Capstone Project Vinooj Kumar Naduparambil Viswanathan

Machine Learning Engineer Nanodegree May 30th, 2021

DOG BREED CLASSIFIER USING CNN

Domain Background

Dogs are one of the most common domestic animals and there are more than 200 breeds currently. It is essential to identify the dog breed to help differentiate their characteristics and select training methods based on the individual breed. Classifying the dog breeds from the image is tedious and a time-consuming process for large datasets, but with the help of Machine Learning and trained algorithms, it is much easier to predict the dog breed.

In this project, the most widely used Deep learning - Convolutional Neural Network is implemented using Transfer Learning to classify the dog breed.

Problem Statement

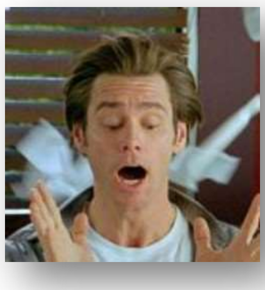
The end goal of the project is to build a ML model to process real-world, user-supplied images and predict the dog breed based on the input:

a. if a dog is detected in the image, return the predicted breed.
b. if a human is detected in the image, return the resembling dog breed.
c. if neither is detected in the image, provide output that indicates an error.

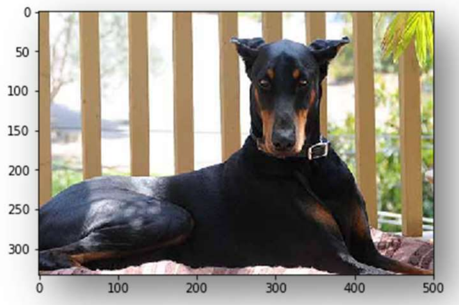
Datasets and Inputs

Human Dataset and Dog Dataset are downloaded and saved as numpy arrays. There are 13233 human images in Human dataset and 8351 dog images in Dog dataset.

Human Datasets – All the images are sorted in alphabetical order and of size 250x250. Data is imbalanced as some of the images contain more than 1 human face or half faces.



Dog dataset – Images are segregated into train (6,680 Images), test (836 Images) and valid (835 Images) folders each with 133 sub-folders corresponding to dog breeds. The dataset is imbalanced as the mean of images varies in each dog breed class.



Solution Statement

The standard procedure is to convert the images to grayscale before using any Face Detector. This image is then fed into the pre-trained face detector - OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images, which returns a Boolean value as output (TRUE if a human face is detected).

Load and transform the data into Train, Valid and Test Datasets. Select pre-trained model VGG16, resize image to 224x224 as VGG16 model requires input to be of 224x244 image size. Perform image augmentation on all three datasets by random flipping, random rotation, and normalization of the given image.

Benchmark Model:

- The CNN model created from scratch must have accuracy of at least 10%. This can confirm that the model is working because a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

  CNN Model scratch: Benchmark Test Accuracy > 10%

- The CNN model created using transfer learning must have accuracy of 60% and above

  CNN Model using Transfer Learning: Benchmark Test Accuracy > 60%

Evaluation Metrics

Multi class log loss will be used to evaluate the model. Log loss takes into the account of uncertainty of prediction based on how much it varies from actual label.

Project Design

- Download and import Dog and Human Datasets.
- Detect Human faces in images using Open CV's implementation of Haar feature-based cascade classifiers.
- Import pre-trained TorchVision and VGG-16 models and implement a Detect Detector.
- Create a CNN to Classify Dog Breeds (from Scratch) with at least 10% test accuracy.
- Create a CNN to Classify Dog Breeds (using Transfer Learning) with at least 60% test accuracy.
- Implement the Algorithm to accept a file path to an image and determine whether image contains a dog, human or neither. Then,
  d. if a dog is detected in the image, return the predicted breed.
  e. if a human is detected in the image, return the resembling dog breed.
  f. if neither is detected in the image, provide output that indicates an error.

References

- GitHub Repository:
  https://github.com/udacity/deep-learning-v2- pytorch/blob/master/project-dog-classification/

- Resnet101:
  https://pytorch.org/vision/stable/_modules/torchvision/models/resnet.html#resnet101

- Transfer Learning:
  https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

- GitHub Imagenet:
  https://github.com/pytorch/examples/tree/master/imagenet

- PyTorch Documentation:
  https://pytorch.org/docs/master/

- CNN guide:
  https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neuralnetworks-the-eli5-way-3bd2b1164a53