MACHINE LEARNING ENGINEER NANODEGREE

CAPSTONE PROJECT REPORT

DOG BREED CLASSIFIER

1. PROJECT OVERVIEW

Machine learning is an application where computers are able to think and act with less human intervention; deep learning is about computers learning to think using structures modelled on the human brain.

Deep learning can analyse images, videos, and unstructured data in ways machine learning can’t easily do. In this project we are performing Image Processing.

Image Processing is a very powerful technique used today to convert an image into a form which is either digital or analog so that it can be used to extract some important and useful data. Image processing when used in ML algorithms such as CNN can be used to get very interesting results such as image recognition or creating a model to predict some feature from image.

The problem we are going to solve using image classification is to train a deep learning model to identify dog breed.

This can have many applications that can be both helpful and fun especially for dog lovers, dog related industries and companies that offer products and services for dog owners. It also has a fun part where when the model is given a human image it mentions dog breed that the human resembles to.

Source: *https://flatironschool.com/blog/deep-learning-vs-machine-learning*

2. PROBLEM STATEMENT

- In this project we build a model using deep learning algorithms and image processing from scratch to predict the breed of 133 breeds of dogs by taking their image as input.
- Then use transfer learning to build a model that the same task in lesser training time.
- We finally create an app that takes an image, if it is of a dog it identifies the dog breed. If the image contains a human face the app returns the breed of dog that most resembles this person.
- The project uses Convolutional Neural Networks (CNNs) implemented on Pytorch.

3 METRICS

For a classification problem accuracy will be a straight forward method to evaluation the model. Accuracy is the measure the number of correct decisions the classifier makes, divide by the total number of test examples. We could Recall, Precision, and F1- Score, but the accuracy will be enough for this problem statement.

4 & 5 DATA EXPLORATION AND VISUALIZATION

The following datasets provided by Udacity will be used:

- The dog dataset with 8351 dog images

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

- The human dataset has 5750 human images

Source: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip

- Other datasets available:
  - Kaggle dataset : https://www.kaggle.com/c/dog-breed-identification/data
  - Stanford University, Dogs Dataset: http://vision.stanford.edu/aditya86/ImageNetDogs

THE DOG DATASET

- The dataset has image of 133 breeds of dogs. It a training set and a test set of images of dogs. Each image has a filename that is its unique id.
- Number of dog images in the dataset

  Total Files in /data/dog_images/train/: 6680

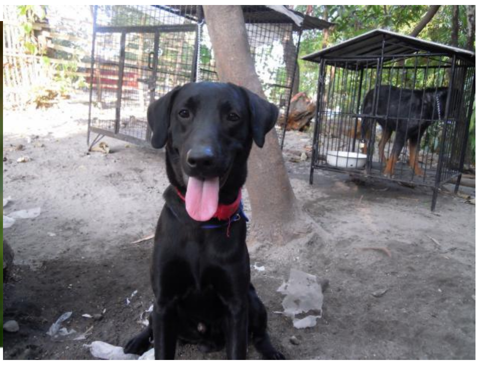
  Total Files in /data/dog_images/test/: 836

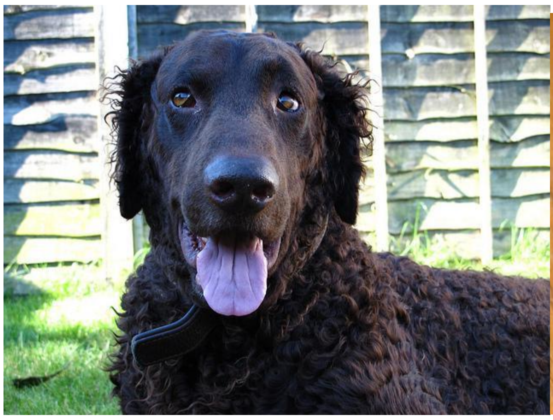
  Total Files in /data/dog_images/valid/: 835

  Images are stored and read in a .jpg format.

