Machine Learning Engineer Nanodegree

Capstone Project – Dog Breed Classifier

Chang-Chun Lee
July 7th, 2019

I. Project Overview

Introduction to image recognition

Image recognition is one of the exploding topics in machine learning [1-7]. By building and training state-of-the-art convolutional neural networks, machine learning algorithms can be trained to classify different objects in the images. One of the most famous image recognition projects is ImageNet project, which is an ongoing research effort to provide researchers around the world an easily accessible image database. With the great amount of data in the ImageNet database, researchers participated in the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) had built many useful ML algorithms to classify images into one of 1,000 different categories. One great thing for ILSVRC is that several state-of-the-art algorithms and their pre-trained weights can be easily access through Keras deep learning library and can be further used for transfer learning on new image recognition project.

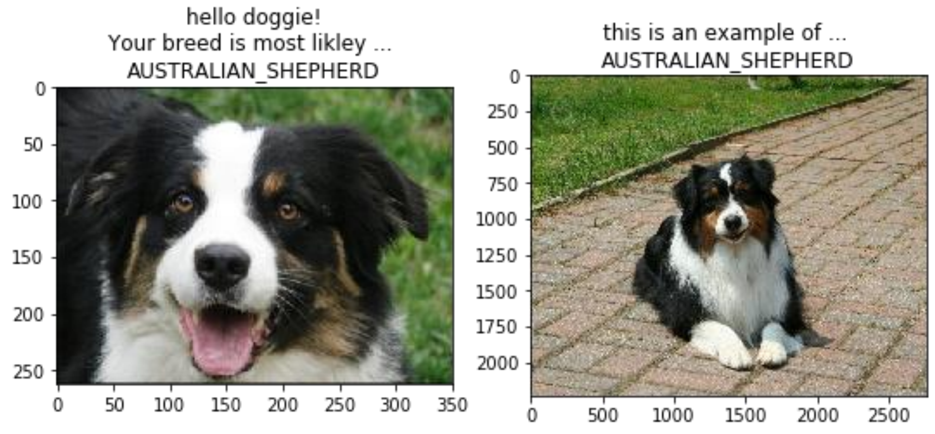
Problem Statement: A ML algorithm to identify dog breed

Inspired by ImageNet project [8], I work on a CNN project which can be used not only to recognize object in the image but also to learn some interesting features about the recognized image. In this project, I will develop several algorithms that could be used to identify the human, the dog, and also the dog breed from images. The task of assigning breed to dogs from images is considered exceptionally challenging. To see why, consider that even a human would have trouble distinguishing between a Brittany and a Welsh Springer Spaniel and distinguishing between Curly-Coated Retrievers and American Water Spaniels.

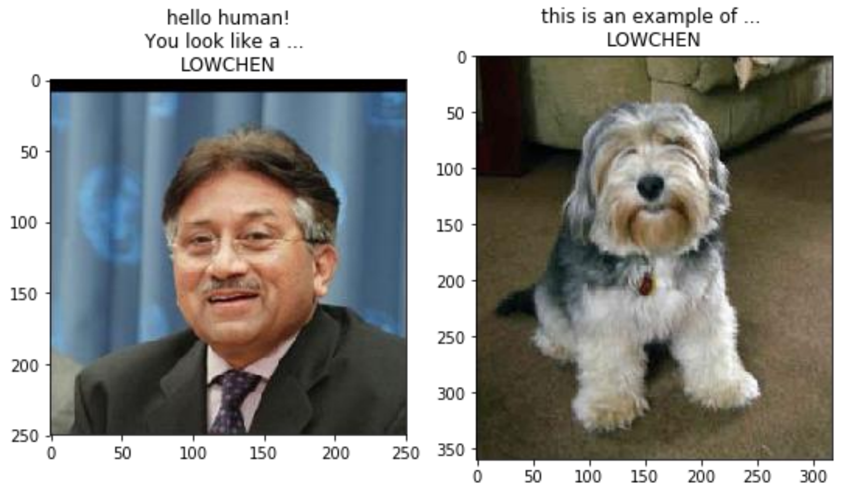
Although to identify dog breed from image is challenging, I believe I can build a decent algorithm by learning from ILSVRC and others who had put great efforts on image recognition. At the end of this project, I will build an ML algorithm, which will accept any user-supplied image as input. If a dog is detected in the image, it will provide an estimate of the dog's breed. If a human is detected, it will provide an estimate of the dog breed that is most resembling.

The images below show example outputs for my ML dog breed classifier. User will supply one image as input, and my classifier will identify its dog breed if human or dog is detected. The output will contain two images: the image on the left will be the original image user supplied with the title showing whether human or dog is detected and the predicted dog breed for this image; the image on the right is an example of dog image having the same breed as that of the original image.

Input dog image:



Input human image:



Input datasets

To build a dog breed classifier for human and dog images with high accuracy, lots of human and dog images are needed. The datasets and inputs used in this project contains dog images and human images, each image contains RGB three channels.

- Dog images were provided by Udacity; 8351 dog images in total. There are 133 dog breed classes in the dataset, and the samples in each class ranges from 26 to 77; this is not a fully balanced data. Also, the image sizes are varied between images; therefore, resize is needed in order to feed in to the CNN model.
- Human images, 13233 images in total, were downloaded from Labeled Faces in the Wild Project by the Computer Vision Laboratory established in the Computer Science Department at the University of Massachusetts. Human images in this project are mainly used for validation purpose.

