CAPSTONE PROJECT REPORT

DOG BREED CLASSIFICATION

By:

QASIM HASSAN

DEFINITION:

Project Overview

Image Classification is a significant and developing area of Machine Learning yet it is additionally one of the most testing and CPU escalated processes. In the perfect past, Convolutional Neural Network has become out to be a valuable instrument for such CPU escalated task.

Convolution Neural Network

Convolutional systems were motivated by natural procedures in that the network design between neurons takes after the association of the creature visual cortex. Individual cortical neurons react to improvements just in a limited district of the visual field known as the responsive field. The responsive fields of various neurons incompletely cover with the end goal that they spread the whole visual field [1].

Typical Architecture

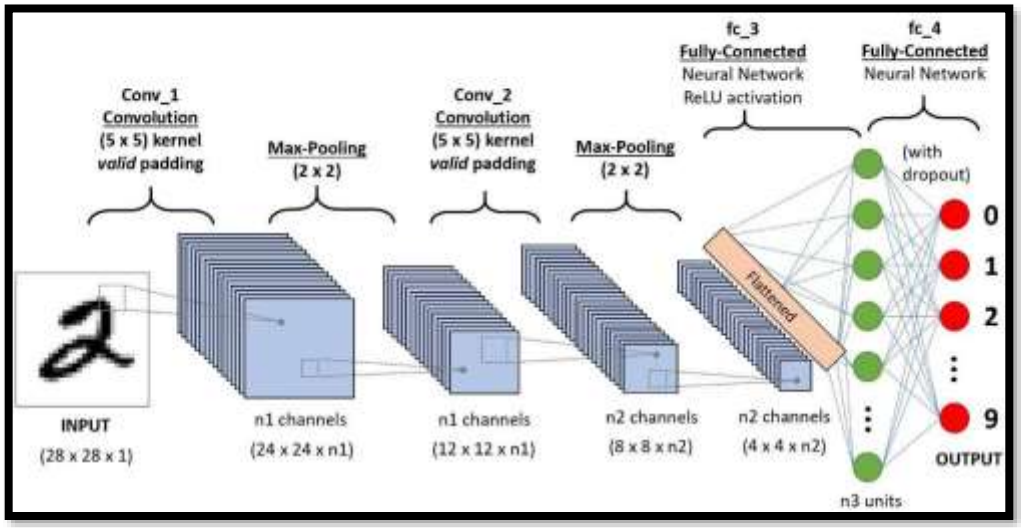


Fig: 1.1 A typical type of Convolutional neural network

PROBLEM STATEMENT:

In this venture task, I am going to code a calculation to recognize the variety of canine in the given pooch picture. To acquire the goal of a dog breed classification is done by using CNN. The task involved are the following:

1. Import Datasets
2. Detect Humans
3. Detect Dogs
4. Create a CNN to Classify Dog Breeds (from Scratch)
5. Create a CNN to Classify Dog Breeds (using Transfer Learning)
6. Write your Algorithm
7. Test Your Algorithm

The final model is expected to detect the dog breed from the input image. If the image contains a human face, then output the dog breed that is most similar to the face.

