Capstone Project Report

Machine Learning Engineer

Ashu Saini 20/11/2020

Domain Background

We love dogs. Right ? Yes, most people love dogs. So sometimes we go to the park or travel somewhere we mostly meet dogs. When we go trekking dogs usually follow us :) isn't it. Sometimes we like a dog so much and a thought comes into our mind that is let's have a dog. And we don't know it's breed and this dog will be compatible with our atmosphere where we lived so we are tackling this problem by creating a classifier that will tell us the dog's breed using a dog's image and it will help us the dog's owners as well who are unsure about there dog's breed. We are going to use a Convolutional neural network (CNN) to deal with this problem as we all know CNNs are a better choice then Fully connected Neural networks for image classification.

Problem Statement

We are going to use an image as an input and based on an algo we design will give us an estimate of the dog's breed if the image is of a dog or if the image is of a human we will tell the most similar dog's breed for that human.

So in this project we will know

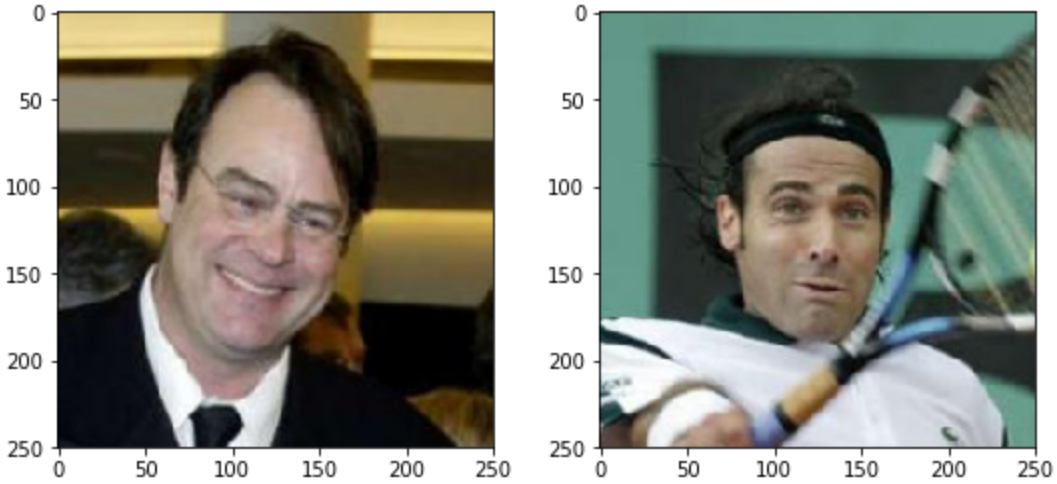
1. if an image is of a human or dog or none.
2. If dog then it's breed
3. If human then most similar dogs breed
4. Our algorithm should be at least 60% efficient

Datasets and Inputs

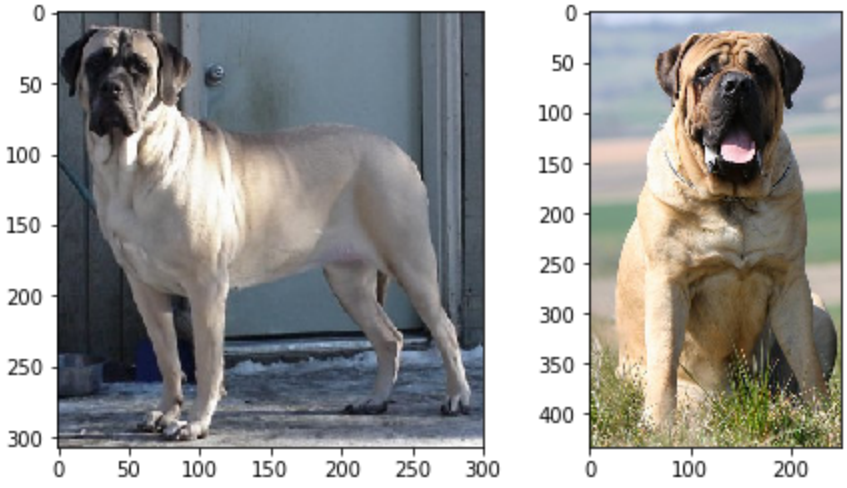
We are using image datasets provided by udacity

Dog Dataset Url : https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip (8351 images)
Human Dataset Url : https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip (13233 images)

Human Image Example



Dog Image Example



Solution Statement

CNN model which will estimate dog's breed if there is a dog in the picture.
We may use transfer learning for better results as there are pre existing models on large datasets like imagenet and we finetune that model to get the desired result of our problem.
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.

Benchmark Model

I am going to use VGG-16 as a benchmark model. It's available in pytorch. I will modify the last Fully connected layer and freese all the convolution layer weights.

Evaluation Metrics

I am going to use accuracy as the evaluation matrix.

