Machine Learning Engineer Nanodegree

Capstone Project

Topic : - Dog Breed Classifier

Saurabh Dubey

19th April 2020

Project Report

1. Definition

Project Overview:

- Dogs are one of the most common domestic animals you will find at most of the homes. As the number of dogs is increasing, a lot of issues associated with dogs are also increases. Issues such as Rabies, vaccination control, etc. So to control such issues we want to identify each dog breed so that we can take appropriate measures to solve this issue. So the idea here is to build the model pipeline which can be trained on a lot of different dogs breed images and then identify or classify the given real-world image of a dog in different breed types. This is a label type of problems and it has multi-class output so it's a multi-class classification problem so we will use a supervised learning approach to solve this problem.

- I want to deploy this project model on the web app so that it can take real-world images of dogs and be able to give predictions. By deploying it can be used for a lot of animal shops to classify and make sure that some different breeds of dogs to be put in a different cell.

Problem Statement:

- The main objective of this project will be to use Deep Learning Convolutional Neural Network with Transfer Learning to classify the dogs breed.

- Given an image of a dog, we want to be able to identify an estimate of the canine's breed with a corresponding likelihood score. If supplied with an image of a human, the code will identify the resembling dog breed.

Metrics:

- Accuracy is the most general metrics for binary classifiers, it takes into account both true positives and true negatives with equal weights

  Accuracy = correct classifications / number of classifications

- Here we will also use F1 score for evaluation purpose because our dataset contains images of dogs have an imbalanced dataset. So when we face such a problem of data imbalanced then we will also use F1 score as a metric for our model evaluation.

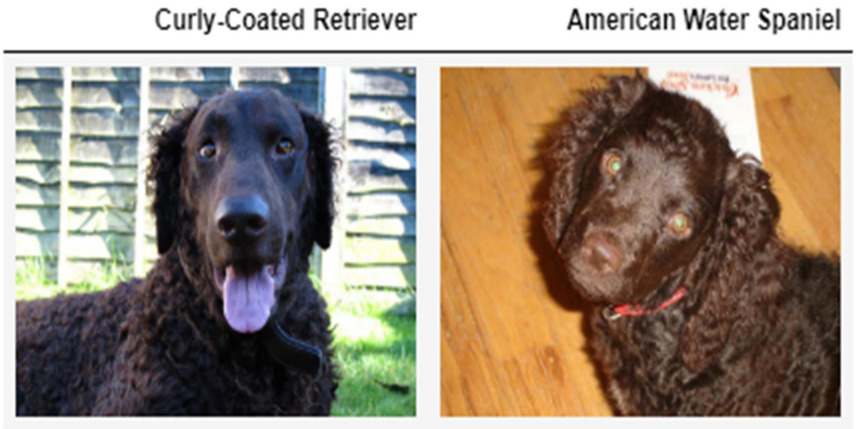
2. Analysis

Data Exploration:

- The dataset for this project was provided by Udacity. The dataset has pictures of Dogs and Humans.

- The dataset is in Image format. The dog dataset contains train, valid and test folders and each folder contains every 133 samples of different dogs canine's breed. The number of images in each folder is not the same i.e some dogs breed folder, the number of images is less and in some, the number of images is more. Pixel size also varies for a lot of images.

- The human dataset contains 5749 items of humans. The human dataset, when supplied to our model will give a resembling dog breed. Each folder has 1 image of input. Each image of size 250x250.
- There were 13233 total human images
- There were 8351 total dogs images



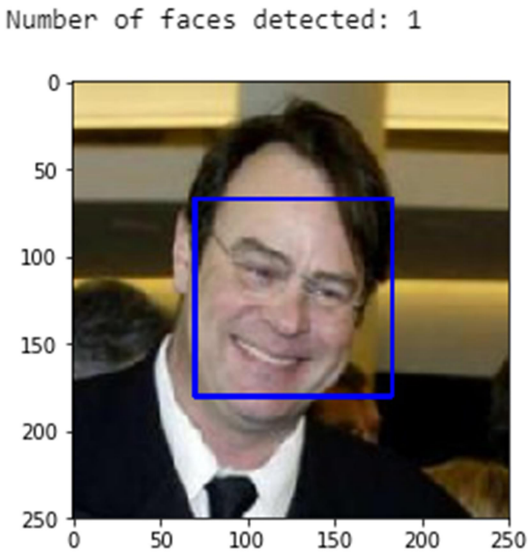
Above Figure shows the dogs images samples. Here we have done the data exploration of dogs images dataset to see how dogs images looks like.

From Above figures we get an idea of how human images looks like in our human images dataset. That's how we have done the data exploration of human images dataset.

Exploratory Visualization:

- We have used OpenCV's implementation of Haar feature cascade classifiers to detect or extract the human faces from the image.

