20
Eskimo_dog	0.95	0.76	0.84	25
American_Staffordshire_terrier	1.00	0.57	0.72	23
kuvasz	0.68	0.65	0.67	23
Lakeland_terrier	0.80	0.44	0.57	18
miniature_pinscher	0.71	0.67	0.69	30
Airedale	0.68	0.71	0.70	21
borzoi	0.76	0.96	0.85	23

Old_English_sheepdog	0.90	0.90	0.90	21
Staffordshire_bullterrier	0.69	0.69	0.69	29
Doberman	0.96	0.85	0.90	27
chow	0.85	0.71	0.77	24
Tibetan_terrier	0.58	0.90	0.70	29
cairn	0.67	0.82	0.74	17
cocker_spaniel	0.61	0.61	0.61	23
Pembroke	0.40	0.58	0.47	24
Kerry_blue_terrier	0.75	0.58	0.65	26
Boston_bull	0.85	0.55	0.67	20
Gordon_setter	0.72	0.88	0.79	24
Australian_terrier	0.96	1.00	0.98	26
English_foxhound	0.76	0.80	0.78	20
African_hunting_dog	0.81	0.85	0.83	20
standard_schnauzer	0.77	0.59	0.67	29
soft_coated_wheaten_terrier	0.75	0.78	0.76	27
black_and_tan_coonhound	0.72	0.84	0.78	25
Norwich_terrier	0.34	0.50	0.41	24
miniature_poodle	0.71	0.26	0.38	19
Sealyham_terrier	0.71	0.69	0.70	32
basenji	0.71	0.67	0.69	15
malinois	0.61	0.42	0.50	26
Blenheim_spaniel	0.67	0.76	0.71	21
accuracy			0.74	3086
macro avg	0.75	0.73	0.73	3086
weighted avg	0.76	0.74	0.74	3086

[54]:
```
plot_confusion_matrix(conf_mat=cf,
                      class_names=target,
                      figsize=(50,50));
```



16 Inception V3 Model for Groups Label

[55]:
```
batch_size = 8
jup_path = '/home/ec2-user/SageMaker/Images/'
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=jup_path,
    x_col="image_path",
    y_col="group",
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode="categorical",
```

```
    subset='training',
    shuffle=True,
    seed=42
)
valid_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=jup_path,
    x_col="image_path",
    y_col="group",
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode="categorical",
    subset='validation',
    shuffle=True,
    seed=42
)
test_generator = test_datagen.flow_from_dataframe(
    dataframe=test_df,
    directory=jup_path,
    x_col="image_path",
    y_col=None,
    target_size=(224, 224),
    batch_size=1,
    class_mode=None,
    shuffle=False,
)
```

```
Found 12347 validated image filenames belonging to 9 classes.
Found 3086 validated image filenames belonging to 9 classes.
Found 5145 validated image filenames.
```

[56]:
```
model = Sequential()

for layer in inception_model.layers:
    layer.trainable= False
#     print(layer,layer.trainable)

model.add(inception_model)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.2))
model.add(Dense(9, activation='softmax'))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, None, None, 2048)  21802784
```

```
_________________________________________________________________
global_average_pooling2d_2 ( (None, 2048)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 9)                 18441
=================================================================
Total params: 21,821,225
Trainable params: 18,441
Non-trainable params: 21,802,784
_________________________________________________________________
```

[57]:
```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy', metrics=['accuracy'])
```

[58]:
```
EPOCHS = 10
incept = model.fit(train_generator,
                   steps_per_epoch = nb_train_samples//batch_size,
                   validation_data = valid_generator,
                   validation_steps = nb_valid_samples//batch_size,
                   epochs = EPOCHS,
                   verbose = 1)
```

```
Epoch 1/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.4506 - acc:
0.4892Epoch 1/10
1543/1543 [==============================] - 232s 150ms/step - loss: 1.4501 -
acc: 0.4893 - val_loss: 0.9505 - val_acc: 0.7695
Epoch 2/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.3093 - acc:
0.5553Epoch 1/10
1543/1543 [==============================] - 225s 146ms/step - loss: 1.3093 -
acc: 0.5554 - val_loss: 0.8594 - val_acc: 0.7971
Epoch 3/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.2910 - acc:
0.5612Epoch 1/10
1543/1543 [==============================] - 228s 148ms/step - loss: 1.2911 -
acc: 0.5611 - val_loss: 1.0325 - val_acc: 0.7714
Epoch 4/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.2858 - acc:
0.5674Epoch 1/10
1543/1543 [==============================] - 226s 146ms/step - loss: 1.2856 -
acc: 0.5674 - val_loss: 0.9454 - val_acc: 0.7951
Epoch 5/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.2645 - acc:
0.5748Epoch 1/10
1543/1543 [==============================] - 224s 145ms/step - loss: 1.2646 -
acc: 0.5749 - val_loss: 0.9106 - val_acc: 0.7938
```

```
Epoch 6/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.2719 - acc:
0.5800Epoch 1/10
1543/1543 [==============================] - 226s 147ms/step - loss: 1.2719 -
acc: 0.5801 - val_loss: 0.9941 - val_acc: 0.7870
Epoch 7/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.2791 - acc:
0.5742Epoch 1/10
1543/1543 [==============================] - 227s 147ms/step - loss: 1.2794 -
acc: 0.5740 - val_loss: 0.9574 - val_acc: 0.7994
Epoch 8/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.2644 - acc:
0.5815Epoch 1/10
1543/1543 [==============================] - 227s 147ms/step - loss: 1.2642 -
acc: 0.5813 - val_loss: 1.0339 - val_acc: 0.7857
Epoch 9/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.2662 - acc:
0.5791Epoch 1/10
1543/1543 [==============================] - 229s 149ms/step - loss: 1.2664 -
acc: 0.5790 - val_loss: 0.9929 - val_acc: 0.7909
Epoch 10/10
1542/1543 [============================>.] - ETA: 0s - loss: 1.2440 - acc:
0.5916Epoch 1/10
1543/1543 [==============================] - 226s 146ms/step - loss: 1.2439 -
acc: 0.5917 - val_loss: 0.9912 - val_acc: 0.7964
```

17 Check Performance for Inception V3 Model (Groups)

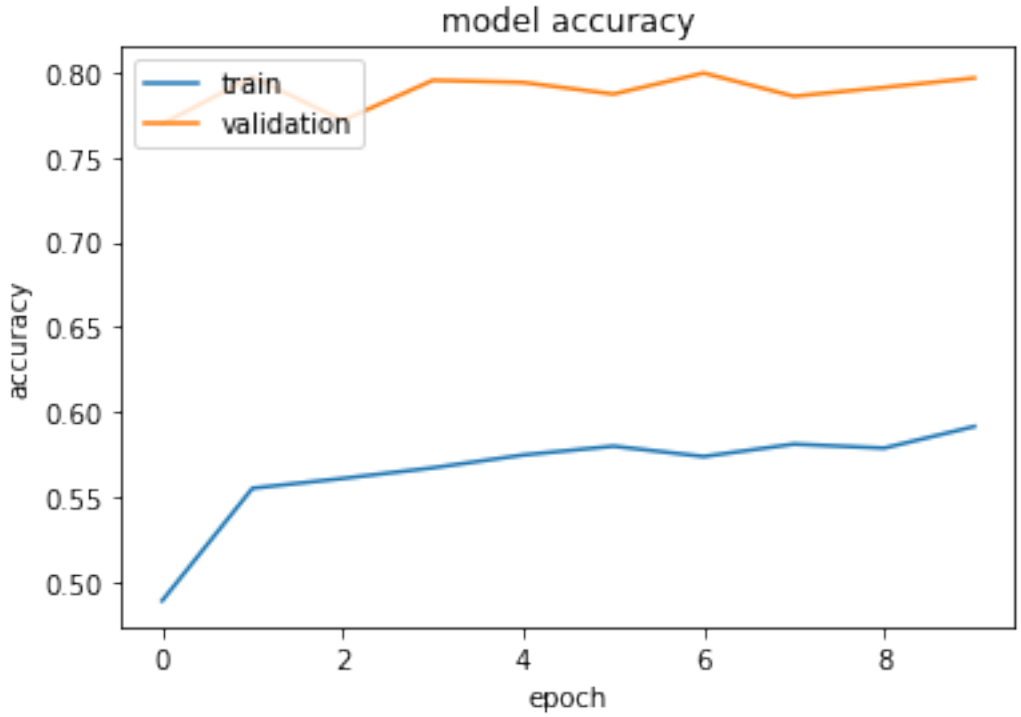
Similarly, the Inception V3 model increase dramatically compares to our CNN model. It reaches 80% accuracy.

