Import libraries and load data

In [2]:

```
!pip install tensorflow
!pip install keras
```

```
Requirement already satisfied: tensorflow in c:\users\zhaklin\anaconda3\lib
\site-packages (2.5.0)
Requirement already satisfied: astunparse~=1.6.3 in c:\users\zhaklin\anacond
a3\lib\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: wrapt~=1.12.1 in c:\users\zhaklin\anaconda3\l
ib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: gast==0.4.0 in c:\users\zhaklin\anaconda3\lib
\site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: protobuf>=3.9.2 in c:\users\zhaklin\anaconda3
\lib\site-packages (from tensorflow) (3.17.0)
Requirement already satisfied: tensorflow-estimator<2.6.0,>=2.5.0rc0 in c:\u
sers\zhaklin\anaconda3\lib\site-packages (from tensorflow) (2.5.0)
Requirement already satisfied: wheel~=0.35 in c:\users\zhaklin\anaconda3\lib
\site-packages (from tensorflow) (0.35.1)
Requirement already satisfied: six~=1.15.0 in c:\users\zhaklin\anaconda3\lib
\site-packages (from tensorflow) (1.15.0)
Requirement already satisfied: keras-nightly~=2.5.0.dev in c:\users\zhaklin
\anaconda3\lib\site-packages (from tensorflow) (2.5.0.dev2021032900)
Requirement already satisfied: typing-extensions~=3.7.4 in c:\users\zhaklin
\anaconda3\lib\site-packages (from tensorflow) (3.7.4.3)
Requirement already satisfied: opt-einsum~=3.3.0 in c:\users\zhaklin\anacond
a3\lib\site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: h5py~=3.1.0 in c:\users\zhaklin\anaconda3\lib
\site-packages (from tensorflow) (3.1.0)
Requirement already satisfied: termcolor~=1.1.0 in c:\users\zhaklin\anaconda
3\lib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: numpy~=1.19.2 in c:\users\zhaklin\anaconda3\l
ib\site-packages (from tensorflow) (1.19.2)
Requirement already satisfied: flatbuffers~=1.12.0 in c:\users\zhaklin\anaco
nda3\lib\site-packages (from tensorflow) (1.12)
Requirement already satisfied: keras-preprocessing~=1.1.2 in c:\users\zhakli
n\anaconda3\lib\site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: absl-py~=0.10 in c:\users\zhaklin\anaconda3\l
ib\site-packages (from tensorflow) (0.12.0)
Requirement already satisfied: tensorboard~=2.5 in c:\users\zhaklin\anaconda
3\lib\site-packages (from tensorflow) (2.5.0)
Requirement already satisfied: grpcio~=1.34.0 in c:\users\zhaklin\anaconda3
\lib\site-packages (from tensorflow) (1.34.1)
Requirement already satisfied: google-pasta~=0.2 in c:\users\zhaklin\anacond
a3\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\users\zha
klin\anaconda3\lib\site-packages (from tensorboard~=2.5->tensorflow) (1.8.0)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\zhaklin\anaco
nda3\lib\site-packages (from tensorboard~=2.5->tensorflow) (2.24.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\u
sers\zhaklin\anaconda3\lib\site-packages (from tensorboard~=2.5->tensorflow)
(0.6.1)
Requirement already satisfied: werkzeug>=0.11.15 in c:\users\zhaklin\anacond
a3\lib\site-packages (from tensorboard~=2.5->tensorflow) (1.0.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\users
\zhaklin\anaconda3\lib\site-packages (from tensorboard~=2.5->tensorflow) (0.
4.4)
Requirement already satisfied: setuptools>=41.0.0 in c:\users\zhaklin\anacon
da3\lib\site-packages (from tensorboard~=2.5->tensorflow) (50.3.1.post202011
07)
Requirement already satisfied: google-auth<2,>=1.6.3 in c:\users\zhaklin\ana
```

```
conda3\lib\site-packages (from tensorboard~=2.5->tensorflow) (1.30.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\zhaklin\anaconda3
\lib\site-packages (from tensorboard~=2.5->tensorflow) (3.3.4)
Requirement already satisfied: idna<3,>=2.5 in c:\users\zhaklin\anaconda3\li
b\site-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflow) (2.
10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\zhaklin\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tens
orboard~=2.5->tensorflow) (1.25.11)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\zhaklin\anacon
da3\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflo
w) (2020.6.20)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\zhaklin\anacond
a3\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflo
w) (3.0.4)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\zhaklin
\anaconda3\lib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorb
oard~=2.5->tensorflow) (1.3.0)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in c:\users\zhaklin\an
aconda3\lib\site-packages (from google-auth<2,>=1.6.3->tensorboard~=2.5->ten
sorflow) (4.2.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\zhaklin\ana
conda3\lib\site-packages (from google-auth<2,>=1.6.3->tensorboard~=2.5->tens
orflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3.6" in
c:\users\zhaklin\anaconda3\lib\site-packages (from google-auth<2,>=1.6.3->te
nsorboard~=2.5->tensorflow) (4.7.2)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\zhaklin\anaconda3
\lib\site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,
>=0.4.1->tensorboard~=2.5->tensorflow) (3.1.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\zhaklin\anac
onda3\lib\site-packages (from pyasn1-modules>=0.2.1->google-auth<2,>=1.6.3->
tensorboard~=2.5->tensorflow) (0.4.8)
Requirement already satisfied: keras in c:\users\zhaklin\anaconda3\lib\site-
packages (2.4.3)
Requirement already satisfied: pyyaml in c:\users\zhaklin\anaconda3\lib\site
-packages (from keras) (5.3.1)
Requirement already satisfied: scipy>=0.14 in c:\users\zhaklin\anaconda3\lib
\site-packages (from keras) (1.5.2)
Requirement already satisfied: numpy>=1.9.1 in c:\users\zhaklin\anaconda3\li
b\site-packages (from keras) (1.19.2)
Requirement already satisfied: h5py in c:\users\zhaklin\anaconda3\lib\site-p
ackages (from keras) (3.1.0)
```