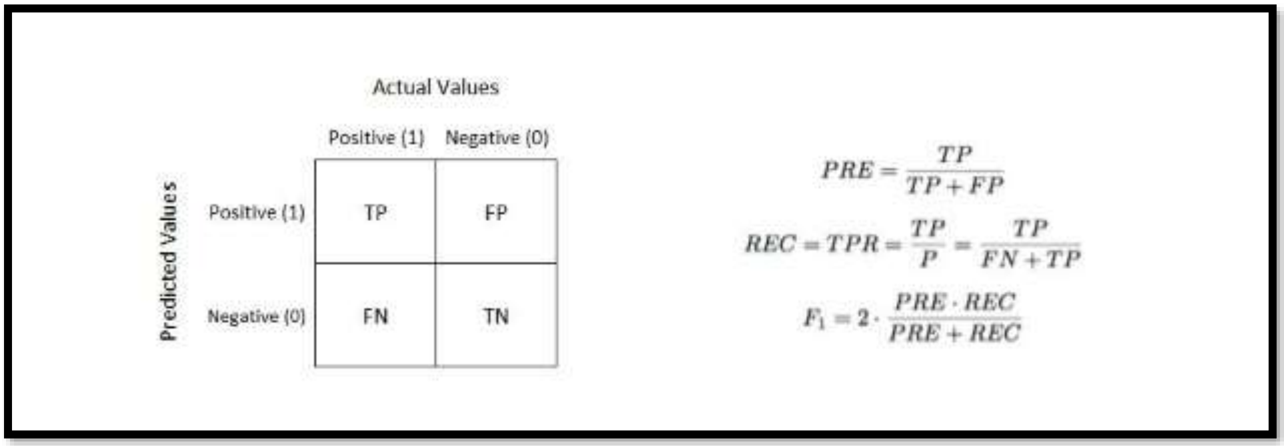
EVALUATION METRICS:

Assessment measurements will be the test set exactness for example how well the model performs on the inconspicuous information. Moreover, how well it can distinguish the Dog-breed chose arbitrarily from the web. So, the evaluation metrics that can be used to evaluate the performance of the machine learning models are:

● **Accuracy**: The ratio of correct predictions to the total size of the data (i.e. (TP+TN)/Data Size).

● **Recall:** The ratio of true positives to the true positive and false negative (i.e. TP/(TP+FN)).

● **Precision**: The ratio of true positives to the true positive and false positive (i.e. TP/(TP+FP)).



$$PRE = \frac{TP}{TP+FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN+TP}$$

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

Figure 1.2 shows the Explanation of Evaluation metrics

In our case, we will be using the accuracy as the metric of measurement to evaluate the performance of the model. There are cases where the metrics such as accuracy might be misleading in the case of imbalance data. One such example is the fraudulent data in which we dealt in the earlier studies of this course. We have seen the percentage of fraud cases being drastically low compared to non-fraudulent cases in the data. However, in this data of dog breeds, as we observe in the visualization part, the data in each class is beyond certain threshold and looks ideally balanced to carry the classification work. Therefore, the accuracy can be used as the metric of evaluation in our work.

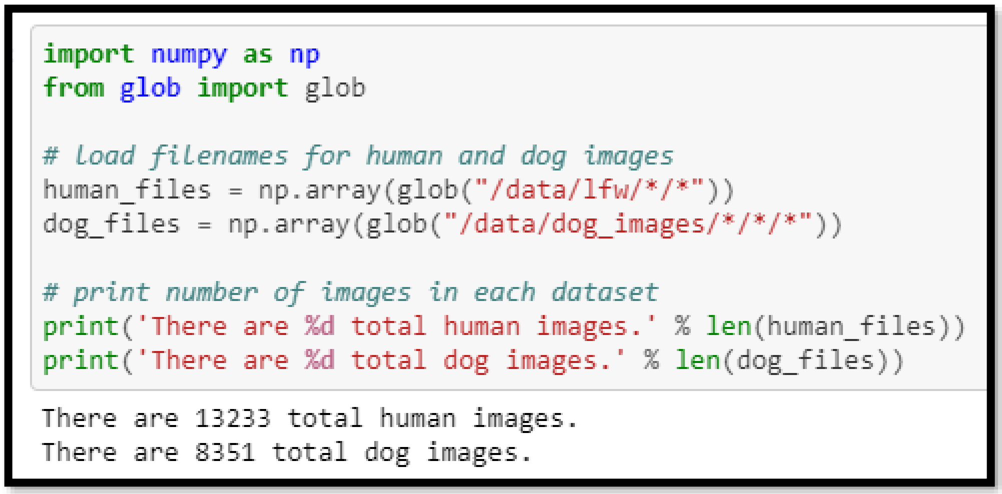
ANALYSIS:

Data Exploration

There are two datasets provided by Udacity in the workspace, although the datasets are public and can be accessed by the following links:

