Dog Breed Classifier using Convolutional Neural Networks

Middi Venkata Sai Rishita
Department of Electronics and Communication
RNS Institute of Technology
Bangalore,India
rishimiddi@gmail.com

Tanvir Ahmed Harris
Department of Electrical Engineering
IIT Bombay
Mumbai,India
tanvirahmed.harris@gmail.com

Abstract—**In the present world, we have wide varieties of species and organisms. This brings into light, the criticality of classification of various Tangible objects. Also, keeping in mind, the ongoing research on genetics and evolution by various scientists across the world, discerning the resemblance among different classes also becomes very crucial. This paper is based on a project which builds a CNN (Convolutional Neural Network) to classify different dog breeds. If the image of a dog is found, this algorithm would find the estimate of the breed. The resembling dog breed is identified if the image of a human is supplied. We have built a pipeline to process real-world images.**

Keywords—pipeline, Processing, models, user experience, implementation, breed, detector

I. Introduction

The ultimate aim of this project is to explore the Convolutional Neural Network models for classification. We explore the different challenges involved in aligning multiple models designed to perform distinguished tasks in a data processing pipeline. The algorithm that we have built could be used as a part of a mobile application or a web application. The CNN to classify Dog breeds is created from scratch and by using transfer learning as well. The accuracy attained from the two methods is compared. The entire paper is divided into eight sections as follows: -

(1) Import the dataset which is the user supplied images.

(2) Detect the humans from the provided images.

(3) Detect the dogs from the provided images

(4)Build a Convolutional Neural Network in order to classify the breeds of dogs from the start.

(5)Employ a Convolutional Neural Network for classification of the breeds of dogs by making use of transfer learning.

(6) Build our own Convolutional Neural Network to do the classification.

(7) Write the devised algorithm.

(8) Results and conclusion

II. RELATED TOPICS

A. CNNs

Convolutional Neural Networks constitute neurons with certain learnable weights and biases. Every neuron receives some inputs, performs a dot product. What distinguishes them from the other neural networks is that the convolutional neural network architectures, make an assumption that the inputs which are fed are indeed images. This aids in instilling certain features into the architecture. They reduce the number of measurable factors that define a network. The neurons in a single layer function are independent of previous layers.

B. Transfer Learning

It is very gruelling to train an entire convolutional Network from scratch. It is arduous to get a dataset of ample expanse. Therefore, we train a network prior itself, on a huge dataset. There are three major Transfer Learning scenarios: - Use a ConvNet as fixed feature extractor i.e as CNN codes, Fine Tune the ConvNet, Pretrained models.

The next point to vacillate on is to decide on the type of transfer learning we use, to operate on a dataset. This largely depends on the expanse of the new dataset. It also depends on its resemblance to the novel one. Navigating through the scenarios we get: -

(1)If we have a dataset that is small and similar to the authentic dataset, then, it isn't advisable to fine-tune the ConvNet. This is due to the overfit issue. Therefore, we train a linear classifier on the convolutional neural network codes.

(2)If we have a dataset that is large but similar to the original one, then we can fine-tune through the full network.

(3)If the new dataset is miniscule, however easily distinguishable from original one, then we could train a linear classifier but not from the top of the network. Instead, train a Support Vector Machine (SVM) classifier from initial stages of the network.

(4) If the new dataset is humongous and quite distinguishable from the authentic one, then we would have to train the network from scratch by initializing the weights from a pretrained model.

The constraints from pretrained models and Learning rates also have to be kept in mind while performing transfer learning.

III. PART 1:IMPORT DATASETS

In this section, it is required to load the dataset of various images of dogs. We employ the use of load_files function from the scikit-learn library for the presence of certain variables. These include numpy arrays that contain file paths to the images, arrays that contain labels encoded for classification .It also contains a list of breed names of dogs which are of the string datatype.

First, define a function to load train, test and to validate datasets. After that, load the list of dog names. Also print the statistics about dataset. We have used onehot-encoded classification labels.

```
There are 133 total dog categories.
There are 8351 total dog images.

There are 6680 training dog images.
There are 835 validation dog images.
There are 836 test dog images.
```

Figure 1: Statistics about dog dataset

Next, we could import the images of humans (Human Dataset). This time we load the filenames in shuffled human dataset and print the statistics about the dataset. A total of **13233** human images were identified.