Metrics to measure performance of a model

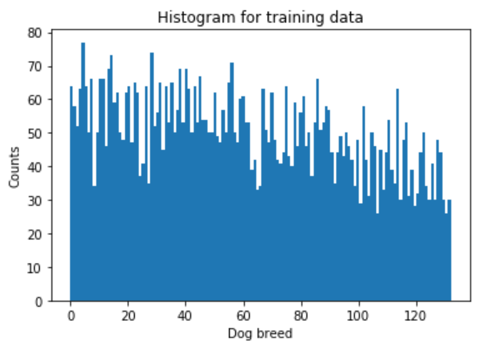
In order to measure the performance for face detector or dog detector algorithms, I use a subset of training data, including both human and dog images, to evaluate the percentage that one algorithm can detect target object in the target group images (true positive rate) and also the percentage that algorithm can detect the target object in non-target group images (false positive rate). For example, for a human detector, the first metric is the percentage that human images have been successfully detected as human (given 1000 human images) and second metric is the percentage that dog images have been detected as human (given 1000 dog images). Therefore, for a good algorithm, it is necessary to have high percentage for the first metric (true positive rate) and low percentage for the second metric (false positive rate). Besides, F1 score, which is the harmonic mean of precision and recall, is also generated to make judgement easier. By comparing F1 score, one can easily pick best algorithm which has F1 score closes to 1.

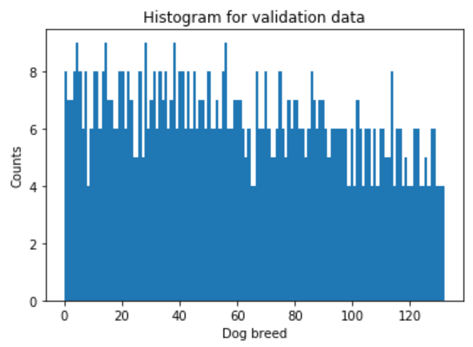
II. Analysis

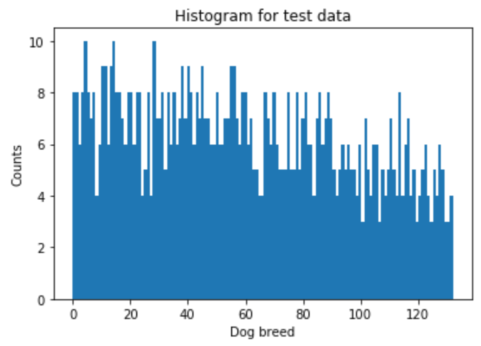
Exploratory Data Analysis

The datasets used in this project includes 8351 dog images and 13233 human images. Human images will be used mainly for validation purpose since the human detector model will be loaded from several open sources pre-trained model. Dog images, on the other hand, will be used to train the model for dog breed classifier. There are 133 dog breed classes in the dog images, and the samples in each class ranges from 26 to 77 (see figure below). Also, the image sizes are varied between images; therefore, resize is needed to make all image sizes consistent for input of the CNN model. In this project, all the images will be resized to 244 by 244 pixel.

Each image contains RGB three different channels; value in each channel ranges from 0 to 255. The following figures show the histogram of training, validation and test data. We can see the data is slightly imbalanced for 133 dog breed classes.







Algorithms and Techniques

In order to predict the dog breed on human or dog images, I have built 3 main functions in this project: human detector, dog detector and dog breed classifier. The task for human detector is to detect if there are humans in the image, and the task for dog detector is to detect if there are dogs in the image. The dog breed classifier is to predicted the most possible dog breed for the object in the image.

Human detector:

First, I build a ML algorithm to detect human in the image. 3 different algorithms from OpenCV and MTCNN are used and compared to find a most accurate way to detect human faces in the image. An ensemble method will also be used by voting the result from 3 algorithms.

- HAAP cascade classifiers: one of the pre-trained face detectors provided by OpenCV.
- LBP cascade classifier: one of the pre-trained face detectors provided by OpenCV.
- MTCNN (Multi-task Cascaded Convolutional Neural Networks): This is essentially several convolutional networks put together that give out several pieces of information.
- Ensemble method will also be used by voting the result from 3 algorithms

Dog detector:

Second, I use 3 different pre-trained ImageNet models to detect dogs in the images. Again, an ensemble method will also be used by voting the result from 3 algorithms

- Pre-trained VGG16
- Pre-trained InceptionV3
- Pre-trained NASNetMobile
- Ensemble method will also be used by voting the result from 3 algorithms

Dog breed classifier:

Here I use transfer learning to train a CNN model that can identify dog breed from images. I use pre-trained VGG16 model followed by few dense and dropout layers to predict 133 dog breed classes. The input dog images need to go through the same preprocess in order to use pre-trained VGG16 model. There are ~138M trainable parameters in the original VGG16 models, and last dense layer of original VGG16 model shows that there are 1000 classes for the final prediction. To build my dog breed classifier, which only has 133 classes, I remove the last dense layer from original VGG16 model, and I add few more layers of Dense, BatchNormalization, and Dropout. I also freeze the layers from original VGG16 model so that their weights will not change during my training. In this way, I only do the training to obtain the weights for the layers I add into the model.

