IMAGE CLASSIFICATION USING CNN (KERAS/TENSORFLOW)

Final Report - 24th Feb 2018

By: Subir Das

Objective, Clients and Goal

Objective: To assign a human identifiable label say alphabets, book, cat, human or dog etc. to an Image. If Computer Vision can attain human accuracy, then practical applications are limitless.

In this project we use computer vision and machine learning techniques to predict dog breeds from images. An image classification model taken an image and assigns a probability to a label. An image is a 3-dimensional array such as X (pixels width) * Y (pixels height) and three color channels (RBG) thus X * Y * 3. Please note that each number in pixel has a range from 0 (presence of light i.e. black) to 255 (all colors white light). In Computer terms – Computer has to assign a human identifiable label to this large 3D array.

Clients:

The client for this project is kaggle.com(https://www.kaggle.com/c/dog-breed-identification)

Goal:

Is to create a 1st generation classifier for image processing. Practical applications of such a model are limitless. It can used in manufacturing, healthcare to life science domain to reduce cycle time, to improve patient care, self driving cars and planes and lip reading etc.

Data Source

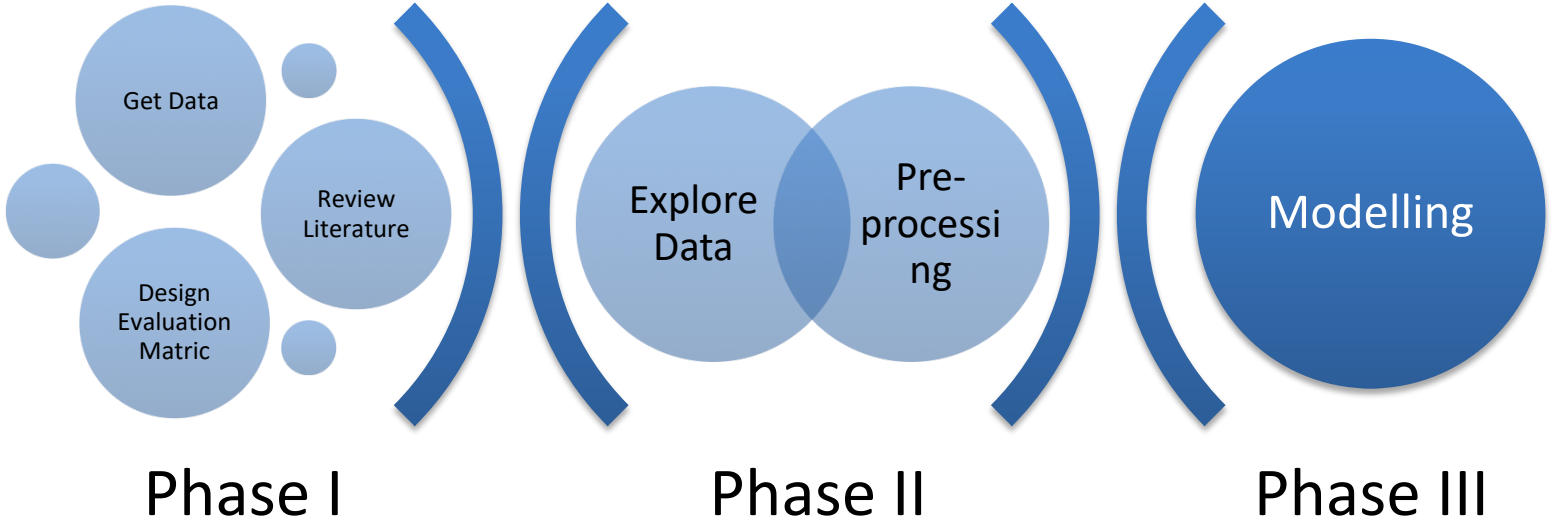
The data for this project was collected from internet sources namely Kaggle, Stanford and ImageNet. The Stanford Dogs dataset contains images of 120 breeds of dogs from around the world. This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization

Data Sources used:
Kaggle: https://www.kaggle.com/c/dog-breed-identification
Stanford: http://vision.stanford.edu/aditya86/ImageNetDogs/
ImageNet: http://image-net.org/download-API

Solution Approach

The solution plans to use CNN to help with image selection and dog breed identification. A deep learning algorithm will be developed using Keras with Tensorflow as the backend running in GCP. It will be trained with the training and validation dataset. Specifically, a CNN will be implemented in Keras and will be optimized to minimize multi-class logarithmic loss. Predictions will be made on the test data set and will be evaluated.

