Machine Learning Engineer Nanodegree

Capstone Project

Carlos Zapapata Huanca
February 27, 2021

Dog Breed Classification

Definition

Project Overview

Thanks to technological advancement in recent decades, increasing computer processing capacity, ML evolves using models that are more accurate and reliable than humans themselves. One of the areas where progress has been most notable is image recognition. It is a discipline that calls my attention by all to the different Applications that it is possible to build.
To Dog breed Identification, basically the algorithm be able to identify the breed of the dog according to a provided image, for this applies techniques of classification of images based on Convolutional Neural Networks (CNN), which need to be trained with a significant number of samples in order to capture unique characteristics.
After this project, I plan to continue research in order to develop projects that contribute to Health and Medicine, imaging based on comparison of previous diagnoses.

Problem Statement

The problem consist to determine the breed of a Dog where an image is the input data. It's a problem of Supervised Learning for which Convolutional Neural Network (CNN) how classification model will be applied.

The first thing is to make a distinction between a human face and a dog image based on detection and classification algorithms.

The second thing is to be able to determine the breed of a dog given the previous condition, that is to say, firstly to recognize that the image is about a dog.

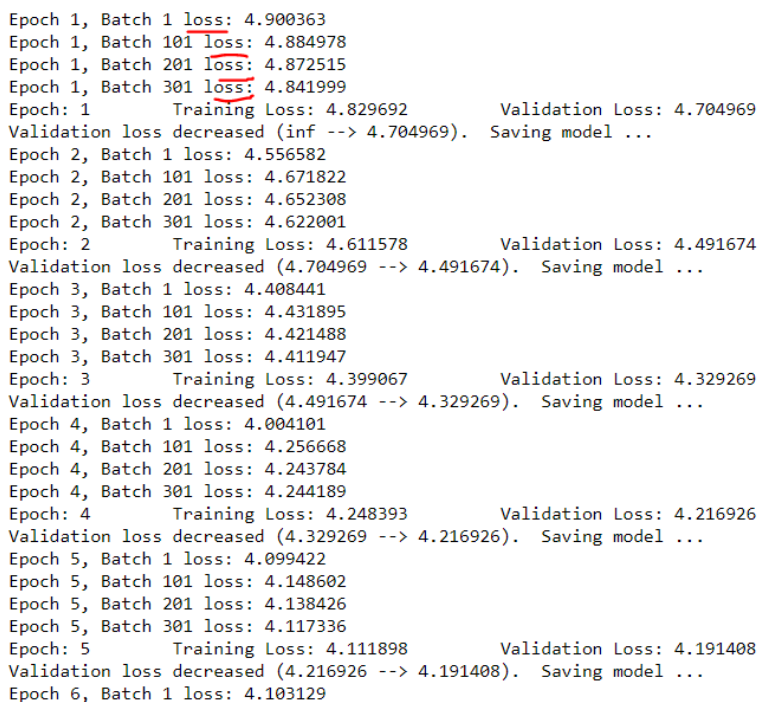
Metrics

In this case, 2 metrics wiil be used:

1.-Accuracy to measure the performance of our algorithm and a way to test our Model because we have multiple categories associated to dog breeds:

Accuracy = (Number of images classified correctly)/(Total Number of Images)

2.- Since we have dataset unbalanced (the number of images in each sub-directory varies), whereby Log Loss will help in evaluate the model.



```
Epoch 1, Batch 1 loss: 4.900363
Epoch 1, Batch 101 loss: 4.884978
Epoch 1, Batch 201 loss: 4.872515
Epoch 1, Batch 301 loss: 4.841999
Epoch: 1        Training Loss: 4.829692         Validation Loss: 4.704969
Validation loss decreased (inf --> 4.704969).  Saving model ...
Epoch 2, Batch 1 loss: 4.556582
Epoch 2, Batch 101 loss: 4.671822
Epoch 2, Batch 201 loss: 4.652308
Epoch 2, Batch 301 loss: 4.622001
Epoch: 2        Training Loss: 4.611578         Validation Loss: 4.491674
Validation loss decreased (4.704969 --> 4.491674).  Saving model ...
Epoch 3, Batch 1 loss: 4.408441
Epoch 3, Batch 101 loss: 4.431895
Epoch 3, Batch 201 loss: 4.421488
Epoch 3, Batch 301 loss: 4.411947
Epoch: 3        Training Loss: 4.399067         Validation Loss: 4.329269
Validation loss decreased (4.491674 --> 4.329269).  Saving model ...
Epoch 4, Batch 1 loss: 4.004101
Epoch 4, Batch 101 loss: 4.256668
Epoch 4, Batch 201 loss: 4.243784
Epoch 4, Batch 301 loss: 4.244189
Epoch: 4        Training Loss: 4.248393         Validation Loss: 4.216926
Validation loss decreased (4.329269 --> 4.216926).  Saving model ...
Epoch 5, Batch 1 loss: 4.099422
Epoch 5, Batch 101 loss: 4.148602
Epoch 5, Batch 201 loss: 4.138426
Epoch 5, Batch 301 loss: 4.117336
Epoch: 5        Training Loss: 4.111898         Validation Loss: 4.191408
Validation loss decreased (4.216926 --> 4.191408).  Saving model ...
Epoch 6, Batch 1 loss: 4.103129
```

