



Dog Breed Classifier

Capstone Project Report

Machine Learning Engineer Nanodegree

Ahmed Karawia

July ~ 2021

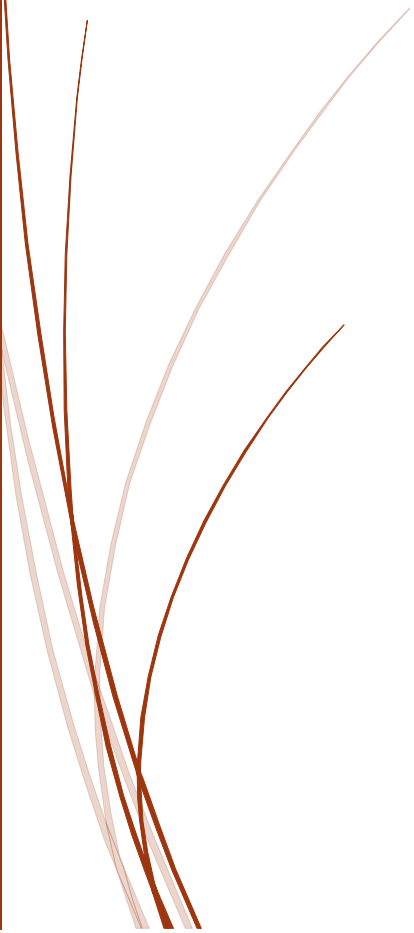


Table of Contents

Definition	2
Project Overview	2
Problem Statement	2
Metrics	2
Analysis	2
Data Exploration.....	2
Benchmark Model.....	5
Methodology	5
Data Preprocessing.....	5
Implementation.....	6
Refinement.....	8
Results	8
Model Evaluation and Validation	8
Justification	9

Definition

Project Overview

Identifying dogs according to breeds is a crucial issue for dogs' buyers, sellers, and even for veterinarian doctors. Each dog breed has certain physical and behavioural characteristicsⁱ. Furthermore, knowing the breed will provide information such as future size and required care. These all at the end are reflected on the price and value of your dog.

Having a handy solution such as an application which can identify the breed will improve and facilitate this task. This project shows a machine learning model which can predict the breed when provided a dog image and even it can predict the most resemble breed when the provided image is for human utilizing Convolutional Neural Networks (CNNs) technique.

Problem Statement

This project provides a solution to predict dogs' breed when provided with a dog image and it can predict the most resemble breed when provided with a human image.

It involves the utilization of Convolutional Neural Networks (CNNs) technique to build image classifier to achieve this goal.

Metrics

Assessing loss and accuracy as follows:

Assessing loss (Cross-entropy loss, or log loss)ⁱⁱ

- Measures the performance of classification model with output probability range between 0 and 1.
- When predicted probability shown more deviation from the actual label, the cross-entropy loss increases.

The evaluation metrics are as illustrated by the Kaggle dog breed identification competition as multi-class log loss between the target and the probability which is between 0 and 1. The goal is to minimize the loss (ideally to zero).

Analysis

Data Exploration

In this project the following datasets provided by Udacity will be used:

- Dog dataset contains 8351 images.ⁱⁱⁱ
- Human dataset contains 5750 images.^{iv}

All inputs including training, testing and validation are of image type to full fill the goal.

Dog dataset has 8351 images (6,680 Images for training the model, 836 Images for testing, 835 Images for validation). Number of images provided for each breed is not the same which leads to imbalance and there is variation in sizes and background among the images.

