Machine Learning Engineer Nanodegree

Dog Breed Classifier

Capstone Project Report

Albert Kuc - 2021.01.03

Project Overview

The purpose of the project is to create an application capable of categorizing dogs based on their breed. This dog breed classification is performed for an image provided by the end user. The model is capable of recognizing and assigning dog breed to any uploaded image.

In an event where a human image is provided, the application recognizes there is no dog but human. In this scenario the app informs the user what is the most associated dog breed to the person in the image.

Problem Statement

Based on the project overview we can differentiate two main requirements for the application. At first, it has to be able to distinguish between human and dog, based on which it communicates the appropriate final output to the end user. A third possibility can occur, when no human nor dog is detected, which needs to be considered as well.
The above is achieved by defining human face detector and dog detector.

The second part of the application is to assign associated dog breed to an image provided. At this stage the algorithm assigns dog breed to any content present in the image. No matter the image content the app returns a predicted dog breed. Depending on the result of the previous step (first part above), this will be differently communicated to the app user.
Dog classification is performed using convolutional neural networks, where the model is trained using dog images to learn specific characteristics of each breed.

Metrics

Common metric to evaluate classification model performance is accuracy. It is a ratio of number of correct predictions to the total number of predictions.

$$accuracy = \frac{correct\ predictions}{total\ number\ of\ predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

The result is reported in percentages.

Accuracy is a basic metric which works well unless the input dataset is heavily unbalanced.

Data Exploration

Datasets used in this project are provided by Udacity.

Dog dataset is divided into train, validation and test categories. Inside there are 133 folders which contain images of each dog breed. The number of images in each of the 3 folders are respectively 6680, 835 and 836 files which is equivalent to 80%, 10% and 10%.

Images are of unequal size and shape. Some images contain more than a single dog, humans or dressed dogs. The number of images is not evenly distributed within breed folders.
Out of 6680 images for training the highest number of images represent Alaskan malamute (77) and the lowest Norwegian buhund (26). Based on that we see that the dog dataset is not ideally balanced, as there are almost 3x more images of the Alaskan Malamute than Norwegian buhund.

Human dataset contains 13233 images. The images are of square shape. Some images contain more than a single person or are taken during sport events which in some cases results in dynamic facial expressions. The total number of directories in the human dataset is 5749 which represent different people in the dataset. Most directories consist of a single image of a person. On the contrary there are 530 images of George W. Bush (4% of the entire dataset), which states the dataset is not well balanced.

Algorithms and Techniques

Haar Cascades algorithm is used to define human `face_detector`. It is an object detection method imported from OpenCV library. The uploaded image is first converted to grayscale and detectMultiscale function is executed. The result is a numpy array of detected faces, where each row corresponds to a detected face. If there are no faces detected the length of the result is 0.

`dog_detector` is defined using *VGG-16*, which is a pre-trained model on *ImageNet* dataset, used for image classification. Given an image the model returns a prediction out of 1000 possible categories, for an object that is present in the image. Image prediction is an integer where numbers within 151-268 correspond to dogs.

Model from scratch is a simple convolution neural network algorithm created from scratch for the project. The model consists of a combination of convolution layers, activation functions, pooling layers, dropout layers and linear layers in defined sequence. The main goal of this model was to establish certain learning capability which leads to 10% accuracy during evaluation.

Model transfer is another pre-trained model on _ImageNet_ dataset used for image classification. The model used for this task is ResNet50 where 50 corresponds to model architecture which is 50 layers deep. The model outputs 1000 categories, which is tuned to 133 to match the number of dog breeds. The model then is trained on the provided dog dataset, however only weights of the additional layer are allowed to change.

Benchmark

The CNN for model scratch is required to result in performance with accuracy of minimum 10%. The value is achievable only in case the model is truly capable to learn during training.

The pre-trained model is required to result in accuracy over 60%.

Data Preprocessing

- Face detector (Haar Cascades): input images are transformed to grayscale.
- Dog detector (VGG-16): images are resized, converted to tensor and normalized.
- Dog classification models: all images are resized to 224x224.
    - Loading training dataset includes random image rotation, resize with crop, vertical and horizontal flip and normalization.
    - Loading Test and validation datasets consist of resize, center crop and normalization.

  Images are converted to tensor prior normalization.

Implementation

For the app implementation the majority of work was done for the model from scratch.
Face detector and dog detector are both described in Algorithms and Techniques.

Model from scratch

The final architecture was defined with experimental trials with simple CNN architecture examples, found in resources such as Deep Learning literature, online tutorials or other github profiles. Initially each model was trained with 5 `epochs` and evaluated. Test accuracy varied with mostly disappointing results while in few cases the model didn't improve during training.