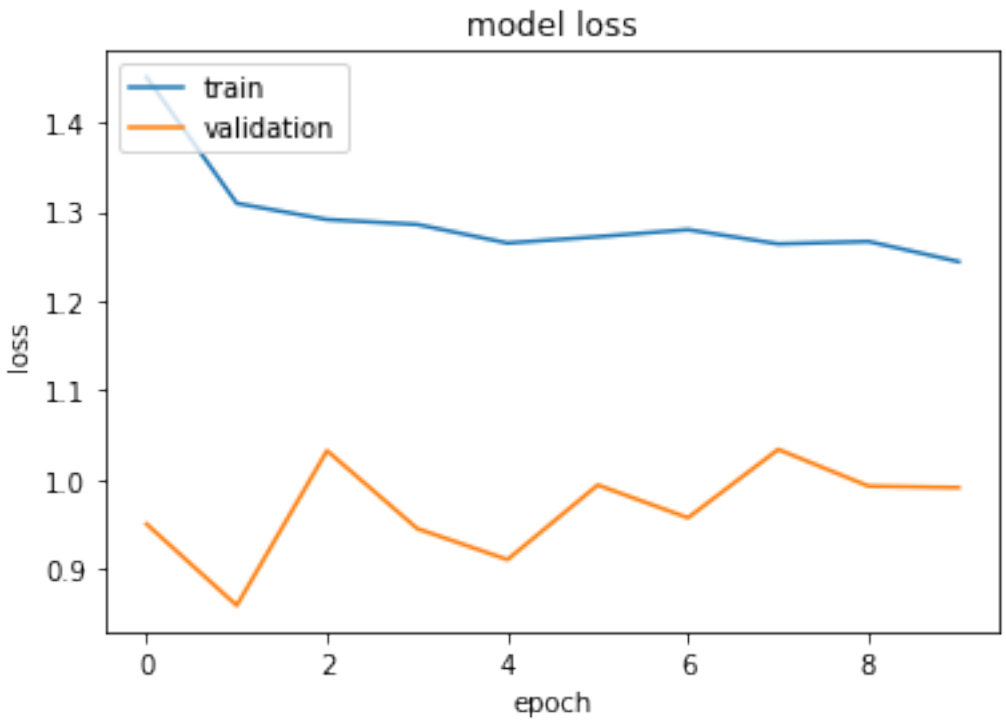
[59]:
```
print('For training set:')
report_accuracy(incept, train_generator)
print('------------------')
print('For validation set:')
report_accuracy(incept,valid_generator)
```

```
For training set:
Loss: 0.9024185325833874
Accuracy: 0.8075646
------------------
For validation set:
Loss: 0.9491798921569354
Accuracy: 0.81432277
```

[60]:
```
plt.plot(incept.history['acc'])
plt.plot(incept.history['val_acc'])
plt.title('model accuracy')
```

```
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(incept.history['loss'])
plt.plot(incept.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```





18 Making Prediction with Our Test Dataset

We didn't have a chance to use our Test Dataset since the other models didn't perform well. But finally, we got a model that we feel satisfied and we can test that with our test dataset.

The result comes pretty well too, in which we have 4230 images predicted correctly and 915 images predicted incorrectly.

[65]:
```
test_generator.reset()
pred=model.predict_generator(test_generator,
                             steps=test_generator.n//test_generator.batch_size,
                             verbose=1)
```

```
5145/5145 [==============================] - 178s 35ms/step
```

[66]:
```
predicted_class_indices=np.argmax(pred,axis=1)
labels = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels.items())
predictions = [labels[k] for k in predicted_class_indices]
```

[70]:
```
filenames=test_generator.filenames
results=pd.DataFrame({"filename":filenames,
                      "prediction":predictions})
```

[72]:
```
test_result = pd.merge(left=image, right=results, left_on='image_path',␣
↪right_on='filename')
```

[73]:
```
test_result = test_result.drop(['filename'], axis=1)
```

[74]:
```
test_result['Compare'] = np.where(test_result['group'] ==␣
↪test_result['prediction'],'Correct','Not Correct')
```

[75]:
```
correct = test_result[test_result['Compare'] == 'Correct']
not_correct = test_result[test_result['Compare'] != 'Correct']
```

[76]:
```
correct
```

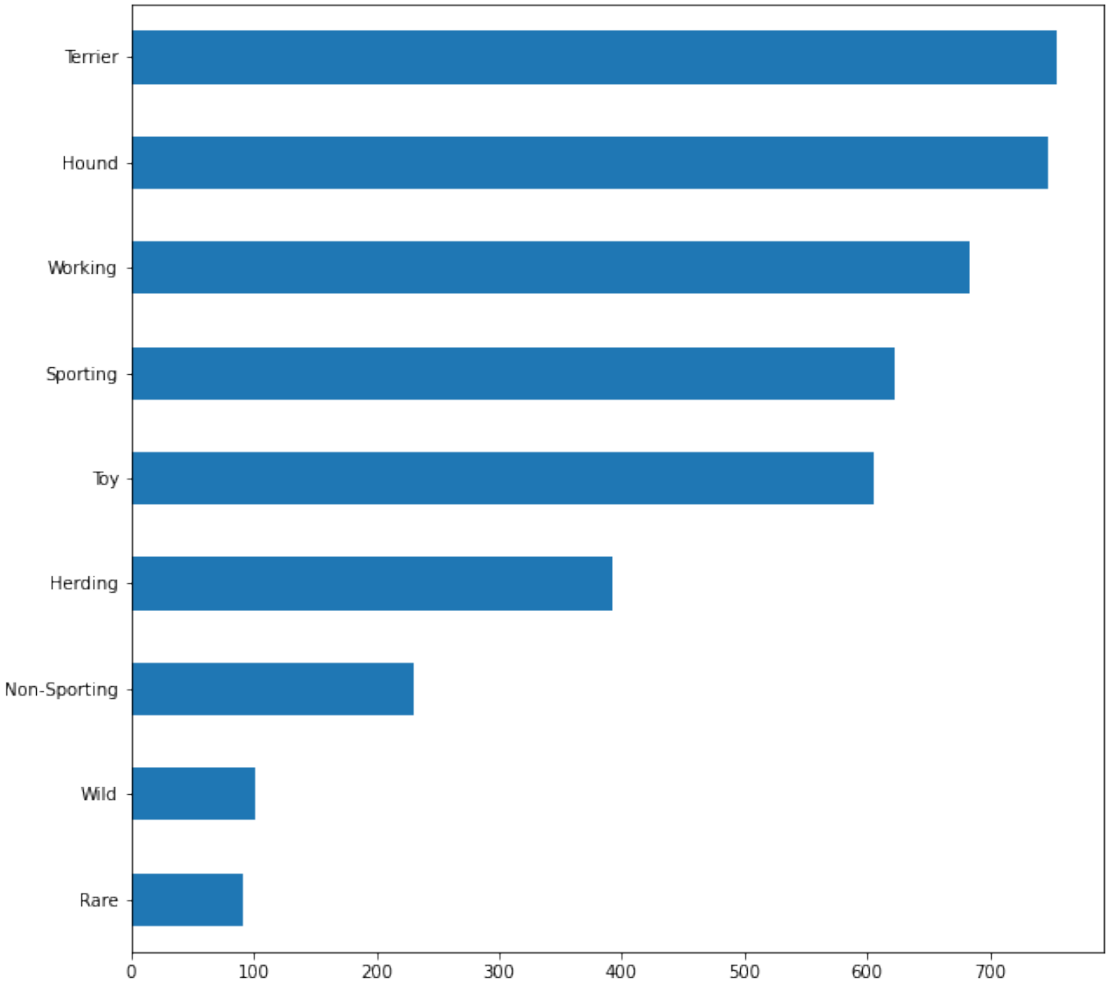
[76]:
```
                                          image_path             breed  \
0                n02108089-boxer/n02108089_9076.jpg             boxer
1                n02108089-boxer/n02108089_8739.jpg             boxer
3                n02108089-boxer/n02108089_1619.jpg             boxer
4                n02108089-boxer/n02108089_7259.jpg             boxer
6               n02108089-boxer/n02108089_10901.jpg             boxer
…                                                 …                …
5140  n02086646-Blenheim_spaniel/n02086646_4045.jpg  Blenheim_spaniel
5141   n02086646-Blenheim_spaniel/n02086646_166.jpg  Blenheim_spaniel
5142  n02086646-Blenheim_spaniel/n02086646_3688.jpg  Blenheim_spaniel
5143  n02086646-Blenheim_spaniel/n02086646_2498.jpg  Blenheim_spaniel
5144  n02086646-Blenheim_spaniel/n02086646_1586.jpg  Blenheim_spaniel

        group prediction  Compare
0     Working    Working  Correct
1     Working    Working  Correct
3     Working    Working  Correct
4     Working    Working  Correct
6     Working    Working  Correct
…       …          …       …
5140      Toy        Toy  Correct
5141      Toy        Toy  Correct
5142      Toy        Toy  Correct
5143      Toy        Toy  Correct
5144      Toy        Toy  Correct