IV. PART 2: DETECT HUMANS

OpenCV provides a multitude of previously trained face detectors. In this work we witness the implementation of Haar feature-based cascade classifiers.

It is an approach where a cascade function is trained from a lot of positive and negative images. We have extracted the detectors from the haarcascades directory. The next task is to load the colour (BGR) image. Convert this BGR image into a grayscale image by fixing the grey levels. The detectMultiScale function takes the grayscale image as a parameter. Now, find the faces in the images loaded and print the number of faces detected.

We have added a bounding box to the colour image. For the purpose of plotting, convert the BGR image into RGB.

```
Number of faces detected: 1
```

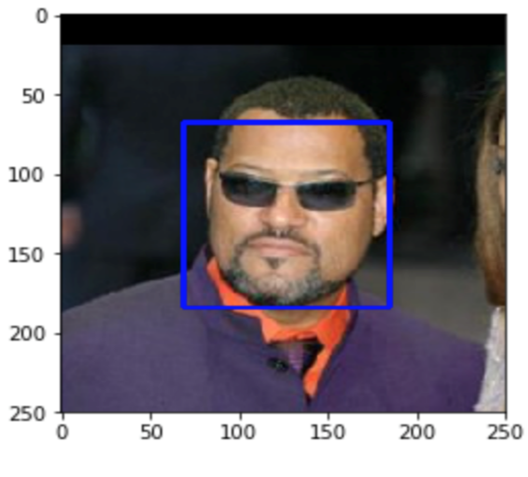


Figure 2: Display of the image with bounding box

The numpy array which we have used to detect faces is "faces". Every detected face is considered to be a one-dimensional array with four parameters. Two of them specify the vertical and horizontal coordinates of the left corner at the top of the box. Other two parameters determine the height and width of the box.

A. Human Face Detector

Make a function that returns 1 only if a human face is detected in the image, else it returns 0. 1 indicates True and 0 indicates False The function takes the file path to the image in string valued form as input.

Assess the performance of the Human Face Detector by determining the percentage of first 100 images in the file that has a detected human face. Also determine the percentage of the first 100 images in the dog file that has a detected human face. From each of the dataset, Extract the paths to the files for 1st 100 images.

```
% faces detected in human_files_short: 99.00%
% faces detected in dogs_files_short: 11.00%
```

Figure 3:Output of performance testing

OpenCV consists of Face Detector is an excellent way for detection of human images in the algorithm above. The report performance on each of the datasets is obtained.

```
% faces detected in human_files_short: 100.00%
% faces detected in dogs_files_short: 10.00%
```

Figure 4: Output after using face detector from OpenCV

V. PART 3: DETECT DOGS

In order to detect the various dogs in images, a pre-trained ResNet-50 model is used. Also, ImageNet is a well-known dataset used classifying the images and tasks apart from that, related to vision. So, if given an image the ResNet-50 model returns a prediction. The program downloads the model. The downloaded models consists of weights which have gotten trained on ImageNet.

A. Pre-process the Data

Since TensorFlow is being used as the backend and Keras is being used for building the convolutional neural network, it is required to provide a 4-dimensional tensor as the input. The shape of the input would be of the form,

(nb_samples, rows, columns, channels)
nb_samples represents total number of images

A function named path_to_tensor is defined. The input to this function is a file path(string-valued) and it returns a 4D tensor. This particular 4D array is suitable to apply to the CNN of Keras. The function initially uploads the picture and alters the image to squared shape of 50,176 pixels. This square image is converted to an array and then reshaped to a 4D sensor. If the processing is on a single image, then the returned tensor would have the shape (1, 224, 224, 3). If multiple images are present, then 1 is replaced by the number of samples/images.

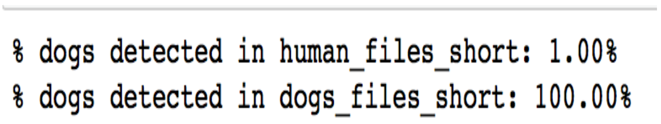
B. Making Predictions with ResNet-50

Additional processing has to be done on 4D tensor.

The RGB image is initially transformed to BGR with the aid of re ordering of channels. All previously trained models undergo normalization. The mean pixel is subtracted from every other pixel in each of the images. After the steps mentioned above, the model could be used to extract the predictions.