It is specifically designed to address large dataset with biodiversity and quality issues like image quality, multiple faces, missing key features etc. The solution is sub-divided into three phases as listed below.



Solution Approach…

1. Data Assembly - Phase I: This phase of the project is designed to gather and do basic cleanup to create dataset.

2. Explore and Preprocessing – Phase II: This phase of the project is designed to validate and explore the dataset for all the problems listed in the “Challenges” section of the proposal.

3. Modelling and Evaluation Phase III: In this phase of the project I will be using CNN from using from scratch and transfer learning to find the best validation loss model to classify the image.

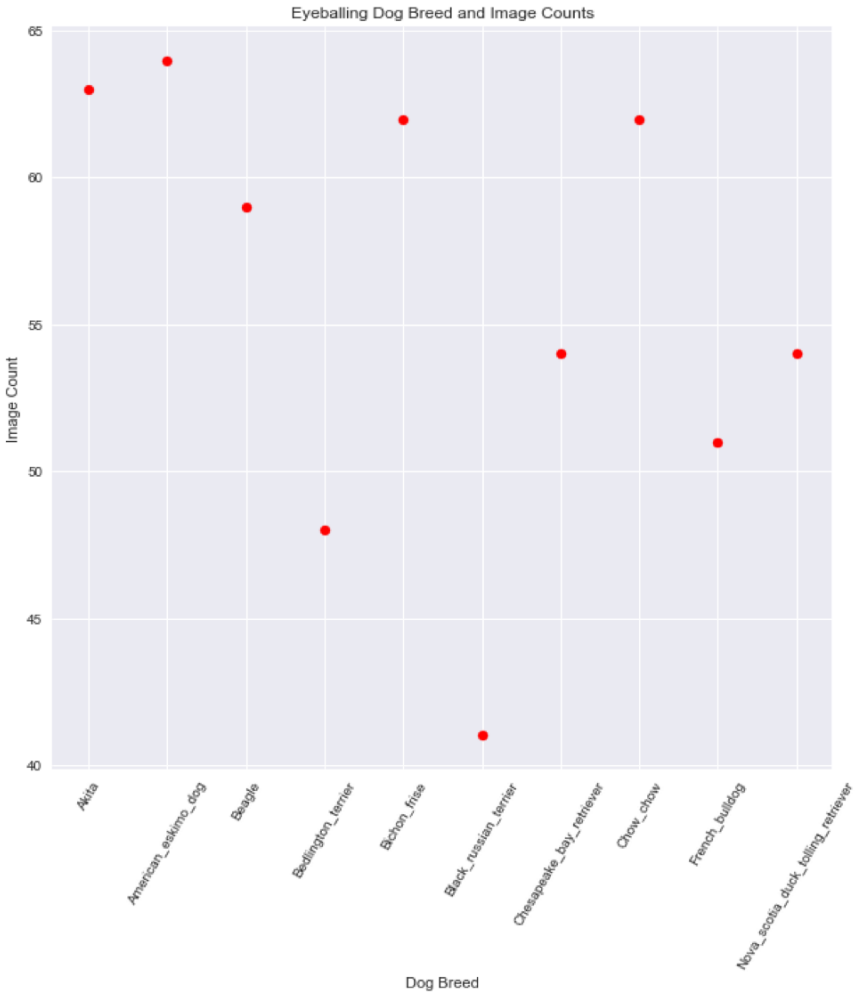
Technology Stack: GCP Storage, Google Cloud Platform, Python, Atom
Configuration Management System: GitHub
Testing and Traceability: IPython Notebook