[4230 rows x 5 columns]
```

[77]:
```
not_correct
```

[77]:
```
                                          image_path             breed  \
2                n02108089-boxer/n02108089_1410.jpg             boxer
5                n02108089-boxer/n02108089_5301.jpg             boxer
7               n02108089-boxer/n02108089_13839.jpg             boxer
```

```
12                n02108089-boxer/n02108089_1072.jpg              boxer
15                n02108089-boxer/n02108089_6429.jpg              boxer
…                                                  …                 …
5124  n02086646-Blenheim_spaniel/n02086646_3536.jpg  Blenheim_spaniel
5128  n02086646-Blenheim_spaniel/n02086646_1920.jpg  Blenheim_spaniel
5129  n02086646-Blenheim_spaniel/n02086646_1182.jpg  Blenheim_spaniel
5136  n02086646-Blenheim_spaniel/n02086646_2621.jpg  Blenheim_spaniel
5138   n02086646-Blenheim_spaniel/n02086646_303.jpg  Blenheim_spaniel

        group    prediction      Compare
2     Working  Non-Sporting  Not Correct
5     Working       Terrier  Not Correct
7     Working       Terrier  Not Correct
12    Working         Hound  Not Correct
15    Working       Terrier  Not Correct
…       …           …          …
5124      Toy      Sporting  Not Correct
5128      Toy      Sporting  Not Correct
5129      Toy      Sporting  Not Correct
5136      Toy      Sporting  Not Correct
5138      Toy      Sporting  Not Correct

[915 rows x 5 columns]
```

[78]:
```
correct['group'].value_counts().sort_values(ascending=True).plot.
↪barh(figsize=(10,10));
```

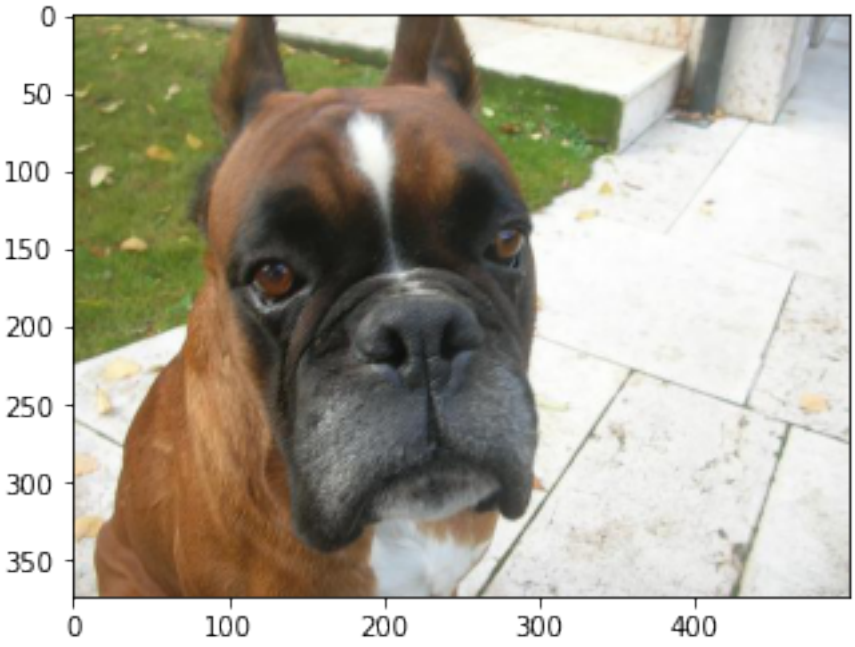


19 To Show Images with Correct and Not Correct Prediction

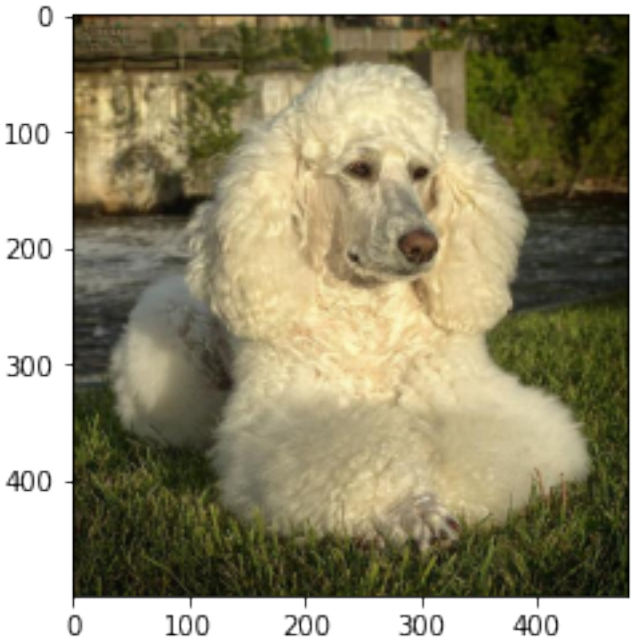
[80]:
```
def predict_and_show(file):
    img = Image.open(file)
    plt.imshow(img)
    plt.show()
```

[82]:
```
img_path = '/home/ec2-user/SageMaker/Images/'
for breed, group, path in zip(correct.breed[::400], correct.group[::400],␣
 ↪correct.image_path[::30]):
    print(breed + ' - ' + group)
    predict_and_show(img_path + path)
```

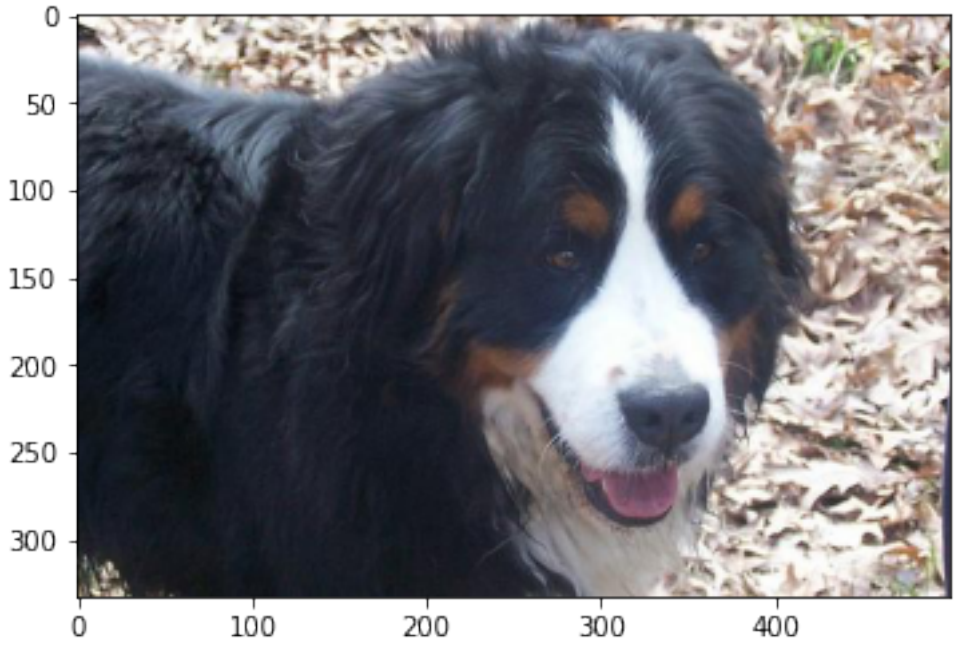
```
boxer - Working
```



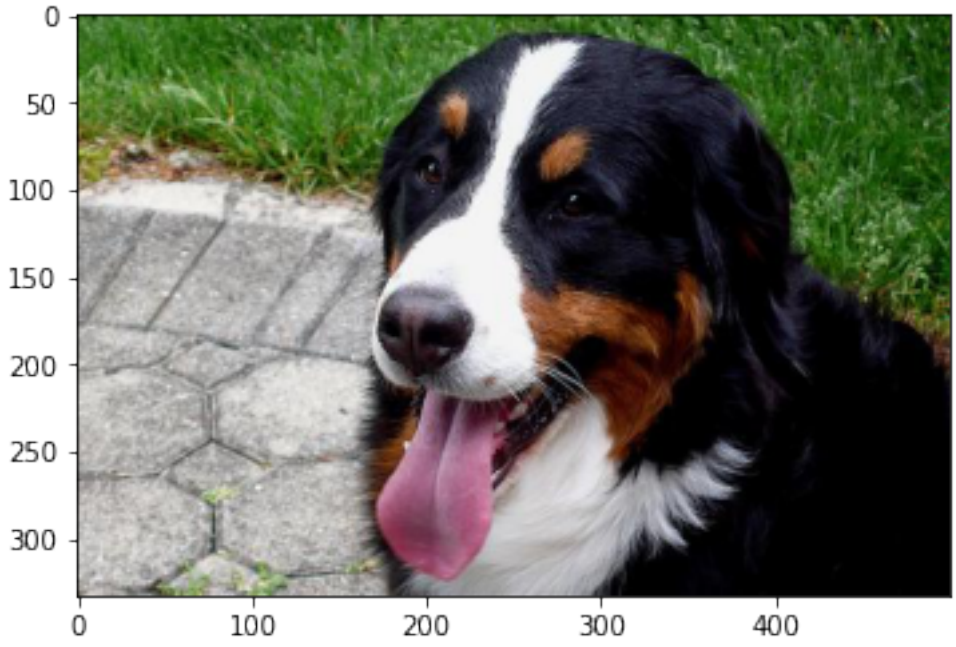