Analysis

Data Exploration

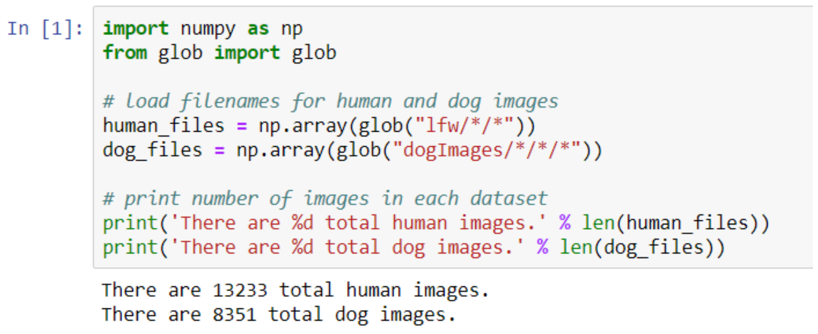
The dataset is provided by Udacity, where the Original repo is possible to find on GitHub. It contains 2 directories, associates with Dog Images and human faces.

- /dogImages → dog dataset
- /lfw → human dataset

The Dog images dataset contains 8351 total dog images that are separated into directories to train(80%), test(10%) and valid(10%). Each directory has 133 Sub-directories and each sub-directory is associated with a breed of Dog. Image have different sizes and different backgrounds. Thus also, the number of images in each sub-directory varies (unbalanced).

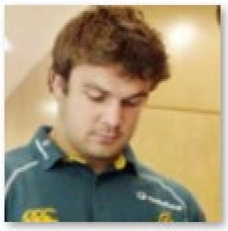
The Human image dataset contains13233 total faces of human images, 5749 directories with human names, and each dorectory can have a different number od images (unbalanced). These images are 250x250 px in size.

Number de images:



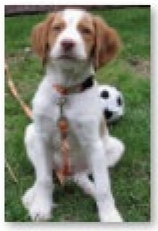
```
In [1]: import numpy as np
        from glob import glob

        # load filenames for human and dog images
        human_files = np.array(glob("lfw/*/*"))
        dog_files = np.array(glob("dogImages/*/*/*"))

        # print number of images in each dataset
        print('There are %d total human images.' % len(human_files))
        print('There are %d total dog images.' % len(dog_files))

        There are 13233 total human images.
        There are 8351 total dog images.
```

Sample images dataset:


