Machine Learning Nanodegree Capstone Project

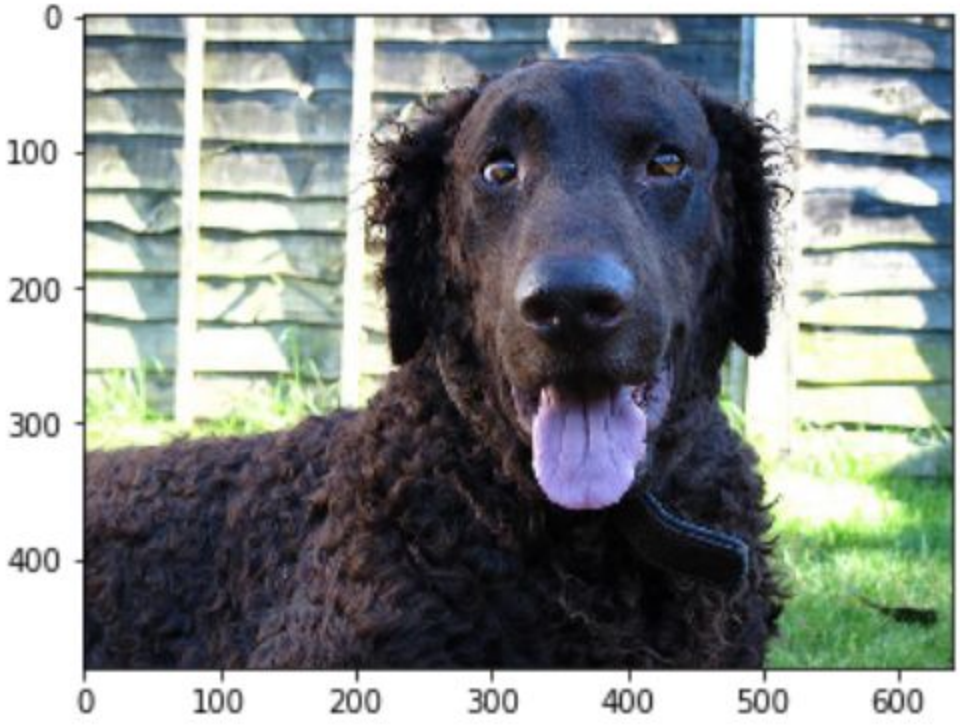
Dog's Breed Classifier

Satya Prakash Biswal **10.13.2020**

Definition

Project Overview

This is a convolutional neural network project where user supplied images are identified as human or dog. And then classified as the closely resembling dog breed. This algorithm basically identifies a dog's breed after being trained on thousands of pictures of the dog. The main goal of this project is, to understand the challenges involved in piecing together a series of models designed to perform various tasks in a data processing pipeline.




```
Dogs Detected: Curly-coated retriever
```

According to the database of The Fédération Cynologique Internationale is the World Canine Organisation, 368 dog breeds exist. Classification of the dog breeds became important centuries ago, the breeders attempted to select dogs based on desirable characteristics and strengths. Modern dog breeds formation was driven by dog shows in the late 19th century. Breeds classification remains a relevant problem for the dog owners who search for a show-class puppy or for a working dog with particular performance characteristics. Deep Learning has been proved to be suitable for image classification problems, therefore the task can be tackled by a Deep Learning algorithm.

Problem Statement

The primarily investigated problem is dog breed classification via Convolutional Neural Network (CNN). The project also covers the tasks:

- Dog face detection
- Human face detection

The steps to achieve the desired results:

1. Import Datasets
2. Detect Humans
3. Detect Dogs
4. Create a CNN to Classify Dog Breeds
5. Create a CNN to Classify Dog Breeds
6. Write your Algorithm
7. Test Your Algorithm

The expected behaviour of the application:

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

Metrics

The evaluation metrics that can be used to evaluate the performance of the machine learning models are:

● Accuracy: The ratio of correct predictions to the total size of the data (i.e. (TP+TN)/Data Size)
● Recall: The ratio of true positives to the true positive and false negative (i.e. TP/(TP+FN))
● Precision: The ratio of true positives to the true positive and false positive