```
clumber - Sporting
```



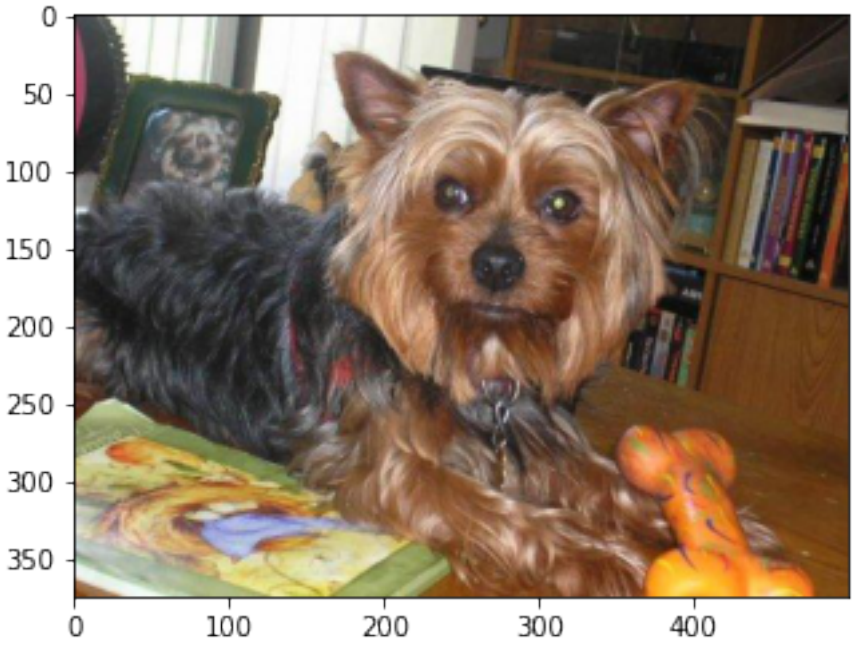

```
Japanese_spaniel - Toy
```



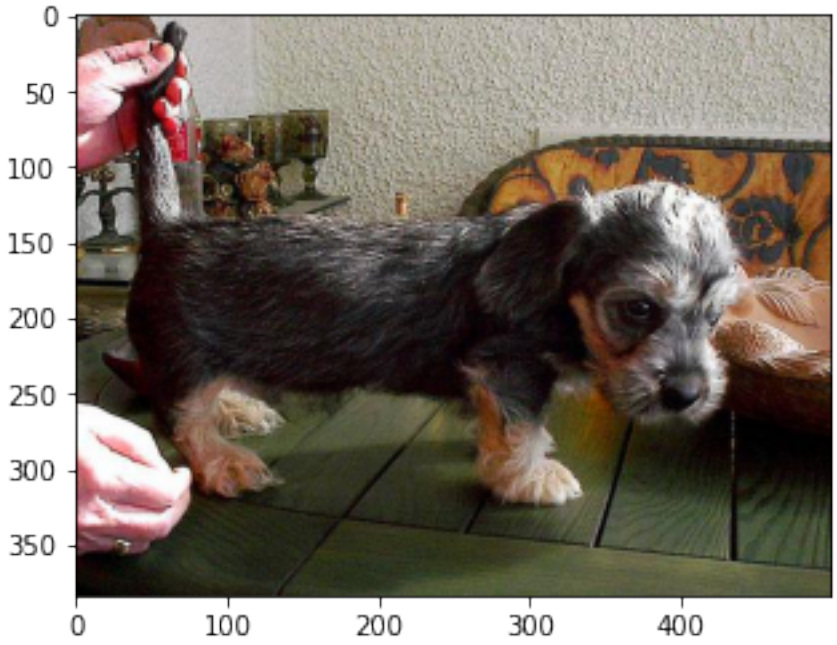

```
Saint_Bernard - Working
```



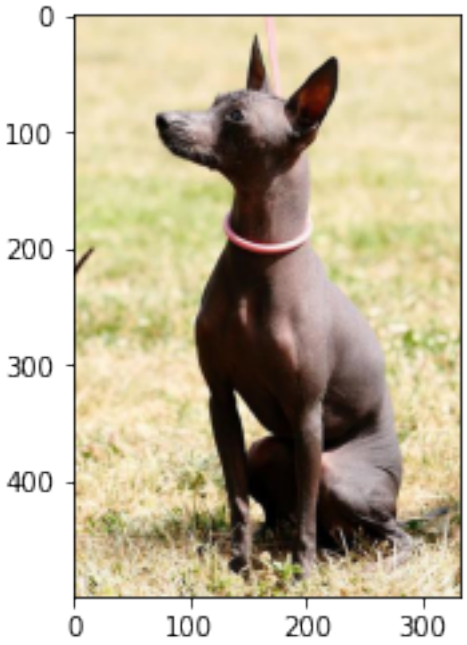

```
Pomeranian - Toy
```



Leonberg - Working

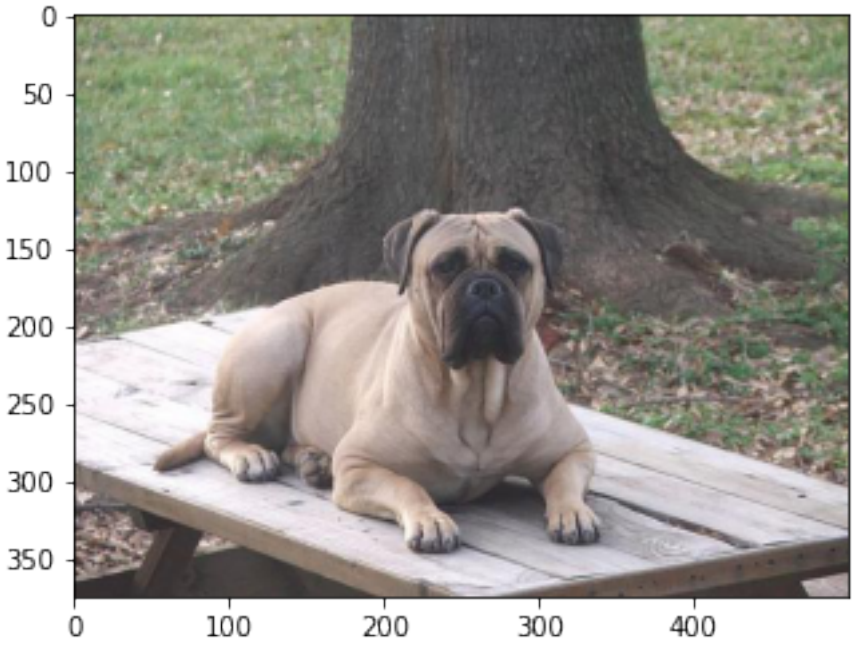


dingo - Wild

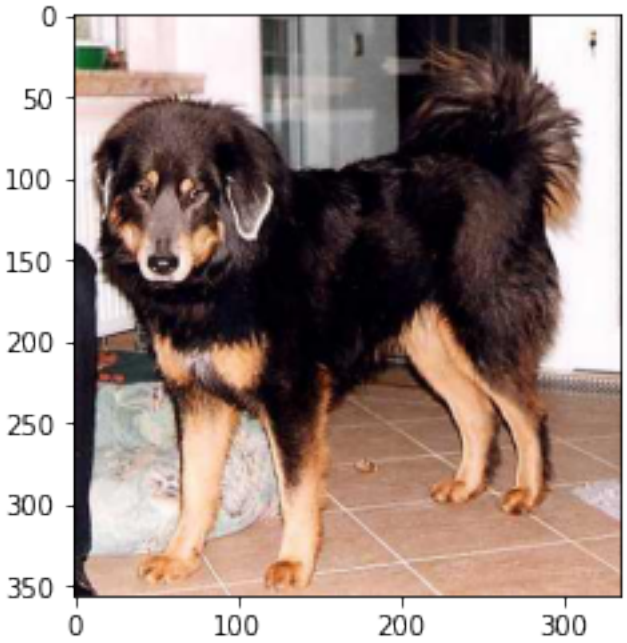