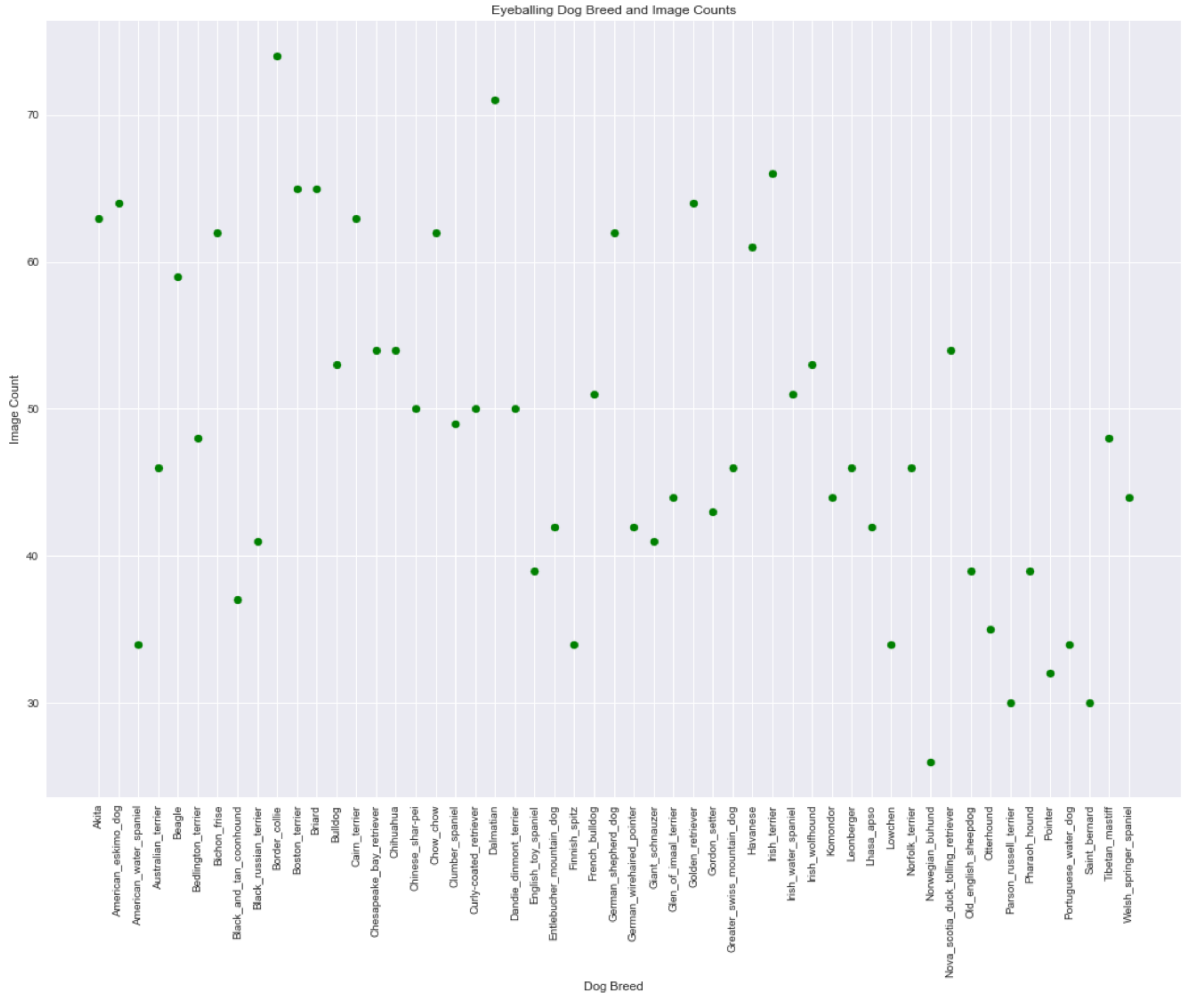
Data Wrangling – Phase I

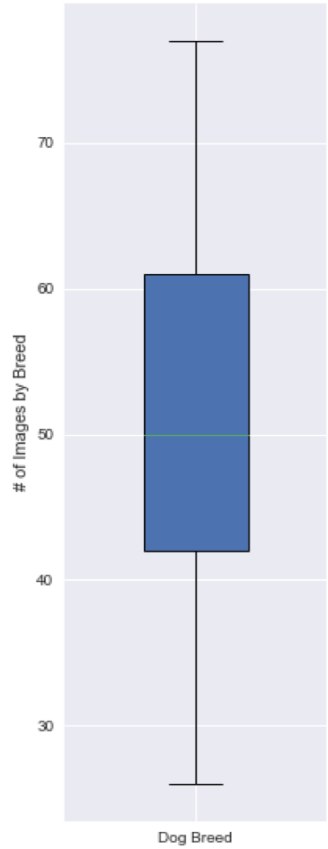
Final Dataset:

The final dataset was divided into three categories train, valid and test. The images were not altered so that neural network can learn better.