Adam_Freier_0001.jpg

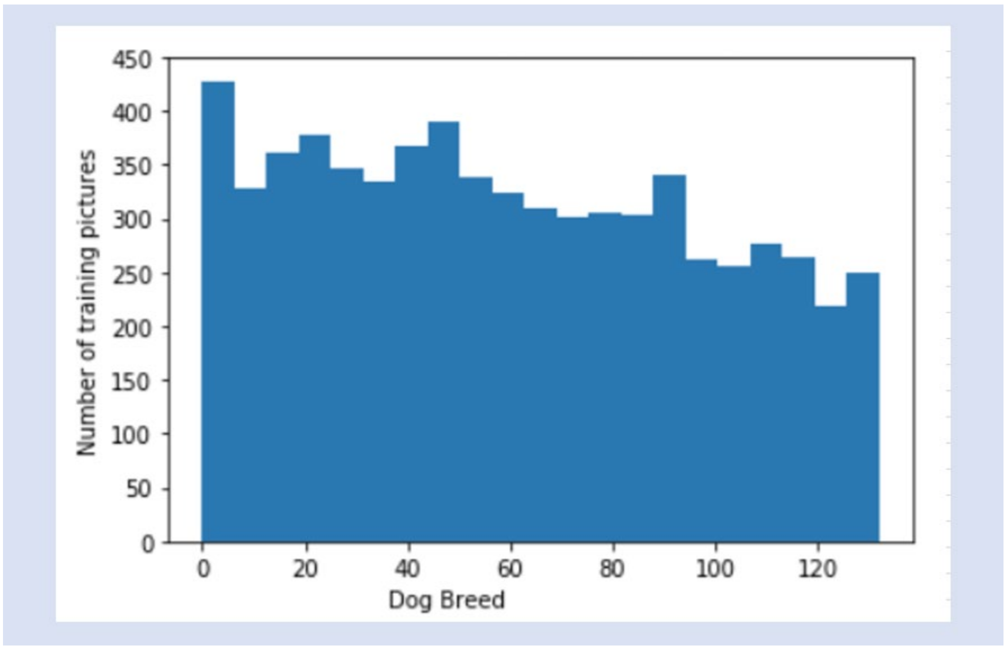
Adrianna_Zuzic_0001.jpg

American_water_spaniel_00648.jpg


Brittany_02625.jpg

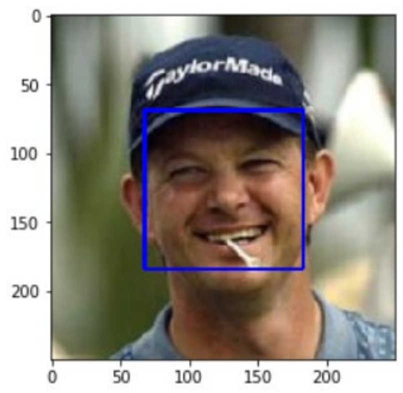
Visualization

In accordance the distribution of data, we notice that dataset is unbalanced. This makes us think that shuffling and splitting is necesary.



Algorithms and Techniques

The first is to be able to to determine wheter the image is human or not based on Face Detection model of OpenCV, it´s a techniques of Haar feature based cascade classifier. So also, it´s a standard procedure to convert the images to grayscale that is used how parameter for the classifier method.



It will permit work with a method called “face_detector” to detect if it´s human or dog image.

```
Humans: 100%|██████████| 100/100 [00:05<00:00, 18.17it/s]
Dogs: 100%|██████████| 100/100 [00:21<00:00,  4.73it/s]

100.00% of humans face detected in human files
9.00% of humans face in dog Files
```

So also, to detect images of dogs is pre-trained using the VGG16 Model.

Once determined that image is about a dog, it proceeds to be classified by Breed, basing in CNN classification Model which process the images and predict matching with one of the 133 breeds provided in dataset.

For training parameters we must take into account:

- Taining length(number of epoch)
- Batch Size(how many images to look at once during a single training step)
- Lr (Learning rate, how fast to learn; could be dynamic)
- Data augmentation(It´s possible resize, strech, rotate, mirror them to obtain more images)

For Neural Network Arquitecture:

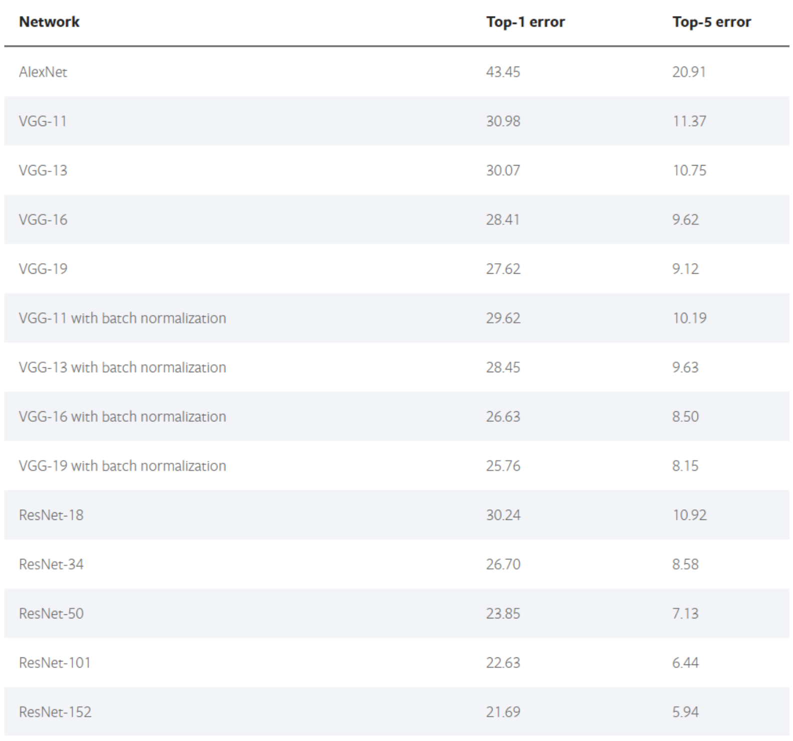
- Number of Layers
- Layer types(convolutional, fully-connected, or poolling)

So also, for training processing is a common practice to use GPU memory.

Benchmark

For Benchmark Model, the Convolutional Neural Neworks model created from scratch require an accuracy of more than 10%. So also a model Conv2d (2 dimensional CNN) is used.

The Convolutional Neural Networks model created using Transfer Learning must have an accuracy of more than 60%. To choose a appropriate model pretrained with ImageNet, we are based on top error the following table provided by pytorch:



Network	Top-1 error	Top-5 error
AlexNet	43.45	20.91
VGG-11	30.98	11.37
VGG-13	30.07	10.75
VGG-16	28.41	9.62
VGG-19	27.62	9.12
VGG-11 with batch normalization	29.62	10.19
VGG-13 with batch normalization	28.45	9.63
VGG-16 with batch normalization	26.63	8.50
VGG-19 with batch normalization	25.76	8.15
ResNet-18	30.24	10.92
ResNet-34	26.70	8.58
ResNet-50	23.85	7.13
ResNet-101	22.63	6.44
ResNet-152	21.69	5.94

ResNet152 will be used, however is possible to test other models and see how it affects the performance.

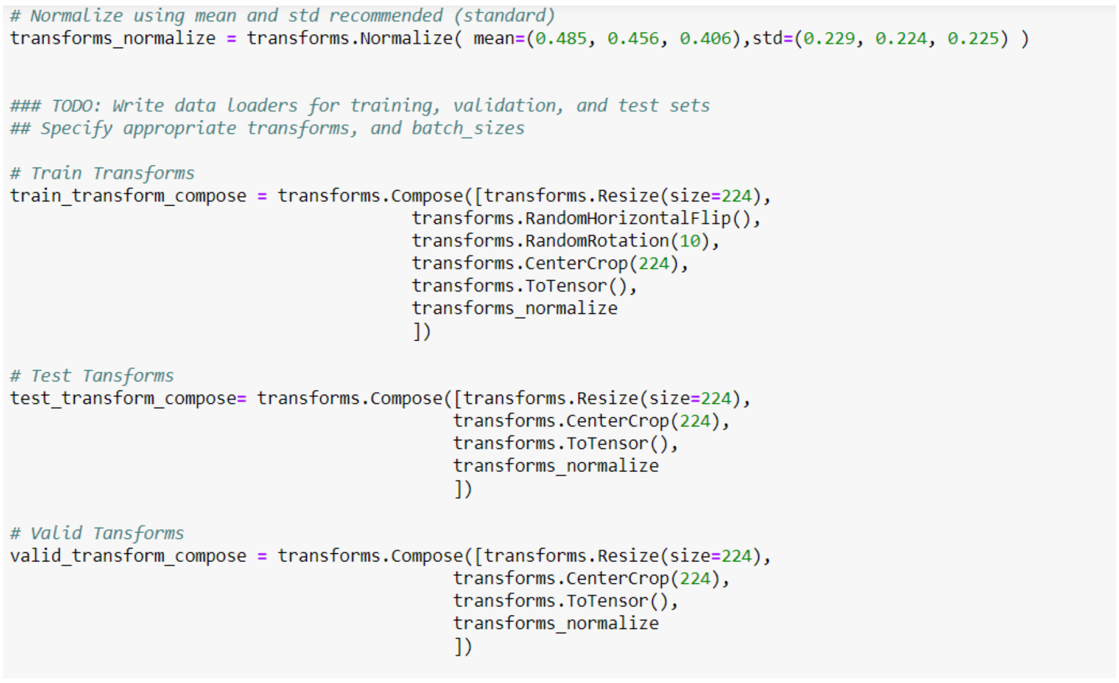
Methodology

Data Preprocessing

Previously to pass the image a CNN model, is good practice to Normalize, Crop , Horizontal Flip, Resize the image of dog dataset , this transformation is necessary due dog dataset not has same size. In accordance with pytorch documentation, the images have to be loaded in to a range of [0-1] and then can use the following transform to normalize:

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

The following transforms is applied to train, test and validation:



```
# Normalize using mean and std recommended (standard)
transforms_normalize = transforms.Normalize( mean=(0.485, 0.456, 0.406),std=(0.229, 0.224, 0.225) )


### TODO: Write data loaders for training, validation, and test sets
## Specify appropriate transforms, and batch_sizes

# Train Transforms
train_transform_compose = transforms.Compose([transforms.Resize(size=224),
                                       transforms.RandomHorizontalFlip(),
                                       transforms.RandomRotation(10),
                                       transforms.CenterCrop(224),
                                       transforms.ToTensor(),
                                       transforms_normalize
                                       ])

# Test Tansforms
test_transform_compose= transforms.Compose([transforms.Resize(size=224),
                                           transforms.CenterCrop(224),
                                           transforms.ToTensor(),
                                           transforms_normalize
                                           ])

# Valid Tansforms
valid_transform_compose = transforms.Compose([transforms.Resize(size=224),
                                           transforms.CenterCrop(224),
                                           transforms.ToTensor(),
                                           transforms_normalize
                                           ])
```

Implementation

Following we present a general scheme for the implementationof the project:

1.-Dataset.
Udacity provided a repo in GitHub where we find folders of Human faces and dog images separatly (/dogImages and /lfw) , since here we will obtain the dataset for train,validation and test our Model.

2.-Human Face Detection.

We are going to use OpenCV model to Human image recognition, model based in Haar Cascade Classifiers to detect human face. It tell us is the image is human or not.

3.-Dog Detection.

In this case we use a pre-trained model (VGG16) to detect Dogs images.

Something important is to use GPU for better performance in processing.

4.-Dog Breed Classification.

- CNN will be created to Classify Dog Breeds from Scratch. Test accuracy must be of at least 10%.
- CNN will be created to Classify Dog Breeds using Tansfer Learning. We will use ResNet152 as Model Arquitecture and obtein at least 60% of accuracy.

To load Train, test and valid, I applied batch_size=20, it´s in accordance with resNet152 to do faster the training proccess, so also batch_size=10 is tested.

```
train_DataLoader = torch.utils.data.DataLoader(train_dataset,batch_size=20,shuffle=True,num_workers=0)
test_DataLoader = torch.utils.data.DataLoader(test_dataset,batch_size=20,shuffle=False,num_workers=0)
valid_DataLoader = torch.utils.data.DataLoader(valid_dataset,batch_size=20,shuffle=True,num_workers=0)
```

In order to implement Network Arquitecture, all convolutiona layers have kernel size of 3, stride is 2 in conv1 y conv2, all paddings is 1. The pooling layer of (2,2) is used to reduce the dimensionality. Dropout of 0.3 used to avoid overfitting.

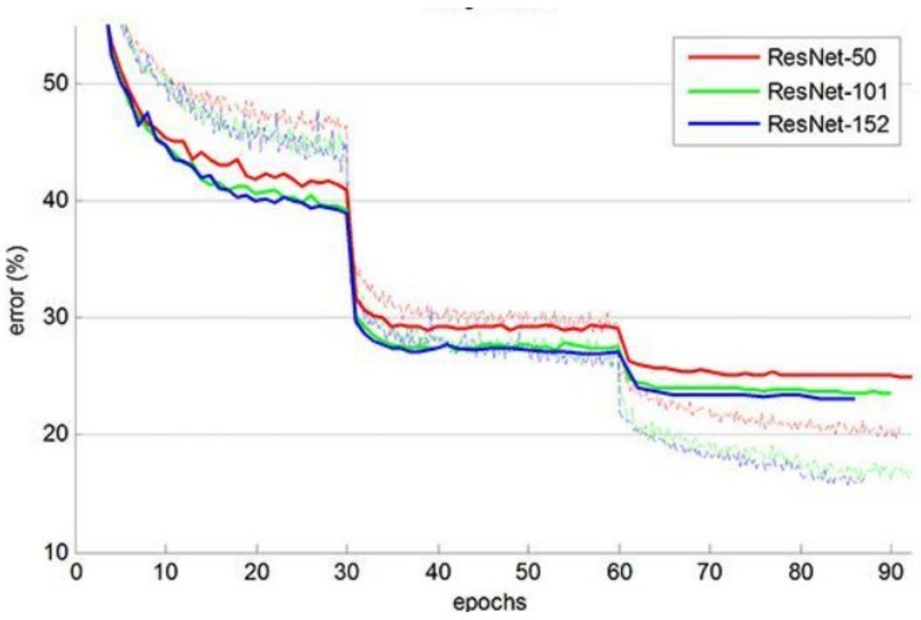
```
# define the CNN architecture
class Net(nn.Module):
    ### TODO: choose an architecture, and complete the class
    def __init__(self):
        super(Net, self).__init__()
        ## CNN Layer
        self.conv1 = nn.Conv2d(3, 32, 3, stride=2, padding=1)
        self.conv2 = nn.Conv2d(32, 64, 3, stride=2, padding=1)
        self.conv3 = nn.Conv2d(64, 128, 3, padding=1)

        # pool
        self.pool = nn.MaxPool2d(2, 2)

        # Linear Layer
        self.Fc1 = nn.Linear(7*7*128, 500)
        self.Fc2 = nn.Linear(500, 133)  # Number of classes of dog breeds=133

        # drop-out
        self.dropout = nn.Dropout(0.3)

    def forward(self, x):
        ## Define forward behavior
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = F.relu(self.conv3(x))
        x = self.pool(x)

        # flatten
        x = x.view(-1, 7*7*128)

        x = self.dropout(x)
        x = F.relu(self.Fc1(x))

        x = self.dropout(x)
        x = self.Fc2(x)
        return x
```

Refinement

The accuracy to the CNN from scratch is :

```
Test Loss: 3.815346


Test Accuracy: 15% (131/836)
```

Is higher that required 10%, it will be improved using Tansfer Learning. I selected the ResNet152 Arquitecture pre-trained on ImageNet dataset. ResNet-152 converges faster, the deeper ResNet achieve better trainig result as copared to the shallow network.



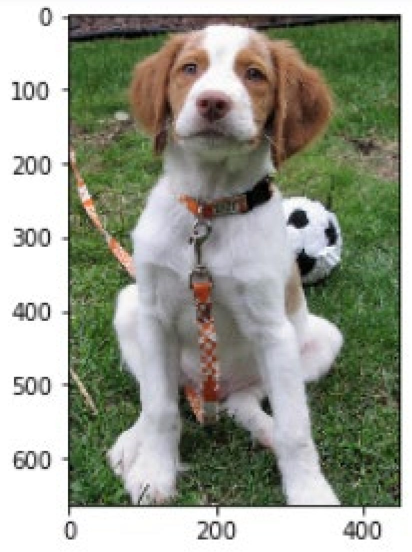
Full connected layer has 133 dimensional output, this is by each category in dog dataset. So with o nly 1 Epoch, the model got:

```
 Test Loss: 0.807759


 Test Accuracy: 80% (671/836)
```

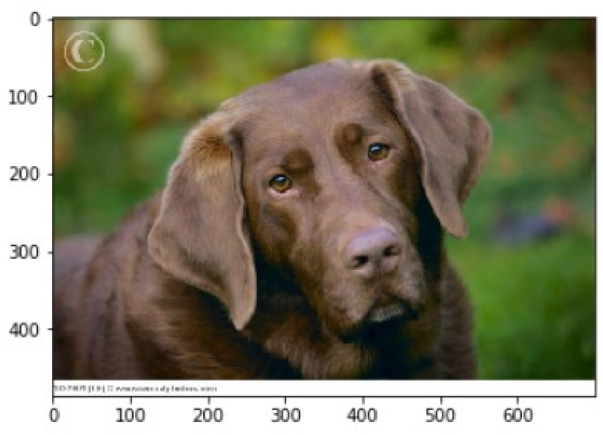
So also, accuracy using 2 epochs:

```
2 epoch - batch size 10

Test Loss: 0.516812


Test Accuracy: 83% (695/836)
```

Some Sample outputs predicted using 1 epoch:



```
Dogs Detected!!
Almost certainly is Brittany
```



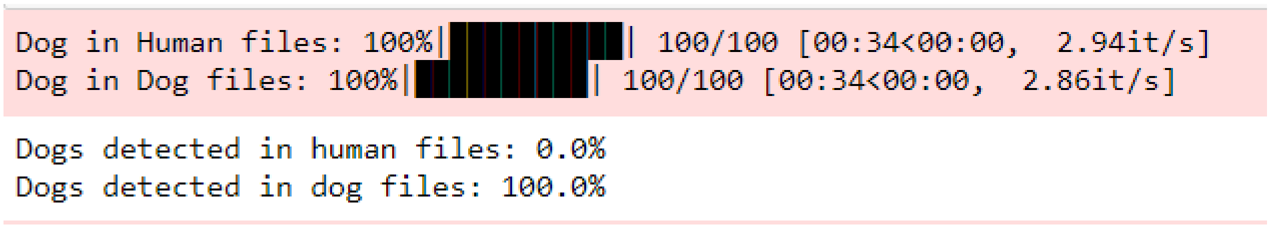
```
Dogs Detected!!
Almost certainly is Labrador retriever
```