```
Scottish_deerhound - Hound
```

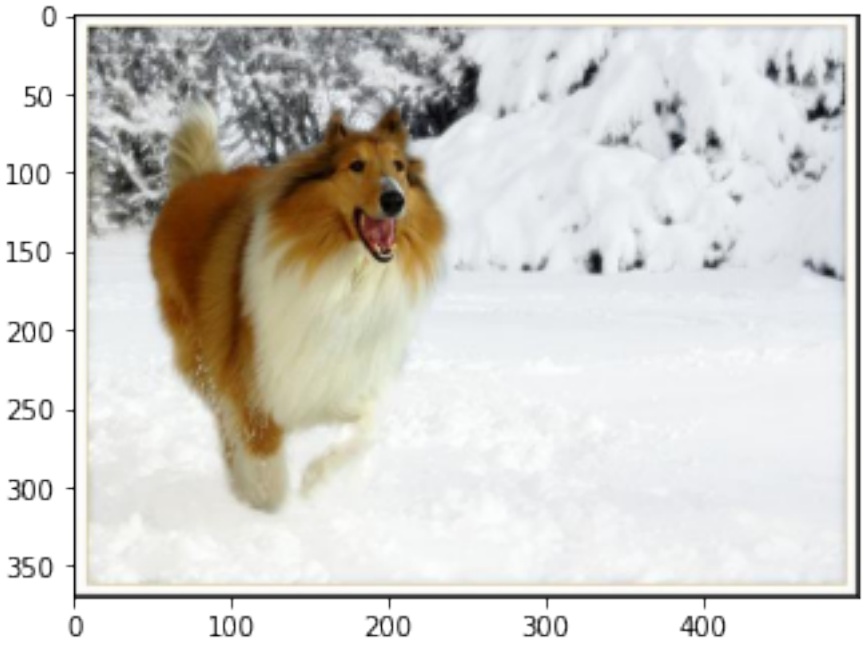

```
kuvasz - Working
```




```
cairn - Terrier
```




```
Norwich_terrier - Terrier
```

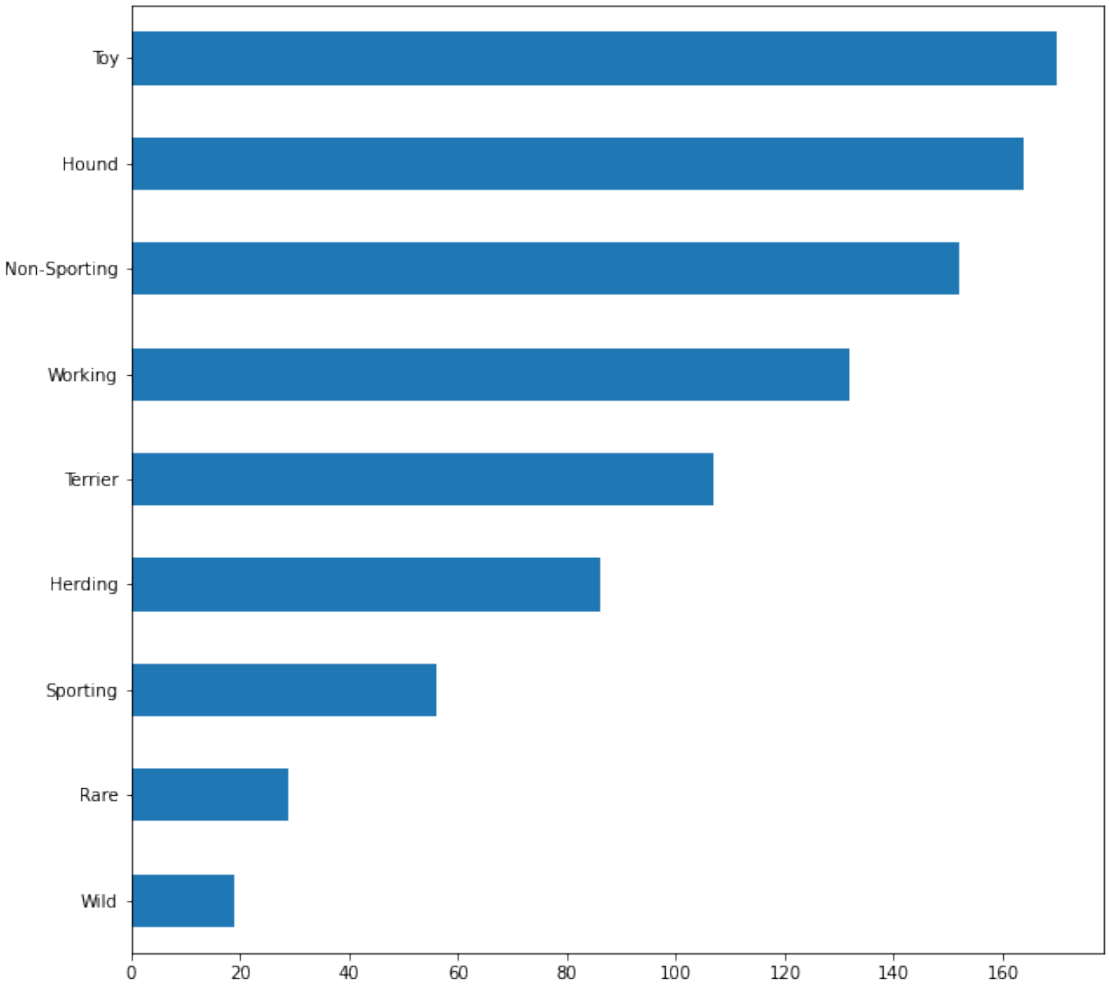


[83]:
```
not_correct
```

[83]:
```
                                        image_path             breed  \
2                n02108089-boxer/n02108089_1410.jpg             boxer
5                n02108089-boxer/n02108089_5301.jpg             boxer
7               n02108089-boxer/n02108089_13839.jpg             boxer
12               n02108089-boxer/n02108089_1072.jpg             boxer
15               n02108089-boxer/n02108089_6429.jpg             boxer
…                                              …                 …
5124  n02086646-Blenheim_spaniel/n02086646_3536.jpg  Blenheim_spaniel
5128  n02086646-Blenheim_spaniel/n02086646_1920.jpg  Blenheim_spaniel
5129  n02086646-Blenheim_spaniel/n02086646_1182.jpg  Blenheim_spaniel
5136  n02086646-Blenheim_spaniel/n02086646_2621.jpg  Blenheim_spaniel
5138   n02086646-Blenheim_spaniel/n02086646_303.jpg  Blenheim_spaniel

        group    prediction      Compare
2     Working  Non-Sporting  Not Correct
5     Working       Terrier  Not Correct
7     Working       Terrier  Not Correct
12    Working         Hound  Not Correct
15    Working       Terrier  Not Correct
…       …             …           …
5124      Toy      Sporting  Not Correct
5128      Toy      Sporting  Not Correct
5129      Toy      Sporting  Not Correct
```

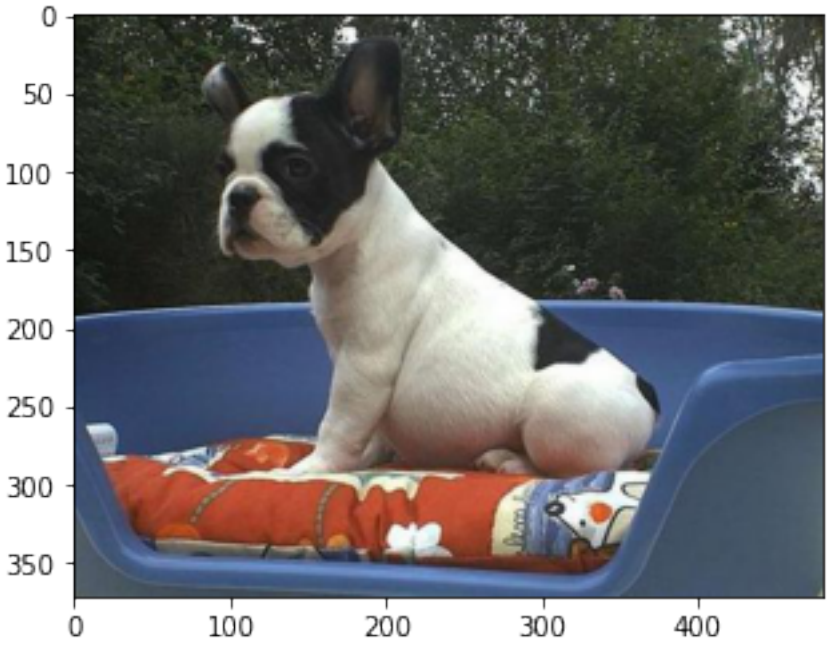
```
5136      Toy      Sporting  Not Correct
5138      Toy      Sporting  Not Correct