- Human Datasets
- Dog Datasets

Dog Dataset contains 8351 images of 133 dog breed of which 6680 is allocated for training while 835 and 836 is allocated for validation and testing respectively. On the other hand, human dataset contains 13,233 human images.



```
import numpy as np
from glob import glob

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

# 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))
```

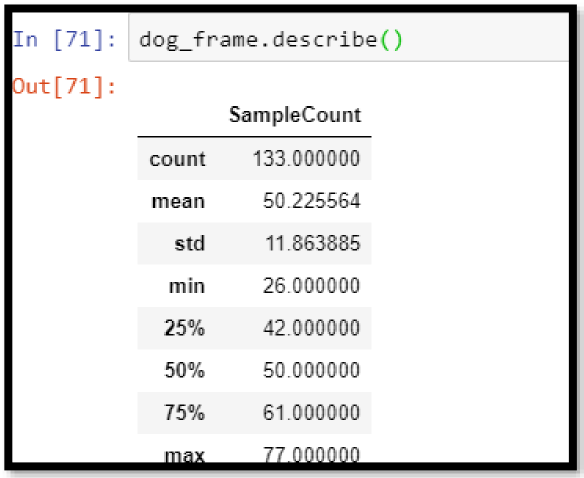
```
There are 13233 total human images.
There are 8351 total dog images.
```

Fig: 1.3 Piece of code from Code

Below are a few example images from the dataset

Fig 1.4 Images from the Data Set

I performed some initial visualization of data set and found every category has at least 26 images at minimum and 77 while average was 50 that shows data is centered sample wise.



```
In [71]: dog_frame.describe()
Out[71]:
```

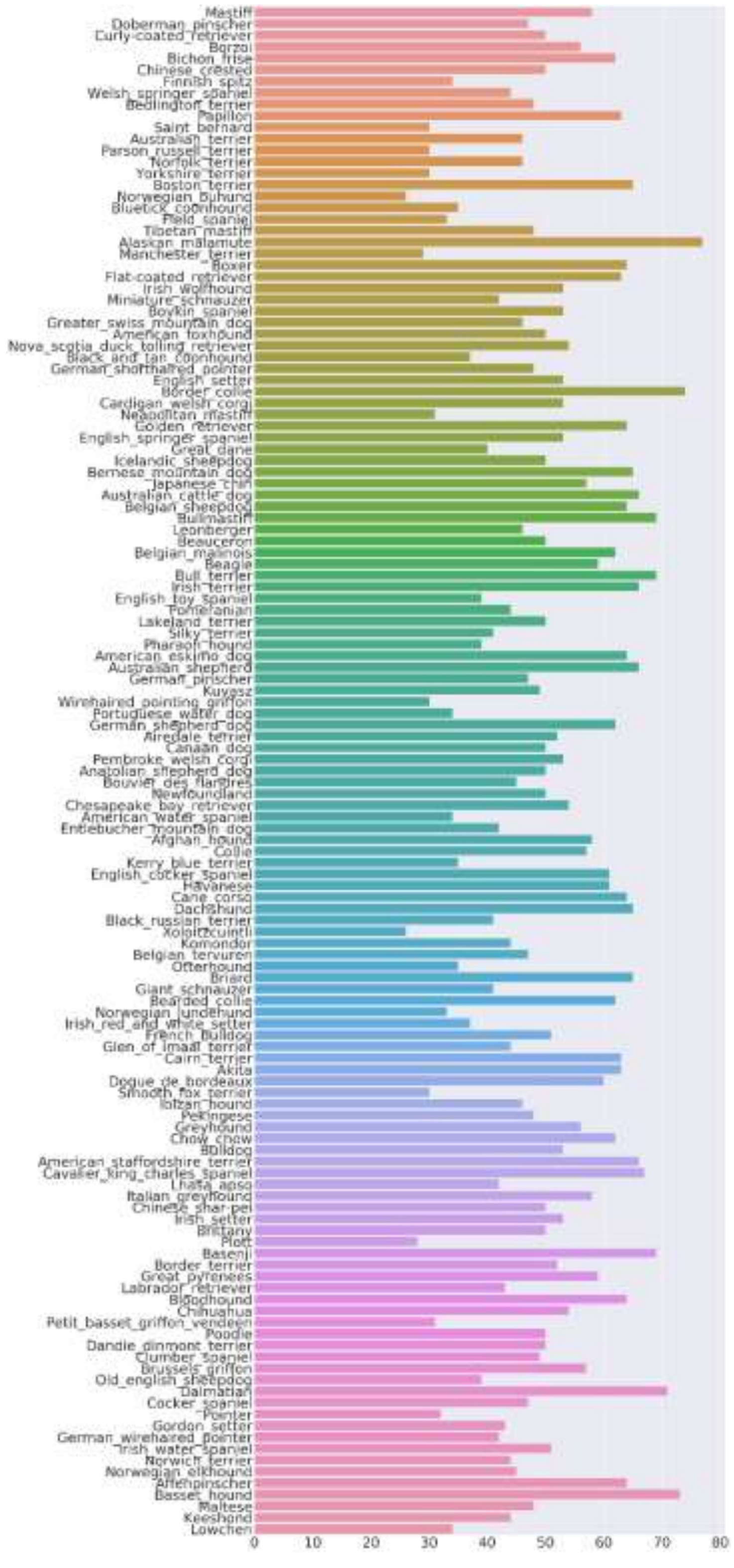
	SampleCount
count	133.000000
mean	50.225564
std	11.863885
min	26.000000
25%	42.000000
50%	50.000000
75%	61.000000
max	77.000000