Labrador_retriever Brittany

Labrador_retriever

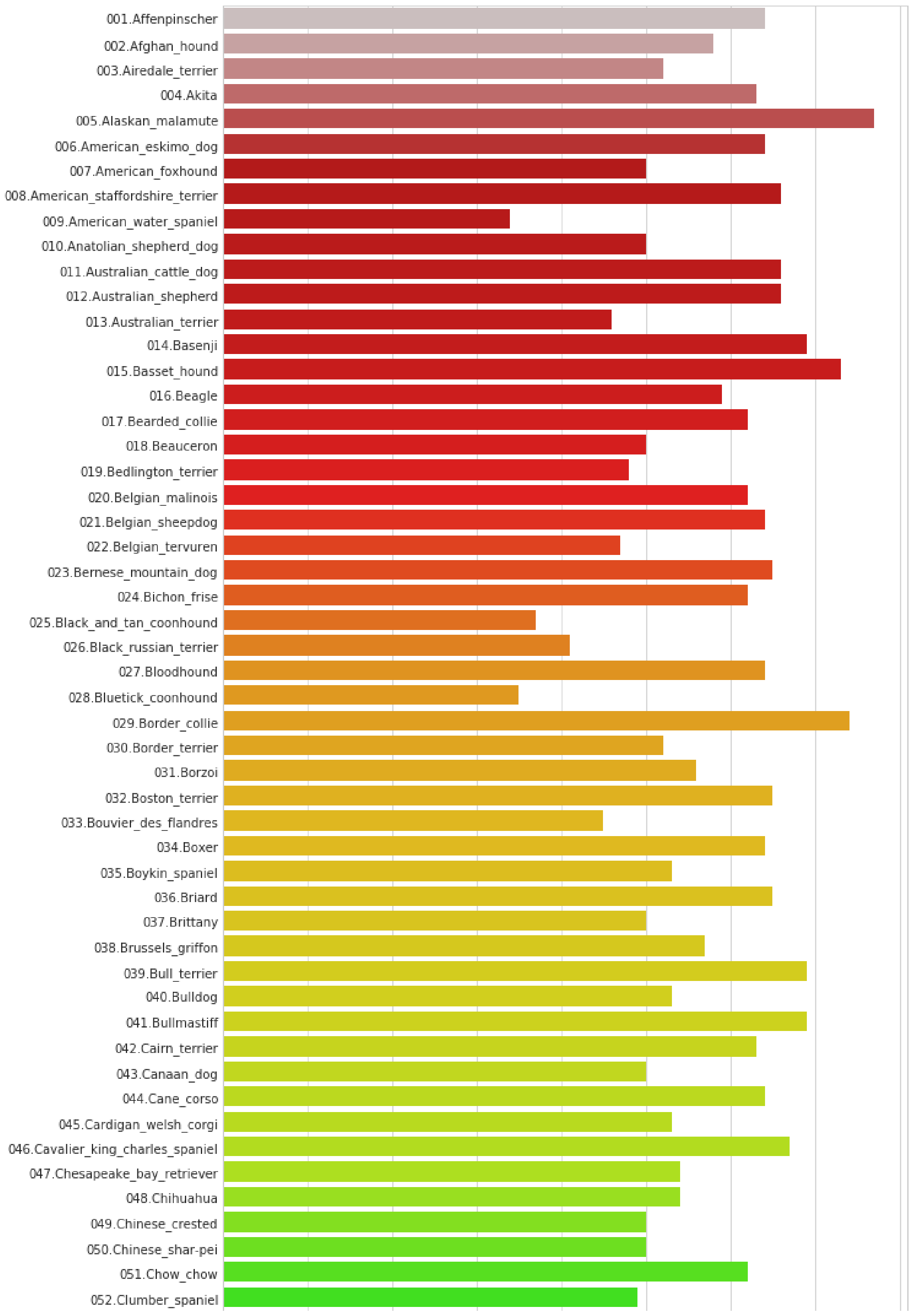

Labrador_retriever


Curly-coated_retriever

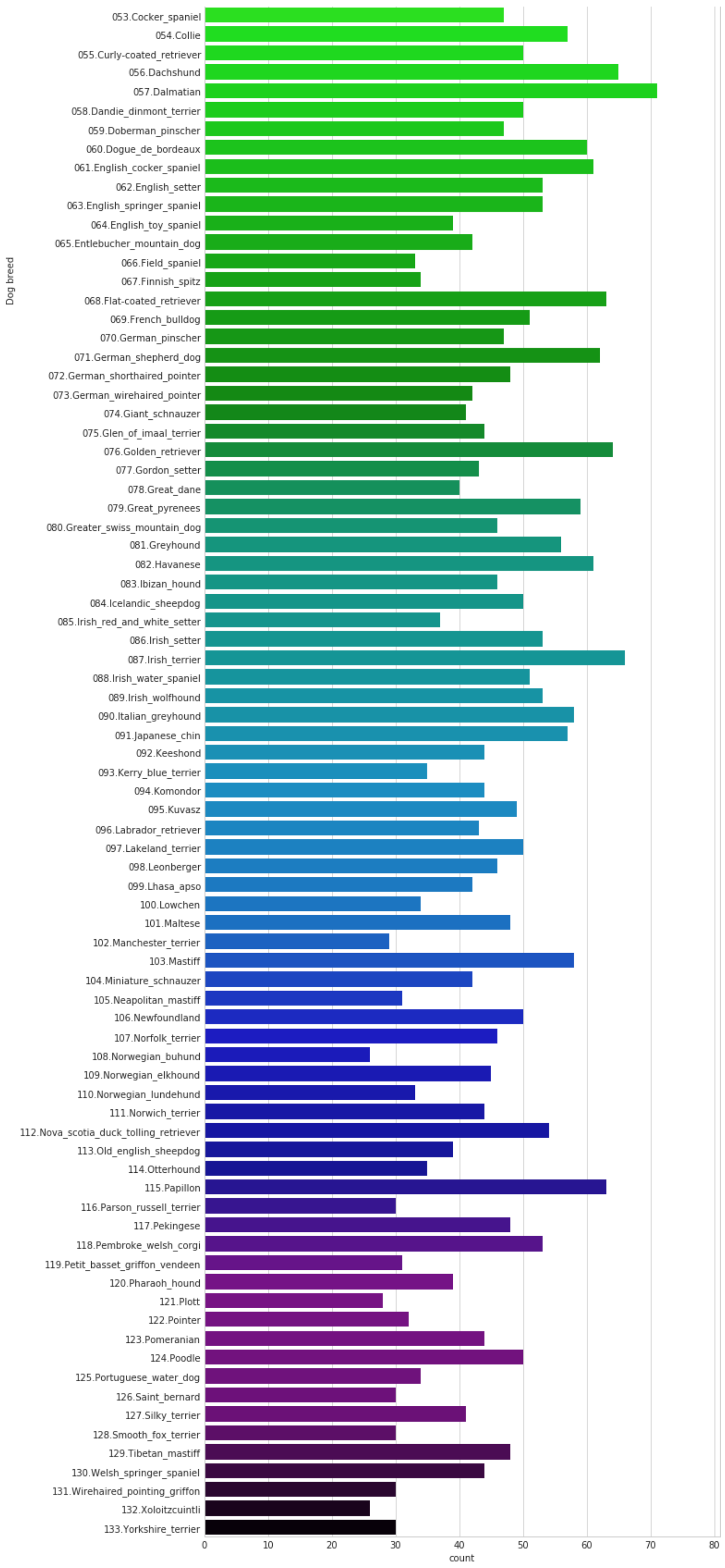
American_water_spaniel

Welsh_springer_spaniel

- Number of classes in the dataset and the number of each classes is shown in the plot in the next 2 pages.



001.Affenpinscher
002.Afghan_hound
003.Airedale_terrier
004.Akita
005.Alaskan_malamute
006.American_eskimo_dog
007.American_foxhound
008.American_staffordshire_terrier
009.American_water_spaniel
010.Anatolian_shepherd_dog
011.Australian_cattle_dog
012.Australian_shepherd
013.Australian_terrier
014.Basenji
015.Basset_hound
016.Beagle
017.Bearded_collie
018.Beauceron
019.Bedlington_terrier
020.Belgian_malinois
021.Belgian_sheepdog
022.Belgian_tervuren
023.Bernese_mountain_dog
024.Bichon_frise
025.Black_and_tan_coonhound
026.Black_russian_terrier
027.Bloodhound
028.Bluetick_coonhound
029.Border_collie
030.Border_terrier
031.Borzoi
032.Boston_terrier
033.Bouvier_des_flandres
034.Boxer
035.Boykin_spaniel
036.Briard
037.Brittany
038.Brussels_griffon
039.Bull_terrier
040.Bulldog
041.Bullmastiff
042.Cairn_terrier
043.Canaan_dog
044.Cane_corso
045.Cardigan_welsh_corgi
046.Cavalier_king_charles_spaniel
047.Chesapeake_bay_retriever
048.Chihuahua
049.Chinese_crested
050.Chinese_shar-pei
051.Chow_chow
052.Clumber_spaniel


