Capstone Project Report: Dog Breed Classification

(By Piyush Dak)

Project Overview:

Image classification is a supervised machine learning problem in which given a set of input images with associated labels or classifications, we decipher or build a model which can classify unknown images. Image classification is very important in broad range of fields including but not limited to robotics, self-driving cars, medical diagnostics, face recognition, etc. State of art classification models use deep convolution neural networks for image classification. The best algorithm for image classification so far is EfficientNet [Ref1] which achieves an accuracy of 84.4% for top-1/97.1% for top-5 on ImageNet while being 8X smaller and 6X faster than other state-of-art image classifiers.

Problem Statement:

This project aims to build a machine learning pipeline to classify real-world, user-supplied images into different category of dog-breeds. The project involves taking important design decisions for building the neural net to classify the images accurately. The project is divided into two main parts:

1. **Classify an image into Dog. Vs. Human vs. Neither**:
   Given an image classify it as either dog or human or neither of the two.
2. **Dog Classifier**:
   The goal of this part is twofold:
   1) Given a dog image, predict the breed of the dog.
   2) Given a human image, predict the dog breed most resembling with the human.

Evaluation metrics

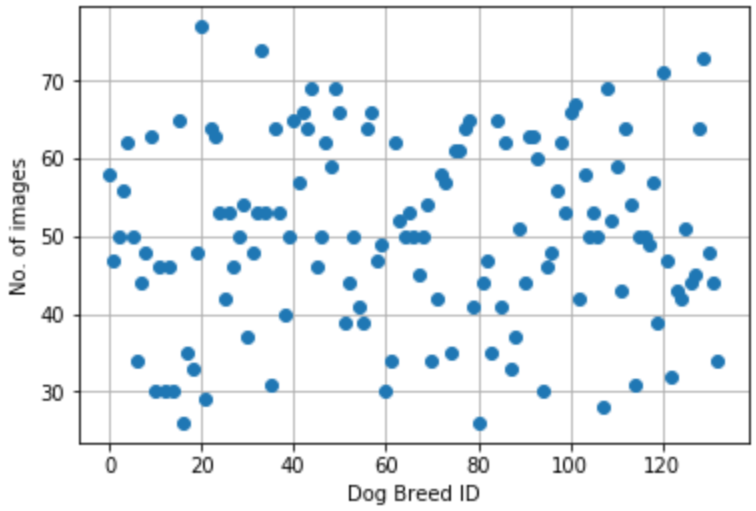
Since, this is multi-class classification problem, I'll be using cross-entropy (log-loss) as the loss function for optimizing problem and finding the best possible parameters for the chosen architecture. Log-loss explodes when the objects are incorrectly classified and approaches zero as the probability of the classification becomes 1. For evaluating the model, I'll use classification accuracy to determine the accuracy of classification. The classification accuracy is defined as:

$$\text{Accuracy} = \frac{\text{Total Correct Predictions}}{\text{Total Sample Size}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Note that even though accuracy is a good predictor of the overall quality of fit, it does tell about the precision i.e., how many selected items are relevant (=TP/(TP+FP)) and recall i.e. how many relevant items are selected (=TP/(TP+FN)). For sake of simplicity of this project, we just use accuracy of the prediction, and treat metrics on precision and recall as future work.

Data Exploration:

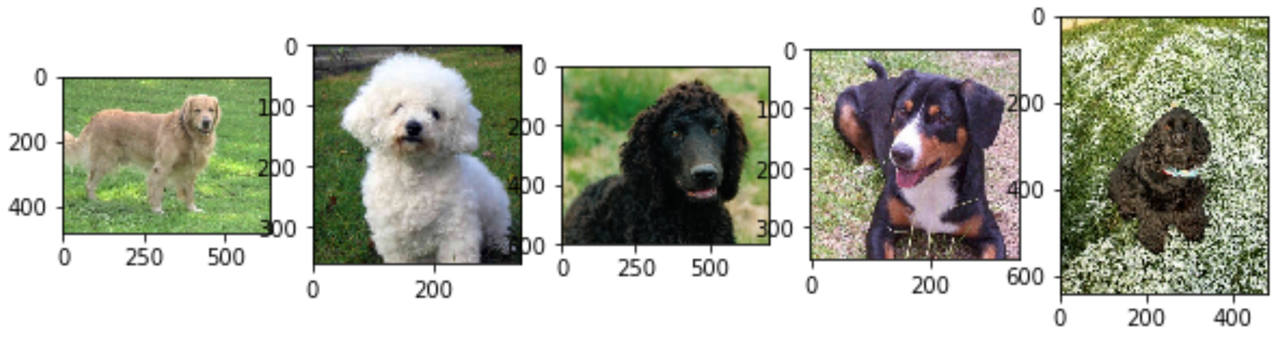
The data source is obtained from Udacity Course Resources for Machine Learning Nanodegree Project. The dataset contains images of 5749 humans (total: 13233 human images) and 133 dogs (total 8351 dog images). For both dog and human images, the data is imbalanced. For example, a significant portion of humans only have 1 image. Luckily, the dog data is not as imbalanced as humans. Each dog has atleast 26 images. Both dog and human images are in different angles and could be either full size or have only face visible. Plot below shows the no. of images for each image id. It shows that there is a huge variation in image counts for each breed. While some breeds have more than 70 images, some others have less than 26. The model is expected to perform worst for breeds with limited or no data.



