Convolutional Neural Networks(CNN) Project: Dog Breed Classifier

Faruk Geles

December 2020

Abstract

The second project of the Udacity Deep Learning Nanodegree is about how to build a pipeline to process real-world, user-supplied images, i.e. if a image of a dog is given the algorithm has to identify the dog's breed. In the case of human image given, the algorithm has will identfy the resembling dog breed

1 The Goal of the Project

In this project we build a machine learning model, which can be used within a web app to process real-world, user-supplied images. Here the algorithm has to do two tasks:

- Dog face detector: if a image of a dog is given the algorithm has to identify the dog's breed.
- Human face detector: if a human image given, the algorithm has will identfy the resembling dog breed

1.1 Used Data Set

The dog and human images datasets, which we use for this project are available at:

- https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
- https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip

The zip-folders contain following data

dogImages.zip:

- In this dataset there are 8351 total images, which are sorted into train directory(6689 images), test directory(836 images) and valid directory(835 images). Each of these directories have 133 folders corresponding to the dog breeds. The images here are of different size and have also different backgrounds. The number of images for each dog breed delivered is also differing.

lfw.zip:

- In this dataset there are 13233 total human images. The structure of the datasat is following. We have 5750 folders sorted by the names of humans. All images are 250x250. The number of images for each human delivered is also differing i.e., each folder contain different number of images corresponding to a human.

2 The Approach

For the multi-class classification problem we use a Convolutional Neural Network(CNN)

- We use an opencv implementation of Haar feature based cascale classifier with an input "...", for the detection of human faces
- For detecting dog faces we will use a pretained VGG16 model
- After the differentiating of the dog and human images, we can pass the image to a Convolutional Neural Network model to predict, which of the 133 breed classes belong the image

2.1 Image Preprocessing

The following transformations are applied to all images. We resize them all to 224x224 square and normalize them all. Then for the training data , image augmentation is done in order to reduce overfitting. The train data are horizontally flipped. Finally all the images has to be coverted into tensors, in order to put them into the model

2.2 Simple Network Architecture

- I choose 3 convolutional layerss with a kernel size of 3x3 and padding 1, which increases the number of feature maps
- In between every convolutional layer is a maxpool layer with 2x2 kernel and stride 2, that halfs the size of all feature maps
- After 3 Convolutional and maxpool layers we end up with 128*7x7 feature maps
- Then the feature maps are flattened to a vector of length 128*7x7 and put into the fully connected (FC) layers for classification. We have for this problem 133 classes
- A dropout of 0.2 is added to the network to reduce the overfitting

This simple network architecture gives us 15% accuracy after 30 epochs of training. So next we are going to improve the model using transfer learning.

2.3 Complicated Network Architecture using Pretrained Model-Transfer Learning

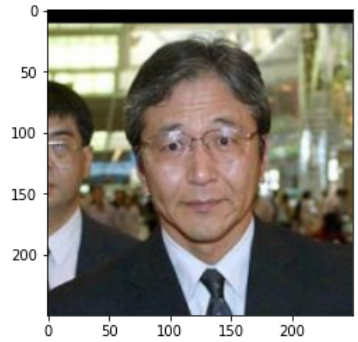
We used for the transfer learning the ResNet50 architecture

- 1) The ResNet50 has been trained on ImageNet, a very large popular dataset which is used for image classification and other vision tasks. ImageNet contains over 10 million URLs, each linking to an image
- 2) It also contain from one of 1000 categories.
- 3) Given an image, this pre-trained ResNet50 model returns a prediction (derived from the available categories in ImageNet) for the object that is contained in the image.
- 4) I have changed the last fully connected layer from 1000 to 133 categories

2.4 Results of Transfer Learning

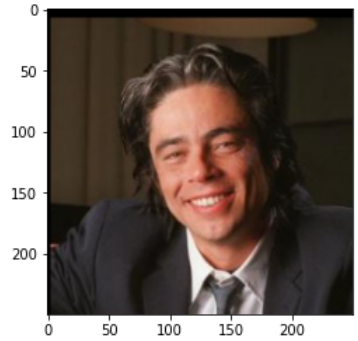
We get an accuracy of 77% after 30 episodes

Human!