Dog breed
053.Cocker_spaniel
054.Collie
055.Curly-coated_retriever
056.Dachshund
057.Dalmatian
058.Dandie_dinmont_terrier
059.Doberman_pinscher
060.Dogue_de_bordeaux
061.English_cocker_spaniel
062.English_setter
063.English_springer_spaniel
064.English_toy_spaniel
065.Entlebucher_mountain_dog
066.Field_spaniel
067.Finnish_spitz
068.Flat-coated_retriever
069.French_bulldog
070.German_pinscher
071.German_shepherd_dog
072.German_shorthaired_pointer
073.German_wirehaired_pointer
074.Giant_schnauzer
075.Glen_of_imaal_terrier
076.Golden_retriever
077.Gordon_setter
078.Great_dane
079.Great_pyrenees
080.Greater_swiss_mountain_dog
081.Greyhound
082.Havanese
083.Ibizan_hound
084.Icelandic_sheepdog
085.Irish_red_and_white_setter
086.Irish_setter
087.Irish_terrier
088.Irish_water_spaniel
089.Irish_wolfhound
090.Italian_greyhound
091.Japanese_chin
092.Keeshond
093.Kerry_blue_terrier
094.Komondor
095.Kuvasz
096.Labrador_retriever
097.Lakeland_terrier
098.Leonberger
099.Lhasa_apso
100.Lowchen
101.Maltese
102.Manchester_terrier
103.Mastiff
104.Miniature_schnauzer
105.Neapolitan_mastiff
106.Newfoundland
107.Norfolk_terrier
108.Norwegian_buhund
109.Norwegian_elkhound
110.Norwegian_lundehund
111.Norwich_terrier
112.Nova_scotia_duck_tolling_retriever
113.Old_english_sheepdog
114.Otterhound
115.Papillon
116.Parson_russell_terrier
117.Pekingese
118.Pembroke_welsh_corgi
119.Petit_basset_griffon_vendeen
120.Pharaoh_hound
121.Plott
122.Pointer
123.Pomeranian
124.Poodle
125.Portuguese_water_dog
126.Saint_bernard
127.Silky_terrier
128.Smooth_fox_terrier
129.Tibetan_mastiff
130.Welsh_springer_spaniel
131.Wirehaired_pointing_griffon
132.Xoloitzcuintli
133.Yorkshire_terrier
0
10
20
30
40
50
60
70
80
count

THE HUMAN DATASET

This dataset has 5750 ordinary images of humans. We implement a Face Detector to identify human faces in a given image.

6 Algorithms and Techniques

- Human face detector: The project uses OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images.

  Reference: https://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html
- Pre-trained VGG model as a Dog detector: We use a pretrained VGG model as a dog detector. VGG is a classical convolutional neural network architecture. It was based on an analysis of how to increase the depth of such networks. The network utilises small 3 x 3 filters. Otherwise the network is characterized by its simplicity: the only other components being pooling layers and a fully connected layer.

  Source: https://paperswithcode.com/method/vgg
- Dog breed classifier using CNN model implementation from scratch. Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply. ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

Cross Entropy loss function is used for classification and it is the best estimator in terms of convergence rates.

SGD optimizer of used since they are better for classification problems.

Source: *https://cs231n.github.io/convolutional-networks/*

- Dog breed classifier using Transfer Learning: In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest.
  I used resnet50 as a pretrained model. The parameters of the RESNET 50 pretrained model were freezed so that that they are not trained again and these weights are reused. The last fully connected layer was replaced by a new one to output for the classes of dog_breeds.
  Source: *https://cs231n.github.io/transfer-learning/*

7. BENCHMARK

We are looking to understand how Convolutional Neural Networks can solve this problem. Hence the goals are tailored according to the resources and time available. Two model shall be implemented, one from scratch and the other using a pre-trained network.