In [18]:

```
from sklearn.datasets import load_files
from keras.utils import np_utils
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
```

In [4]:

```
# define function to load train, test, and validation datasets
def load_dataset(path):
    data = load_files(path)
    dog_files = np.array(data['filenames'])
    dog_targets = np_utils.to_categorical(np.array(data['target']), 133)
    return dog_files, dog_targets
```

In [5]:

```
# load train, test, and validation datasets
train_files, train_targets = load_dataset('./data/train')
valid_files, valid_targets = load_dataset('./data/valid')
test_files, test_targets = load_dataset('./data/test')
```

In [6]:

```
from keras.preprocessing import image
from tqdm import tqdm

def path_to_tensor(img_path):
    # loads RGB image as PIL.Image.Image type
    img = image.load_img(img_path, target_size=(224, 224))
    # convert PIL.Image.Image type to 3D tensor with shape (224, 224, 3)
    x = image.img_to_array(img)
    # convert 3D tensor to 4D tensor with shape (1, 224, 224, 3) and return 4D tensor
    return np.expand_dims(x, axis=0)
def paths_to_tensor(img_paths):
    list_of_tensors = [path_to_tensor(img_path) for img_path in tqdm(img_paths)]
    return np.vstack(list_of_tensors)
```

Pre-process data

In [7]:

```
from PIL import ImageFile

ImageFile.LOAD_TRUNCATED_IMAGES = True

# pre-process the data for Keras
train_tensors = paths_to_tensor(train_files).astype('float32')/255
valid_tensors = paths_to_tensor(valid_files).astype('float32')/255
test_tensors = paths_to_tensor(test_files).astype('float32')/255
```

```
100%|██████████████████████████████████████████████████████████████████████████
██████| 6680/6680 [00:41<00:00, 159.74it/s]
100%|██████████████████████████████████████████████████████████████████████████
████████| 835/835 [00:04<00:00, 187.75it/s]
100%|██████████████████████████████████████████████████████████████████████████
████████| 836/836 [00:04<00:00, 194.72it/s]
```

In [8]:

```
print(len(train_files))
```

```
6680
```

Modelling

A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptrons, a machine learning unit algorithm, for supervised learning, to analyze data. CNNs apply to image processing, natural language processing and other kinds of cognitive tasks. That is why I will be using it in this project for identification of the breed through the images of dogs.

