Convolutional Neural Network (CNN) Project

Capstone Project Proposal : Dog Breed Classification

Yongyang Sun Udacity

29th Dec 2020

Domain Background

Thanks to the increase in computational power and the amount of availabe training data in the last few decades, Convolutional Neurol Network (CNN) has managed to achieve suoperhuman performance on some complex visual tasks. They power image and video recognition, recommender systems,image classification, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

The dog breed classification copstone project is one of the most popular programs that can be used to start exploring the CNN models. It provides a great introduction to CNN and deep learning as the goal of this project is to classify images of dogs according to their breed. It is a multi-class problem and easy to follow. For machine learning beginners, it is a great opportunity to explore the world of CNN and deep learning.

Problem Statement

Image that you are a dog lover and you have a picture from the website or just taken by your phone. A fantastic thing is to develp a model that can be used to distinguish the dog breeds. But before we approach that step, the first thing we need to figure out is that whether there is a dog on the picture. If the answer is yes, which type does the dog belong to? If the answer is no, is it an human ? and if the human were a dog, which type does the human look like?

This is an supervised learning problem as we already know that the dogs are divided into different breeds. The key solution to this provblem is to train an multi-class predictive model

Datasets and Inputs

To solve the problem of dog breed classification, we need to explore and visualize the datasets availabel. All the datasets used for this project are provided by Udacity. We have 13233 human images and 8351 dog images. Dog images are divided into three sub datasets, "training" (6680 images), "validation" (835 images) and "test" (836 images). In the three sub-datasets, all the images are sorted into 133 different dog breed folders.

As we can see from the images below, dog images have many different characteristics such as different backgrounds, different lighting, and different image size. Some dogs are with human while some others are not.

In [1]:

```
import numpy as np
from glob import glob

#load filenames for humans and dog images
human_files = np.array(glob('lfw/*/*'))
dog_files = np.array(glob('dogImages/*/*/*'))

# print number of images in each dataset
print('There are %d total human images.' % len(human_files))
print('There are %d total dog images.' % len(dog_files))
```

```
There are 13233 total human images.
There are 8351 total dog images.
```

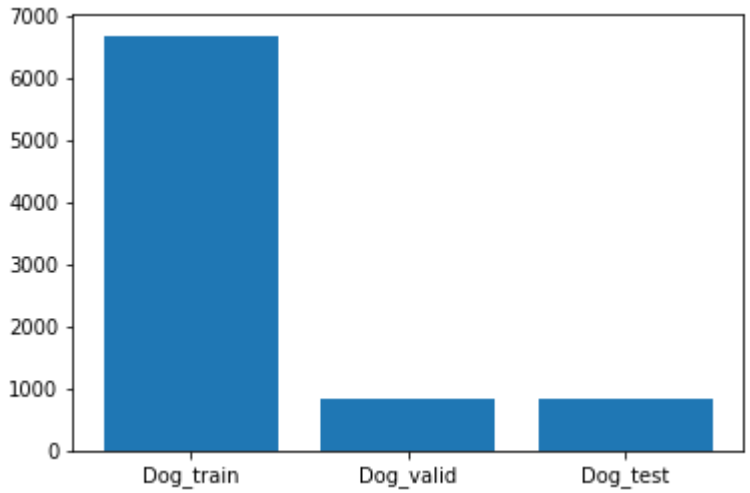
In [2]:

```
import matplotlib.pyplot as plt
%matplotlib inline
from PIL import Image
fig = plt.figure(figsize=(16, 8))

for i in range(1, 11):
    img=Image.open(dog_files[i])
    plt.subplot(2,5,i)
    plt.imshow(img)
plt.tight_layout()
plt.show()
```

In [3]:

```
Dog_train = dog_files = np.array(glob('dogImages/train/*/*'))
Dog_valid = dog_files = np.array(glob('dogImages/valid/*/*'))
Dog_test = dog_files = np.array(glob('dogImages/test/*/*'))
numbers = [len(Dog_train), len(Dog_valid), len(Dog_test)]
x = ['Dog_train', 'Dog_valid', 'Dog_test']
plt.bar(x, height=numbers)
plt.show()
```



Human images are sorted by the names. We have 13233 human images which are stored in 5749 folders. As we can see from the images below, the dataset of human images are imbalanced. Some of them have only one image (row 1, column 1) while some others have many images (row1, column 2-5). Besides this, human images have so many different characteristics such as different backrounds, multiple colors, and different hair styles, different facial expressions, and so on. In addition, some images have only one person while some others have several persons on the same image. Will this affect the modeling result? Is it essential to remove these images from the dataset? It would be interesting to see the model perforamnce in these cases.