Benchmark Dog Breed Classifier

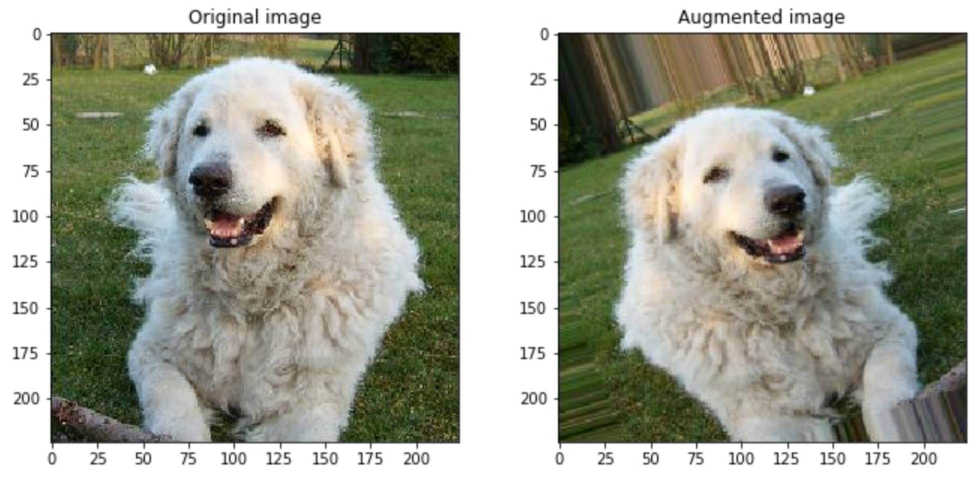
The most naïve benchmark for dog breed classifier is performing a random guess, which will provide a correct answer with the probability 1/133 (less than 1%), assuming the classes are more or less balanced. Instead of random guessing, I will build a simple CNN model to classify dog breeds from scratch as the baseline benchmark model. With 8000+ dog images, it could be difficult to train the CNN model from scratch. I can get some help from image augmentation to generate more images for training dataset by rotation, shift, zoom, and horizontal flip. This can be easily achieved by using ImageDataGenerator function in Keras deep learning library. The preprocess steps applied to the image are resize and normalization (rescale values to 0-1). The simple CNN benchmark model has the following architure: I repeat Conv2D and MaxPool layers 5 times to greatly reduce the output shape and then flatten layers followed by dense layers. After training for less than 1 hour, my dog breed classifier can have an accuracy of ~23% on the unseen test data. This simple benchmark model tells us that the problem of dog breed classifier is very likely solvable, and by using a more complex CNN model and more data, the classification accuracy can definitely be improved

III. Methodology

Data Preprocessing

For the image recognition task, the input datasets are all kinds of images, which consist 3 channels of 2D array with different sizes. The data preprocessing steps for image data are resize and normalization. The images used in this project may have varied sizes. Therefore, to make image size consistent for the CNN model, all the images are resized to 244 by 244. Besides, normalization is also performed for the images depend on the preprocess of the pre-trained model. For example, when using VGG16 pre-trained model, I use the function preprocess_input from Keras; basically, subtract the mean value of each RGB channel from the its original value. However, when using InceptionV3 and NASNetMobile pre-trained model, a minmaxscaler ranging from 0 to 1 is used for the preprocessing.

Besides resize and normalization, image augmentation is used for the training dataset to generate more training example. All the images in the training dataset could have several functions applied, like rotation, shift, zoom, and horizontal flip, in order to generate different images from the original ones.



Implementation

Human detector:

Three different pre-trained algorithms, including HAAR cascade, LBP cascade, and MTCNN, are used to detect the human faces in the images. These pre-trained algorithms can not only detect multiple number of faces in the images but also output the bounding boxes for each face. Therefore, by counting the numbers of face exist in the image, we can know if there is any human face in the image.

Dog detector:

Three different pre-trained algorithms, including VGG16, InceptionV3, and NASNetMobile, are used to detect dog in the images. These 3 models along with weights that have been trained in ImageNet can be loaded from Keras deep learning library. ImageNet contains over 10 million images, each image containing an object from one of 1000 categories. Given an image, one pre-trained ImageNet model returns a prediction (a value from 0-999 representing 1000 possible categories in ImageNet) for the object that is in the image. Looking at the label dictionary, we can find that the categories corresponding to dogs appear in dictionary keys 151 (Chihuahua) - 268 (Mexican hairless), and also dict. key 273 (Canis dingo). Thus, in order to check if an image is predicted as a dog by the pre-trained ImageNet model, I can check if the pre-trained model predicts an output index between 151 and 268, or an index 273.

Dog breed classifier:

To improve the model performance with limited numbers of training images, I use transfer learning, and then further train my CNN model that can identify dog breed from images. The pre-trained VGG16 model and its weights, which had been trained by several million of images, is used as the base of my CNN model, by removing the last layer containing 1000 nodes corresponding to 1000 classes for the ImageNet. Then two dense layers are added to predict 133 dog breed classes. There are ~ 138M trainable parameters in the original VGG16 models, but after freezing the layers from VGG16 models, the total trainable parameters in my CNN model are ~1.1M. To make the training converges faster and also train the model efficiently, learning rate reduction is used to reduce the learning rate during the training when necessary, and early stopper is used to stop the train if there is no further improvement on the accuracy. Since I have ~6600 training images and batch size is 16, the steps per epoch is set to 400 so that roughly all the training images are used once per epoch.