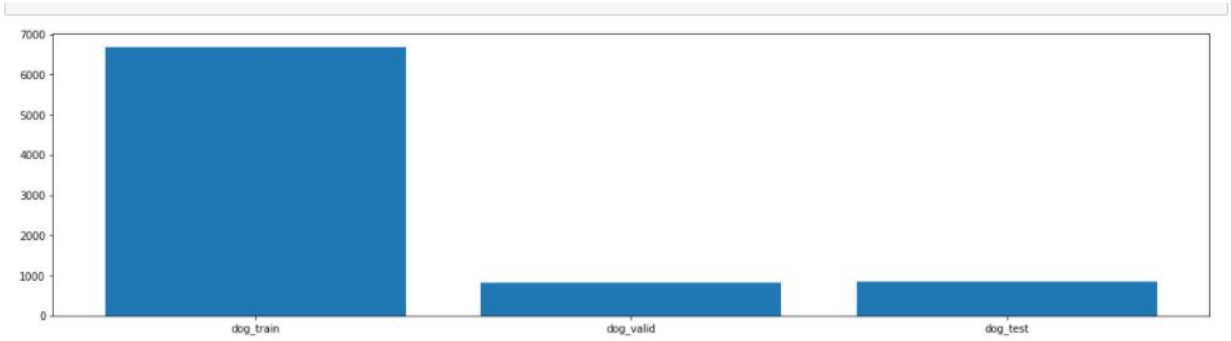
In our case, we will be using the accuracy as the metric of measurement to evaluate the performance of the model

Analysis

Data Exploration

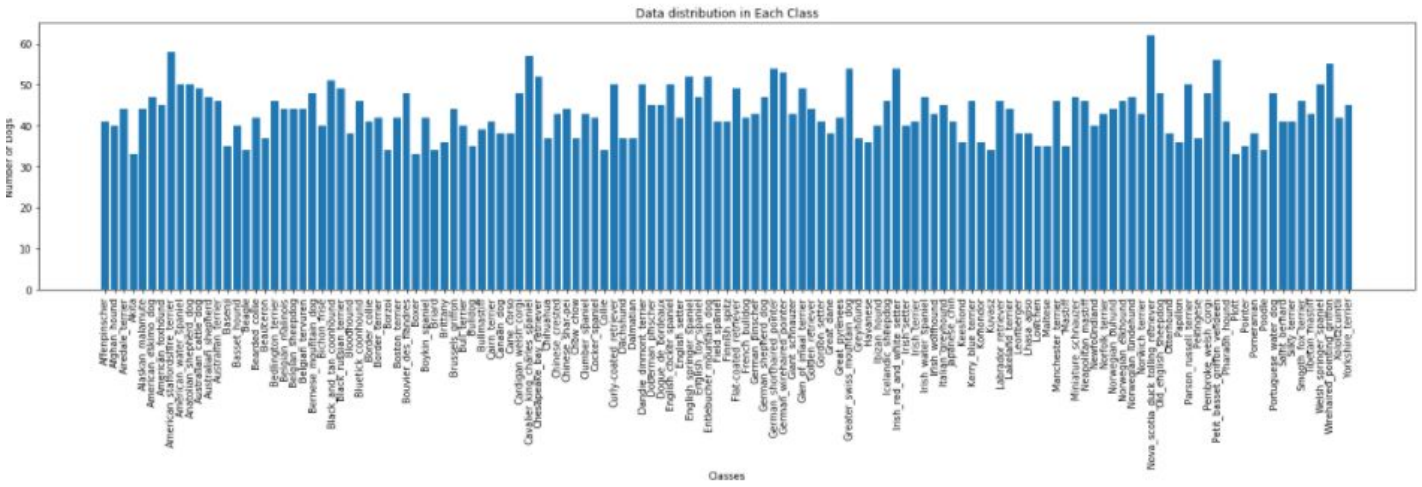
The dataset contains the images of Dogs and Humans. There are a total of 133 breeds, 8351 images for dogs. Using these images as data, it has to be processed according to our needs and a model has to be designed to train our machine. On making the analysis on the data, we see that the resolution of the images are not the same for all images of dogs in their respective breeds. In our case, we observe that the split of train and test data is 90%-10%. The resultant split of data can be observed in the below graph. From the below plot we can observe that a total of 6680 images will be used to train our machine, to further fine-tune the parameters we use another 835 images for validating it. And, lastly, we will be using 836 images to test our model's performance for the evaluation of metric i.e. Accuracy.

Two image sets were selected for the project:
1. Dog dataset [4]. The folder contains 133 folders, each corresponding to a different dog breed. There are 8351 total dog images. The dataset is split into train (6680 images), test (836 im.) and validation (835 im.)



2. Human dataset [5] for testing on human images.
There are 13233 total human images. The images are organized in folders by the names. There are 5750 different people (1 or more images per person). All images are the same in size - 250 by 250 pixels

Distribution of breed wise counts. This looks like a uniform distribution across all breeds.



Algorithms and Techniques:

The algorithm used outputs an assigned probability for each class; this can be used to reduce the number of false positives using a threshold.