Human dataset has 13233 images, and all have the same size 250x250 but have different backgrounds.

```

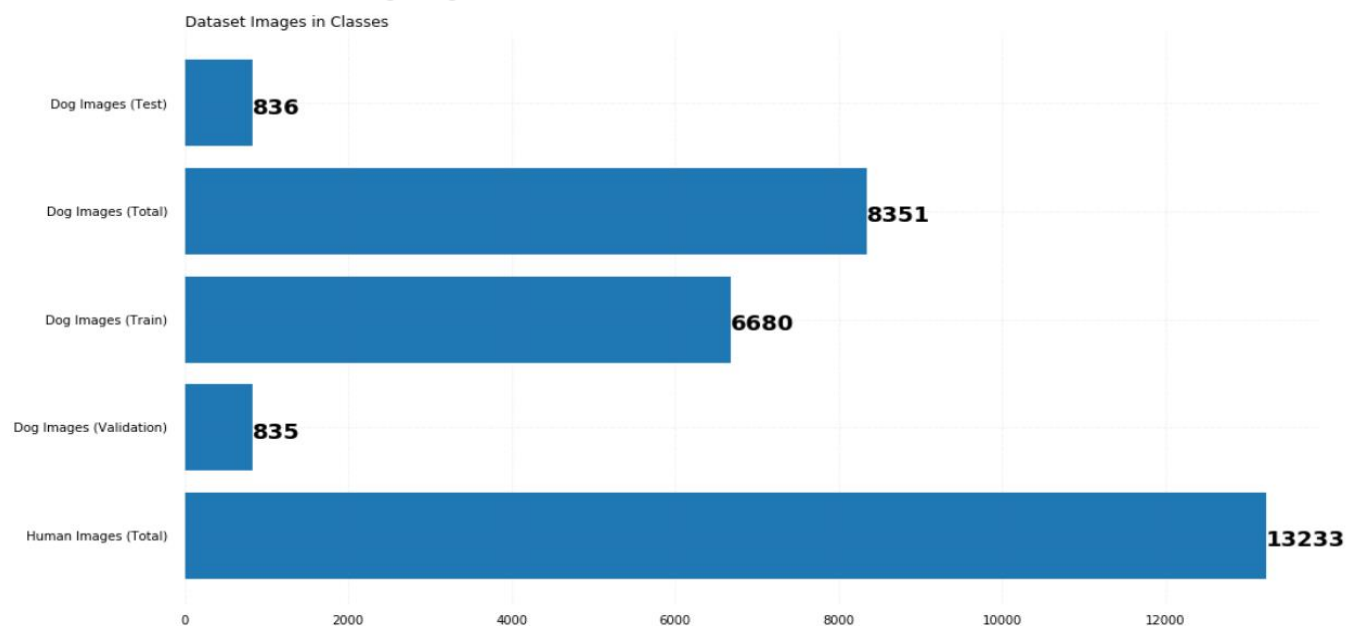
import numpy as np
from glob import glob
import os, os.path

# Load filenames for human and dog images
human_files = np.array(glob("/data/lfw/*/*"))
dog_files = np.array(glob("/data/dog_images/*/*/*"))
dog_files_train = np.array(glob("/data/dog_images/train/*/*"))
dog_files_test = np.array(glob("/data/dog_images/test/*/*"))
dog_files_valid = np.array(glob("/data/dog_images/valid/*/*"))

# 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))
print('There are %d train dog images.' % len(dog_files_train))
print('There are %d test dog images.' % len(dog_files_test))
print('There are %d validation dog images.' % len(dog_files_valid))

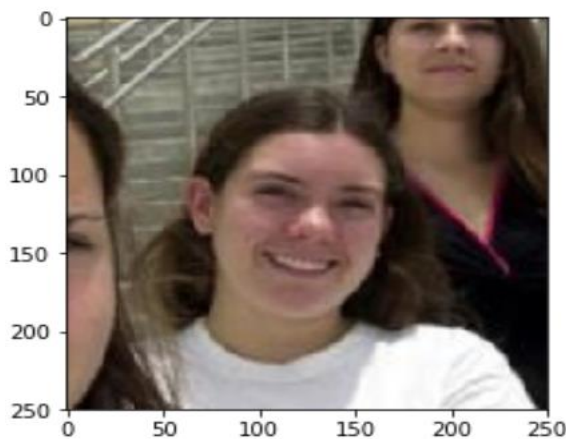
```

There are 13233 total human images.
 There are 8351 total dog images.
 There are 6680 train dog images.
 There are 836 test dog images.
 There are 835 validation dog images.



Exploratory Visualization

Most of images shows single dog or single human however, some images show both human and dogs, or more than one human.



Algorithms and Techniques

It involves the following steps:

1. Detect human using OpenCV's with Haar feature based cascade classifiers.
2. Detect dog using pretrained VGG16 model.
3. Predict the breed: classified images are passed to CNN to return the best breed match.