The 'predict' method returns an array. The j-th term of that particular array is the anticipated probability of the model, that it belongs to the corresponding category on ImageNet. The maximum argument of the probability vector that has been predicted, the integer analogous to the predicted object class of the model is obtained out of an identified object category from a dictionary containing 1000 images from ImageNet.

In the dictionary the breeds of dogs appear from keys 151-268 (both inclusive) specifically from 'Chihuahua' to 'Mexican hairless'. The dog detector function returns True if a dog is detected, False otherwise. Lastly, Assess the performance by determining the percentage of dog detected in human files.



```
% dogs detected in human_files_short: 1.00%
% dogs detected in dogs_files_short: 100.00%
```

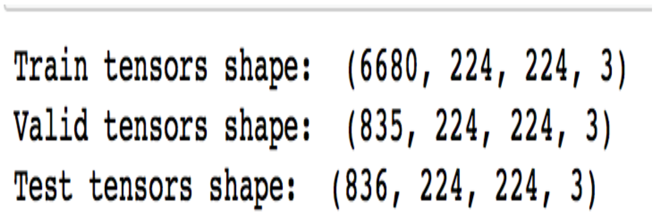
Figure 5:Performance of dog detector

VI. PART 4 : CREATE A CNN TO CLASSIFY DOG BREEDS (FROM SCRATCH)

Multiple challenges come up when the attempt to assign the breed of the dogs. To mention a few : -

- The breed Brittany and Welsh Springer are difficult to distinguish even for humans.
- Breeds like Labradors are present in black, yellow and chocolate. The algorithm must decipher this high intra-class variation.
- Occurrence of a correct answer through random chance is exceptionally low.

Pre-processing of data is required. Re-scaling of the images is done. Dividing each pixel by 255 in all the images. Print the tensors shapes.



```
Train tensors shape:  (6680, 224, 224, 3)
Valid tensors shape:  (835, 224, 224, 3)
Test tensors shape:  (836, 224, 224, 3)
```

Figure 6:Output after pre-processing the data

A. Model architecture

The depth of a CNN has to be given utmost importance while designing the model architecture in order to get a god recognition system. This is due to the high levels of hierarchy in the structure of images. In order to augment the capacity of the network to learn the complex features, choose a larger network in terms of number and size of layers.

Regular ReLU computes a function defined by:

$$f(x)=\max(0,x)$$

The pros of the function are that it expedites the convergence of stochastic gradient descent compared to sigmoid/tanh functions and the implementation can be easily done by fixing the threshold of activations matrix to 0.

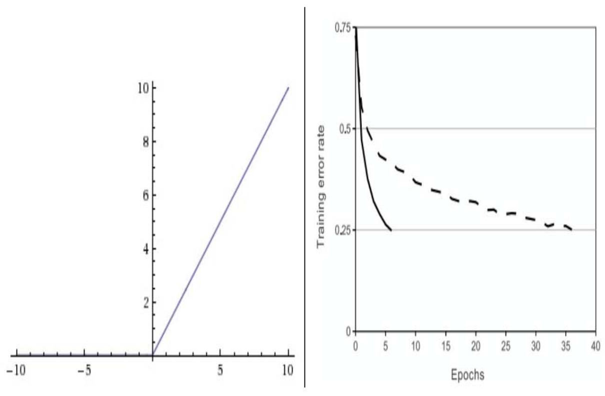


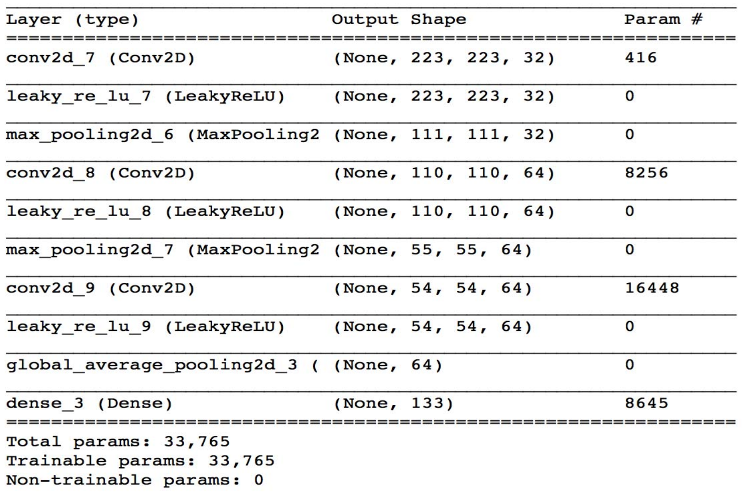
Figure 7:Left: Rectified Linear unit activation function, Right: 6x improvement in convergence compared to tanh function.