The table below shows the summary of my final CNN architecture for dog breed classifier. Layers above fc2 (Dense) are from pre-trained VGG16 model, and their pre-trained weights are frozen meaning that they will not change during the training process. After layer fc2 (Dense), I add two layers of dense + batch norm + dropout before making the prediction for 133 classes. The total trainable parameters for this CNN model are about 1.1M.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856     
_________________________________________________________________
```

```
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
dense_8 (Dense)              (None, 256)               1048832
_________________________________________________________________
batch_normalization_97 (Batc (None, 256)               1024
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               32896
_________________________________________________________________
batch_normalization_98 (Batc (None, 128)               512
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 133)               17157
=================================================================
Total params: 135,360,965
Trainable params: 1,099,653
Non-trainable params: 134,261,312
_________________________________________________________________
```

Refinement

For human and dog detector task, 3 different algorithms are used to perform the task. However, the performance from different algorithm varies for different images. One trick I did to improve the performance of human or dog detection task is to perform ensemble method by voting from three different prediction results. Instead of make 1 prediction for human detector or dog detector, 3 predictions are made for each case and voting from 3 predictions is used to get more accurate prediction result.

For dog breed classifier, when connecting two dense layers after pre-trained VGG16 model, two dropout layers with dropout rate = 0.25 are also added to reduce the potential overfitting when training the model. Another way to reduce the overfitting is to use more data during the training process, and data augmentation is therefore used to achieve this purpose.

IV. Results

Model Evaluation and Validation

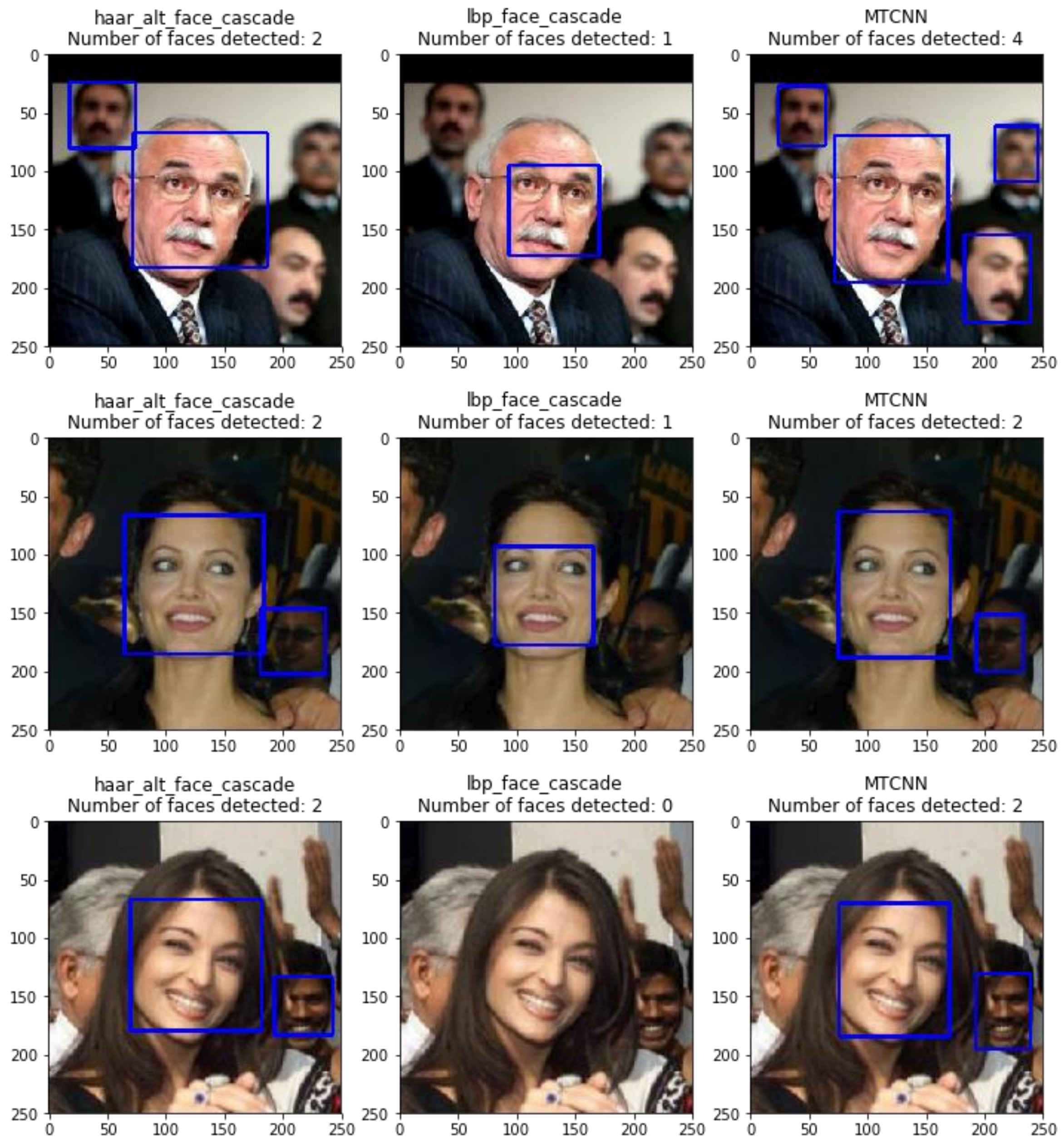