- Haar feature cascade classifier will make the bounding box in the image where the human face is present.



Above figures shows how human face will be detected after using Haar Cascade Classifier.

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

Algorithms and Techniques:

- We have used OpenCV implementation of Haar cascade classifier to first detect the human faces and then we have design our model Convolutional Neural Networks from scratch to detect the dogs breed out of the 133 breed outcomes.
- Our Convolutional Neural Networks uses 3 layer architecture implementation. We used pre-trained weights for training our model. VGG16 is the pre-trained weights we have used.
- We then used Transfer Learning to increase our model accuracy. Transfer Learning will use Resnet50 pre-trained model for training our model. So by doing Transfer Learning our model accuracy is gets increased

Benchmark :

- The CNN model created from scratch must have an accuracy of at least 10%. This can confirm that the model is working because a random guess will provide a correct answer roughly 1 out of 133 times, which corresponds to an accuracy of less than 1%.

- The Convolutional Neural Networks model created using Transfer Learning must have an accuracy of 60% and above.

3. Methology

Data Preprocessing:

- Pre-processing images is all about **standardizing** input images so that you can move further along the pipeline and analyze images in the same way. In machine learning tasks, the pre-processing step is often one of the most important.
- If the images are different sizes, or even cropped differently, then this counting tactic will likely fail! So, it's important to pre-process these images so that they are standardized before they move along the pipeline
- We have standardized our images with mean and median and then we have performed RandomResizedCrop(224) to resize our images and set its pixel value to 224 because VGG16 works with 224-pixel size image value.
- Standard normalization was applied on whole dataset i.e on train, test and validation sets

```
standard_normalization = transforms.Normaliza(mean=[0.485, 0.456,0.406], std=[0.229, 0.224, 0.225])

data_transforms = {'train': transforms.Compose([transforms.RandomResizedCrop(224),
                                    transforms.RandomHorizontalFlip(),
                                     transforms.ToTensor(),
                                     standard_normalization]),
'val': transforms.Compose([transforms.Resize(256),
                                    transforms.CenterCrop(224),
                                    transforms.ToTensor(),
                                    standard_normalization]),
 'test': transforms.Compose([transforms.Resize(size=(224,224)),
                                    transforms.ToTensor(),
                                    standard_normalization]) }
```

Implementation:

- We have used 3 layer architecture for implementing this Convolutional Neural Networks.
- Both 1st and 2nd layer uses kernel size of value 3,stride=2 and padding =1. Except for the 3rd layer i.e., last layer padding was applied on every layer.
- We have also applied the max-pooling layer after every layer of the architecture. The max-pooling layer is of the size (2,2).
- Dropout used in this 3-layer architecture layer to overcome the problem of overfitting. Dropout value was set to 0.3
- 2 Fully connected layer is implemented towards the end of our architecture. It gives the prediction of 1 class out of 133 class.

```
Net(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=6272, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=133, bias=True)
  (dropout): Dropout(p=0.3)
)
```

Compiling the model:

For compiling our model we will use the following 2 parameters:-

- Loss function – we will use Cross Entropy Loss for this problem st atement.
- Optimizer – we will use the Adam optimizer which is the best as w ell as fast optimizer in many problems.
  import torch.optim as optim

  ### TODO: select loss function
  criterion_scratch = nn.CrossEntropyLoss()

  ### TODO: select optimizer
  optimizer_scratch = optim.Adam(model_scratch.parameters(), lr=0 .0001)

Training the model:

For training our model we will start with 40 epochs which is the number of times our model will go through the cycle. For our custom model we Get less accuracy. We get an accuracy of about 16% even after 40 epoch s. We will send our dataset in batches to our model. Doing batching mak es execution fast as well as improves the model faster. We will save our model weights when our validation loss gets lesser as compare to our tra ining loss. At epoch 40 we get validation loss of about 3.666922.

```
Test Loss: 3.666922


Test Accuracy: 16% (138/836)
```

4.Refinement:

- When we create a CNN model from scratch with 3 layer architectu re then we get less accuracy.
- So then we will use the Transfer Learning approach for training ou r model with a pre-trained model. By doing this we get accuracy g reater than our custom model accuracy.
- By using the resnet50 model we get an accuracy of about 85% eve n at epoch 20, this shows how powerful our resnet50 model is.
- We add our own fully connected layer at the end of the resnet50 m odel to produce outcomes of 133 classes of dogs.

```
    ,
    (2): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stat
      (relu): ReLU(inplace)
    )
  )
  (avgpool): AvgPool2d(kernel_size=7, stride=1, padding=0)
  (fc): Linear(in_features=2048, out_features=133, bias=True)
)
```

Results:

Our model gives accuracy of 85% when using learning rate 0.0001 . We can get even more accuracy by training it at more number of epochs.