The following parameters can be tuned to optimize the classifier:

- Classification threshold
- Training parameters
    - Epochs
    - Batch Size
    - Learning Rate
    - Momentum
- Neural network architecture
    - Number of layers
    - Layer Types
- Preprocessing parameters

During training, both the training and the validation sets are loaded into the RAM. After that, random batches are selected to be loaded into the GPU memory for processing. The training is done using the Mini-batch gradient descent algorithm

Benchmark

The benchmark model helps us to make a comparison and reduce the overfitting or underfitting condition and tune the model for a better outcome. Logistic Regression, KNN are such examples of the benchmark. We can also use the predefined image classifiers such as ImageNet, ResNet, VGG16, etc. to classify our images and later optimize our pipeline for better evaluation of metrics.

The solution model was compared to a benchmark CNN with:

- 4 convolutional layers
- flattening layer,
- drop-out layer,
- fully-connected layer,
- ReLU,
- drop-out
- fully-connected layer.

The architecture was designed to follow the common structure of CNN classifiers and VGG in particular: the first layers are convolutional layers, the number of features increases in higher layers. The feature extractor is followed by flattening of the feature tensor and the classifier: 2 dense layers with activation functions. The kernel size of (3, 3) is the most popular, having a number of features as a power of 2 is also a standard.

Methodology

Data Preprocessing

The images given as input to the model were preprocessed before passing it to the network. Images are resized to 258 then cropped to a size of 224 by 224. Since the images come in various sizes resizing and cropping them to an image of 224 by 224 is a considerable size especially while training, in terms of compute. The augmentation is also useful for generalizing the model, reduce overfitting and have better predictions. The transformation used are (only for the training images; we don't need to modify the test/val images): horizontal flip, random rotation by (-60 , 60) degrees. Then the image is converted into tensors. In this step, there are some parameters we can choose, like the parameter for resizing, center crop, and normalize methods.

Implementation

For the final model using Tranfer Learning I used ResNet50. I changed the last layer of the architecture to:
Linear(in_features=2048, out_features=133, bias=True)
This modification is made because the output nodes of ResNet50 is 2048 but we have 133 dog breeds.
The loss function used is: Cross Entropy.
The optimizer used is: Stochastic gradient descent (SGD).
The learning rate is set to 0.001.

Refinement

The initial solution was a baseline CNN with 4 convolutional layers. It was required to be better than a random prediction with 1/133 accuracy and achieve test accuracy greater than 10%. Passing those requirements meant that the data preprocessing is reasonable.

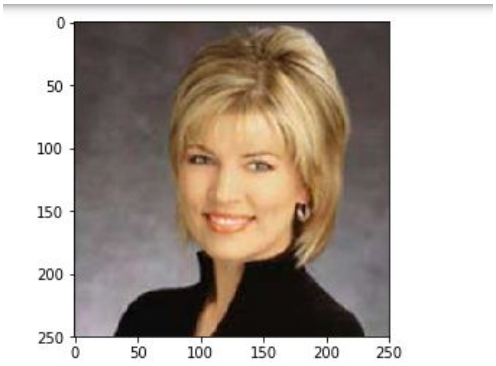
For the custom CNN and ResNet50 models, some hyperparameters tuning was made during all the training and design process.
These are some parameters toned during the process:
• Training epochs, learning rate, dropout value, etc.

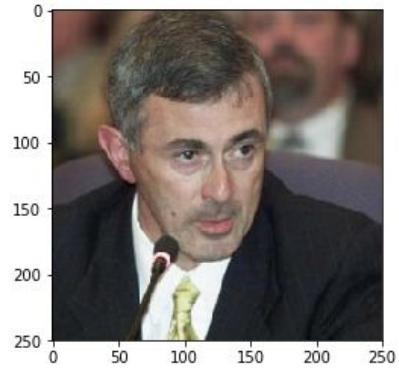
Results

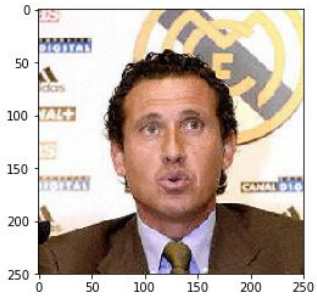
Model Evaluation and Validation

In the implementation phase of the pre-trained model VGG 16, we test the prediction of the model and we also observe that the results show the right prediction value of the dog. On implementing the custom Convolutional Neural Network model, we construct out the relative stack of layers required for our purpose and tune the necessary hyperparameters of the model. On testing the performance of the custom model through the metric evaluation i.e. accuracy in our case, we observe that the results give us a result of 15% in terms of accuracy. The project had the condition to achieve the results of more than 10% accuracy. After the implementation and training of the transfer learning model for a total of 30 epochs, we need to evaluate the performance of the given test data of dog images so that the model should predict the breed of the dog with utmost accuracy. We see that the model achieved an accuracy of 68%, and we see that there is a considerable rise in the performance of the model after combining the predefined model with our custom model.The human face detector was validated on the first 100 images of human and dog datasets: 98% human faces were detected in the human subset and 17% human faces were detected in the dog subset. The dog detector was validated on the first 100 images of human and dog datasets as well: 96% dogs were detected in the dog subset and 0% dogs were detected in the human subset.

