Capstone: Dog Breed Identification

In the capstone for Machine Learning Engineer, I will be working on building a dog breed classification model that can hopefully work within the browser itself. I have chosen to do this particular project as this gives me opportunity to explore the realm of unstructured data in a little more detail, and will also allow me to experiment with PyTorch.

Domain Background:

The project falls in the image classification domain where I use some of the modern deep learning framework and CNN architecture-based models.

In the last decade, machine learning (ML) becomes more popular thanks to very powerful computers that can handle to process lots of data in a reasonable amount of time. Machine learning concept is introduced by Arthur Samuel in 1959 so it is not new, but today we can use lots of its potential. One more reason for this is that today there is a lot of digitized data that we need to successfully implement good ML models.

Dog breed identification problem is well known in the ML community. We can find it on Kaggle: https://www.kaggle.com/c/dog-breed-identification/overview/description

Here is a link to an article where some similar academic work is done but on flowers:

https://www.researchgate.net/publication/320746968_Flower_species_recognition_system_using_convolution_neural_networks_and_transfer_learning

One more reason for choosing this problem is because I believe this is a good path to solve a bigger problem: How many calories and carbs have food in a picture? Solving this problem is the main reason I want to learn ML.

Problem Statement:

Elaborating on the background, I will specifically build a model that, given an image of a dog, classify into a breed. We will be working with specific breeds to keep the problem statement simple.

The base reference will be the [Udacity repository] (https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification).

Datasets and inputs:

To solve our problem our input data must be images because we want to user takes an image of a dog (or human) with his phone, sent it to our server and we would return what dog breed is most likely in a picture (or to which dog breed is human most look like).

All data for this project is provided by Udacity. We have pictures of dogs and pictures of humans.

All dog pictures are sorted in train (6,680 Images), test (836 Images) and valid (835 Images) directory, and all the images in these directories are sorted in breed directories. We have 133 folders (dog breeds) in every train, test and valid directory.

Human pictures are sorted by name of each human. We have 13,234 Files (Images), 5,749 Folders (Humans)

Our data is not balanced because we have one image of some people and several for others. The same is for dog images. (the difference is from 1 to 9 images in most cases)

Dog images have different image sizes, different backgrounds, some dogs are in full sizes and some just ahead. Lightning is not the same. That is actually ok because we don't know how users' images will be, and we want that our model works on different types of images. Human images are all of the same size 250×250. Images are with different backgrounds, light, from different angles, sometimes with few faces on the image. Here are a few samples of our dog and human images:

Affenpinscher

Afghan_hound

Airedale_terrier

Cane_corso

Lowchen

Norwich_terrier

Plott

Silky_terrier

Tibetan_mastiff

Xoloitzcuintli

Solution statement:

I will be building the solution in two parts.

1. Use a simple, few layers deep, CNN architecture that is trained from scratch to classify the breeds. Once a good enough model is built, I will try deploying it within the browser itself.
2. I will use one of the pretrained networks available and fine tune them for the task.

Benchmark model:

For our benchmark model, we will use the Convolutional Neural Networks (CNN) model created from scratch with an accuracy of more than 10%. This should be enough to confirm that our model is working because random guess would be 1 in 133 breeds which are less than 1% if we don't consider unbalanced data for our dog images.

Evaluation metrics:

I will be looking at the accuracy and confusion matrix to determine how well the model is performing. The loss function for training will be cross entropy loss.

Project design:

After getting a dataset that is provided by Udacity second thing we want to do is to detect humans on images.

We will use the OpenCV model to get faces from the image and that will tell us is on the image human or not. To do this we will implement Haar feature-based cascade classifiers. We can find more about this here: Object Detection using Haar feature-based cascade classifiers.

Our workflow on detecting faces:

- initialize pre-trained face detector
- load image
- convert image to grayscale
- find faces in the image
- return true if the number of faces is more than 0 else return false

Then we detect dogs on images. We will use the pre-trained model VGG16 for this.

- first, we define our VGG16 model
- we will use GPU for better performance
- load and pre-process the image
- send an image to the VGG16 model
- model return index from 0 to 999 (dog classes are from 151 to 268)
- return true if the index is >=158 and <=268 else return false

Our data is already divided into training, validation and test partitions so we can now use our train data to make a benchmark model using Convolutional Neural Networks. After creating a model, we will test it with test data. When we get accuracy over 10%, we will proceed on building a new model using transfer learning. With transfer learning, we can build our model with fewer data to give us a better result. We will use the same training data as before. We will then test our model with the same test data as before but know we expect our accuracy to be over 80%. Then we can try to experiment with different model parameters to get better results. We will use f1 score and log loss to evaluate our models.

I would have loved to do something similarly sophisticated. If you can guide me on some other ideas that I can work on, specifically around reinforcement learning, I can rework on my submission accordingly. I believe that the dog breed classifier is not that unique an idea to begin with, but if nothing else works, I am happy to work on this problem.