In [9]:

```
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Dropout, Flatten, Dense
from keras.models import Sequential
```

In [10]:

```
model = Sequential() #the model uses one after another
model.add(Conv2D(filters=16, kernel_size=2, padding='same',activation='relu',input_shape=(2
model.add(MaxPooling2D())
model.add(Conv2D(filters=32, kernel_size=2, padding='same',activation='relu'))
model.add(MaxPooling2D()) # MaxPooling2D is a pooling operation that calculates the maximum
model.add(Conv2D(filters=64, kernel_size=2, padding='same',activation='relu'))
model.add(MaxPooling2D())
model.add(GlobalAveragePooling2D())
model.add(Dense(133,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 16)      208
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 32)      2080
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 64)        8256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
global_average_pooling2d (Gl (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 133)               8645
=================================================================
Total params: 19,189
Trainable params: 19,189
Non-trainable params: 0
_________________________________________________________________
```

In [11]:

```
#Compile and train the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

In [12]:

```
from keras.callbacks import ModelCheckpoint

epochs = 5 #one pass over the entire dataset
checkpointer = ModelCheckpoint(filepath='saved_models/weights.best.from_scratch.hdf5',
                               verbose=1, save_best_only=True)
model.fit(train_tensors, train_targets,
          validation_data=(valid_tensors, valid_targets),
          epochs=epochs, batch_size=20, callbacks=[checkpointer], verbose=1)
```

```
Epoch 1/5
334/334 [==============================] - 71s 180ms/step - loss: 4.8859 - a
ccuracy: 0.0099 - val_loss: 4.8695 - val_accuracy: 0.0108

Epoch 00001: val_loss improved from inf to 4.86948, saving model to saved_mo
dels\weights.best.from_scratch.hdf5
Epoch 2/5
334/334 [==============================] - 62s 184ms/step - loss: 4.8659 - a
ccuracy: 0.0109 - val_loss: 4.8555 - val_accuracy: 0.0120

Epoch 00002: val_loss improved from 4.86948 to 4.85554, saving model to save
d_models\weights.best.from_scratch.hdf5
Epoch 3/5
334/334 [==============================] - 61s 183ms/step - loss: 4.8464 - a
ccuracy: 0.0137 - val_loss: 4.8291 - val_accuracy: 0.0192

Epoch 00003: val_loss improved from 4.85554 to 4.82907, saving model to save
d_models\weights.best.from_scratch.hdf5
Epoch 4/5
334/334 [==============================] - 62s 185ms/step - loss: 4.8131 - a
ccuracy: 0.0201 - val_loss: 4.7999 - val_accuracy: 0.0204

Epoch 00004: val_loss improved from 4.82907 to 4.79995, saving model to save
d_models\weights.best.from_scratch.hdf5
Epoch 5/5
334/334 [==============================] - 62s 184ms/step - loss: 4.7774 - a
ccuracy: 0.0197 - val_loss: 4.7822 - val_accuracy: 0.0251

Epoch 00005: val_loss improved from 4.79995 to 4.78225, saving model to save
d_models\weights.best.from_scratch.hdf5
```

Out[12]:

```
<keras.callbacks.History at 0x2cba8efd8b0>
```

Model using transfer learning

Using bottleneck features to convert the input X (image of size 224 x 224 x 3, for example) into the output Y with size 512 x 7 x 7.

In [ ]:

```
import requests

url = 'https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogResnet50Data.npz'
r = requests.get(url)
```

In [ ]:

```
with open('DogResnet50Data.npz', 'wb') as f:
    f.write(r.content)

bottleneck_features = np.load('DogResnet50Data.npz')
train_Resnet50 = bottleneck_features['train']
valid_Resnet50 = bottleneck_features['valid']
test_Resnet50 = bottleneck_features['test']
```

In [15]:

```
#Adding dropout to prevent the model from overfitting
Resnet50_model = Sequential()
Resnet50_model.add(GlobalAveragePooling2D(input_shape=train_Resnet50.shape[1:]))
Resnet50_model.add(Dropout(0.3))
Resnet50_model.add(Dense(1024,activation='relu'))
Resnet50_model.add(Dropout(0.4))
Resnet50_model.add(Dense(133, activation='softmax'))

Resnet50_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 133)               136325
=================================================================
Total params: 2,234,501
Trainable params: 2,234,501
Non-trainable params: 0
_________________________________________________________________
```

test accuracy of 80.8612%

Conclusion

I used CNN and then applied transfer learning with the intention to improve the model's accuracy from 3.2% to 81% because of the small number of train images.

In [ ]:

```
```