Data Exploration – Phase II – Data Diversity

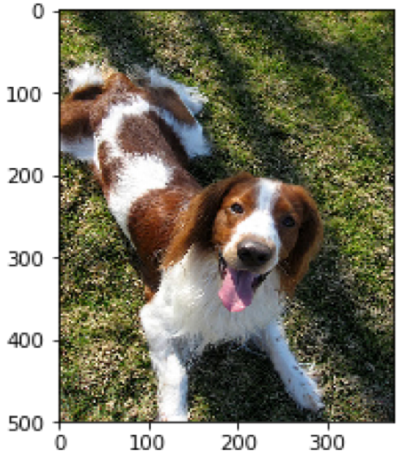




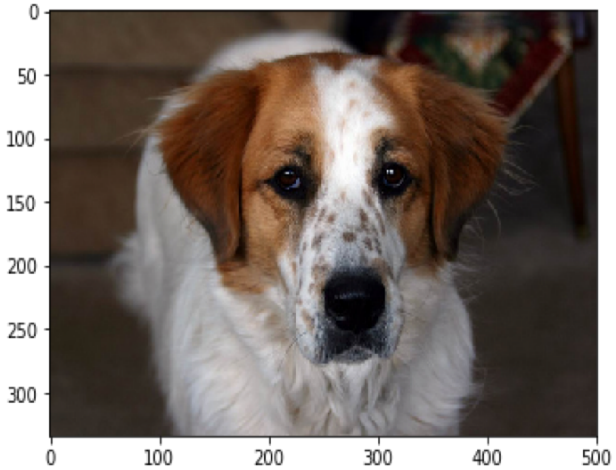


Based on initial data review, it appears that we good amount of images/Breed to train our CNN model.

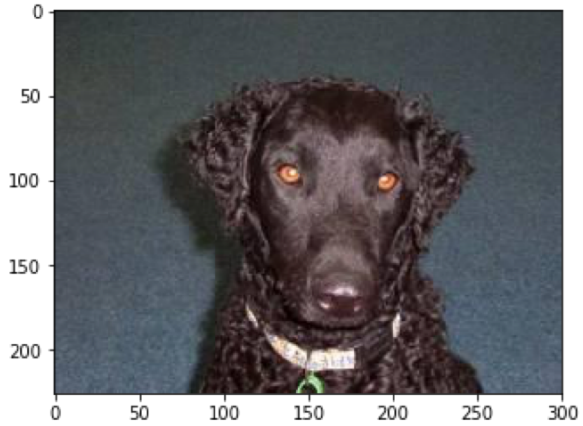
Data Exploration – Breed Identification Challenges



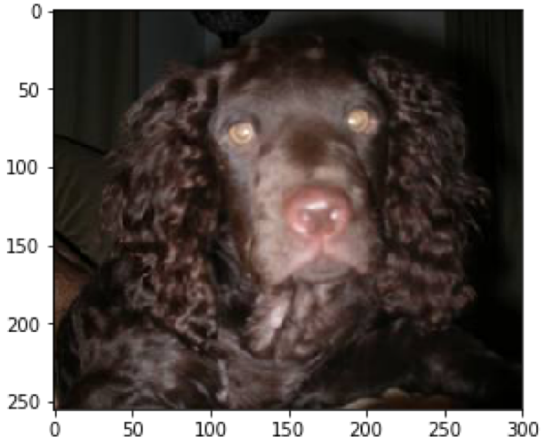
Brittany



Welsh Springer Spaniel



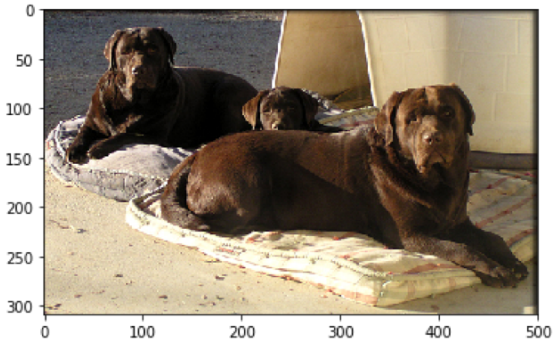
Curly Coated Retriever

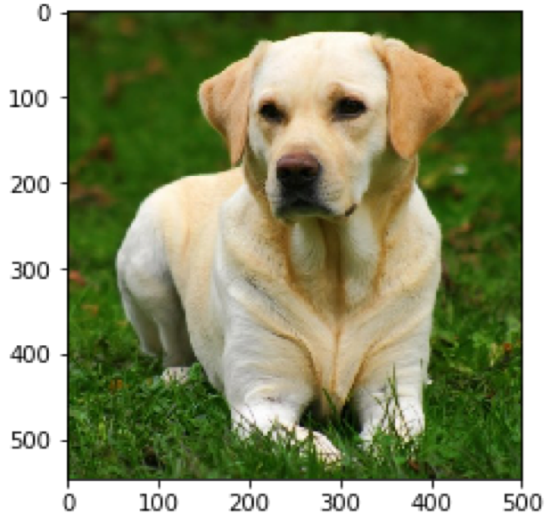


American Water Spaniel

Observations:

Identification of dog breeds is a difficult as there are relatively few differences between breeds and relatively large differences within breeds ex. size, shape, and color. Dogs are both the most morphologically and genetically diverse species on Earth. The difficulties of identifying breeds because of diversity are compounded by the stylistic differences of photographs used in the dataset, which features dogs of the same breed in a variety of lightings and positions.