Project Design Phases

As we are using a predefined project here in the course I am going to follow the same.

Phase 1: Data collections

For Data collections usually in ML projects we have to do a lot of work but in our case formatted data is provided by udacity.

Data can be downloaded from the following links

Dog Images

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

Human Images

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

In our dataset we have 13233 human images and 8351 dog images with their breeds.

Phase 2: Detect Humans

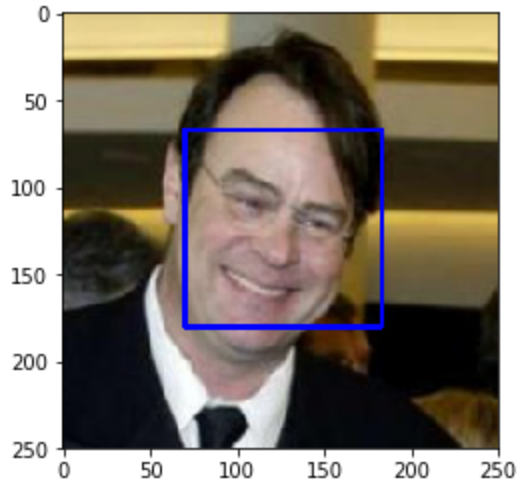
Detecting humans is a lot of work standalone. We can create a custom model but this was not the main target of this project and there are so many libraries available for human detection and we are not recreating the wheel here.

We use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images.

OpenCV is the best computer vision library available on the internet. OpenCV provides many pre-trained face detectors. We have downloaded one of these detectors and stored it in the haarcascades directory.

Before using any of the face detectors, it is standard procedure to convert the images to grayscale.

And we can see beautiful human faces are detected seamlessly :D. That's the magic of machine learning we loved about.



Phase 2: Detect Dogs

In this phase we used a pretrained VGG-16 model. 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.

Before passing the images to vgg model we have to preprocess the images as dogs images are raw all different sizes. So I transformed and cropped images using RandomResizedCrop and the transform to tensor array and then changed the array to single array and passed to the model.

Vgg-16 has 1000 categories and dogs lie between 151 to 268 inclusive so it was pretty easy to just check the index is between or not for a dog image.

So we randomly tested the images and found out 100 images. 70 to 80 percent of images it correctly identified.

Phase 3: Create a CNN to Classify Dog Breeds from scratch

This was the pretty crucial step in this project. I have created a CNN that classifies dog breed from scratch.

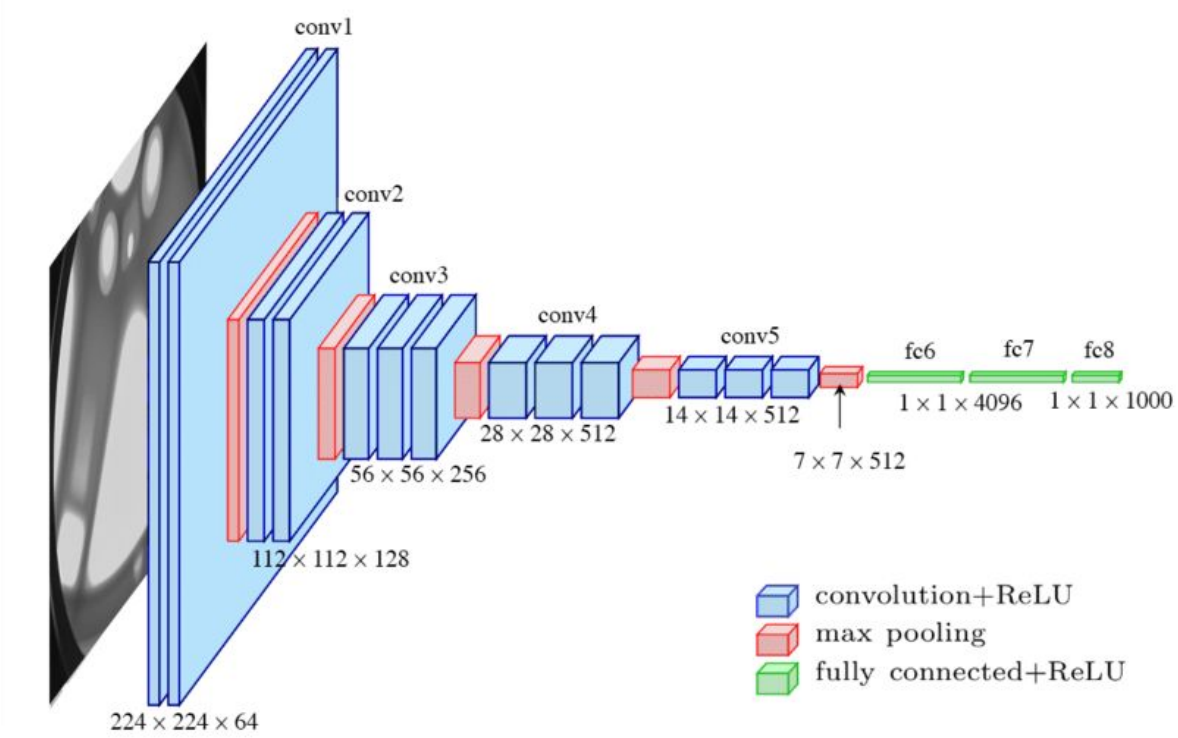
Data Preprocessing

First we transform data Resize and crop to give our model a consistent input. For a robust model we can translate and rotate the data as well.