Unfortunately, ReLU units are quite fragile during and may never activate again at a datapoint if they "die".

Therefore, Leaky ReLU is used. It is defined as:

$$f(x)= 1\ (x<0)(\alpha x) + 1\ (x>=0)\ (x)$$

The function entirely doesn't become zero for x<0, instead it has a small negative slope. α has the value 0.3 as default in Keras.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 223, 223, 32)      416
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 223, 223, 32)      0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 110, 110, 64)      8256
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 110, 110, 64)      0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 55, 55, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 54, 54, 64)        16448
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 54, 54, 64)        0
_________________________________________________________________
global_average_pooling2d_3 ( (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 133)               8645
=================================================================
Total params: 33,765
Trainable params: 33,765
Non-trainable params: 0
```

Figure 8:Output after implementation of model architecture

Train the model on 6680 samples, validate on 835 samples with 100 epochs. This induces model checkpointing to get the model with best validation loss. Load the model with best validation loss by indicating the file path(string-valued).

Test the model using the test dataset of dog images.

- Get the index of predicted dog breed for each image in test dataset.
- Report the test accuracy.

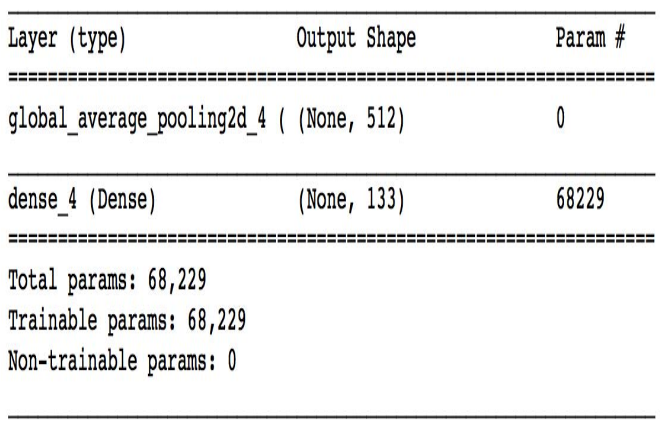
The test accuracy was found to be 10.8852%.

VII. PART V: USE CNN TO CLASSIFY DOG BREEDS

Initially obtain the bottleneck features.

A. Model Architecture

This needs to use a fixed feature extractor. Thus, pre-trained VGG-16 model is used where its last convolutional output is given as input to the present one. Addition of (i) Global average pooling layer (ii) Fully connected layer is done. The fully connected layer contains 1 node for each and every dog category. The activation is softmax.



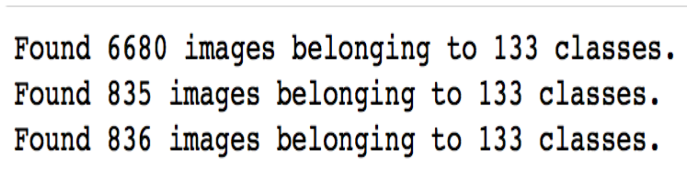
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_4 ( (None, 512)               0

_________________________________________________________________
dense_4 (Dense)              (None, 133)               68229
=================================================================
Total params: 68,229
Trainable params: 68,229
Non-trainable params: 0

_________________________________________________________________
```

Figure 9:Output after model architecture

Compile and train the model with data checkpointing. The number of epochs taken are 20. The model with best validation loss is loaded. The test accuracy is found to be 36.6029%.
Return dog breed predicted by the model.

VIII. PART 6: CREATE A CNN TO CLASSIFY DOG BREEDS USING TRANFER LEARNING

Initially, extraction of the bottleneck features is performed. The features are consistent with the train sets, test sets and validation sets.



```
Found 6680 images belonging to 133 classes.
Found 835 images belonging to 133 classes.
Found 836 images belonging to 133 classes.
```

Figure 10: Output after feature extraction

A. Model Architecture

The steps involved in reaching the final CNN architecture are as follows:

- The dataset is relatively small, so choose InceptionV3.
- Slice off at the end of InceptionV3 and add a fully connected layer.
- Train the network to update only the weights of the fully connected layer.
- Finally fine-tune the last 140 layers(approx.) of the model using a stochastic gradient descent.