To achieve the goal of identifying the dog breed in the human or dog images, three different algorithms are used to perform three different tasks: human detector, dog detector, and dog breed classifier. In order to achieve high accuracy of dog breed prediction on human or dog images, I need to make sure each of this algorithm can make good prediction for its own task. Human detector or dog detector could be a relatively easy task for ML algorithms since humans can perform these tasks with pretty high accuracy. However, dog breed classifier could be relatively difficult since even a human may have troubles distinguishing between some dog breeds. Let’s take the performances of these algorithms.

Human detector:

The performance of human detector from 3 different algorithms and 1 ensemble method can be seen in the following table. We can see all three algorithms have pretty good F1 score, and the performance can be further improved by ensemble method.

Metrics	haar_alt_face_cascade	lbp_face_cascade	MTCNN	Ensembling
Detect face in human images	97.8%	82.7%	97.3%	97.0%
Detect face in dog images	9.8%	3.0%	14.9%	3.5%
F1_score	0.942	0.891	0.917	0.968

Below are few output images of human detector with face bounding box displayed. We can see that multiple faces can be detected more accurately in some algorithms, like HAAR cascade and MTCNN. For some human images (for example, 3rd example below), LBP cascade human detection algorithm could have difficulty to identify faces in the image, but other algorithms can detect even multiple faces correctly. Therefore, by using ensemble method with the cost of making two more predictions for each input image, the task of human detection can be performed better. We can see the F1 score on the unseen test dataset improves to 0.968 after ensemble method.

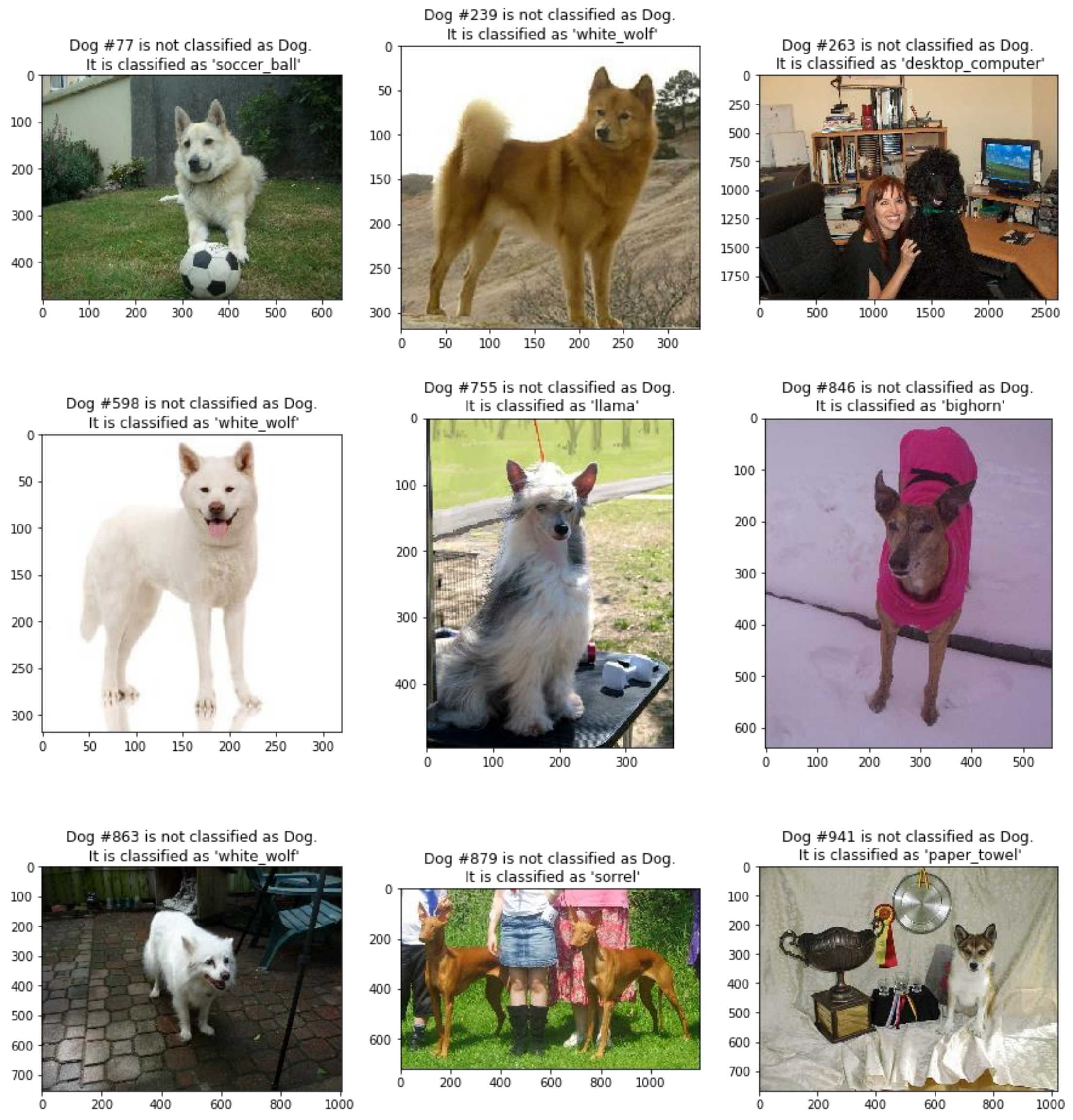

haar_alt_face_cascade
Number of faces detected: 2
lbp_face_cascade
Number of faces detected: 1
MTCNN
Number of faces detected: 4
haar_alt_face_cascade
Number of faces detected: 2
lbp_face_cascade
Number of faces detected: 1
MTCNN
Number of faces detected: 2
haar_alt_face_cascade
Number of faces detected: 2
lbp_face_cascade
Number of faces detected: 0
MTCNN
Number of faces detected: 2

Dog detector:

Similar to the result of human detector, the dog detectors for 3 different algorithms have very high F1 score, and ensemble method can even improve the F1 score to near 1.

Metrics	VGG16	InceptionV3	NASNetMobile	Ensembling
Detect dog in human images	0.3%	1.6%	1.0%	0.2%
Detect dog in dog images	99.4%	99.8%	99.7%	99.7%
F1_score	0.995	0.991	0.994	0.997