We split the dataset in three parts: train, test and validation data.

Model Architecture

I followed simple vgg-16 model architecture.



for convolution layers. I did some tries. I started with 2 convolution layers and then 3 but it was not enough for my experiment and then I stuck to 5 convolution layers and 2 fully connected layers.

I found that It's better to use batch normalization after each channel for faster training and regularize the network.
Batch normalization normalizes the activations of the network between layers in batches so that the batches have a mean of 0 and a variance of 1.

The mean and standard deviation are calculated for each batch and for each dimension/channel. γ and β are learnable parameters which can be used to scale and shift the normalized value, so that we can control the shape of the data when going into the next layer (e.g., control the percentage of positive and negative values going into a ReLU).

Model Layers

```
Net(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (conv4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
  (conv5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=True)
  (norm2d1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (norm2d2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (norm2d3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (norm2d4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (norm2d5): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (dropout): Dropout(p=0.2)
  (fc1): Linear(in_features=9216, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=133, bias=True)
)
```

Loss Function and Optimizer

Loss function used CrossEntropyLoss and optimizer used SGD optimiser.

Results

After testing and training we do get test accuracy of **16%**. Yes it's pretty less but as per data available and project requirement it is enough for this model.

Phase 4 : Create a CNN to Classify Dog Breeds (using Transfer Learning)

Data Preprocessing

Reused the same data as in scratch model.

Model Architecture

I have used Resnet50 pretrained model for transfer learning initially i thought of vgg16 but as it is already used in classify dogs so i want to experiment a different one.

I have changed the last fully connected layer of Resnet50 and changed the output classes to 133 as we have 133 breads of dogs.

Loss Function and Optimizer

Loss function used CrossEntropyLoss and optimizer used SGD optimiser.

Results

After training and testing we do get a pretty good accuracy of **74%** in 25 epochs.

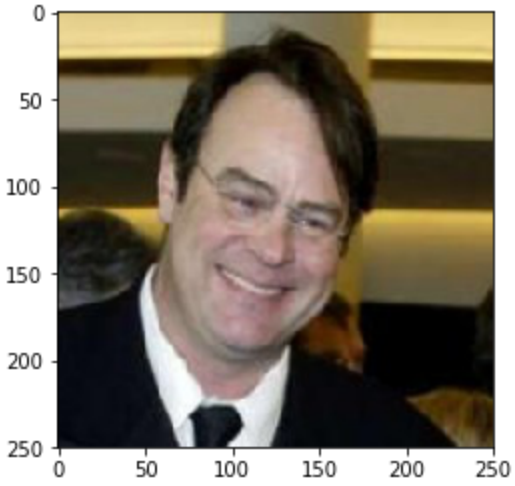
Phase 5 : Combining all together

In the end I combined a dog detector , human detector and bread detector all together.

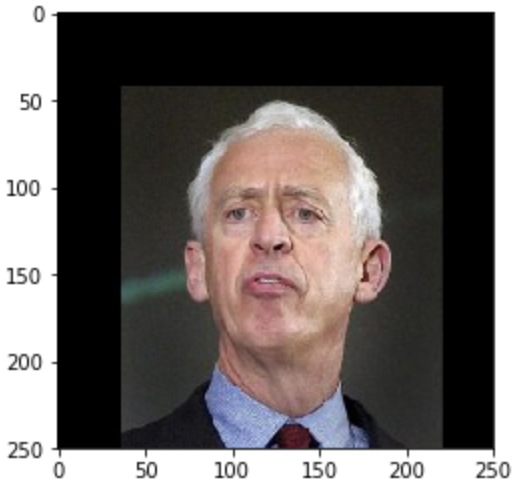
So the final output of our algo is able to detect if an image is human or dog or none of these.

If it’s human then which dog breed it’s looks like
If it's a dog then identify the actual dog’s breed.

Examples from our app



Human : look like a Chihuahua



human: look like a Basenji

Dog : Doberman pinscher

Not Human Not dog

So yes we found some good matches and some false negatives

Improvements we can do :

1. **we can have more dogs image dataset.**
2. **we can train our model for more epochs**
3. **we can add more dropout in between to avoid overfitting.**
4. **we can transform images in the training set move left , right, rotate , flip for more robust model.**