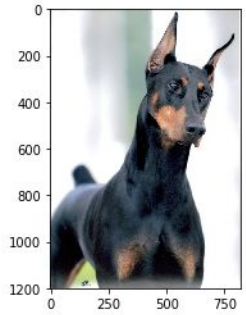
At last, I used 6 images to test the model and found that the model correctly classified the images



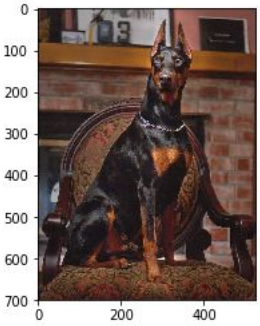




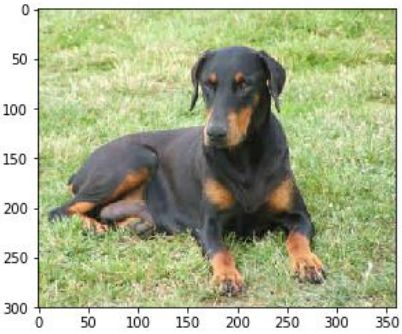
```
Hi, human
 You look like a {'American eskimo dog'}
```



```
Hi, dog
 Your predicted breed is: {'American eskimo dog'}
```



```
Hi, dog
 Your predicted breed is: {'Greyhound'}
```



```
Hi, dog
 Your predicted breed is: {'Beauceron'}
```

Justification

The model's performance can be improved but it has already shown a higher accuracy of around 68% than a random classifier and the benchmark model that showed an accuracy of around 18%. So we can conclude that the final solution is capable for the dog classification problem.

Reflection

The project was conducted in 7 steps:

- Step 0: Import Dog and Human datasets
- Step 1: Detect humans by OpenCV's implementation of pretrained Haar feature-based cascade classifiers.
- Step 2: Detect dogs by trained on ImageNet VGG-16 model.
- Step 3: Create a CNN from scratch to classify dog breeds. Train and test it.
- Step 4: Create a CNN to classify dog breeds using Transfer Learning. Train and test the solution model.

- Step 5: Design an algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither. Then, - if a dog is detected in the image, return the predicted breed. - if a human is detected in the image, return the resembling dog breed. - if neither is detected in the image, provide output that indicates an error.
- Step 6: Test the algorithm on sample images.

The hardest part of the project was step 3: find the suitable optimizer and the architecture leading to the loss' decrease. The most enjoyable part of the project was step 6: test the app of the images of my own dog.

Improvement

Below are some steps that could result in a better model.

- Model Tuning:
  - Increase the number convolutional layer
  - Adding more dropout layer
  - Convolutional Network Parameters
  - Altering the Multi-Layer Feed-Forward NN at the end of the network

- Provide probabilities of top-5 predicted breeds
- Explain the net's decision using interpretability techniques such as Saliency maps, Grad-CAM, Ablation-CAM, and Occlusion. It would be insightful for classification of mix-breed dogs and funny for human "breed" predictions.
- Build a nice web app to upload an image and get an instant result without running the notebook.

Conclusion

In this project, we see the implementation of Dog Breed Classification. This gives us a detailed description of how the classification of dogs based on their breeds are carried out and implemented.

From the data, we study and make an analysis of the split of data, type of data, distribution of data, and also the visualization of how the data is distributed. From the split, we see that 80% of the data is used for training, 10% for validation, and the remaining 10% for the test data.

In the preprocessing phase, we see how the data has been preprocessed using resize, center crop, and normalization.

In the implementation phase, we implemented the pre-trained model, custom model, and transfer learning model. In the pre-trained model, we use the model to directly make a prediction on the given data. Building our own custom model, we see that the model is implemented using the convolutional architecture.

In the transfer learning, we see the involvement of a custom model with the pre-defined model and training it for the whole data and see a significant rise in the accuracy level to an output of 68% in test accuracy. On making the comparative study with the benchmark model, we see our model has outperformed the benchmark and also satisfying the condition of above 60% accuracy.

References

1. **http://www.fci.be/en/Nomenclature/Default.aspx**
2. **https://en.wikipedia.org/wiki/Dog_breed**
3. **Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 2015**
4. **Marcel Simon, Erik Rodner, Joachim Denzler, ImageNet pre-trained models with batch normalization, ArXiv, 201**