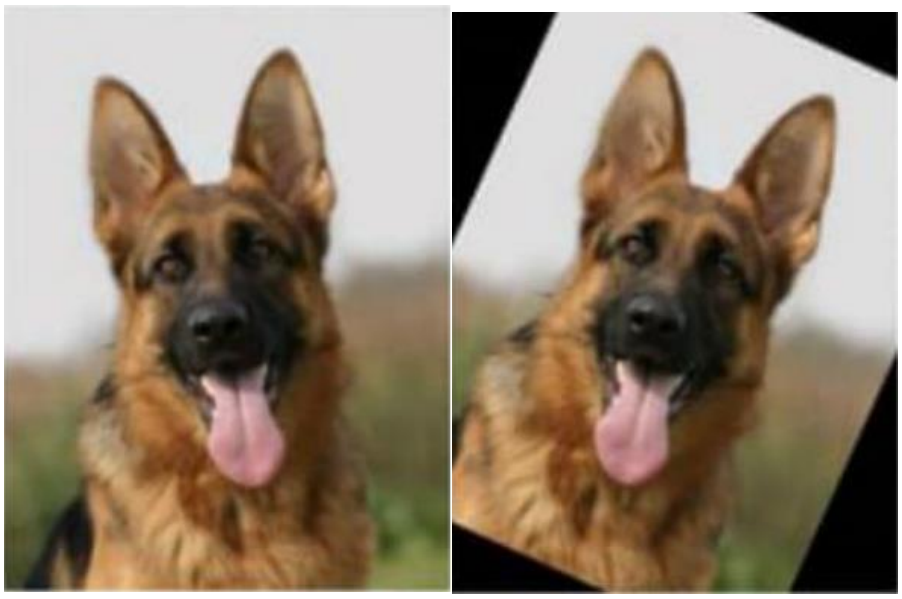
To prove that the model is learning features from the image we looking at the accuracy of at least 10% using the model created from scratch and 60% from the pre-trained network. The pre-trained model can be used to implement an interesting app.

8. DATA PREPROCESSING

We will be using the following data augmentation on the dataset:

- RandomResizedCrop, which is a type of image data augmentation where a crop of random size of the original size and a random aspect ratio of the original aspect ratio is made. This crop is finally resized to given size. This augmentation helps in training the model to learn to predict well even when the objects are partially cutout. Randomness adds to the variety of the images and hence makes the model robust.
- RandomHorizontalFlip, which mirrors images horizontally.

- Random rotation rotate the images to 10 degrees.



- I could have applied more transformations, but that will hinder the performance of the model and make the training slow.
- I am using the same transformations for validation set since I am splitting the data later. For all training, valid and test dataset I am reszing the image to 224.The choice for a 224 is from AlexNet where it was seen as the best choice for applying data augmentation.
- Normalization with mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]

9. IMPLEMENTATION

HUMAN FACE DETECTOR:

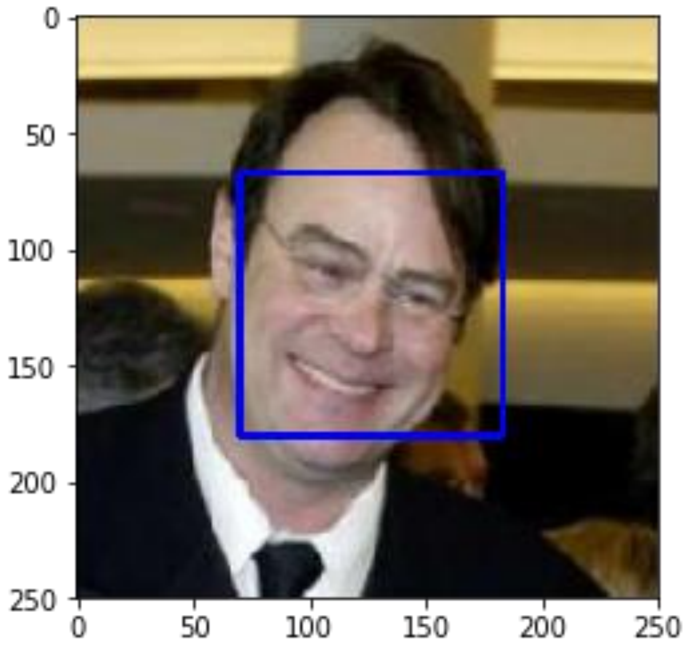
The project uses OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images. Reference: https://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html

OpenCV provides many pre-trained face detectors, stored as XML files on github. We have downloaded one of these detectors and stored it in the haarcascades directory. The project demonstrates how to use this detector to find human faces in a sample image. Reference: https://github.com/opencv/opencv/tree/master/data/haarcascades

Before using any of the face detectors, it is standard procedure to convert the images to grayscale. The detectMultiScale function executes the classifier stored in face_cascade and takes the grayscale image as a parameter.