Note that due to unbalanced nature of dataset, accuracy is not a sufficient metrics for classification for all dog-breeds. A more appropriate approach is to consider the classification accuracy for each of the dog-breeds. However, to meet the project goals within a short duration, I'll keep use overall accuracy metrics for this project.

Image sizes:

The figure blow shows a random sample of dog images. Please note that the dog images are of different size and dog face is in different direction. For building a ML model, we need to resize all these images to same size. Further, dominant part of the dog body is located in different part of the image (center/left/right), it would be good to have a random crop across images to enable us to detect dog body.



Model Benchmark:

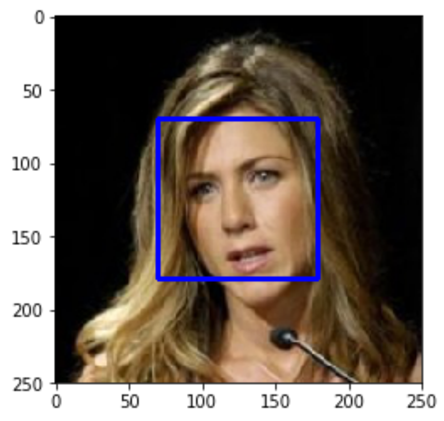
Scratch Model:
For the scratch model, my benchmark is heuristic guess. Since, there are 133 classes of dogs, the probability of a random guess being correct is 1/133 or less than 1%. Both the models i.e. the CNN model built from scratch as well as the model built using transfer learning should perform better than the heuristic approach.

Transfer Learning based Model:
Further, the model built from transfer learning should perform better than the scratch model as it is already pre-trained using a very deep convolution neural network.

Project workflow:

Human Face Detection:

- I have used a pretrained classifiers to classify whether a user supplied image is of a human or a dog. For identifying humans in images, I have used an open CV implementation of “**Haar feature-based cascade classifiers**” [Ref2] and for identifying whether an image is a dog or not, I’ve used VGG16, a well-known convolution neural network for image classification. Plot below shows a sample result for Haar feature-based cascade classifier.



The algorithm does a pretty good job in locating the face of Jennifer Aniston, and other human beings (not shown).

Evaluation:

In notebook, I have evaluated the recall of the first 100 human images (out of all relevant items, how many items does this classifier correctly identify?). This comes out to be about 98% which is quite good. Finally, amongst dog images it only qualifies 0.2% as the human images. So, the algorithm works fine in ignoring the dog images (not human images) as well.

Dog Detection:

I used a pretrained model of VGG16 (Very Deep Convolutional Networks for Large-Scale Image Recognition) for identifying whether image is a dog or not. The model was proposed by K. Simonyan and A. Zisserman. The model showed an accuracy of 92.7% top-5 in ImageNet challenge. The model works quite well for dog detection as well. It was able to classify 100% of dog images as dog images and classified only 1% of human images as dogs.

Dog breed Classifier:

- **Data-Preprocessing:**

  Image Size: I follow the VGG-16 network strategy for the image size, i.e. I use the final image size of 224*224. This helps us to use the same dataloader for both VGG16 and the scratch model.

  Image Augmentation:
  During the pre-processing step, we saw that thee images are of different sizes and dog-face is located in several different regions, and different orientation. Therefore, we should do a random resized crop as well as a random horizontal flip.