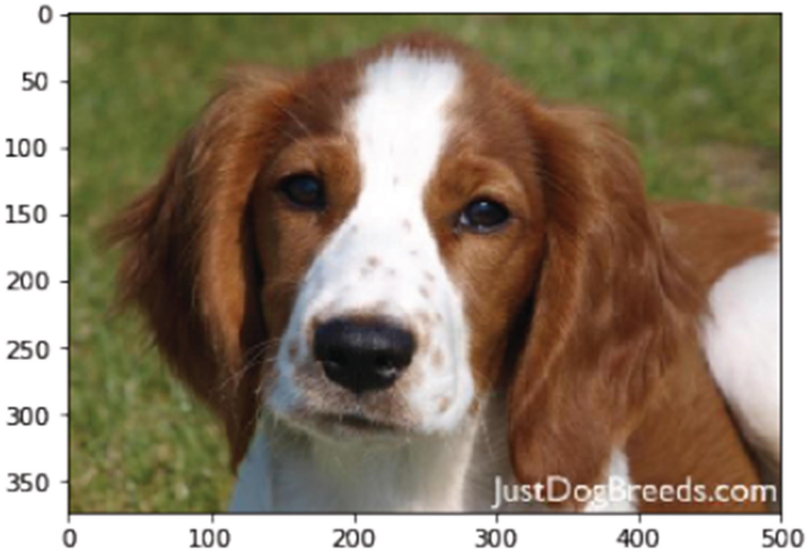
```
In [31]: test(loader_transfer, model_transfer, criterion_transfer, use_cuda)

Test Loss: 0.493131


Test Accuracy: 85% (718/836)
```

We will test the evaluation of our model on some images to see how our model is doing and predicting the dogs breed as in figure below.

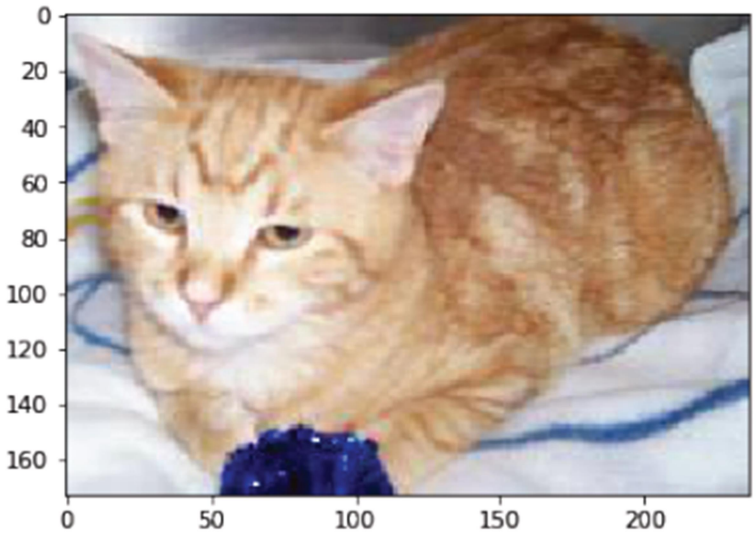
```
image_file_name: ./images/Welsh_springer_spaniel_08203.jpg,         predition breed: Afghan hou
image_file_name: ./images/sample_human_output.png,         predition breed: Portuguese water do
image_file_name: ./images/Labrador_retriever_06457.jpg,         predition breed: Giant schnauze
image_file_name: ./images/Curly-coated_retriever_03896.jpg,         predition breed: Portuguese
image_file_name: ./images/sample_cnn.png,         predition breed: Portuguese water dog
```




```
Dog is detected !!!!!! Ohh i think it looks Afghan hound
```

Above figure accurately predicts the dogs breed. The prediction breed of dogs is Afghan hound

I have even try and test our model by passing even more different types of images of dogs then also our model is behaving accurately. I even pas sed images of cats and humans to see how it will predict the different typ es and kinds of images




```
Error!!!! Neither Dogs are found nor Human in the given supplied image
```

5.Conclusion:-

- We get the accuracy of 16% on our custom model versus the benchmark accuracy for custom model is 10%.
- We even get accuracy of 85% as compare to benchmark set for model is 60%.
- By making such types of models where we can predict the dog's breed will help us a lot in the real world. We can separate dogs who have such kinds of diseases and those who are dangerous for human beings and then we can provide treat them.
- At animal shops also, even humans face problems of identifying the breed of dogs. They just buy that dog and when they take that dog at their home then they came to know that, that dog is dangerous for them. So by making such types of models, we can give peoples such apps so that they can buy less harmful and even more funny behaving dogs for their homes.