UDACITY CAPSTONE PROJECT

DOG CLASSIFIER

Pankaj Patil | Machine Learning Engineer Nanodegree

Project Overview

Convolutional Neural Networks have transformed the way machines interpret images. While classical neural networks have shown promising results, one important drawback of them is the need to flatten out the input. In image, this essentially translates to throwing away useful information which could have been used for learning more about the input. An image of a dog is an image of a dog regardless of which part of the image dog appears in. CNNs are able to tackle this problem by have a series of filters and pooling layers in the network. The task is to learn these filters such that they can be used to predict output labels.

ImageNet is a huge dataset of labelled images and is widely used for comparing current benchmarks in the field of computer vision. ILSVR competition involves training a model on 1.2 million training images, 50k validation images and 10k test images to predict the correct category for an image among 1000 available categories. The target classes include various objects from day-to-day lives like dogs, cats, various household projects, vehicle types etc.

In 2014, the 1st runner up for the ILSVR competition was VGGNet. VGGNet is invented by Visual Geometry Group (by Oxford University). The reason to understand VGGNet is that many modern image classification models are built on top of this architecture. It is a convolutional neural network made of architecture shown in column D of image below:

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Image Source : Table 1 of Very Deep Convolutional Networks for Large Scale Image Recognition, Simonyan and Zisserman (2014).

Authors of VGGNet knew that deeper neural networks struggle with convergence. So they trained shallower networks and used weights from them to initialize the deeper networks.

While better models are now available, VGGNet still remains one of the best known algorithms in image classification tasks.

Problem Statement

The problem statement I propose to solve is for a given input image, developing a model to classify between human and dogs. Then if a dog, the model would further classify the breed of the dog. The user would be able to upload an image via a web app and then the webapp would communicate with the model for inference. This would involve use of pre-trained state-of-the-art like VGGnet to do classification tasks.

Datasets and Inputs

There would be 2 datasets for this project:

a) Dog Dataset: This would contain RGB images of various dogs and a dataset specifying which image corresponds to which breed. The dataset I'll be using has 8,351 dog images and has been provided by Udacity
b) Human Dataset: Same as dog dataset but it would contain images of humans. Since all images would correspond to humans, there would be no labels required. The dataset I'll be using has 13,233 human images and has been provided by Udacity.

The dataset is a subset of the ImageNet

Since the task at hand is to first predict if a human/dog is present in the give image, it is important that we have a model which is able to classify either. Thus, we require a dataset which has images of both humans and dogs. I plan to work with CNNs to solve this problem

Metrics

The primary metric for this problem statement was accuracy. I also checked class wise accuracy to get a better understanding if the model had any bias towards specific kind of class.

Analysis

Starting with human dataset, it didn't matter what the image label for it as we were not classifying humans based on their appearance but rather our aim is to successfully detect humans. While on the other hand, we need to specifically classify dog breeds.

First, I checked the category wise split in training data for dog Images. I have pasted some specifics about the dataset below :

	Training_Image_Count
count	133.000000
mean	50.225564
std	11.863885
min	26.000000
25%	42.000000
50%	50.000000
75%	61.000000
max	77.000000

Table : Training Image statistics

Table : Histogram showing #input samples for each breed

As seen above, for a total of 133 dog breeds we have 26-77 images per breed with an average of 50 images per breed. While the class imbalance is not huge, we need to be aware that this imbalance might lead to a bias in the model.

Algorithms and Techniques

The primary algorithms that were used for this project were Convolutional Neural Networks. CNNs are exceptionally useful when dealing with spatial data like images. In classical neural networks we'd have to flatten an image and feed that into a neural network to learn patterns in the data but this presents some problems when dealing with image data. An object can appear at different places in an image or be upside down, sideways etc. in an image and the label would still remain the same. This information is thrown out when we flatten the image. While defining how CNNs work is out of scope of this document, a great point to begin with will be this article from data science central.

Another technique used in this project is use of Flask for deploying the trained model. Using a combination of Flask API, HTML, javascript and css, I created a web interface where

end user can upload an image and the interface would return prediction based on the given image.

Benchmark

I benchmark performance of my model with a simple probabilistic classifier. For a total of 133 breeds, a random guess model will be accurate 0.75%. Thus I expect my model to have at least that much accuracy. While this is a very low bar to achieve, the aim of this project is successful deployment and not to get the highest accuracy model. If latter was the case, I'd have benchmark it against the performance of one of the famous models like VGG16, Inception v3 etc.

Methodology

Data Preprocessing

Like any machine learning task, the data would need to be pre processed before it can be input to a model. I did some regular preprocessing tasks like resizing the image so all the inputs are of the same shape (224 x 224, I chose this because as later explained, I used a VGG16 model which was originally trained using 224x224 size images. Though VGG26 can handle other shaped images as well). I also augmented my initial dataset by adding translational, rotational and scale variance. This would allow my model to learn features with greater stability.