Compile and train the model. As done previously, use model checkpointing to load the one that achieves best validation loss.
The number of epochs used are 10. The number of layers to freeze are 172 while fine-tuning.
Finally test the model to get the training accuracy. The training accuracy was found to be 87.42%.

B. Predict Dog Breed with the model(Implementation)

First, extract the bottleneck features corresponding to chosen Convolutional Neural Network model, supply these as input to the model, use the dognames array to return corresponding breed.

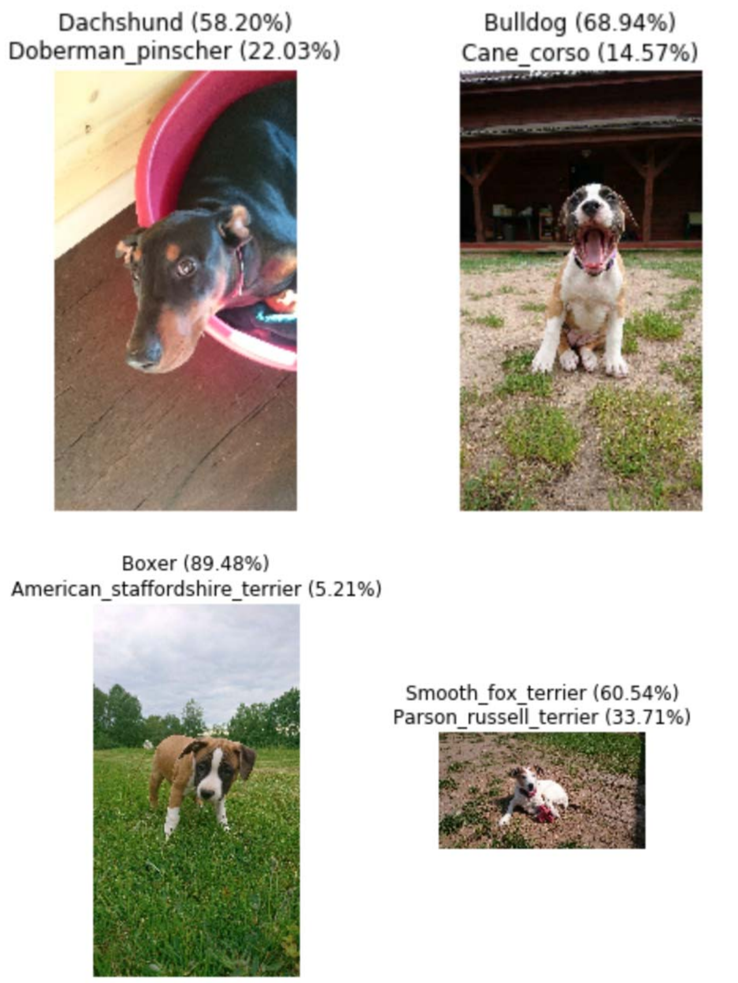

Dachshund (58.20%)
Doberman_pinscher (22.03%)
Bulldog (68.94%)
Cane_corso (14.57%)
Boxer (89.48%)
American_staffordshire_terrier (5.21%)
Smooth_fox_terrier (60.54%)
Parson_russell_terrier (33.71%)

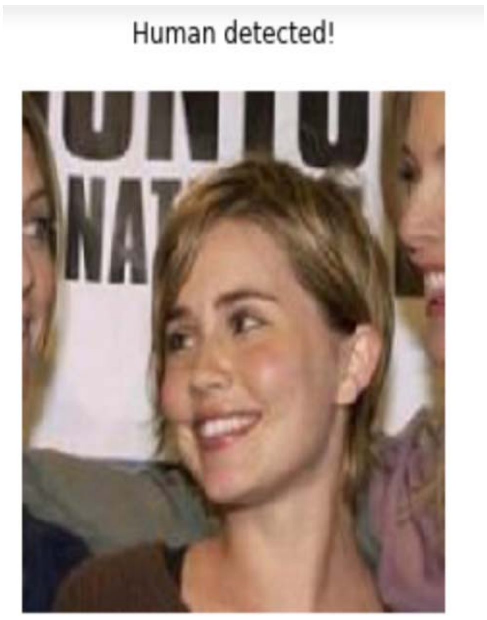
Figure 11:Prediction of dog images
Clearly transfer learning provides a better accuracy