Different Types of Labradors

Modeling – Phase III – CNN Keras w/ tensorflow

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 224, 224, 16)	208
max_pooling2d_2 (MaxPooling2	(None, 112, 112, 16)	0
conv2d_2 (Conv2D)	(None, 112, 112, 32)	2080
max_pooling2d_3 (MaxPooling2	(None, 56, 56, 32)	0
conv2d_3 (Conv2D)	(None, 56, 56, 64)	8256
max_pooling2d_4 (MaxPooling2	(None, 28, 28, 64)	0
dropout_1 (Dropout)	(None, 28, 28, 64)	0
flatten_2 (Flatten)	(None, 50176)	0
dense_1 (Dense)	(None, 500)	25088500
dropout_2 (Dropout)	(None, 500)	0
dense_2 (Dense)	(None, 133)	66633

Total params: 25,165,677
Trainable params: 25,165,677
Non-trainable params: 0

CNN: Keras CNNs with TensorFlow need a 4D tensor as input, with shape as (samples,rows,columns,channels) where samples represents total number of images and rows, columns, and channels(RBG) are number of rows, columns, and channels for each image. We take color image file path convert that into a 4D tensor suitable for supplying to a Keras CNN.

```
from keras.preprocessing import image
from tqdm import tqdm
# loads RGB image as PIL.Image.Image type
img = image.load_img(file, 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
np.expand_dims(x, axis=0)
```

Layer (type)	Output Shape	Param #
global_average_pooling2d_1 (	(None, 512)	0
dense_3 (Dense)	(None, 133)	68229

Total params: 68,229
Trainable params: 68,229
Non-trainable params: 0

Model Architecture 2 - Using Pre-Trained VGG-16 model.

The model 2 uses the the pre-trained VGG-16 model as a fixed feature extractor, where the last convolutional output of VGG-16 is fed as input to our model. We only add a global average pooling layer and a fully connected layer, where the latter contains one node for each driver category and is equipped with a softmax.

Modeling – Phase III – CNN Keras w/ tensorflow

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_2 ( (None, 2048)              0
_________________________________________________________________
dense_4 (Dense)              (None, 133)               272517
=================================================================
Total params: 272,517
Trainable params: 272,517
Non-trainable params: 0
_________________________________________________________________
```

Modeling using Inception V3 with bottleneck features to improve the accuracy:

We are going to use the bottleneck features from a different pre-trained model. To make things easier, we have pre-computed the features for all of the networks that are currently available in Keras: Inception bottleneck features. The files are encoded as such: DogInceptionV3Data.npz download the corresponding bottleneck features, and store it in the folder.

Checking Model Accuracy

In [40]:
```
InceptionV3_predictions = [np.argmax(InceptionV3_model.predict(np.expand_dims(feature, axis=0)))
                           for feature in test_InceptionV3]

# report test accuracy
test_accuracy = 100*np.sum(np.array(InceptionV3_predictions)==
                           np.argmax(y_test_targets, axis=1))/len(InceptionV3_predictions)
print('\nTest accuracy: %.4f%%' % test_accuracy)
```

```
Test accuracy: 83.6124%
```

Model Accuracy:

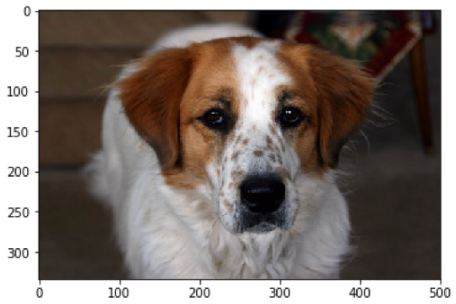
Finally we can see that Model accuracy has improved from 7% (initial) to 84% after using pre-trained with bottleneck features.

Testing Boundary Conditions – Challenges

Predict Images using the fine tuned model

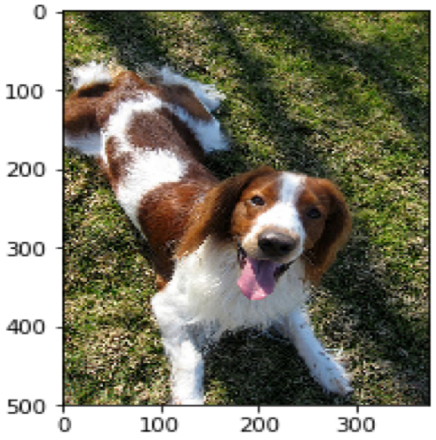
```
In [43]: image_prediction('Images/train/037.Brittany/Brittany_02592.jpg')
```