In [4]:

```
fig = plt.figure(figsize=(16, 8))
for i in range(1, 11):
    img=Image.open(human_files[i])
    plt.subplot(2,5,i)
    plt.imshow(img)
plt.tight_layout()
plt.show()
```

Solution Statement

In this project, we will test the performance of two Convolutional Neural Network (CNN) models. CNN is a class of deep neural networks, most commonly applied to analyzing visual imagery. To solve the key problem, the first step is to determine if the picture is a human or dog. In this case, we will use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images. Then, we will test the performance of several pre-trained models on detecting dog faces such as VGG16, ResNet_50 and ResNet_18. After these works, we will create a CNN model from scratch to see its performance on distinguishing the dog breeds. However, a CNN model from scratch is more preferred when we have a massive number of images. The number of images (6680) for training in this project is far less behind "massive". Hence, a CNN model from scratch will have relatively low accuracy. Instead, we will test the performance of a CNN model created by transfer learning. The main advantage is that transfer learning is referred to when such a deep network was already built and ready to use for any application. Hence, it will have relatively higher accuracy and run much faster than creating a CNN model from scratch.

Benchmark Model

As we claim in "Solution Statement", we will use a CNN model created from scratch with an accuracy of over 10% as a benchmark model. Since we know that there are 133 different dog breeds in our dataset, randomly picking 1 from 133 will give a probability of 0.75%, which is much smaller than 10%.

Evaluation metrics

The metrics that can be used to evaluate the performance of the trained CNN model are many, such as Accuracy, Recall, Precision, F1 score and so one. In our case, we use cross-entropy loss to optimize the trained model and accuracy to evaluate the performance of the models in classifying the dog breeds. Cross-entropy loss awards lower loss to predictions which are closer to the class label. The accuracy, on the other hand, is a binary true/false for a particular sample. That is, Loss here is a continuous variable i.e. it's best when predictions are close to 1 (for true labels) and close to 0 (for false ones). While accuracy is kind of discrete. In our case, there are 133 classes for dog breeds and the datasets are relatively balanced. Hence, accuracy can be used as a metric to evaluate the performance of the trained model. Accruacy is calculated by the ratio of the correct predictions to the total number:

$Accuracy = \frac{TP}{TP+FP}$

Project Design

With all the knowledge obtained above, we have the outline to conduct the project as shown below

1. Import the datasets required (dog and human datasets);
2. Detect human using a pre-trained detector (here we will use the OpenCV's implementation of Haar feature-based cascade classifiers t to detect human faces from the image. Please refer to https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html (https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html) to get more information);
3. Detect dogs using several pre-trained detectors (here we will use VGG16, ResNet50 and ResNet18 to test their performance)
4. Since our data has been divided into "training", "validation" and "test" subsets, we can use the "training" dataset to create a Convolutional Neural Network (CNN) from scratch as a benmark model. We will test the model on the "test" dataset to get an accuracy of over 10% by adujusting the input parameters such as epoch value and the number of convolutional layersused. CrossEntropyLoss is used to evalute the quality of the created CNN model.
5. Create a CNN model by transfer learning using a pre-trained dog detector based on the same "training" dataset. By adjusting the parameters, we will develop a model with an expected acccuracy of over 60%. CrossEntropyLoss is used to evalute the quality of the created CNN model.

References

1. https://en.wikipedia.org/wiki/Convolutional_neural_network (https://en.wikipedia.org/wiki/Convolutional_neural_network)
2. https://pytorch.org/docs/stable/nn.html (https://pytorch.org/docs/stable/nn.html)
3. https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/ (https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/)
4. https://github.com/KwokHing/udacity-project-dog-classification (https://github.com/KwokHing/udacity-project-dog-classification)
5. Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media.
6. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2). Cambridge: MIT press.