[915 rows x 5 columns]
```

[84]:
```
not_correct['group'].value_counts().sort_values(ascending=True).plot.
↪barh(figsize=(10,10));
```



[86]:
```
for breed, group, prediction, path in zip(not_correct.breed[::90],
                                          not_correct.group[::90],
                                          not_correct.prediction[::90],
                                          not_correct.image_path[::90]):
    print(f'{breed} - Actual {group} vs Predict {prediction}')
    predict_and_show(img_path + path)
```

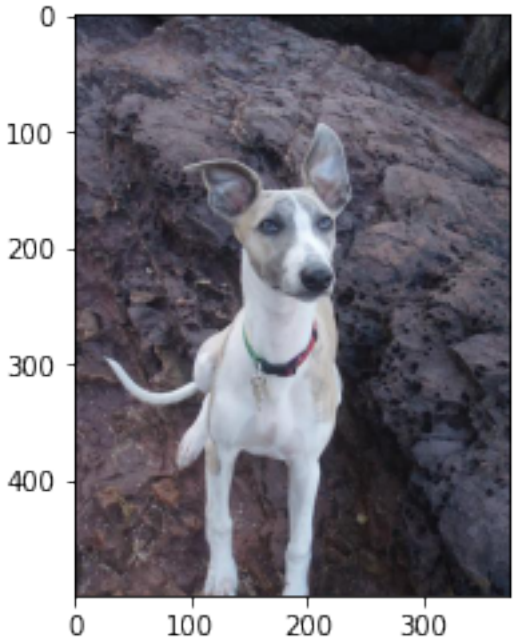
```
boxer - Actual Working vs Predict Non-Sporting
```



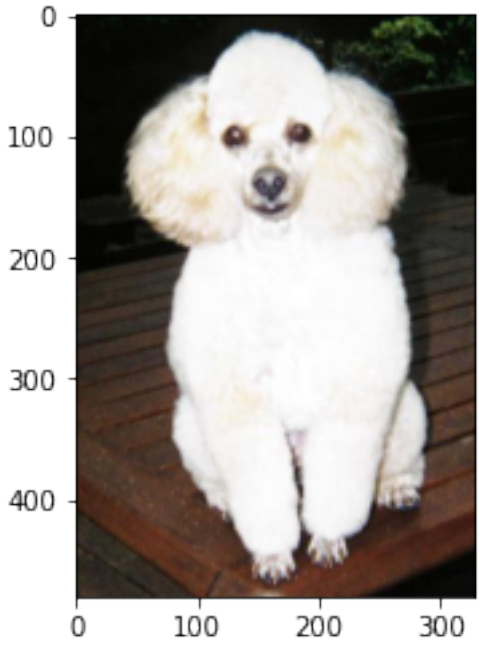
miniature_schnauzer - Actual Terrier vs Predict Working

schipperke - Actual Non-Sporting vs Predict Rare

whippet - Actual Hound vs Predict Toy

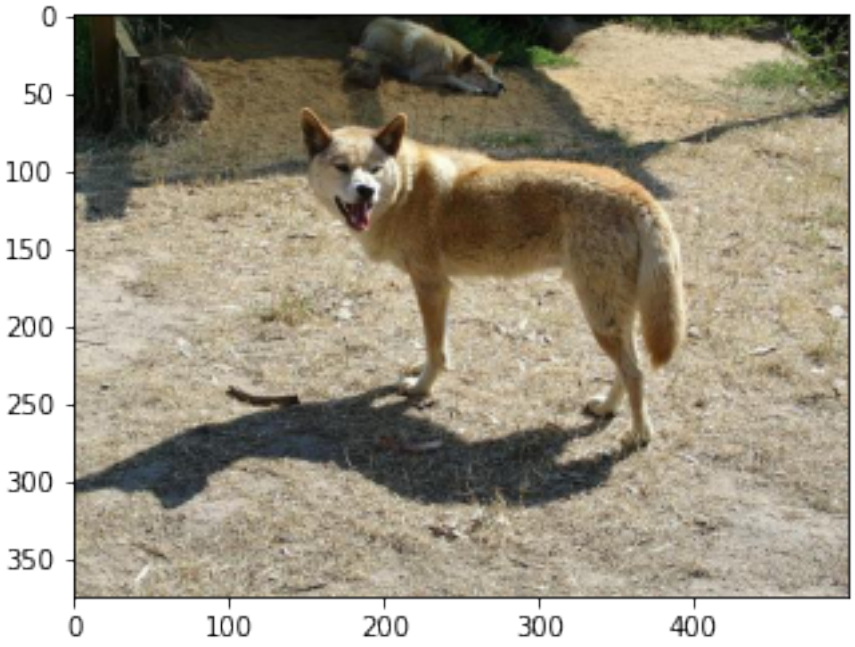