```
Hello, Human's Best Friend!
The breed of dog is a Brittany
```




```
In [45]: image_prediction("Images/train/130.Welsh_springer_spaniel/Welsh_springer_spaniel_08195.jpg")
```

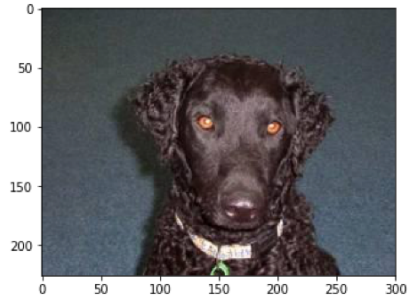
```
Hello, Human's Best Friend!
The breed of dog is a Welsh_springer_spaniel
```



Testing Boundary Conditions – Challenges

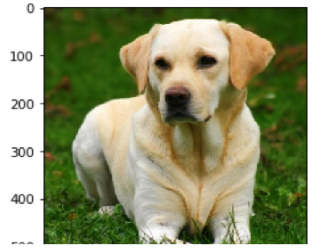
```
In [46]: image_prediction("Images/train/055.Curly-coated_retriever/Curly-coated_retriever_03860.jpg")

Hello, Human's Best Friend!
The breed of dog is a Curly-coated_retriever
```



```
In [48]: image_prediction("Images/train/096.Labrador_retriever/Labrador_retriever_06457.jpg")

Hello, Human's Best Friend!
The breed of dog is a Labrador_retriever
```

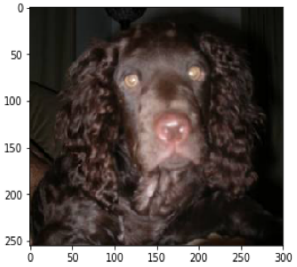


```
In [50]: image_prediction("Images/train/096.Labrador_retriever/Labrador_retriever_06492.jpg")

Hello, Human's Best Friend!
The breed of dog is a Labrador_retriever
```

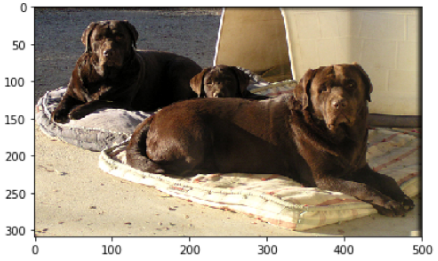
```
In [47]: image_prediction("Images/train/009.American_water_spaniel/American_water_spaniel_00623.jpg")

Hello, Human's Best Friend!
The breed of dog is a American_water_spaniel
```



```
In [51]: image_prediction("Images/train/096.Labrador_retriever/Labrador_retriever_06486.jpg")

Hello, Human's Best Friend!
The breed of dog is a Labrador_retriever
```



Summary and Next Steps

Summary:

The classification model was developed to identify Dog Breed using Deep Learning techniques like CNN(Keras with Tensorflow as the backend). We used transfer learning with bottleneck features with different pre-trained models to improve our accuracy score from initial model accuracy of 5%.

- In-order for further improve the loss metric, pre-trained model such as VGG16 was used and transfer learning is applied
- Technique such as “Using the bottleneck features of a pre-trained network” is applied to VGG16 for two types of model architectures as the top layers
- Bottleneck features of a pre-trained network for VGG16 did not result in any improvement of loss metric.
- Finally, InceptionV3 model with Bottleneck was applied to Model Architecture 3 as the top layer. This resulted in further improvement of loss metric.

Next Steps:

The accuracy score should be further improved to 95-98% that is close to human accuracy level by increasing the dataset variety and size –

a) Consider using bigger size 640x480
b) Try using the different pre-trained models - ResNet-50, Xception
c) Model Ensembles - Image Segmentation algorithms such as R-CNN, Fast R-CNN, Faster R-CNN and Mask R-CNN can also be investigated.