Below show few false negative examples for dog detector. Some of the images are actually quite challenging (for example, first image is classified as soccer ball which is correct).


Dog #77 is not classified as Dog.
It is classified as 'soccer_ball'
Dog #239 is not classified as Dog.
It is classified as 'white_wolf'
Dog #263 is not classified as Dog.
It is classified as 'desktop_computer'
Dog #598 is not classified as Dog.
It is classified as 'white_wolf'
Dog #755 is not classified as Dog.
It is classified as 'llama'
Dog #846 is not classified as Dog.
It is classified as 'bighorn'
Dog #863 is not classified as Dog.
It is classified as 'white_wolf'
Dog #879 is not classified as Dog.
It is classified as 'sorrel'
Dog #941 is not classified as Dog.
It is classified as 'paper_towel'

Dog breed classifier:

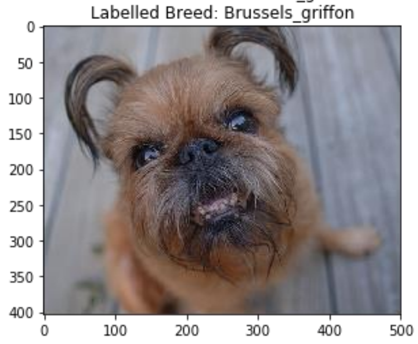
Dog breed classifier is used to identify the predicted dog breed in the image. The accuracy on training data is much higher than validation or test data showing overfitting in training data.

Metrics	Training	Validation	Test
Accuracy	90.2%	81.8%	81.8%

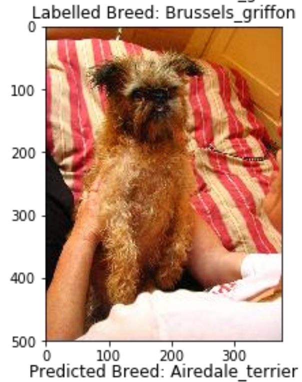
Below show few examples of predicted dog breed for the dog images. Surprisingly, most predicted results are correct. For the first image, it would be challenging to classify its dog breed correctly since even I cannot distinguish dogs between irish red and white setter and Brittany.

Predicted Breed: Irish_red_and_white_setter
Labelled Breed: Brittany

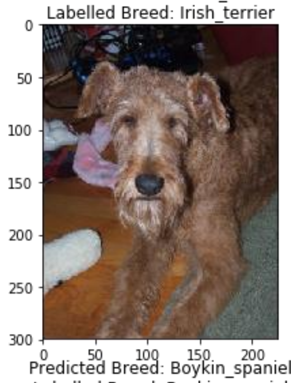
Predicted Breed: Brussels_griffon
Labelled Breed: Brussels_griffon



Predicted Breed: Brussels_griffon
Labelled Breed: Brussels_griffon



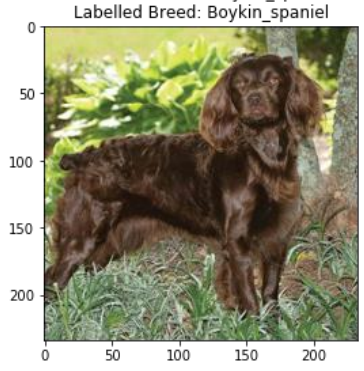
Predicted Breed: Irish_terrier
Labelled Breed: Irish_terrier



Predicted Breed: Havanese
Labelled Breed: Havanese

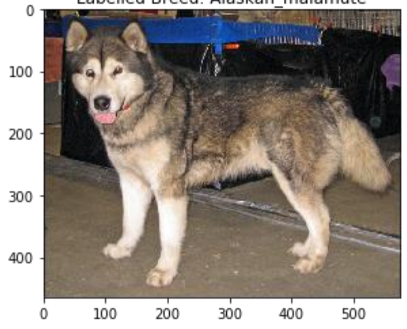
Predicted Breed: Airedale_terrier
Labelled Breed: Airedale_terrier

Predicted Breed: Boykin_spaniel
Labelled Breed: Boykin_spaniel



Predicted Breed: Newfoundland
Labelled Breed: Newfoundland

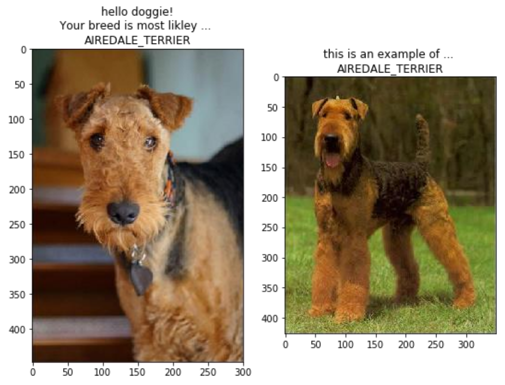
Predicted Breed: Alaskan_malamute
Labelled Breed: Alaskan_malamute

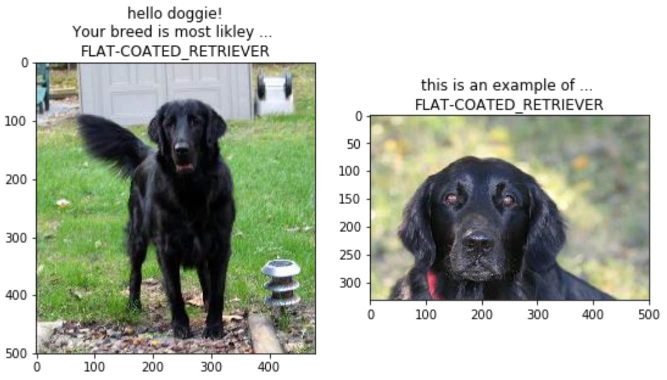


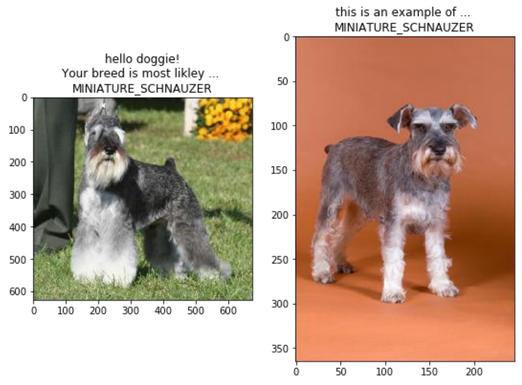
Dog breed prediction on dog images:

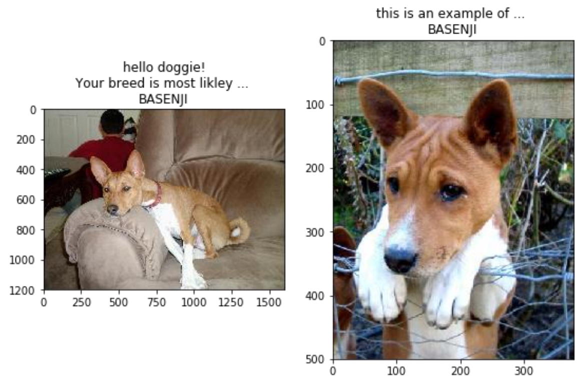