The faces is a numpy array of detected faces, where each row corresponds to a detected face. Each detected face is a 1D array with four entries that specifies the bounding box of the detected face. The first two entries in the array (extracted in the above code as x and y) specify the horizontal and vertical positions of the top left corner of the bounding box.

The last two entries in the array (extracted here as w and h) specify the width and height of the box.



Percentage of humans detected: 98

Percentage of dog detected: 17

PRE-TRAINED VGG MODEL AS A DOG DETECTOR:

The VGG-16 model, along with weights that have been trained on ImageNet, a very large, very popular dataset used for image classification and other vision tasks. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of 1000 categories.

We use a pretrained VGG model as a dog detector. The dog detector is a function that accepts a path to an image (such as 'dogImages/train/001.Affenpinscher/Affenpinscher_00001.jpg') as input and returns the index corresponding to the ImageNet class that is predicted by the pre-trained VGG-16 model. While looking at the dictionary, you will notice that the categories corresponding to dogs appear in an uninterrupted sequence and correspond to dictionary keys 151-268, inclusive, to include all categories from 'Chihuahua' to 'Mexican hairless'. Thus, in order to check to see if an image is predicted to contain a dog by the pre-trained VGG-16 model, we need only check if the pre-trained model predicts an index between 151 and 268 (inclusive).

Results:

Percentage of humans detected: 2

Percentage of dog detected: 100

CNN MODEL IMPLEMENTATION FROM SCRATCH TO CLASSIFY DOG BREEDS

- The model has 5 convolution layers, because we need atleast that many to get the detailed features of the image. It used stride = 1 to get the finer details, kernel size of 3 to keep the performance good.
- Relu activation applied to the hidden layers for non-linearity
- Pooling layers are used after convolution to filter only important features and reduce the size of feature maps for better performance.
- Dropout layers are added to regularize the network except to the last layer.
- Batch Normalization is added for faster convergence except to the last layer.
- Cross Entropy loss function is used for classification and it is the best estimator in terms of convergence rates.
- SGD optimizer of used since they are better for classification problems.

TRANSFER LEARNING

- I used resnet50 as a pretrained model. RESNET 50 allows a model to be deep without suffering the issues of vanishing gradient because of the skip connections. It also helps in communicating the features detected in top layers to the deeper layers which in turn helps in better learning of the model.
- The parameters of the RESNET 50 pretrained model were freezed so that that they are not trained again and these weights are reused.
- The last fully connected layer was replaced by a new one to output for the classes of dog_breeds.
  Reference: *https://arxiv.org/abs/1512.03385*

10. REFINEMENT

I found the output quite good. I think it is because of the pretrained RESNET architecture. It is amazing to see that even though the model was pretrained using other dataset, we could still get a very good performance just by changing the last layer and that too in just 10 epochs. Possible improvements:

More augmentations like cut-out technique, zoom can be added to make the model more robust.

Few last layers of the pre-trained RESNET model can be unfreezed so that parameters of those layers can further be trained to learn more finer details the current dataset.

The dataset can be increased by collected more images and annotating them.

WE can fine-tune the hyperparameters for the given dataset and the model for better accuracy and performance. Start with standard values given in research papers and use grid search for fine-tuning.