Fig 1.5 code

I am also inclosing the Distribution Chart at the end of this document.

Exploratory Visualization

The study on the distribution of data gives us the information on balance or imbalance in the data. If there is an imbalance in the data beyond a certain threshold, we must see that the data is balanced by adding relevant images. If the balance in the data is comparatively near to the threshold, it is good to carry forward with the operation. Let us see how it works with our data in the below figure. The plot shows a clear description on breed class with the number of dogs.



ALGORITHMS & TECHNIQUES:

The classifier is a Convolutional Neural Network, which is the state-of-the-art algorithm for most image processing tasks, including classification. It needs a large amount of training data compared to other approaches; fortunately, the COCO and COCO-Text datasets are big enough. The algorithm outputs an assigned probability for each class; this can be used to reduce the number of false positives using a threshold. (The tradeoff is that this increases the number of false negatives.) The following parameters can be tuned to optimize the classifier :

❖ **Classification threshold** (see above paragraph)

❖ **Training parameters**

● Training length (number of epochs)
● Batch size (how many images to look at once during a single training step)
● Solver type (what algorithm to use for learning)
● Learning rate (how fast to learn; this can be dynamic)
● Weight decay (prevents the model being dominated by a few “neurons”)
● Momentum (takes the previous learning step into account when calculating the next one)

❖ **Neural network architecture**

● Number of layers
● Layer types (Convolutional, fully-connected, or pooling)
● Layer parameters (see links above)

❖ **Preprocessing parameters** (see the Data Preprocessing section) During training, both the training and the validation sets are loaded into the RAM. After that, random batches are selected to be loaded into the GPU memory for processing. The training is done using the Mini-batch gradient descent algorithm (with momentum). In contrast to this, inference (that is, inference in the Android app) is done using only the CPU, because TensorFlow doesn’t support smartphone GPUs.

Custom CNN Model

I have decided to create a five layered CNN model approached with a Max Pooling layer after each Convolution layer in a sandwiched manner. The output will be fed to the two fully connected layers that will eventually predict “1” in any of the 133 dog breeds. Dog detection will be assisted via VGG16 model.