  Training Images: I use a random resized crop using RandomResizedCrop layer to 256 size (should be larger than final desired size). Then, I flip images horizontally randomly using the RandomHorizontalFlip layer.
  Validation and Test Images: For both validation and test images, we do not need to do a random flip as these are more for inference and checking the model accurancy. So, I just resize the images to 256x256 and do a center crop to 224x224 size.

Neural Network:

1. **Neural Network from Scratch:**

Model Architecture Implementation, Refinement and Validation:

- I started a very shallow convolution (ShallowNet) network with a combination of three convolution + max-pooling layer, and a global average pool layer followed by a fully connected layer. This network's accuracy approached 10%, but it was too slow and didn't seem to make much progress.
- Next, I tried a slightly longer neural network (MediumNet) with 4 convolutions + max-pooling layer, and a global average pool layer followed by a fully connected layer in the end. This network showed lower loss and better accuracy and could reach an accuracy of about 20 % on the validation set by epoch 20.
- Finally, I tried a even longer neural network (DeepNet) with 5 convolution + max-pooling + batch-normalization (to make the solution more stable), and a global average layer followed by a fully connected layer. This network showed the lowest loss and best accuracy and could reach about 30% accuracy by epoch 20.

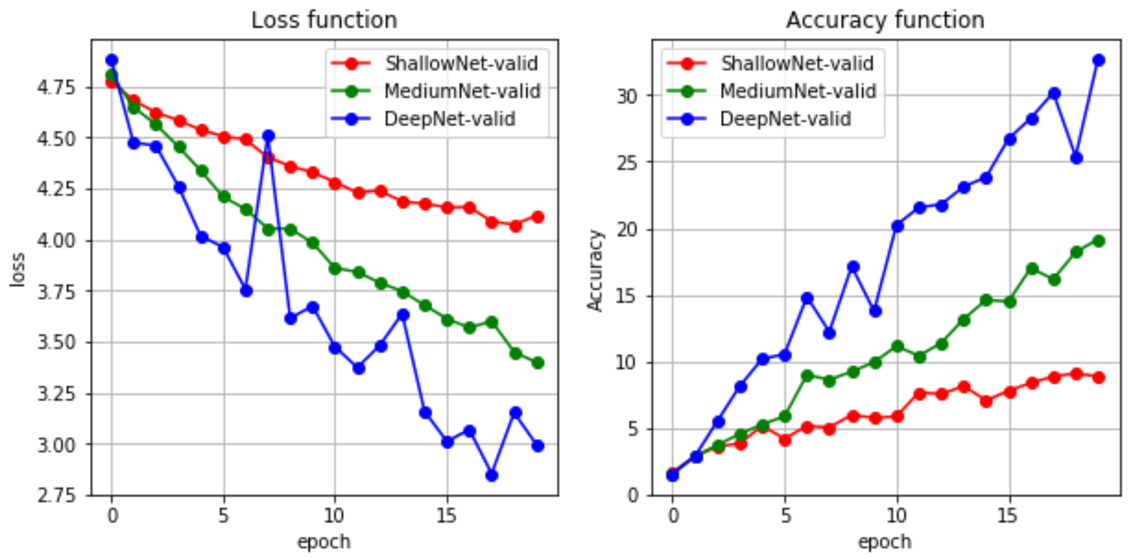


Figure 1 Plot shows a comparison of loss and accuracy of the 3 neural networks tried in this project. Since, the MediumNet and DeepNet both meed the project requirements, I didn't run it even longer for better accuracy.

Hyperparameters:
For optimizer, I use Adam optimizer which is widely used in deep learning community. It is a combination of AdaGrad (Adaptive Gradient Algorithm) and RMS Prop (Root Mean Square Propagation). This is particularly useful for problems where there are sparse gradients or we want to use adaptive learning rate based on the average of recent magnitude of the gradients.

Test set accuracy:

The final test set accuracy of the model is 30%. This beats the project guideline of 10% model accuracy, so in the interest of time and computational resources, I stop at 20 epochs. From the plot above, it is obvious that the loss has not saturated for the deep net at 20 epochs, so it is expected that the model will be even better with higher no. of epochs.

2. Neural Network using Transfer Learning:

Model Implementation:

Resnet18 model has performed well for the image net problems. Further, the example given in [Ref], shows that model performs quite well with transfer learning on the ants vs. bees dataset. This inspired me to use Resnet18 for transfer learning. Similar to reference link, I freeze all the parameters of layers in the Resnet18, and replace the last layer with a fully connected layer linear layer with 133 classes outputs (unlike tutorial where it was a binary classification). On 20 epoch runs, the model shows a validation accuracy of about 81%.