The implemented Convolutional Neural Network Technique involves multiple layers to extract image features such as edges (horizontal / vertical) , shapes, colours, ... etc.

The hyperparameters of the CNN model including epochs (number of cycles through which training set pass through the model), pooling, number of layers are adjusted to full fill the goal:

- First convolution layer: 32 filters with maximum pooling and stride reduced image size to 56x56.

- Second convolution layer:64 filters with maximum pooling and stride reduced image size to 14x14.
- Third convolution layer:128 filters with maximum pooling and stride reduced image size to 7x7.
- Output size: 133 assigned by 2 linear layers.
- Dropout: set to be 0.3 to avoid overfitting.

Benchmark Model

The benchmark model will be according to the Kaggle leaderboard for dog breed ^v less than 0.01 which is in the top 100 of the competition. Another benchmark would be minimum 60% accuracy.

Methodology

The project is designed according to steps provided by the standard template as follows:

- **Step 0:** Import Datasets and required libraries.
- **Step 1:** Detect Humans using OpenCV's implementation of Haar feature-based cascade classifiers.
- **Step 2:** Detect Dogs using pre-trained model (Pre-trained VGG-16 Model)
- **Step 3:** Create a CNN to Classify Dog Breeds (from Scratch).
- **Step 4:** Create a CNN to Classify Dog Breeds (using Transfer Learning).
- **Step 5:** Writing the Algorithm.
- **Step 6:** Testing the Algorithm.

Data Preprocessing

It involves resizing, cropping, stretching to improve the dataset and the outcome as follows:

- All images are resized to 244x244 to full fill the pretrained VGG.
- As for stretched and compressed images, I centred the cropped images.
- The tensor size is (3,224,224), because each image contains 3 colour layers (RGB) and corresponding pixels 224x224 containing the values, I have performed a standard normalization on all the images.

```

import os
from torchvision import datasets
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True

### TODO: Write data loaders for training, validation, and test sets
## Specify appropriate transforms, and batch_sizes

prefix = '/data/dog_images/'
train_dir = os.path.join(prefix, 'train')
test_dir = os.path.join(prefix, 'test')
valid_dir = os.path.join(prefix, 'valid')
batch_size = 20
num_workers = 0
img_transform = {'train': transforms.Compose([transforms.Resize(size=224),
                                              transforms.CenterCrop((224,224)),
                                              transforms.RandomHorizontalFlip(),
                                              transforms.ToTensor(),
                                              transforms.Normalize(mean=[0.485,0.456,0.406],
                                                                    std=[0.229,0.224,0.225])]),
                  'test': transforms.Compose([transforms.Resize(size=224),
                                              transforms.CenterCrop((224,224)),
                                              transforms.RandomHorizontalFlip(),
                                              transforms.ToTensor(),
                                              transforms.Normalize(mean=[0.485,0.456,0.406],
                                                                    std=[0.229,0.224,0.225])]),
                  'valid': transforms.Compose([transforms.Resize(size=224),
                                              transforms.CenterCrop((224,224)),
                                              transforms.RandomHorizontalFlip(),
                                              transforms.ToTensor(),
                                              transforms.Normalize(mean=[0.485,0.456,0.406],
                                                                    std=[0.229,0.224,0.225])])}]

```

Implementation

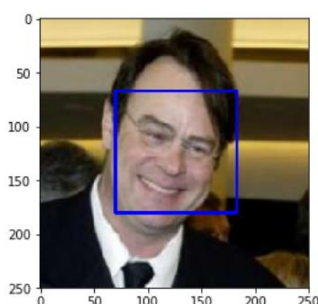
Detect Humans using OpenCV's implementation of Haar feature-based cascade classifiers

```

# returns "True" if face is detected in image stored at img_path
def face_detector(img_path):
    img = cv2.imread(img_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray)
    return len(faces) > 0

```

Number of faces detected: 1



Detect Dogs using pre-trained model (Pre-trained VGG-16 Model)

