Dog Breed Classification

Example:

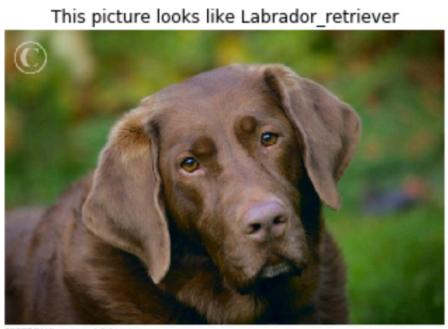
Overview

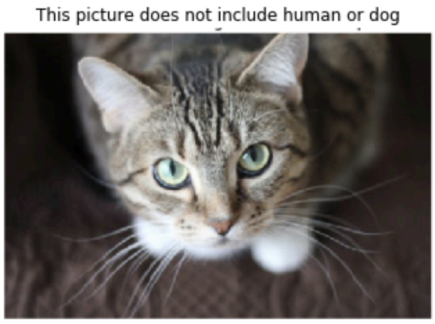
This project is the last part of the Data Scientist Nano Degree of Udacity, which aims to help students with data analysis. This projects is based on CNN, through which I could learn how to build a Deep Learning model for a classifier.

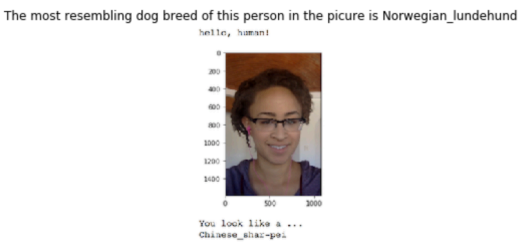
The classifier I accomplished is to detect whether a human face or dog exists in the picture. The whole work includes:

- Step 0: Import necessary datasets and libraries with python
- Step 1: Detect human faces
- Step 2: Detect dogs

The classifier can recognize whether there is a human face or dog in the image. If it is a dog, then it tells you the breed.







- Step 3: Build up a CNN model to classify dog breed if a dog is detected

- Step 4: Use transfer learning to classify dog breed based on CNN
- Step 5: Build up a CNN model with transfer learning for dog breed classification
- Step 6: Write algorithms for classifier
- Step 7: Test the customized classifier

Protocol

Step 0: Import Necessary Datasets and Libraries with Python

Datasets include thousands of dog images and 13233 human face images. What I did is to split the datasets into three parts, which is training datasets, validation datasets and test datasets. There are 133 dog breeds.

To accomplish this project, I used many popular libraries, including scikit-learn, keras, tensorflow, pandas, numpy and so on.

Step 1: Detection of Human Faces

OpenCV's implementation of Haar feature-based cascade classifiers is used to detect human faces in a given picture.

First of all, the colored images has been converted to gray scale in order to process later and easier. Then, Cascade classifier is used to point out the number of human faces in the image. If a human face exists in the image, the program can point it out with a rectangle. At last, with the testing data, I find 100 percent human faces are detected correctly. But 11 percent of 100 dog images are detected as human faces.

Step 2: Detection of Dogs

I used pre-trained ResNet-50 model for this part. The weights from ResNet-50 model are used as pre-trained. Then the classifier works after pre-processing the input image that includes converting the image to an array. This pre-process resizes the image, and makes it a 4D tensor for Keras. After all, the program outputs the probabilities for a particular image from ImageNet category. The dogs index is from 151-268. So we can recognize the result based on the number the program returns.

The results show that the accuracy is 100 percent for both of human faces and dogs.

Step 3: Build up a CNN to Classify Dog Breeds

Here I need to create a CNN model to classify dog breeds reaching at least 1% accuracy. The structure of the model is as follows:

```
_________________________________________________________________
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
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 28, 28, 128)       32896
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 300)               38700
_________________________________________________________________
dropout_1 (Dropout)          (None, 300)               0
_________________________________________________________________
dense_2 (Dense)              (None, 150)               45150
_________________________________________________________________
dropout_2 (Dropout)          (None, 150)               0
_________________________________________________________________
dense_3 (Dense)              (None, 133)               20083
=================================================================
Total params: 147,373
Trainable params: 147,373
Non-trainable params: 0
_________________________________________________________________
```

Our test accuracy in this part is 2.5120%.

Step 4: Use Transfer Learning to Classify Dog Breed based on CNN

In this step, I chose ResNet-50 as pre-trained model for transfer learning. The last convolutional output of ResNet-50 is fed as input to my model through extracting the bottleneck features. I add global average pooling layer and 133 fully connected layer after the ResNet-50 model output with a Softmax function. Through 20 epochs, I got 42.22% accuracy.

Step 5: Build up a CNN Model with Transfer Learning for Dog Breed Classification

After testing the VGG-19 and ResNet-50 model, I chose ResNet-50 for this step. The structure of model in this step is similar to last step. With 20 epochs, I got 78.95% accuracy with dog image testing data. In addition, this step can be used to classify dog breed.

Step 6: Write Algorithms for Classifier

In this step, I write an algorithm that can detect whether a human face or dog in the image. If a dog is detected, then the breed will be returned. This program accepts a file path to an image as input to determine the results.

- If a dog is in the image, the breed will be returned.
- If a human face is in the image, the resembling dog breed will be returned.
- If neither presents, provide the situation description.

Step 7: Test Algorithms

This step is made up of the above steps. It works as a classifier as expected. The example of results is shown above.

Conclusion

More steps needs to be done to optimize the classifier. When training the model, overfitting needs more attention. If it is, adding more dropouts may be helpful. Increasing epochs is helpful for underfitting, but not for overfitting.

In the next step, more adjustments to parameters and hyper-parameters should be applied to. Grid search is also a considerable way to optimize the model.