Implementation

First part of the problem is to detect a human face. There are multiple ways to do it but one of the easiest ways is to use haarcascades from the open-cv library. This model can be used to detect the human face in the given image. So, I tested the performance of `haarcascade_full_frontal.xml` model on the first 100 images of both human and dog training dataset. I observed that this model was able to detect 98/100 faces in human images and 9/100 images in dog files. Provided we have images as shown below in dog dataset, it's safe to say that this model is very accurate for our use case while demonstrating high recall and precision.

Table : Sample image containing both human and dog photo from dog Image folder

For this project, I have decided not to focus on the corner case where both dog and humans are present and let the image get classified as a human though it is very important to know the limitations of our model.

For the next part, which is to predict a dog face, I tried using VGG16 model to detect a dog in the given image and observed very high accuracy (almost 100%) in predicting whether a given image is an image of a dog or not. (note : In this stage, I'm working only to detect whether a dog is present in an image or not, not trying to predict it's breed).

I also tried building a convolutional neural network from scratch to identify the breed of a dog based on a given image. I started with a 32 layer deep CNN and gradually increased it up to 128 layers but only had limited success with this technique and the maximum observed accuracy was at 128 layers at 28%. While I expected the model to perform better if I increased the depth of the model, I decided to leverage transfer learning to get most out of my dataset.

For transfer learning to work, it is important that the model has learned features that are useful in predicting classes in our task. I decided to use the same VGG16 model as it is trained on ImageNet data.

I changed the final classification layer of the VGG16 model to map it s.t. It'd only have 133 output layers and retrained the classification layer weights. This model resulted in a 60% accuracy which is far better than 28% accuracy I got from the model I created from scratch.

Refinement

Since I'm using transfer learning, the only parameter that I could vary was the number of epochs my model trained for. The feature layer weights of the model were frozen. I first trained the model for 10 epochs to see if the model was performing and I achieved 55% accuracy. I then increased the number of epochs to 50 and retrained the model. The model showed lower train and validation losses but I also noticed that it stayed almost flat after epochs 20. In between the iterations I was always saving the model having lowest validation loss. I used that model for the final process.

Results

Model Evaluation and Validation

I used a validation dataset to monitor both training and validation losses during training. The model which ended up using, both had low training and validation loss. I also checked if my model was returning only a specific set of breeds, essentially if it is not able to predict any of the breeds. That wasn't the case.

Justification

The accuracy I got for the test data was around 60%. As discussed before, the precision and recall for this model did not indicate any kind of underfitting or overfitting. The model is also performing significantly better than naive probabilistic classifiers.

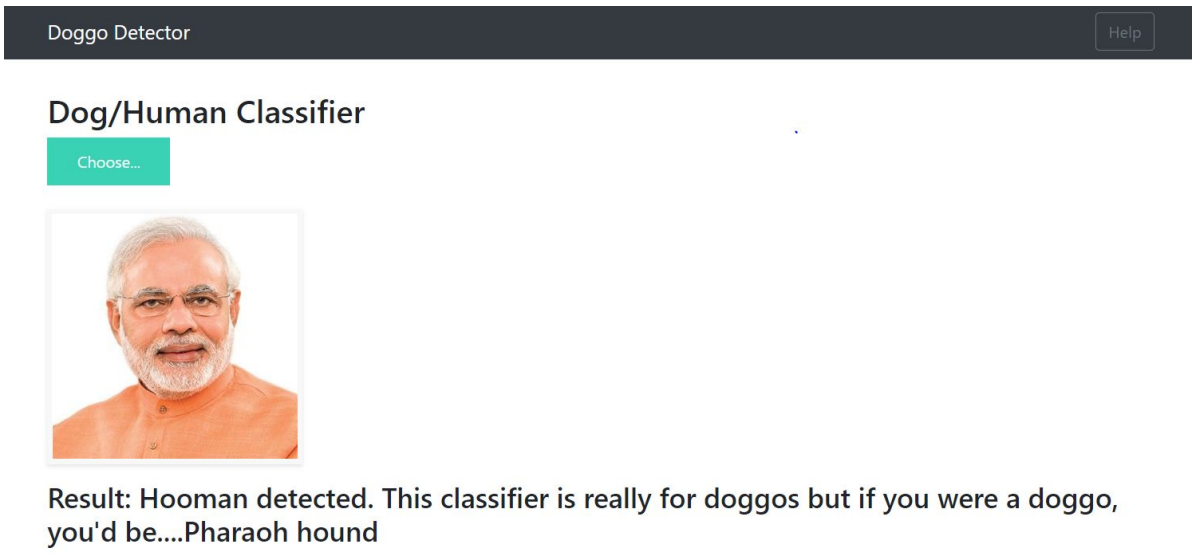
Deployment

Tools used : Flask, HTML, CSS, JavaScript

To deploy the model, I used the models from my analyses earlier to build an app using flask where the user could input an image and the web interface would send the data to the model. If the model thinks that given image is dog, it'd return it's breed; if model thinks that the given image is a human, it'd guess the most resembling dog breed anyway.

Following snips show the deployed model in action :

1) Human image :



2) Dog image :


Doggo Detector
Dog/Human Classifier
Choose...
Result: Hey good boi, My guess is you'd be... Finnish spitz