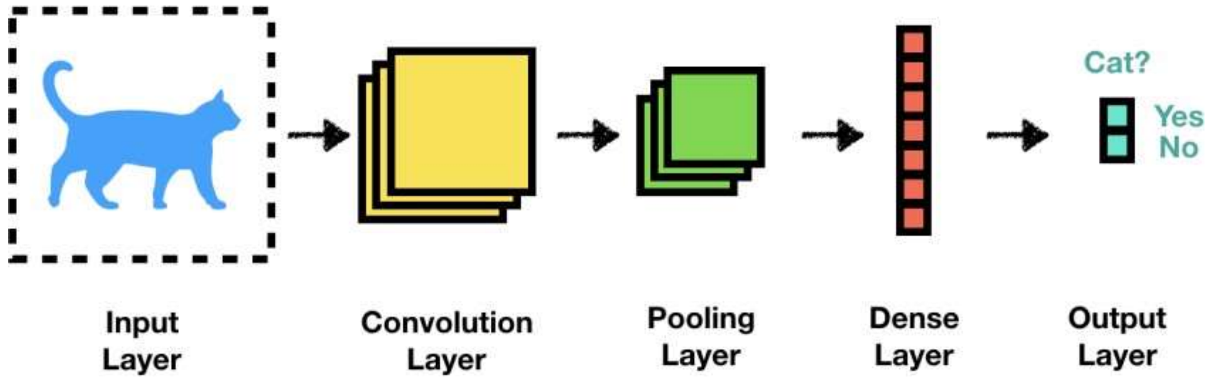


Fig: 1.6

I am then going to compare my custom model with the restnet50, which is exceptionally good in image recognition. Model will be trained further on already learned parameter via transfer learning methodology.

Restnet50:

ResNet-50 is a convolutional neural network that is 50 layers deep. A pre-trained version of the network trained on more than a million images can be downloaded from the ImageNet database. The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. [2]

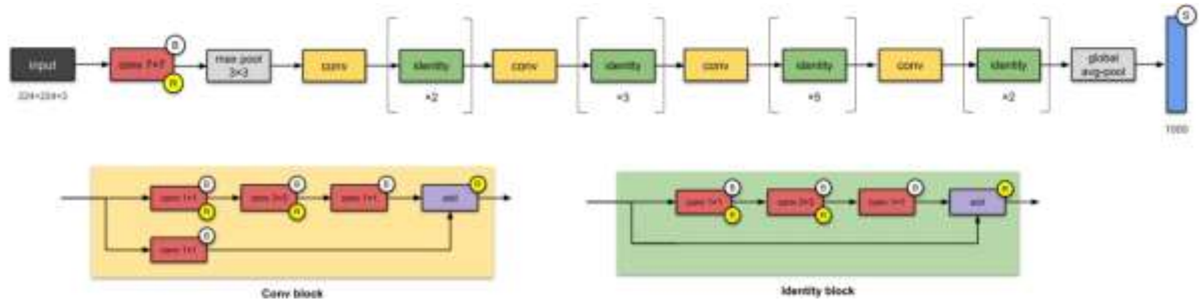


Fig: 1.7

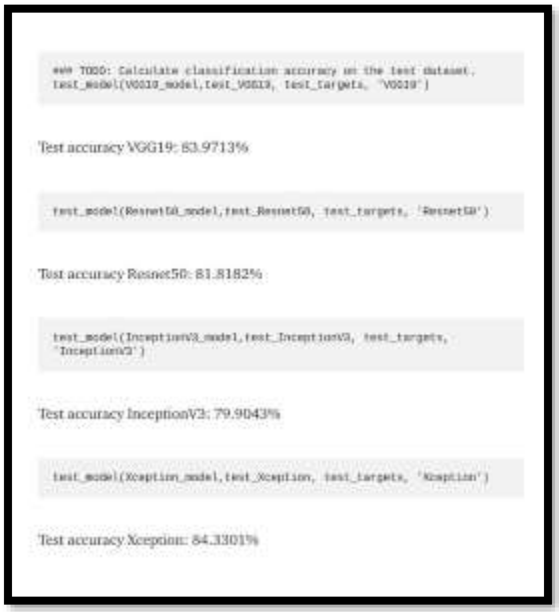
Benchmark

To tackle such data, we use a benchmark model to build a basic pipeline and a well-versioned model to improvise our classification rate. Such a methodology is carried to tune our model for better prediction of results. The benchmark model helps us to make a comparison and reduce the overfitting or underfitting condition and tune the model for a better outcome. Logistic Regression, KNN are such examples of the benchmark. We can also use the predefined image classifiers such as ImageNet, ResNet, VGG16, etc. to classify our images and later optimize our pipeline for better evaluation of metrics.