toy_poodle - Actual Toy vs Predict Non-Sporting



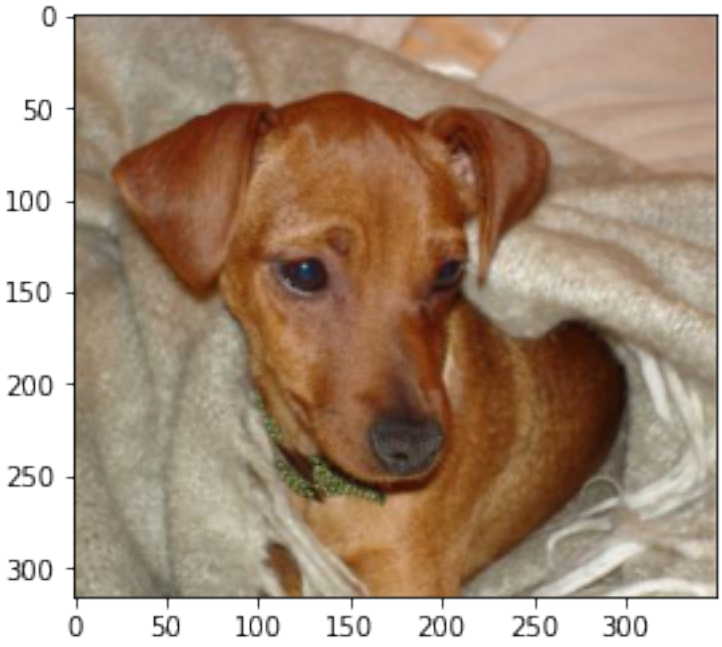
pug - Actual Toy vs Predict Working

dingo - Actual Wild vs Predict Herding

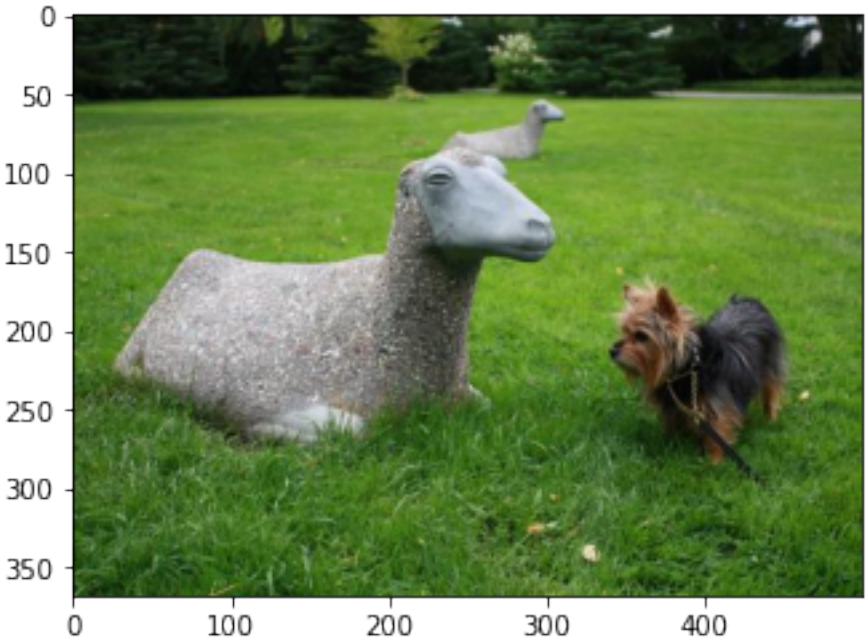


Scotch_terrier - Actual Terrier vs Predict Working

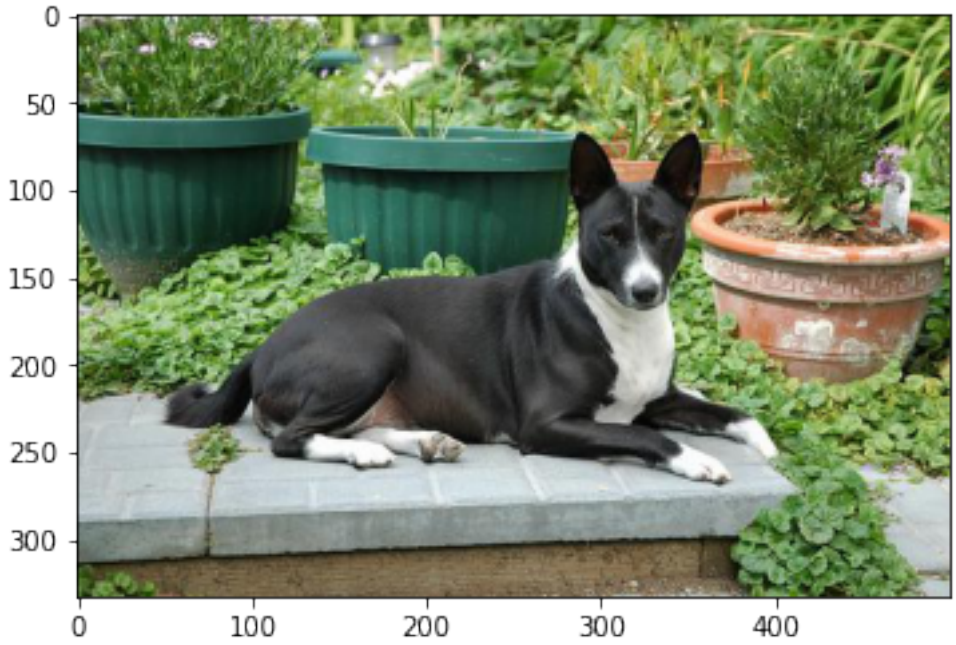
miniature_pinscher - Actual Toy vs Predict Terrier



Australian_terrier - Actual Terrier vs Predict Hound



basenji - Actual Hound vs Predict Rare



20 Confusion Matrix

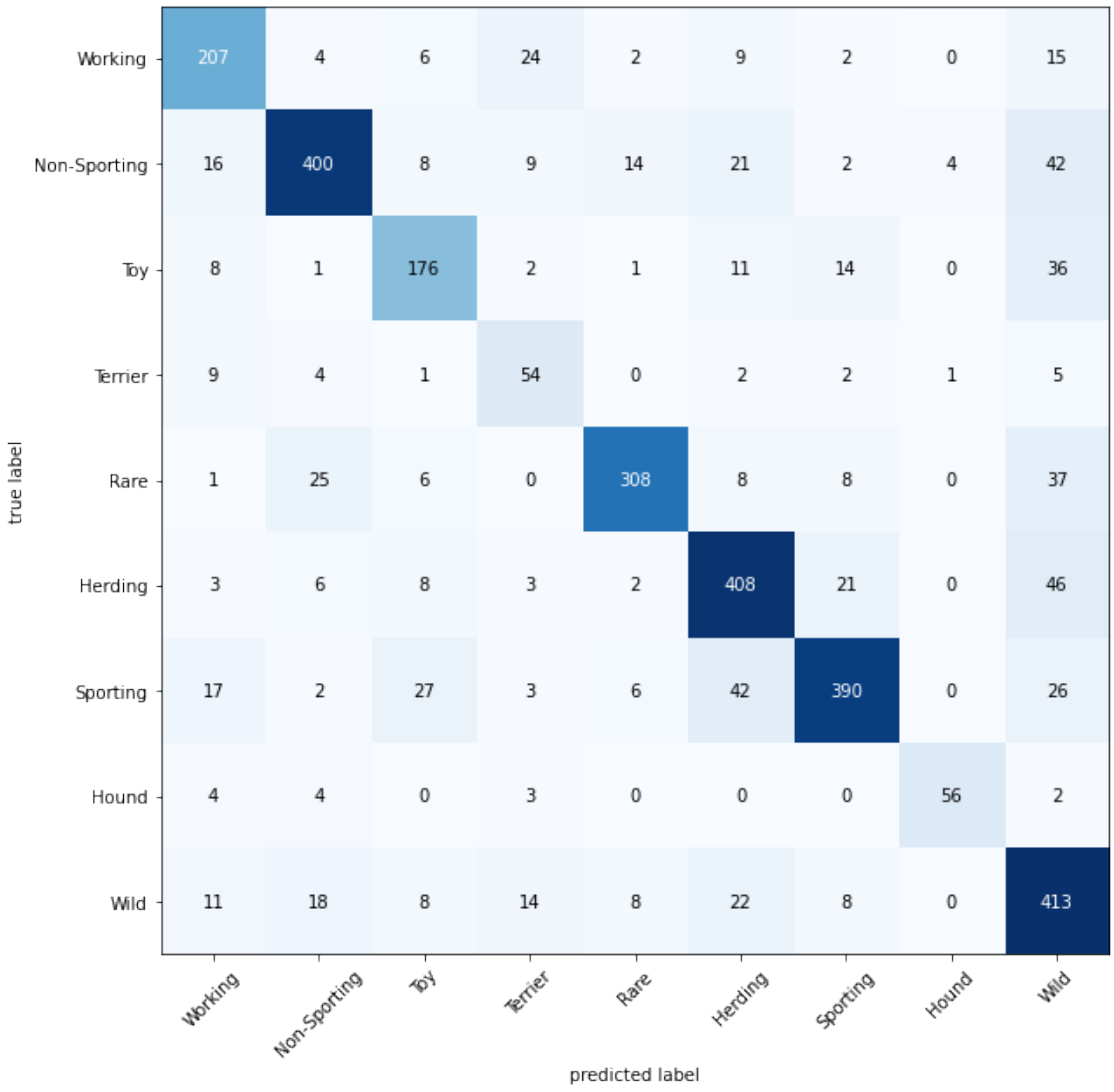
[36]:
```
target = list(image.group.unique())
```

[37]:
```
Y_pred = model.predict_generator(valid_generator)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
cf = confusion_matrix(valid_generator.classes, y_pred)
print(cf)
print('Classification Report')
target_names = target
print(classification_report(valid_generator.classes, y_pred,␣
 ↪target_names=target_names))
```