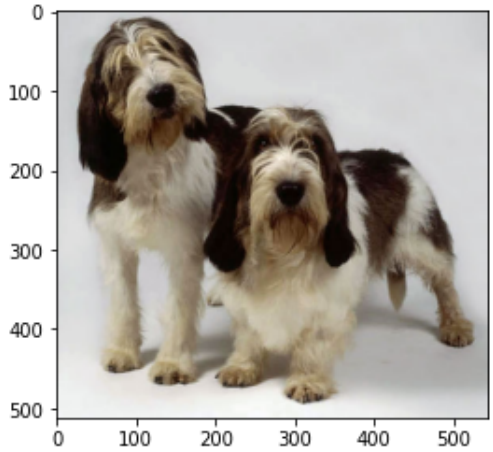

```
You look according the algorithm like a... Glen of imaal terrier
============================================================
```

Human!

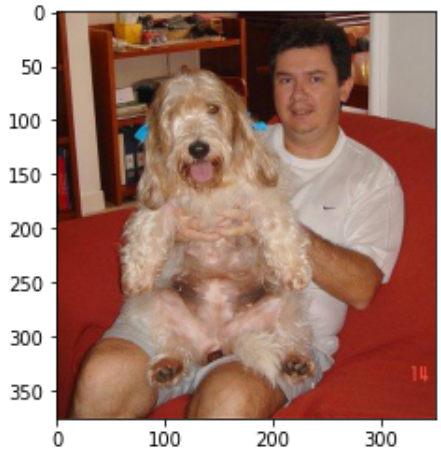



```
You look according the algorithm like a... Dandie dinmont terrier
============================================================
```

Figure 1: Some results for human images.



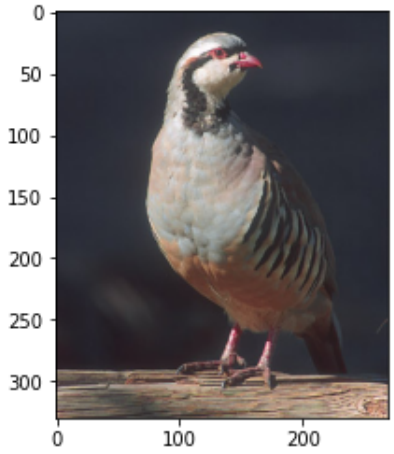

```
This is the image of a ... Bearded collie
============================================================

Human!
```




```
You look according the algorithm like a... Otterhound
============================================================
```

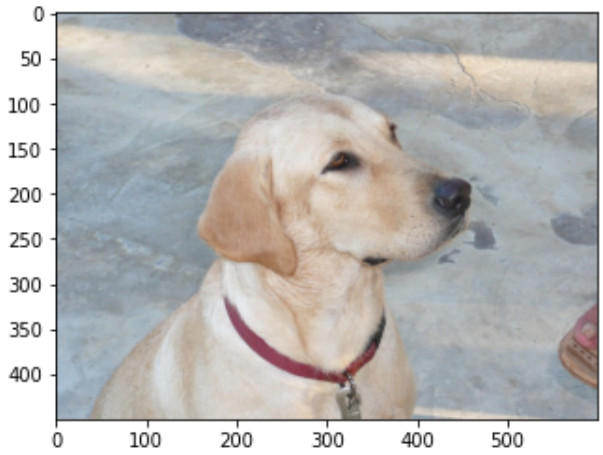
Figure 2: Some results for dog images.

```
In [30]: run_app('my_images/Keklik.jpg')
```




```
Sorry, in this image neither a dog nor human is detected!!!
============================================================
```

```
In [36]: run_app('my_images/dog_labrador.jpg')
```




```
This is the image of a ... Labrador retriever
============================================================
```

Figure 3: Some results of example images.

3 Ideas for Future Work

- 1) improving the face detector algorithm
- 2) Try to use other neural network architectures using transfer learning to achieve a better accuracy
- 3) Increasing the training episodes