In the works of Dog Identification in Kaggle, we see that "Mohamed Sheik Ibrahim" used the VGG19, a predefined base model, and carried various processing techniques such as data augmentation to improvise the results obtained from the predefined model. He also used logistic regression to classify the images of dogs and achieved an accuracy of 68%.

Considering another work performed on the same data, using the inception v3, the pre-trained model for image classification, Carey B achieved an accuracy of 83%, which is considered to be a good classification rate based on the performance of the model.

Using the above understandings, We will be using VGG16 for our data for classifying the breeds of the Dogs, Later we build a Convolutional Neural Network and tune the parameters, Using transfer learning through these models, make a comparative study and analyze the performance of the model.



```
### TODO: Calculate classification accuracy on the test dataset.
test_model(VGG19_model,test_VGG19, test_targets, 'VGG19')
```

Test accuracy VGG19: 83.9713%

```
test_model(Resnet50_model,test_Resnet50, test_targets, 'Resnet50')
```

Test accuracy Resnet50: 81.8182%

```
test_model(InceptionV3_model,test_InceptionV3, test_targets,
'InceptionV3')
```

Test accuracy InceptionV3: 79.9043%

```
test_model(Xception_model,test_Xception, test_targets, 'Xception')
```

Test accuracy Xception: 84.3301%

Figure above is the accuracy results of some pre-trained models. I will be using these values to benchmark the performance of our model.

METHODOLOGY:

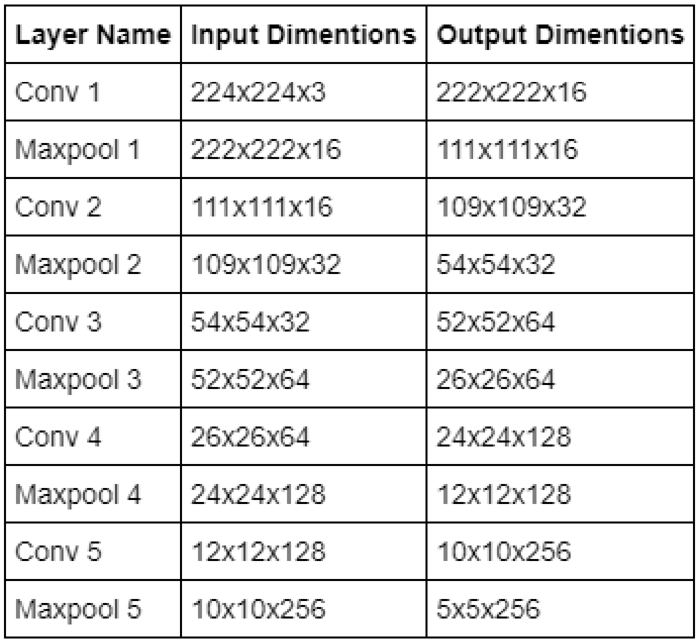
Data Preprocessing

All the images that are fed to the model should have 224 x 224 size, and then standard normalization is applied. To achieve the required size, training images are randomly cropped to this size while validation and test images are resized. Beside this training images are also flipped randomly to increase the model's performance on unseen data.

Data is fed to the model in the PyTorch defined Data Loader format.