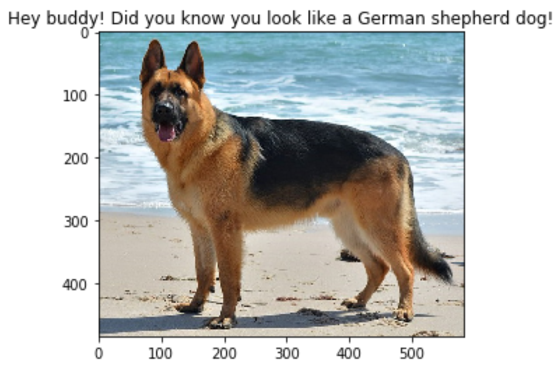
Optimizer:

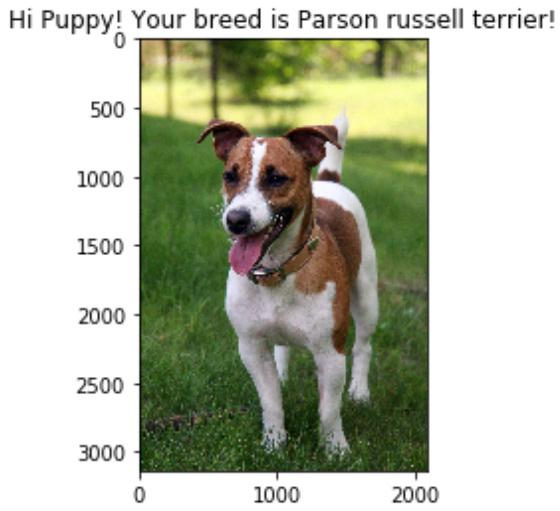
Similar to the tutorial, I try out with the Stochastic Gradient Descent Optimizer which gives us the model accuracy within desired value. SGD is a light weight optimizer as compared to Adam and does a good enough job for meeting project goals.

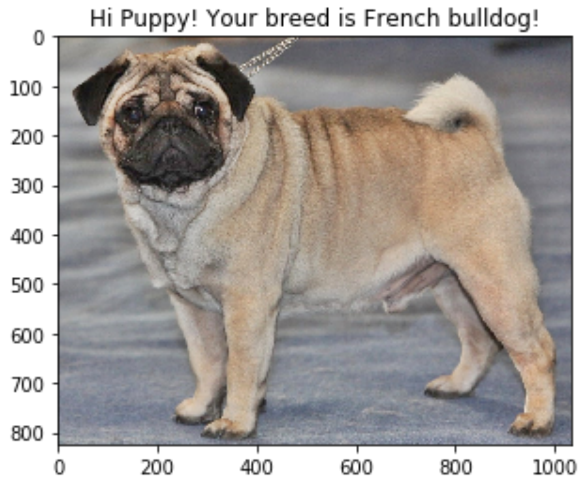
Test set accuracy:

The model shows a test set accuracy of about 80% which is much higher than our project expectations of **60%** accuracy.

Sample Results:


Hey buddy! Did you know you look like a German shepherd dog!
0
100
200
300
400
0
100
200
300
400
500





Out of the above 4 a Pug has been identified incorrectly, and rest 3 are identified correctly.

Scope for future improvements:

- We need to train our algorithm to more breeds of dogs for it to be able to differentiate between them.
- We need to let the model train for longer. For all the networks trained in this project, the loss function was still decreasing and hence we can improve on the accuracy. However, due to time constraints, I just met the project goals.
- The DeepNet is not really very deep. It still has only few layers compared to the well known classifiers. We can have a more complex architecture with better accuracy.
- I only did a manual hyperparameter search. Hence, it is not optimal. For optimal hyperparameters, we can use AWS Sagemaker or some other services.

References:

1. Efficient Net: https://arxiv.org/abs/1905.11946
2. Hass Feature Based Cascade Classifier (https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html
3. Very Deep Convolution Networks for Large-Scale Image Recognition (https://arxiv.org/pdf/1409.1556)
4. VGG in Pytorch (https://pytorch.org/hub/pytorch_vision_vgg/)
5. Image Net (http://www.image-net.org/)
6. Resnet (https://arxiv.org/abs/1512.03385
7. Transfer learning using pytorch (https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html