After putting all the functions together, my algorithms can take one image from user and display two images: the original image with the predicted dog breed + additional image showing an example of dog of the predicted dog breed.

Below show few example of the output results when input dog images. First, my algorithm correctly idenfiy objects in these images as dogs. Second, the predicted dog breed is correctly. Although I don't know all the breed names, I can simply compare the original image of the left to the example image of predicted dog breed on the left. This is a much easy way to make sure the dog breed has been predicted correctly or not.









Justification

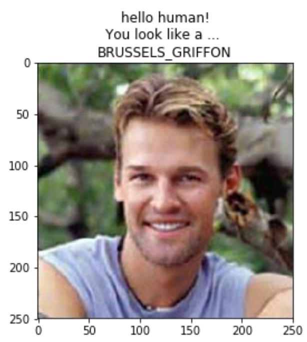
Starting from a simple CNN benchmark model, the dog breed classifier can perform the task with ~ 23% accuracy on the test dataset. The final solution of my dog breed classifier can achieve ~82% accuracy on the unseen test dataset, which is definitely much better than the benchmark model. Although the accuracy of my final solution could be improved further, knowing dog breed identification is a challenging task in image recognition topic, I feel confident to use my final CNN model to make a prediction to classify dog breed in the image.

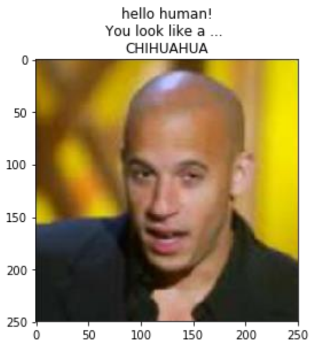
V. Conclusion

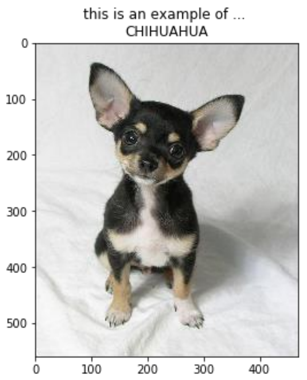
Fun Visualization

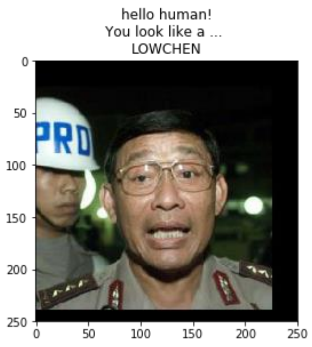
Dog breed prediction on human images:

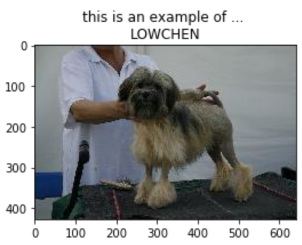
The fun thing to see using my final algorithm is to predict the resembling dog breed for a given human image. Images below show 6 example outputs of dog breed prediction on human images. I feel the human image of the left indeed share some similarities with the dog image on the right. Using my algorithm, user can supply an image to see the predicted dog breed for the input human or dog image.

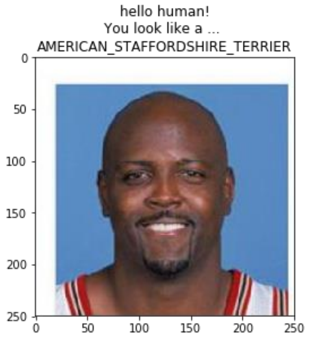


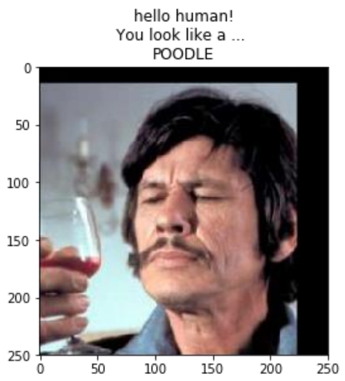


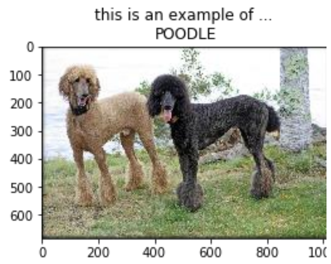


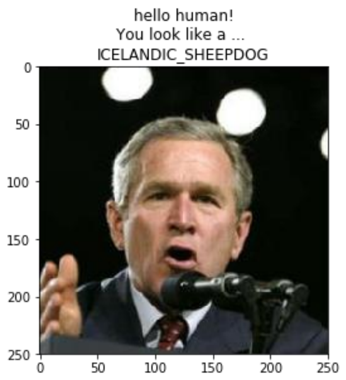


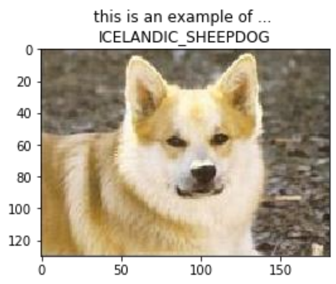












Reflection

Image recognition is one of the challenging tasks in machine learning. However, great efforts have been made to improve the performance of this task every year. Even without a great amount of training data, people can always use pre-trained model or transfer learning to make a robust ML algorithm to perform their image recognition task. In this project, I have utilized both pre-trained model and transfer