Implementation

The custom CNN model has 05 layers, each layer contains one convolution layer and one max pooling layer as recommended in the talk. All convolutional layers have bit size 3 while stride is kept 0. The last yield of these layers is then taken care of to the two completely associated layers which in the end emits 133-dimensional yield each alluding to explicit variety. A drop out can be added to forestall overfitting after the completely associated layer which I have kept away from to keep the model as straightforward as could be expected under the circumstances. The measurement computation of CNN layer is done beneath:



Layer Name	Input Dimentions	Output Dimentions
Conv 1	224x224x3	222x222x16
Maxpool 1	222x222x16	111x111x16
Conv 2	111x111x16	109x109x32
Maxpool 2	109x109x32	54x54x32
Conv 3	54x54x32	52x52x64
Maxpool 3	52x52x64	26x26x64
Conv 4	26x26x64	24x24x128
Maxpool 4	24x24x128	12x12x128
Conv 5	12x12x128	10x10x256
Maxpool 5	10x10x256	5x5x256

Fig: 5

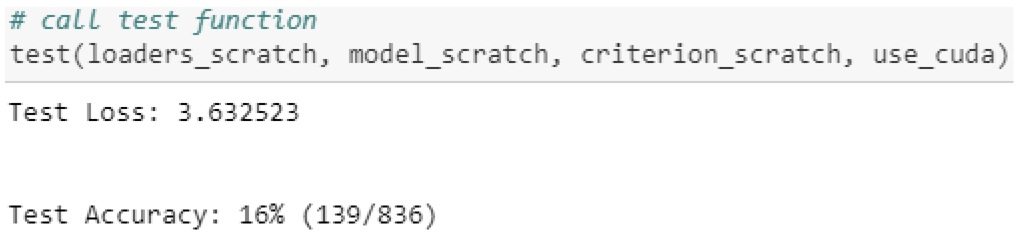
I trained my custom model for 10 epochs and got an accuracy of 9% which was 1% short of the required benchmark. After that started training for 30 more epochs but interrupted the kernel at 8th epoch but eventually got an accuracy of 16% this time which suggest my model surpassed the benchmark in 17 epochs however if I've continued till the last, I could have been a lot better.

Hyper parameters:

- Learning Rate: Default Learning rate of 0.01 is used
- Dropout: I haven't used the dropout in my custom model since the model is not so deep
- Optimization: Stochastic Gradient Decent has been used as an optimization algorithm which is slower than ADAM but performs better
- RELU is used as an activation function for Hidden layer while linear classifier is used for fully connected layer.

Refinement

The CNN model that is created from scratch achieved a test set accuracy of only 16% i.e. it was able to correctly identify only 139 out of 836 images correctly.

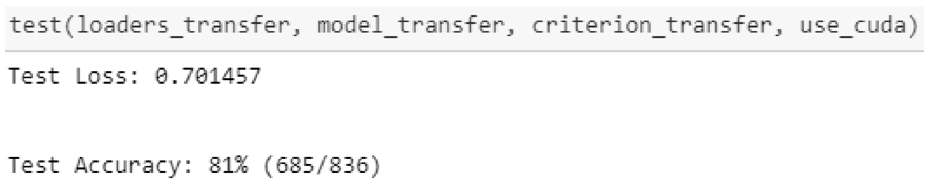


```
# call test function
test(loaders_scratch, model_scratch, criterion_scratch, use_cuda)

Test Loss: 3.632523


Test Accuracy: 16% (139/836)
```

Fig: 6

Although the accuracy score surpassed the required benchmark score, this can be improved by using a pertained model, i.e. a model that has already been trained on some other dataset of images, and training it on our given dataset via a methodology which is termed as transfer learning. The model which I have chosen is ResNet50 which is quite robust and works on the idea called "Identity Shortcut Connection". The final RestNet50 output of 2048 dimensions is fed to the fully connected layer to get the desired 133-dimensional output.

After training for 20 epochs (I trained for 10 epochs two times) with default learning rate of 0.01 with ADAM optimizer, I got a test set accuracy of 81% i.e. the model has correctly predicted 685 images out of 836 from the test set. The obtained accuracy is far better than the benchmark of 60%.



```
test(loaders_transfer, model_transfer, criterion_transfer, use_cuda)

Test Loss: 0.701457


Test Accuracy: 81% (685/836)
```

RESULTS:

In this section, we observe the performance evaluation and results of the final implemented model. Initially, we observe the study made on the detection algorithm and compared their results to see their behavior.