1. The basic structure for convolution steps is common (convolution layer, activation function, pool layer). Initially `kernel size`, `stride` and `padding` were equal within all convolution layers. One of resources suggested to use higher values for the first convolution which is used in the final model.
2. Few `dropout` values or lack of dropout were tested prior to the linear layer.
3. `Softmax` activation function was replaced with `LogSoftmax` based on the performance.
4. Adding a pool layer, either `AvgPool2d` or `MaxPool2d`, prior to the first linear layer made a noticeable difference in the structure.
5. Also tested a few more convolution and linear layers but haven't found any improvement there with the tested parameters.

Note: The following details are not directly related to model structure however were included in model from scratch trails:

6. The best architectures were tested against `Adam` and `SGD` optimizers at `lr=1e-3` and `lr=3e-3`. The second learning rate value was suggested by one of the industry literature, but didn't lead to noticeable improvement.
7. Various data transformers affected the performance:
    - `RandomResizedCrop` resulted in higher accuracy than combination of `Resize` with `RandomCrop` or `CenterCrop` for the training dataset.
    - `ColorJitter` and `RandomGrayscale` didn't seem to have a positive effect either, at least not at the combination used. Finally both were dropped from the transform structure.
8. DataLoader's `batch_size` for training data set was tested with higher value but didn't result with an improvement as well.

The best model achieved 9% once trained with 5 epochs. The model trained with 20 epochs stopped improving which indicates overfitting. Model parameters were saved to the external file every time the validation loss improved, hence it captures the best model parameters.

Model trained

`ResNet50` model used as a pre-trained model required replacing the last linear layer with output of 1000 (representing 1000 categories), by another linear layer with output of 133, to match breed categories. The weights of the new layer were raw and required training. To prevent the pre-trained architecture from adjusting during the additional training, this section of the model was frozen.

The final model results in 81% accuracy.

Model evaluation and validation

Face detector report 99% accuracy with testing using human images. It has almost perfect accuracy with detecting true positives. The detector is less accurate with images without humans as accuracy for true negatives is 91%.

Dog detector reported 100% accuracy for both true positives and true negatives.

Both above tests were performed on a small dataset sample.

CNN model from scratch achieves accuracy of 14% when tested using previously unseen images. The model is trained with 20 epochs with validation loss monitored at each iteration. Model parameters are stored in an external file for the lowest validation loss achieved, which represents the best model parameters, while training with 20 epochs.

Pre-trained model achieves accuracy of 81% using test dataset. The model additionally trained with 7 epochs. Validation loss is monitored during the training and the best model parameters are stored in an external file.

Justification

Model from scratch meets the benchmark requirement of 10% accuracy (14%). At this step the model proves to provide enough training capability. This is a result of not only improvement of model architecture, but also a result of establishing better structure for loading dataset, selecting optimizer and defining training sequence.

Pre-trained model meets the benchmark requirement of 60% accuracy (81%). It is the final model selected for the proposed dog breed application. The pre-trained architecture is adapted to meet the project requirements. It uses the same data loaders, optimizer, criterion and training method defined during trials with model from scratch.

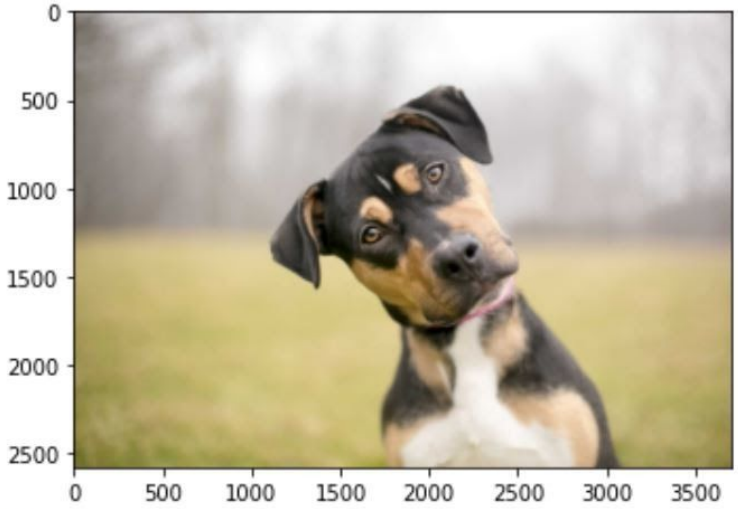
Conclusions

Dog breed app is potentially ready for deployment as it meets the benchmark score, although there is still room for improvement.

- Different face detector might lead to higher performance. Currently there is a small chance that a dog will be detected as human.
- The pre-trained model results in 81% which could be further investigated as well.
- Datasets available for the project are inconsistent and not ideally balanced.

Worth to notice that there are few breeds which are very similar and a single image might not be enough to determine the correct breed.

Sample user output:




```
Hi dog!
You seem to be Entlebucher mountain dog.
---------------------------------
```


```
Hello, human!
You look like Alaskan malamute.
---------------------------------
```