Reference: [*https://www.kaggle.com/willkoehrsen/intro-to-model-tuning-grid-and-random-search*](https://www.kaggle.com/willkoehrsen/intro-to-model-tuning-grid-and-random-search)*.*

11. RESULTS

MODEL EVALUATION AND VALIDATION

MODEL FROM SCRATCH

HYPER-PARAMETERS:

Batch size - 32

Learning rate – 0.05

Epochs – 15

```
Epoch: 1 	Training Loss: 3.868028 	Validation Loss: 4.430374
Validation loss decreased (inf --> 4.430374).  Saving model ...
Epoch: 2 	Training Loss: 3.777621 	Validation Loss: 4.076888
Validation loss decreased (4.430374 --> 4.076888).  Saving model ...
Epoch: 3 	Training Loss: 3.595259 	Validation Loss: 4.040984
Validation loss decreased (4.076888 --> 4.040984).  Saving model ...
Epoch: 4 	Training Loss: 3.431549 	Validation Loss: 4.239084
Epoch: 5 	Training Loss: 3.307569 	Validation Loss: 4.472433
Epoch: 6 	Training Loss: 3.152341 	Validation Loss: 4.242904
Epoch: 7 	Training Loss: 3.051498 	Validation Loss: 3.712976
Validation loss decreased (4.040984 --> 3.712976).  Saving model ...
Epoch: 8 	Training Loss: 2.943359 	Validation Loss: 3.931410
Epoch: 9 	Training Loss: 2.791779 	Validation Loss: 3.761310
Epoch: 10 	Training Loss: 2.672278 	Validation Loss: 3.889188
Epoch: 11 	Training Loss: 2.526016 	Validation Loss: 3.694072
Validation loss decreased (3.712976 --> 3.694072).  Saving model ...
Epoch: 12 	Training Loss: 2.433448 	Validation Loss: 3.547985
Validation loss decreased (3.694072 --> 3.547985).  Saving model ...
Epoch: 13 	Training Loss: 2.295088 	Validation Loss: 3.638066
Epoch: 14 	Training Loss: 2.162600 	Validation Loss: 3.678086
Epoch: 15 	Training Loss: 2.075638 	Validation Loss: 3.599909
```

The validation loss shows consistent decrease which is a sign of good learning pace for the model.

Test Accuracy: 19% (165/836)

MODEL WITH TRANSFER LEARNING

HYPER-PARAMETERS:

Batch size - 32

Learning rate – 0.05

Epochs – 10

Pre-trained model – ResNet50

```
Epoch: 1 	Training Loss: 2.628365 	Validation Loss: 1.290821
Validation loss decreased (inf --> 1.290821).  Saving model ...
Epoch: 2 	Training Loss: 1.033340 	Validation Loss: 0.800192
Validation loss decreased (1.290821 --> 0.800192).  Saving model ...
Epoch: 3 	Training Loss: 0.725893 	Validation Loss: 0.639232
Validation loss decreased (0.800192 --> 0.639232).  Saving model ...
Epoch: 4 	Training Loss: 0.598282 	Validation Loss: 0.643720
Epoch: 5 	Training Loss: 0.520189 	Validation Loss: 0.592856
Validation loss decreased (0.639232 --> 0.592856).  Saving model ...
Epoch: 6 	Training Loss: 0.455146 	Validation Loss: 0.526389
Validation loss decreased (0.592856 --> 0.526389).  Saving model ...
Epoch: 7 	Training Loss: 0.405248 	Validation Loss: 0.521060
Validation loss decreased (0.526389 --> 0.521060).  Saving model ...
Epoch: 8 	Training Loss: 0.386178 	Validation Loss: 0.485380
Validation loss decreased (0.521060 --> 0.485380).  Saving model ...
Epoch: 9 	Training Loss: 0.358793 	Validation Loss: 0.463666
Validation loss decreased (0.485380 --> 0.463666).  Saving model ...
Epoch: 10 	Training Loss: 0.331242 	Validation Loss: 0.465092
```

The validation loss shows consistent decrease which is a sign of good learning pace for the model. The validation loss is much less than that of the model from scratch.

Test Accuracy: 86% (721/836)

CLASSIFIER APP RESULTS:

The app used the transfer learning model to predict the dog breed. It used the face detector with it to do some interesting things like:

- 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.

Here are some interesting results from the App:

12 JUSTIFICATION

MODEL FROM SCRATCH

To prove that the model is learning features from the image we were looking at the accuracy of at least 10% using the model created from scratch and we got:

Test Accuracy: 19% (165/836)

Which has exceeded the expectations.

MODEL WITH TRANSFER LEARNING

With the pre-trained network we were looking at the accuracy of at least 60% and we got

Test Accuracy: 86% (721/836)

Which again has exceeded the expectations.