IX. PART 7: WRITE THE ALGORITHM

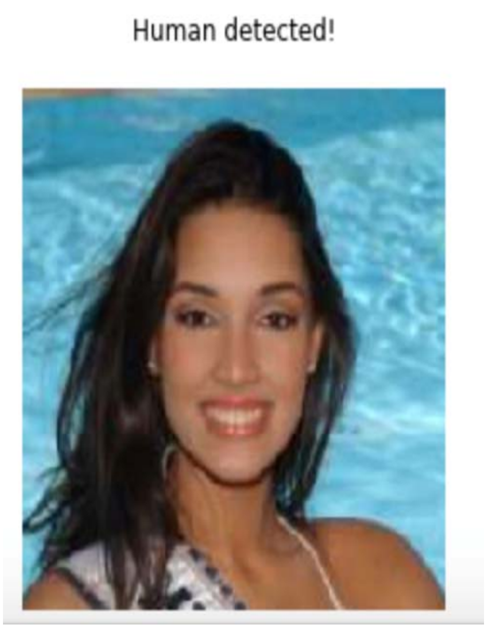
Summarizing all the steps above, write an algorithm that returns a predicted breed if a dog is detected in the image, return the resembling dog breed if a human is detected. If neither is detected, provide the output that indicates an error.

X. PART 8: RESULTS

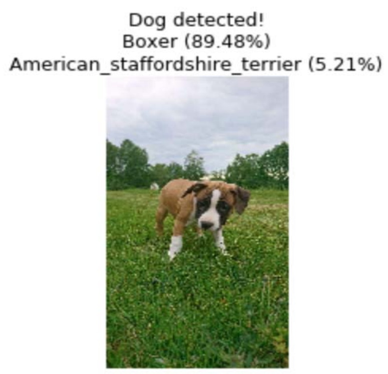
Having tested the algorithm on the results are obtained as given below:



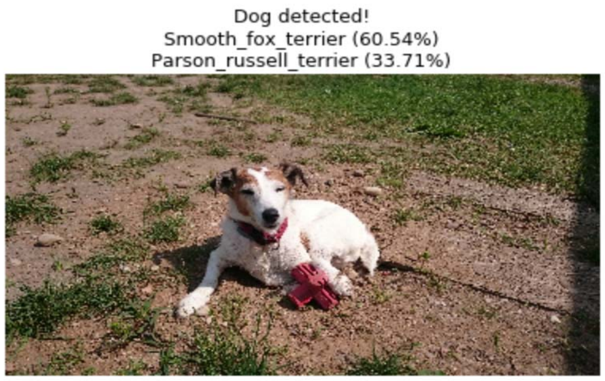
```
Human detected!
```



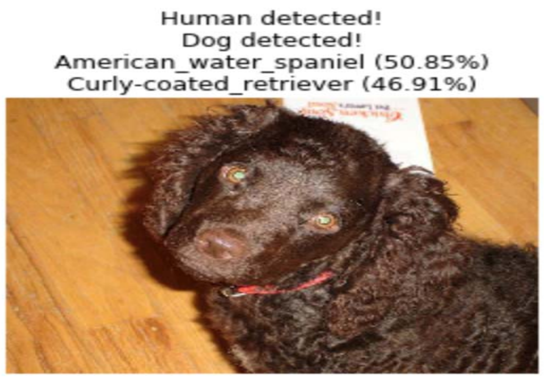
```
Human detected!
```



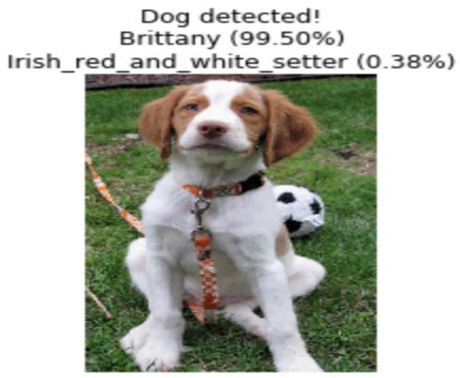
```
Dog detected!
Boxer (89.48%)
American_staffordshire_terrier (5.21%)
```



```
Dog detected!
Smooth_fox_terrier (60.54%)
Parson_russell_terrier (33.71%)
```