```
Confusion Matrix
[[207   4   6  24   2   9   2   0  15]
 [ 16 400   8   9  14  21   2   4  42]
 [  8   1 176   2   1  11  14   0  36]
 [  9   4   1  54   0   2   2   1   5]
 [  1  25   6   0 308   8   8   0  37]
 [  3   6   8   3   2 408  21   0  46]
 [ 17   2  27   3   6  42 390   0  26]
 [  4   4   0   3   0   0   0  56   2]
 [ 11  18   8  14   8  22   8   0 413]]
Classification Report
```

```
                precision    recall  f1-score   support

       Working       0.75      0.77      0.76       269
   Non-Sporting       0.86      0.78      0.82       516
           Toy       0.73      0.71      0.72       249
       Terrier       0.48      0.69      0.57        78
          Rare       0.90      0.78      0.84       393
       Herding       0.78      0.82      0.80       497
      Sporting       0.87      0.76      0.81       513
         Hound       0.92      0.81      0.86        69
          Wild       0.66      0.82      0.73       502

      accuracy                           0.78      3086
     macro avg       0.77      0.77      0.77      3086
  weighted avg       0.80      0.78      0.78      3086
```

[38]:
```
plot_confusion_matrix(conf_mat=cf,
                      class_names=target,
                      figsize=(10,10));
```



21 Making Prediction

We want to see how the prediction work.

The two images we are comparing are Chihuahua and Mexican Hairless, which both images look very similar even from a human perspective, so the model did make sense here.

Also, I included two images from my friend that she is going to adopt, which is a breed that did not appear in our dataset. It is impossible for our model to have the correct prediction. Also, base on the feedback, we can see that different angels taking the images will come out with a different result.

But from a business perspective, we can make the assumption that it is a mix, and the answer is a mix.

Answer: **Formosan Mountain Dog Terrier mix**

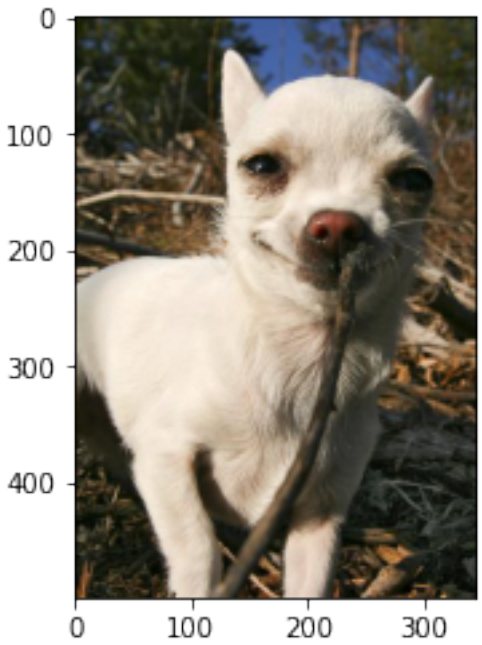
[87]:
```
from tensorflow.python.keras.preprocessing.image import load_img, img_to_array

image_size = 224
def read_and_prep_images(img_paths, img_height=image_size,␣
 ↪img_width=image_size):
    imgs = [load_img(img_path, target_size=(img_height, img_width)) for␣
 ↪img_path in img_paths]
    img_array = np.array([img_to_array(img) for img in imgs])
    return preprocess_input(img_array)
```

[89]:
```
image.head()
```

[89]:

	image_path	breed	group
0	n02108089-boxer/n02108089_11616.jpg	boxer	Working
0	n02108089-boxer/n02108089_9045.jpg	boxer	Working
0	n02108089-boxer/n02108089_9076.jpg	boxer	Working
0	n02108089-boxer/n02108089_3028.jpg	boxer	Working
0	n02108089-boxer/n02108089_3162.jpg	boxer	Working

[90]:
```
predict_and_show('/home/ec2-user/SageMaker/Images/n02085620-Chihuahua/
 ↪n02085620_1073.jpg')
```



[91]:
```
from keras.preprocessing.image import load_img, img_to_array
pic_dog = '/home/ec2-user/SageMaker/Images/n02085620-Chihuahua/n02085620_1073.
 ↪jpg'
pic_dog = load_img(pic_dog,target_size=(224,224))
pic_dog = img_to_array(pic_dog)
pic_dog = pic_dog/255
pic_dog = pic_dog.reshape(1,224,224,3)
result = model.predict(pic_dog)
# print(result)
count = 0
for i in result[0]:
    percent = '%.2f%%'%(i*100)
    print("{} -- probablity {}".format(files[count], percent))
    count += 1
```

```
n02108089-boxer -- probablity 0.01%
n02113799-standard_poodle -- probablity 0.03%
n02107683-Bernese_mountain_dog -- probablity 0.33%
n02094433-Yorkshire_terrier -- probablity 0.00%
n02096437-Dandie_Dinmont -- probablity 0.11%
n02113978-Mexican_hairless -- probablity 97.63%
n02106550-Rottweiler -- probablity 1.87%
n02108422-bull_mastiff -- probablity 0.00%
n02108551-Tibetan_mastiff -- probablity 0.01%

Using TensorFlow backend.
```

[92]:
```
hairless = image.loc[(image['breed'] == 'Mexican_hairless')]
```

[93]:
```
hairless
```

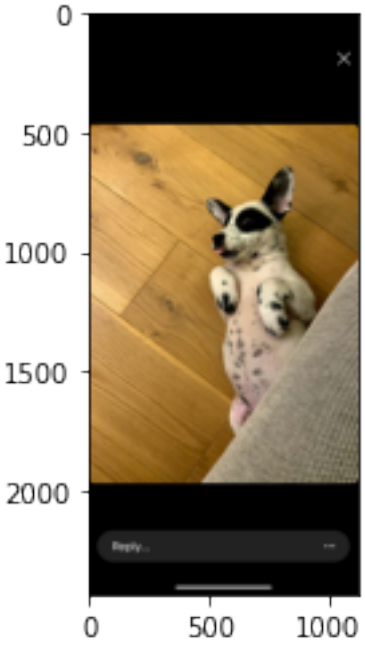
[93]:

	image_path	breed	group
0	n02113978-Mexican_hairless/n02113978_468.jpg	Mexican_hairless	Rare
0	n02113978-Mexican_hairless/n02113978_183.jpg	Mexican_hairless	Rare
0	n02113978-Mexican_hairless/n02113978_1823.jpg	Mexican_hairless	Rare
0	n02113978-Mexican_hairless/n02113978_1030.jpg	Mexican_hairless	Rare
0	n02113978-Mexican_hairless/n02113978_3419.jpg	Mexican_hairless	Rare
..	...	...	...
0	n02113978-Mexican_hairless/n02113978_1957.jpg	Mexican_hairless	Rare
0	n02113978-Mexican_hairless/n02113978_124.jpg	Mexican_hairless	Rare
0	n02113978-Mexican_hairless/n02113978_505.jpg	Mexican_hairless	Rare
0	n02113978-Mexican_hairless/n02113978_356.jpg	Mexican_hairless	Rare
0	n02113978-Mexican_hairless/n02113978_147.jpg	Mexican_hairless	Rare

[155 rows x 3 columns]

[154]:
```
predict_and_show('/home/ec2-user/SageMaker/Images/n02113978-Mexican_hairless/
 ↪n02113978_124.jpg')
```

[96]:
```
predict_and_show('/home/ec2-user/SageMaker/IMG_1061.jpeg')
```



[97]:
```
pic_dog = '/home/ec2-user/SageMaker/IMG_1061.jpeg'
pic_dog = load_img(pic_dog,target_size=(224,224))
pic_dog = img_to_array(pic_dog)
pic_dog = pic_dog/255
pic_dog = pic_dog.reshape(1,224,224,3)
result = model.predict(pic_dog)

count = 0
for i in result[0]:
    percent = '%.2f%%'%(i*100)
    print("{} -- probablity {}".format(files[count], percent))
    count += 1
```

```
n02108089-boxer -- probablity 0.95%
n02113799-standard_poodle -- probablity 0.00%
n02107683-Bernese_mountain_dog -- probablity 0.13%
n02094433-Yorkshire_terrier -- probablity 0.00%
n02096437-Dandie_Dinmont -- probablity 0.54%
n02113978-Mexican_hairless -- probablity 0.07%
n02106550-Rottweiler -- probablity 55.70%
n02108422-bull_mastiff -- probablity 0.00%
n02108551-Tibetan_mastiff -- probablity 42.60%
```

[98]:
```
predict_and_show('/home/ec2-user/SageMaker/IMG_5547.tiff')
```

[99]:
```
pic_dog = '/home/ec2-user/SageMaker/IMG_5547.tiff'
pic_dog = load_img(pic_dog,target_size=(224,224))
pic_dog = img_to_array(pic_dog)
pic_dog = pic_dog/255
pic_dog = pic_dog.reshape(1,224,224,3)
result = model.predict(pic_dog)

count = 0
for i in result[0]:
    percent = '%.2f%%'%(i*100)
    print("{} -- probablity {}".format(files[count], percent))
    count += 1
```

```
n02108089-boxer -- probablity 97.92%
n02113799-standard_poodle -- probablity 0.00%
n02107683-Bernese_mountain_dog -- probablity 0.00%
n02094433-Yorkshire_terrier -- probablity 0.00%
n02096437-Dandie_Dinmont -- probablity 0.00%
n02113978-Mexican_hairless -- probablity 0.00%
n02106550-Rottweiler -- probablity 0.00%
n02108422-bull_mastiff -- probablity 0.00%
n02108551-Tibetan_mastiff -- probablity 2.08%
```

[ ]: