**Capstone Project** Injeti Ravi Varma

Machine Learning Engineer Nano Degree April 22nd , 2020

# **Domain Background**

The Dog breed classifier is a well-known problem in Machine Learning. The task is to identify a breed of dog if dog image is given as input, even if supplied an image of a human, we have to identify the resemblance of dog breed in the image. The idea is to build and develop a pipeline that can process real-world, user-supplied images and identify an estimate of the canine’s breed. This is a multi-class classification problem where we can use supervised machine learning to solve this problem.

# **Problem Statement**

The main aim of this project is to build and develop a Machine learning model that can be used within the webapp to process real world and user supplied images. Here, the algorithm performs two different tasks:

***Dog face detector*:** After giving the image of dog as input the algorithm must identify an estimate of the canine’s breed.

***Human Face Detector*:** after giving the image of human, the algorithm must identify the resemblance of the dog breed.

1. **Metrics**

The data we have is already is split into train, valid and test dataset. From these datasets the train dataset is trained by using the model. The results of this trained model was compared with the test data(unseen data) to predict the performance of the model. To find out the how far the model predicted or classified the data with the test data we calculated the accuracy where,

During the Training we compare the prediction of test data with the validation dataset by calculating the multi class log loss which is used for to find the best performing model Where this log-loss consider the abnormal prediction and determined how much it varies from the actual label.

# **Datasets & Inputs**

For this project, the input must be in the image format, because using the image as input and identification of the dog breed will be estimated. To do this the Dataset was provided by Udacity which contains the dogs and human’s dataset.

So, in total the Dog image dataset has 8351 images where it was splitted in to 6680 images for training, 836 images for testing and 835 images for validation are saved into directories. Each

of this directory (train, test, valid) have 133 folders corresponding to dog breeds. The images are of different sizes and different backgrounds, some images are not full-sized. The data is not balanced because the number of images provided for each breed varies. Few have 4 images while some have 8 images. And, Human images dataset consists of 13233 images which were sorted with names in 5750 folders with the image size of 250x250 and different background and different angles i.e., data is not balanced.

A dog looking at the camera

Description automatically generated A dog standing on grass

Description automatically generated 

Sample Images of Humans and dogs

## **Algorithms and Techniques**

For performing this multiclass classification, we can use Convolutional Neural Network to solve the problem. A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The solution involves three steps. First, to detect human images, we can use existing algorithm like OpenCV’s implementation of Haar feature based cascade classifiers. Second, to detect dog-images we will use a pretrained VGG16 model. Finally, after the image is identified as dog/human, we can pass this image to an CNN which will process the image and predict the breed that matches the best out of 133 breeds.

1. **Bench Mark**

The Convolution Neural Network Model created from scratch must and sgiuld have accuracy greater tan 10%. This can confirm that the model is working because a random guess wil provide a correct answer roughly 1 in 133 times where which corresponds to an accuracy of less than 1%.

1. **Data Pre-processing**

All the image in the data are resized to the shape of (224\*224). To, prevent the information loss we have scaled the data using the Normalization. For Training the data, image Augmentation is done by applying randomly rotated and horizontal Flip. After the augmentation all the images in the data are converted to tensor values using the Transforms Functions in Pytorch and then passes the data to the model as an input.

1. **Implementation**

To solve the Classification problem First I built the CNN model from scratch which consists of 3 convolutional Layers. These 3 Convolutional Layers has the kernel Size of 3 and stride 1. For the 1st convolution layer takes the input size of 224\*224 where the final convolutional layer provides the 128 size of output. ReLU activation function is used which makes ensure that the model trains efficiently. The pooling layer of (2,2) is used to reduce the size by 2. After this we have 2 fully connected layers which produces the 133-dimensional output for the final layer. A dropout layer of 0.25 is used to avoid the over fitting. Before the training the model we have given the loss function for the cross-entropy loss and Stochastic Gradient Momentum (SGDM) as an optimizer Function with the learning rate of 0.01 and momentum of 0.9.

1. **Refinement**

The Convolutional Neural Network Model created from scratch got the accuracy of 14% where it meets the requirements of the benchmarking model. Even though it meets the Bench Marking model the model can be improved if we use the Transfer Learning. So, to create the Transfer Learning, I Have used the Resnet 101 architecture which is pretrained ImageNet Dataset like VGG16 and which consist of 101 layers of deep. In the Resnet the convolutional output of the last layer is fed as the input to our model. Where, we only need to add a fully connected layer to produce the 133-dimensional output. Comparing to the cnn scratch model this model performs well with 82% of accuracy and trained with 7 epochs.

Hello Dog! Hello Dog!

Predicted breed: Mastiff Predicted breed: Mastiff



Outputs predicted by the model

1. **Model Evaluation and Validation**

So, we have evaluated and validated the model in three steps by doing the face detection, dog face Detection and CNN transfer Learning to improve the model significance.

Human Face Detection: For the Face Detection we have created a function called human face detector by using the open CV’s implementation of Haar feature based cascade classifier. Using this the function have detected 98% of human faces are detected in the first 100 images of human dataset and 17% of human faces are detected in the first 100 image of the dog dataset.

Dog Face Detection: The dog detector function was created using the VGG16 pretrained model. Using this dog Detector Function 100% of dog faces are detected in the first 100 images of dog dataset and 1% of dog faces are detected in the first 100 image of the human dataset.

CNN using the Transfer Learning: The CNN model created using transfer learning with ResNet101 architecture was trained with 7 epochs and the final model produced and accuracy of 82% on test data. The model correctly predicted breeds for 690 out of 836 images(690/836)

**11. Justification**

From the overall results I think the performance of the model is well improved by using the transfer learning with an accuracy of 82% compared to the CNN model created from scratch which has the accuracy of 14%. Moreover, the model can be improved with more augmentation like colour jitter, using the saturation of the image with this we can avoid over fitting and tweaking with the epochs and optimization learning rate.

## **12. Reference**

* Udacity Repo for project: <https://github.com/udacity/deep-learning-v2-pytorch/blob/master/project-dog-classification/>
* Explanation of CNN(cs231n): <https://cs231n.github.io/convolutional-networks/#layers>
* Imagenet Training : <https://github.com/pytorch/examples/blob/97304e232807082c2e7b54c597615dc0ad8f6173/imagenet/main.py#L197-L198>
* Pytorch Documentation: <https://pytorch.org/docs/master/>.
* <https://arxiv.org/abs/1409.1556> : VGG16 Pretrained model.
* <https://arxiv.org/abs/1512.03385>: ResNet101 pretrained model and Architecture