learning to speed up the training process in order to identify the dog breed in the image. I start from building 3 blocks to detect dog, detect human, and classifier dog breed, separately. By integrating all 3 functions, I have built my final algorithm to identify the dog breed for the human or dog image. The performance of dog breed classifier is evaluated by test data containing 836 dog images, and my final solution can achieve ~82% accuracy when performing this task.

Improvement

Although my final solution can achieve ~82% accuracy on the dog breed identification task, I think there are still few things I can do to make the final solution even better. First, collect more training data of dog images from public websites. It's clear that overfitting is existed during my training since the accuracy on the training dataset if much higher than test dataset. One way to solve this problem is to use more training data so that the model can be more generalized to the unseen data. Second, use the ensemble method to improve the performance of dog breed classifier. It's obvious that ensemble method can improve the model performance by voting from several predicted result. However, the disadvantage is that I need to train at least two more models using transfer learning.

VI. Reference

1. Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.
2. J. Deng, O. Russakovsky, J. Krause, M. Bernstein, A. Berg, L. Fei-Fei. Scalable multi-label annotation. *ACM conference on human factors in computing (CHI)*, 2014.
3. O. Russakovsky, J. Deng, Z. Huang, A. Berg and L. Fei-Fei, Detecting avocados to zucchinis: what have we done, and where are we going?, *Proceedings of the International Conference of Computer Vision (ICCV). 2013*
4. J. Deng, A. Berg, K. Li and L. Fei-Fei, What does classifying more than 10,000 image categories tell us? *Proceedings of the 12th European Conference of Computer Vision (ECCV). 2010.*
5. O. Russakovsky and L. Fei-Fei, Attribute Learning in Large-scale Datasets. *Proceedings of the 12th European Conference of Computer Vision (ECCV), 1st International Workshop on Parts and Attributes. 2010.*
6. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. *IEEE Computer Vision and Pattern Recognition (CVPR), 2009.*
7. J. Deng, K. Li, M. Do, H. Su, L. Fei-Fei, Construction and Analysis of a Large Scale Image Ontology. *In Vision Sciences Society (VSS), 2009.*
8. ImageNet project, http://image-net.org/index