In the implementation phase of the pre-trained model VGG 16, we test the prediction of the model and we also observe that the results show the right prediction value of the dog. On implementing the custom Convolutional Neural Network model, we construct out the relative stack of layers required for our purpose and tune the necessary hyperparameters of the model.

On testing the performance of the custom model through the metric evaluation i.e. accuracy in our case, we observe that the results give us a result of 18% in terms of accuracy. The project had the condition to achieve the results of more than 10% accuracy.

After the implementation and training of the transfer learning model for a total of 7 epochs, we need to evaluate the performance of the given test data of dog images so that the model should predict the breed of the dog with utmost accuracy. Therefore, getting the model ready for evaluating the performance on the test data, we used the test function that was previously defined for the custom model and evaluated the performance of the transfer learning model. We see that the model achieved an accuracy of 77%, and we see that there is a considerable rise in the performance of the model after combining the predefined model with our custom model. If we compare the results with our benchmarking models (VGG19: 83.9%, ResNet: 81.8%, InceptionV3; 79.9%, Xception: 84.33%), we can see that our model scored the comparable but slightly lesser, but keeping in mind that we only used 7 epochs. It doesn't mean that our model is bad, with proper adjustments the accuracy of our model can be increased.

Reflection:

The process used for this project can be summarized using the following steps:

- An initial problem and relevant, public datasets were found
- The data was downloaded and preprocessed
- Classifier was trained using the data multiple times, until a good set of parameters, were found)

Out of these steps, the 3rd step bit hard compared to other steps since it contains network construction and training. But after many tries, I found a good pair of parameters to train the model and got a good model at the last.

Improvement:

Even though I trained the model to predict good results, it doesn’t mean that we can’t further improve the model. Below are some steps that could result in a better model.

- Increase the number convolutional layer
- Adding more dropout layer
- Convolutional Network Parameters
- Altering the Multi-Layer Feed-Forward NN at the end of the network
- Training for a longer duration
- Trying different CNN architecture
- Tuning hyper parameter.

CONCLUSION:

In this project, we see the implementation of Dog Breed Classification. This gives us a detailed description of how the classification of dogs based on their breeds are carried out and implemented. From the data, we study and make an analysis of the split of data, type of data, distribution of data, and also the visualization of how the data is distributed. From the split, we see that 80% of the data is used for training, 10% for validation, and the remaining 10% for the test data. From the visualization of data distribution, we see that though the data is not completely in balance, but the data in all the classes are above a certain threshold and distributed evenly. We also make a comparative study of the detecting algorithms using cascading and local binary pattern.

In the preprocessing phase, we see how the data has been preprocessed using resize, center crop, and normalization. Selectively for training, we also observe the data augmentation been done using the transformation techniques to increase the dataset size for training purposes.

In the implementation phase, we implement the pre-trained model, custom model, and transfer learning model. In the pre-trained model, we use the model to directly make a prediction on the given data. Building our own custom model, we see that the model is implemented using the convolutional architecture as shown in the implementation phase above. We also observe that the custom model gives us a test accuracy of 18% satisfying the required condition and also improvising on fine-tuning the hyperparameters such as epochs, dropout value, and learning rate. In the transfer learning, we see the involvement of a custom model with the pre-defined model and training it for the whole data and see a significant rise in the accuracy level to an output of 77% in test accuracy. On making the comparative study with the benchmark model, we see our model has outperformed the benchmark and also satisfying the condition of above 60% accuracy. However, by using the improvisation techniques, the model can be further improved in terms of evaluation and performance.

References:

1- https://en.wikipedia.org/wiki/Convolutional_neural_network

2- https://www.mathworks.com/help/deeplearning/ref/resnet50.html#:~:text=ResNet%2D50%20is%20a%20convolutional,%2C%20pencil%2C%20and%20many%20animals.

3- https://fossies.org/linux/pytorch/torch/utils/data/dataloader.py

4- https://docs.opencv.org/trunk/db/d28/tutorial_cascade_classifier.html

5- https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b