If dog detected, it will pass through VGG-16 classifier as shown below...

```
# get a dog breed based on predicted class index
import ast
import requests

label_map_source = "https://gist.githubusercontent.com/yrevar/942d3a0ac09ec9e5eb3a/raw/c2c91c8e767d04621020c30ed31192724b8630"
label_map = ast.literal_eval(requests.get(label_map_source).text)

index2 = VGG16_predict(dog_files_short[55])
dog_img_example2 = Image.open(dog_files_short[55])
plt.imshow(dog_img_example2)
print("The dog in the image is classified as {}".format(label_map[index2]))
```

The dog in the image is classified as bull mastiff



CNN to Classify Dog Breeds

As stated above, the CNN model including epochs (number of cycles through which training set pass through the model), pooling, number of layers are adjusted to full fill the goal (minimum 60% accuracy)

- First convolution layer: 32 filters with maximum pooling and stride reduced image size to 56x56.
- Second convolution layer: 64 filters with maximum pooling and stride reduced image size to 14x14.
- Third convolution layer: 128 filters with maximum pooling and stride reduced image size to 7x7.
- Output size: 133 assigned by 2 linear layers.
- Dropout: set to be 0.3 to avoid overfitting.

Refinement

For the CNN model, the following hyperparameters had been evaluated and tested:

- Filters.
- Pooling.
- Striding.

Using of a pertained model (resnet50) to avoid long time training for the fresh CNN model. The model is readily available with torch and shows excellent performance. Original model did not use the 133 dog breeds, I have replaced the final layer of 2048 with the 133 dog classes.

Results

Model Evaluation and Validation

- Scratch model with 20 training epochs reached 4.046987 as validation loss (11% accuracy).

```
# train the model
model_scratch = train(20, loaders_scratch, model_scratch, optimizer_scratch,
                      criterion_scratch, use_cuda, 'model_scratch.pt')
```

```
Epoch: 1      Training Loss: 4.881172      Validation Loss: 4.867905
Validation loss decreased (inf --> 4.867905). Saving model ...
Epoch: 2      Training Loss: 4.829570      Validation Loss: 4.787581
Validation loss decreased (4.867905 --> 4.787581). Saving model ...
```

```
Epoch: 20     Training Loss: 3.483415      Validation Loss: 4.046987
Validation loss decreased (4.076914 --> 4.046987). Saving model ...
```

```
# call test function
test(loaders_scratch, model_scratch, criterion_scratch, use_cuda)
```

Test Loss: 4.019118

Test Accuracy: 11% (93/836)

- Transferred model with 20 training epochs reached 1.531260 as validation loss (77% accuracy).

```
# train the model
model_transfer = train(20, loaders_transfer, model_transfer, optimizer_transfer, criterion_transfer, use_cuda, 'model_transfer.pt')

# Load the model that got the best validation accuracy (uncomment the line below)
model_transfer.load_state_dict(torch.load('model_transfer.pt'))
```

```
Epoch: 1      Training Loss: 4.781370      Validation Loss: 4.597419
Validation loss decreased (inf --> 4.597419). Saving model ...
Epoch: 2      Training Loss: 4.482446      Validation Loss: 4.303919
Validation loss decreased (4.597419 --> 4.303919). Saving model ...
```

```
Epoch: 20     Training Loss: 1.732892      Validation Loss: 1.531260
Validation loss decreased (1.576130 --> 1.531260). Saving model ...
```

```
test(loaders_transfer, model_transfer, criterion_transfer, use_cuda)
```

Test Loss: 1.510013

Test Accuracy: 77% (645/836)

Justification

The accuracy 77% can be further improved by increasing the number of epochs and increasing the training dataset. In addition to enhancing images of the dataset such as flipping and resizing. Moreover, fine tuning the CNN layers hyperparameters and improve the accuracy and the outcome.

ⁱ Editorial, P. M. D. (2021, February 15). *Importance of Purebred Research*. PetMD.
https://www.petmd.com/dog/general-health/evr_dg_importance_of_purebred_research.

ⁱⁱ *Cross-entropy*. Loss Functions - ML Glossary documentation. (n.d.).
https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

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

^{iv} <https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip>

^v *Dog Breed Identification*. Kaggle. (n.d.). <https://www.kaggle.com/c/dog-breed-identification/leaderboard>