Results

Model Evaluation and Validation

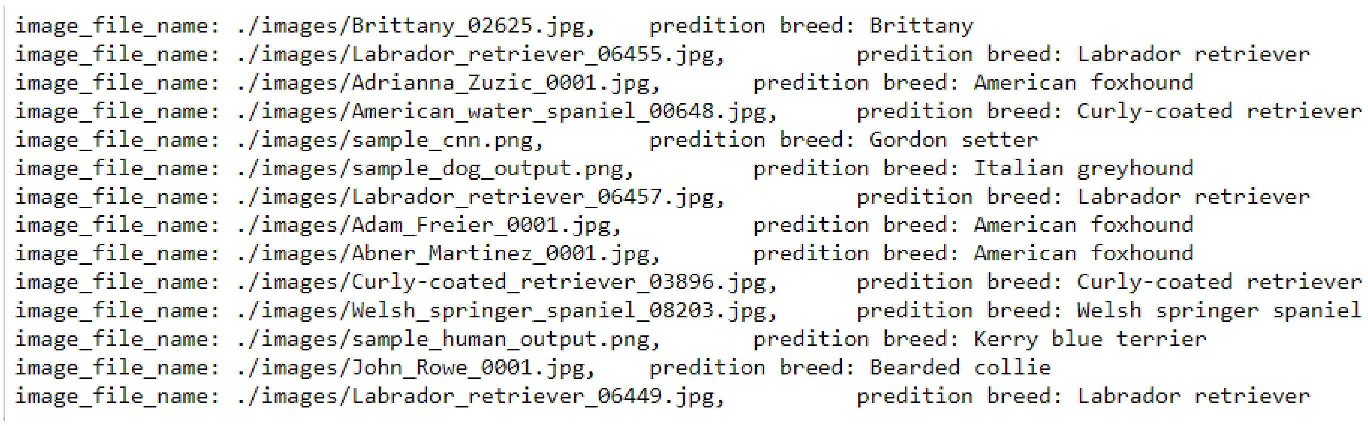
After having provided a set of images, we should be able to evaluate:

a.-If the image provided is a dog, the breeds is predicted. So evaluating the dog detector:



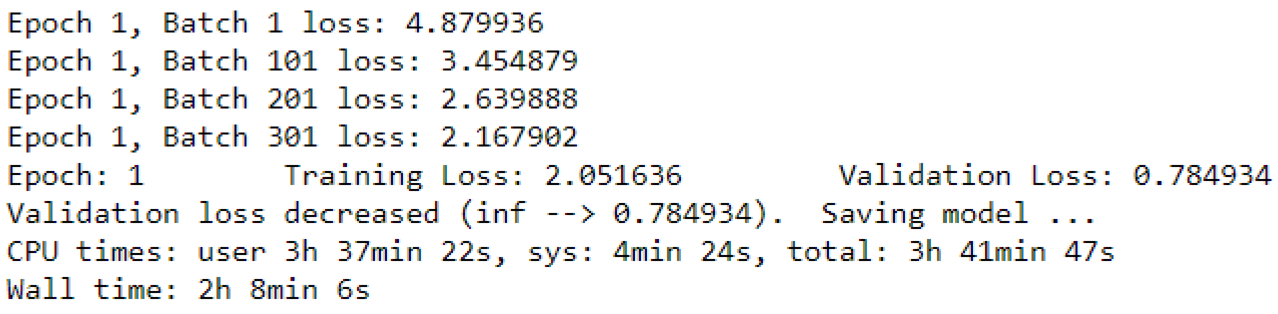
```
Dog in Human files: 100%|██████████| 100/100 [00:34<00:00,  2.94it/s]
Dog in Dog files: 100%|██████████| 100/100 [00:34<00:00,  2.86it/s]

Dogs detected in human files: 0.0%
Dogs detected in dog files: 100.0%
```

Some samples of results:

