Capstone Project

Data Science Nanodegree

Ali UPT

November 18th, 2021

Project Overview

This project uses Convolutional Neural Networks (CNNs)! In this project, I built a ML algorithm to process real-world, user-supplied images. By giving an image, the algorithm gives an estimate of whether it is a dog and its breed or it is a human face or neither. If supplied an image of a human, the code will also identify the resembling dog breed. This is a multi-class classification ML problem that use supervised machine learning to unravel the patterns.

Capstone Project | Ali UPT

Data Science Nanodegree | November 18th, 2021

Problem Statement

The ultimate goal of this project is to build a ML algorithm that can be fed any images so that it can tell us if it is a human, dog or unrelated object.

The algorithm has two detectors:

Dog face detector: Given an image of a dog, return the estimate of the predicted breed.

Human face detector: If supplied an image of a human, return the resembling dog breed.

Used Datasets

I used the following two datasets for training the models:

- The dog dataset = 8351 images with 133 classes

  This dataset included 6680, 835, 836 images for training, validation and test

- The human dataset = 13233 images

System requirements and libraries and Instruction

In order to execute the project, you may need Python 3x, Jupyter Notebook, OpenCV and PyTorch. The detailed list of required libraries can be found in the Jupyter Notebook given in this repository. You can run the Jupyter Notebook using python 3x.

The project Road map

The project includes the following main steps:

- Step 0: Import Datasets

I imported a dataset of dog images. I populated a few variables through the use of the load_files function from the scikit-learn library:

train_files, valid_files, test_files - numpy arrays containing file paths to images

train_targets, valid_targets, test_targets - numpy arrays containing onehot-encoded classification labels

dog_names - list of string-valued dog breed names for translating labels

- Step 1: Detect Human

Subsequently, the OpenCV's implementation of Haar feature-based cascade classifiers was used to detect human faces in images.

- Step 2: Detect Dogs

For detecting different breed of dogs, I used a pre-trained ResNet-50 model to detect dogs in images. The provided block of codes, first downloads the ResNet-50 model along with the weights on the trained ImageNet.

- Step 3: Create a CNN to Classify Dog Breeds (from Scratch)

Through steps one and two, we defined functions that can detect human and dogs faces. Subsequently, I created a CNN that classifies dog breeds. We built a CNN model from the scratch which attained the test accuracy of at least 1%. For the classification purpose, I constructed a CNN with 3 convolutional layers along with 3 max pooling layers. I used max pooling layer to reduce the dimensionality. Additionally, I increased the intralayer filter size from 16 to 64 to boost the accuracy. I used Relu as an activation function to improve neural networks efficiency by speeding up training.

- Step 4: Use a CNN to Classify Dog Breeds (using Transfer Learning)

In this step, I used transfer learning to train a CNN in order to reduce training time without sacrificing accuracy

- Step 5: Create a CNN to Classify Dog Breeds (using Transfer Learning)

In this step, I used transfer learning to create a CNN, which capable of detecting dog breed from raw images. By defining an appropriate architecture of model, I could obtain the accuracy of around 86%. In this step, I used Xception bottleneck features from a pre-trained model.

- Step 6: Write Classifier Algorithm from scratch

For this step, I wrote a classifier algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither. The algorithm is capable of detecting dog images and its breed, human images and its resembling dog breed. Additionally, the classifier can tell the user if the images is neither human nor dog.

- Step 7: Test the Classifier Algorithm

Finally, I tested the Algorithm with some random images found online.

Evaluation Metrics

To evaluate the multiclass classifier, I used the Accuracy, Precision, Recall and F1 score. I used classification_report for obtaining this evaluation metrics. By calculating, how far each prediction is from the true label, the algorithm learns from each prediction and minimize the distances in each iteration.

References:

https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/

https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2

https://en.wikipedia.org/wiki/Precision_and_recall