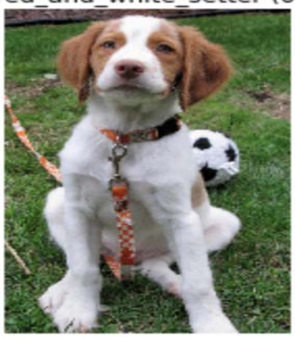


```
Human detected!
Dog detected!
American_water_spaniel (50.85%)
Curly-coated_retriever (46.91%)
```

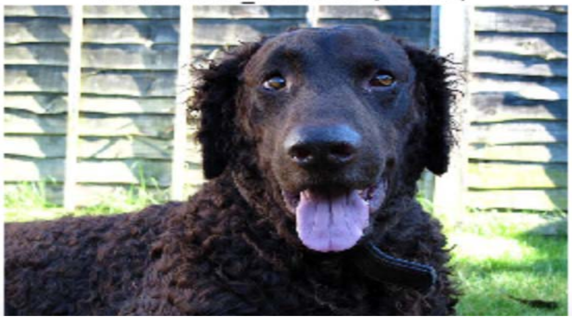


```
Dog detected!
Brittany (99.50%)
Irish_red_and_white_setter (0.38%)
```

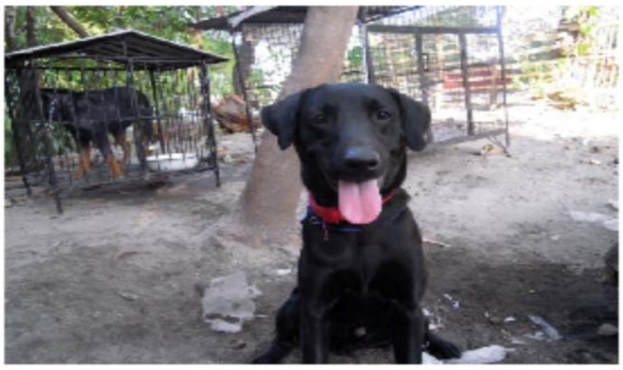
Dog detected!
Brittany (99.50%)
Irish_red_and_white_setter (0.38%)



Dog detected!
Curly-coated_retriever (99.96%)
Flat-coated_retriever (0.02%)



Dog detected!
Labrador_retriever (86.91%)
Plott (7.80%)



Human detected!
Dog detected!
Labrador_retriever (90.37%)
Chesapeake_bay_retriever (6.53%)

Human detected!
Dog detected!
Labrador_retriever (99.74%)
Pointer (0.21%)

Human detected!

```
hello, human!
```

```
You look like a ...
Chinese_shar-pei
```

Dog detected!
Irish_red_and_white_setter (67.16%)
Welsh_springer_spaniel (32.51%)

XI. CONCLUSION

With a very good accuracy the dog breed classifier performed well. The algorithm correctly classified the breeds from step 4 which is a very challenging task for human beings as well.

The possible improvements on this algorithm could be to train it to distinguish humans from dogs. The accuracy can be further enhanced by data augmentation. Data augmentation enables the network to differentiate the features irrespective of the orientation and scale. Clearly, building a convolutional neural network using transfer learning yielded a far better accuracy than building it from the scratch.

However, the model architecture used in Part 4 overcome the cons of the method used in ref.(1) in which the units are so fragile during training which poses the risk of the gradient flowing through the unit being zero forever from a point. It also overcomes the challenges posed in ref(2) where the number of parameters are doubled for every single neuron.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever “ImageNet classification with Deep Convolutional Neural Networks”

[2] Kaiming He et al., 2015 on “Delving Deep into Rectifiers”.

[3] Andrew G. Howard et al. on “Efficient Convolutional Neural networks for Mobile Vision Applications(2017)

[4] Jonas Gehring et al. on “ Convolutional sequence to sequence learning(2017)”

[5] Priya Goyal et al. on “Accurate, Large Minibatch SGD:Training ImageNet in one hour (2017)”.

[6] C. Zhang et al. on “Understanding deep learning requires rethinking generaization”.

[7] Y. Lechun et al. on “Gradient-based learning applied to document recognition (1998)”.

[8] Y. Lechun, Y. Bengio and G.Hinton on “ Deep learning (2015)

[9] A. Conneau t al. on “Very deep Convolutional networks for natural language processing (2015)”

[10] J. Hosang et al. “What makes for effective detection proposals”

[11] M. Malinowski et al. “ Ask your neurons:A neural based approach to answering questions about images (2015)”.