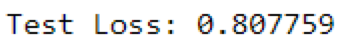


```
image_file_name: ./images/Brittany_02625.jpg,    predition breed: Brittany
image_file_name: ./images/Labrador_retriever_06455.jpg,         predition breed: Labrador retriever
image_file_name: ./images/Adrianna_Zuzic_0001.jpg,       predition breed: American foxhound
image_file_name: ./images/American_water_spaniel_00648.jpg,     predition breed: Curly-coated retriever
image_file_name: ./images/sample_cnn.png,        predition breed: Gordon setter
image_file_name: ./images/sample_dog_output.png,         predition breed: Italian greyhound
image_file_name: ./images/Labrador_retriever_06457.jpg,         predition breed: Labrador retriever
image_file_name: ./images/Adam_Freier_0001.jpg,          predition breed: American foxhound
image_file_name: ./images/Abner_Martinez_0001.jpg,       predition breed: American foxhound
image_file_name: ./images/Curly-coated_retriever_03896.jpg,     predition breed: Curly-coated retriever
image_file_name: ./images/Welsh_springer_spaniel_08203.jpg,     predition breed: Welsh springer spaniel
image_file_name: ./images/sample_human_output.png,       predition breed: Kerry blue terrier
image_file_name: ./images/John_Rowe_0001.jpg,    predition breed: Bearded collie
image_file_name: ./images/Labrador_retriever_06449.jpg,         predition breed: Labrador retriever
```

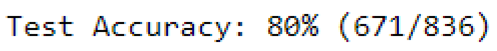
It´s using 1 epoch:



```
Epoch 1, Batch 1 loss: 4.879936
Epoch 1, Batch 101 loss: 3.454879
Epoch 1, Batch 201 loss: 2.639888
Epoch 1, Batch 301 loss: 2.167902
Epoch: 1        Training Loss: 2.051636         Validation Loss: 0.784934
Validation loss decreased (inf --> 0.784934).  Saving model ...
CPU times: user 3h 37min 22s, sys: 4min 24s, total: 3h 41min 47s
Wall time: 2h 8min 6s
```



```
Test Loss: 0.807759
```

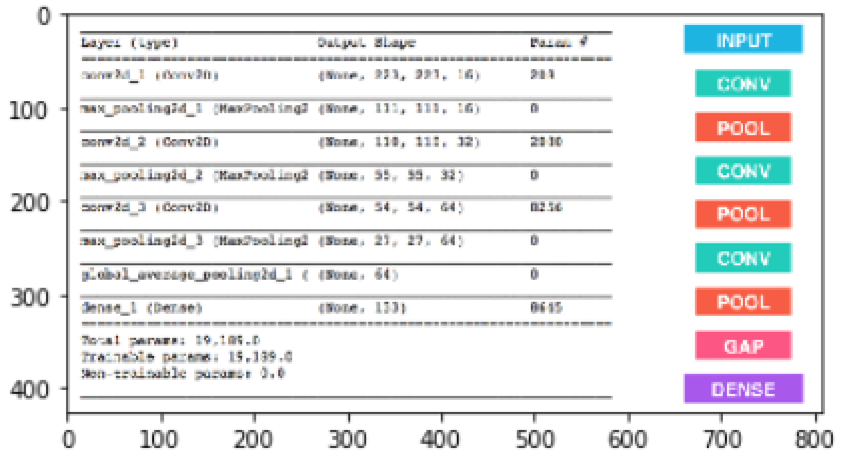


```
Test Accuracy: 80% (671/836)
```

b.- If the image provided is a human, human is predicted and most similar breed of dog returns.

```
Human Detected!!
You look like a Cane corso
```

c.-Otherwise indicates error.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 223, 223, 16)      208
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 110, 110, 32)      2080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 55, 55, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 54, 54, 64)        8256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 27, 27, 64)        0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 133)               8645
=================================================================
Total params: 19,189.0
Trainable params: 19,189.0
Non-trainable params: 0.0
_________________________________________________________________
```

```
Error! Can't detect anything..
```

Justification

The model created using Transfer Learning has an accuracy of 80%(1 epoch) and 83% (2 epoch) in reference to the 15% CNN model created from scratch. So the ResNet152 permit good performance but is lower (2h 8min 6s to first epoch).

Conclusion

Reflection

- Since image processing has estándar parameters, help to determine the model used with more accuracy.
- Due the dataset provided by Udacity help to focus over the selection of good model and tunning of parameters.

Improvement

- Is possible to follow tunning Convolutional Network Parameters
- Evaluate at least 5 epoch using ResNet-152
- Is possible increase the batch size to do faster the processing.
- Increase the nuber convolutional layer.

References:

- Project Repo in GitHub: https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification
- ResNet152: https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html#resnet152
- Pytorch docs: https://pytorch.org/docs/master/
- CNN: https://en.wikipedia.org/wiki/Convolutional_neural_network
- Kaggle-Dog breed identification: https://www.kaggle.com/c/dog-breed-identification/overview/description
- Residual CNN: https://neurohive.io/en/popular-networks/resnet/